Manual de PHP

Stig Sæther Bakken

Alexander Aulbach

Egon Schmid

Jim Winstead

Lars Torben Wilson

Rasmus Lerdorf

Andrei Zmievski

Jouni Ahto

Editado por

Rafael Martínez (Coordinador)

Víctor Fernández

Leonardo Boshell

08-07-2002

Manual de PHP

por Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Andrei Zmievski, y Jouni Ahto

por

Editado por Rafael Martínez (Coordinador)

Editado por Víctor Fernández

Editado por Leonardo Boshell

Publicado 08-07-2002

Tabla de contenidos

Prefacio

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP.

Este manual contiene principalmente una referencia de funciones PHP, tambien contiene una referencia del lenguaje, explicaciones de caracteristicas importantes de PHP y alguna información suplementaria.

Este manual se puede conseguir en diferentes formatos en http://www.php.net/docs.php. Estos ficheros son actualizados a medida que el manual vaya cambiando. Más información sobre como este manual es desarrollado puede encontrarse en el apéndice’Sobre este manual’

Parte I. Conceptos Básicos

Capítulo 1. Introducción

Qué es PHP?

PHP (acronimo de "PHP: Hypertext Preprocessor") es un lenguaje "open source" interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

Una respuesta corta y concisa, pero que significa realmente? Un ejemplo nos aclarará las cosas:

Ejemplo 1-1. Un ejemplo introductorio

```
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
        echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

Podemos ver que no es lo mismo que un script escrito en otro lenguaje de programación como Perl o C -- En vez de escribir un programa con muchos comandos para crear una salida en HTML, escribimos el código HTML con cierto código PHP embebido (introducido) en el mismo, que producirá cierta salida (en nuestro ejemplo, producir un texto). El código PHP se incluye entre etiquetas especiales de comienzo y final que nos permitirán entrar y salir del modo PHP.

Lo que distingue a PHP de la tecnología Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. Si tuviesemos un script similar al de nuestro ejemplo en nuestro servidor, el cliente sólamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar que código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP.

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas caracteristicas avanzadas para los programadores profesionales. No tengais miedo de leer la larga lista de caracteristicas de PHP, en poco tiempo podreis empezar a escribir vuestros primeros scripts.

Aunque el desarrollo de PHP está concentrado en la programación de scripts en la parte del servidor, se puede utilizar para muchas otras cosas. Sigue leyendo y descubre más sobre PHP en la sección Qué se puede hacer con PHP?.

Qué se puede hacer con PHP?

PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o mandar y recibir cookies. Y esto no es todo, se puede hacer mucho más.

Existen tres campos en los que scripts escritos en PHP son usados.

- Scripts en la parte del servidor. Este es el campo más tradicional y el principal campo de trabajo. Se necesitan tres cosas para que esto funcione. El parseador PHP (CGI ó módulo), un servidor web y un navegador. Se necesita correr el servidor web con PHP instalado. El resultado del programa PHP se puede obtener a través del navegador, conectando con el servidor web. Consultar la sección Instrucciones de instalación para más información.
- Scripts en linea de comandos. Podeis crear un script PHP y correrlo sin ningún servidor web ó navegador. Solamente necesitais el parseador PHP para usarlo de esta manera. Este tipo de uso es ideal para scripts ejecutados regularmente desde cron (en *nix ó Linux) ó el Planificador de tareas (en Windows). Estos scripts tambien pueden ser usados para tareas simples de procesado de texto. Consultar la sección Usos de PHP en la linea de comandos para más información.
- Escribir aplicaciones gráficas clientes. PHP no es probablemente el mejor lenguaje para escribir aplicaciones gráficas, pero si sabeis bien PHP, y os gustaria utilizar algunas características avanzadas en programas clientes, podeis utilizar PHP-GTK para escribir dichos programas. Es tambien posible escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal. Si te interesa PHP-GTK, puedes visitar las páginas web del projecto (http://gtk.php.net/).

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluido HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS y probablemente alguno más. PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape y iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

Asi que, con PHP teneis la libertad de escoger el sistema operativo y el servidor de vuestro gusto. Tambien teneis la posibilidad de usar programación de procediminetos ó programación orientada a objetos. Aunque no todas la características estándares de la programación orientada a objetos están implementadas en la versión actual de PHP, muchas librerías y aplicaciones grandes (incluyendo la libreria PEAR) están escritas íntegramente usando programación orientada a objetos.

Con PHP no estais limitados a resultados en HTML. Entre las habilidades de PHP se incluyen, creación de imágenes, ficheros PDF y películas Flash (usando libswf y Ming) sobre la marcha. Tambien podeis presentar otros resultados, como XHTM y ficheros XML. PHP puede autogenerar estos ficheros y grabarlos en el sistema de ficheros en vez de presentarlos en la pantalla.

Quizas la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz via web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente:

Adabas D Ingres Oracle (OCI7 and OCI8)

dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

Tambien tenemos una extensión DBX de abstracción de base de datos que permite usar de forma transparente cualquier base de datos soportada por la extension. Adicionalmente, PHP soporta ODBC (The Open Database Connection standard), asi que podeis conectar a cualquier base de datos que soporte este estandar.

PHP tambien tiene soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. Tambien se pueden crear raw sockets. PHP soporta WDDX para intercambio de datos entre lenguajes de programación en web. Y hablando de interconexión, PHP puede utilizar objetos Java de forma transparente como objetos PHP Y la extensión de CORBA puede ser utilizada para acceder a objetos remotos.

PHP tiene unas características muy útiles para el proceso de texto, desde expresiones regulares POSIX Extended ó Perl hasta parseador de documentos XML. Para parsear y acceder documentos XML, soportamos los estandares SAX y DOM. Podeis utilizar la extensión XSLT para transformar documentos XML.

Si usais PHP en el campo del comercio electrónico, encontrareis muy útiles las funciones Cybercash, CyberMUT, VeriSign Payflow Pro y CCVS para vuestros programas de pago.

Para terminar, tenemos muchas otras extensiones muy interesantes, las funciones del motor de búsquedas mnoGoSearch, funciones para pasarelas de IRC, utilidades de compresión (gzip, bz2)„ conversión de calendarios, traducción .....

Como podeis ver esta página no es suficiente para enumerar todas las características y beneficios que PHP ofrece. Consultar las secciones Instalando PHP y Referencia de las funciones para una explicación de las extensiones mencionadas aqui.

Capítulo 2. Instalación

Bajándose la última versión

El código fuente y las distribuciones binarias para algunas plataformas (incluído Windows) se pueden encontrar en `http://www.php.net/`.

Instalación en sistemas UNIX

Esta sección le guiará a través de la configuración e instalación del PHP. Conocimientos y software necesarios:

- Habilidades básicas en UNIX (ser capaz de manejar el "make" y un compilador de C)
- Un compilador ANSI de C
- Un servidor web

Instrucciones Rápidas de Instalación (Versión Módulo de Apache)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-3.0.x.tar.gz
4.  tar xvf php-3.0.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-3.0.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure --prefix=/www --activate-module=src/modules/php3/libphp3.a
13. make
14. make install

  En lugar de este paso quizás prefiera simplemente copiar el binario
  httpd encima del binario existente. Si lo hace, asegúrese antes de
  cerrar su servidor.

15. cd ../php-3.0.x
16. cp php3.ini-dist /usr/local/lib/php3.ini

  Puede editar el archivo /usr/local/lib/php3.ini para ajustar opciones
  del PHP. Si prefiere tenerlo en otro sitio, utilice
  --with-config-file-path=/path en el paso 8.

17. Edite su archivo httpd.conf o srm.conf y añada:
```

```
            AddType application/x-httpd-php3 .php3
```

```
  Puede elegir la extensión que desee aquí. .php3 es simplemente nuestra
  sugerencia.
```

```
18. Utilice su método habitual para iniciar el servidor Apache (debe detener
    y reiniciar el servidor, no solamente hacerlo recargarse usando una
    señal HUP o USR1.)
```

Configuración

Hay dos maneras de configurar el PHP.

- Utilizando el script de "setup" que viene con el PHP. Este script le hace una serie de preguntas (casi como el script "install" del PHP/FI 2.0) y ejecuta el "configure" al final. Para ejecutar este script, escriba **./setup**.

  Este script también creará un archivo llamado "do-conf", que contendrá las opciones pasadas a la configuración. Puede editar este archivo para cambiar algunas opciones sin tener que re-ejecutar el "setup". Escriba luego **./do-conf** para ejecutar la configuración con las nuevas opciones.

- Ejecutar el "configure" a mano. Para ver las opciones de que dispone, escriba **./configure --help**.

Los detalles sobre las distintas opciones de configuración son listados a continuación.

Módulo del Apache

Para configurar el PHP como módulo de Apache, responda "yes" a "Build as an Apache module?" (la opción --with-apache=*DIR* es la que lo configura) y especifique el directorio base de la distribución de Apache. Si ha desempacado el Apache en `/usr/local/www/apache_1.2.4`, este será su directorio base de la distribución de Apache. El directorio por defecto es `/usr/local/etc/httpd`.

Módulo fhttpd

Para configurar el PHP como módulo fhttpd, responda "yes" a "Build as an fhttpd module?" (la opción --with-fhttpd=*DIR* es la que lo configura) y especifique el directorio base del fuente del fhttpd. El directorio por defecto es `/usr/local/src/fhttpd`. Si está ejecutando fhttpd, configurar PHP como módulo le dará mejor rendimiento, más control y capacidad de ejecución remota.

CGI version

El valor por defecto es configurar el PHP como programa CGI. Si está ejecutando un servidor web para el que el PHP tiene soporte como módulo, debería elegir dicha solución por motivos de rendimiento. Sin embargo, la versión CGI permite a los usuarios del Apache el ejecutar distintas páginas con PHP bajo distintos identificadores de usuario. Por favor, asegúrese de haber leído el capítulo sobre Seguridad si va a ejecutar el PHP como CGI.

Opciones de soporte para Base de Datos

El PHP tiene soporte nativo para bastantes bases de datos (así como para ODBC):

Adabas D

```
--with-adabas=DIR
```

Compila con soporte para Adabas D. El parámetro es el directorio de instalación de Adabas D y por defecto vale `/usr/local/adabasd`.

Página de Adabas (http://www.adabas.com/)

dBase

```
--with-dbase
```

Habilita el soporte integrado para DBase. No se precisan librerías externas.

filePro

```
--with-filepro
```

Habilita el soporte integrado de sólo lectura para filePro. No se precisan librerías externas.

mSQL

```
--with-msql=DIR
```

Habilita el soporte para mSQL. El parámetro es el directorio de instalación de mSQL y por defecto vale `/usr/local/Hughes`. Este es el directorio por defecto de la distribución mSQL 2.0. **configure** detecta automáticamente qué versión de mSQL está ejecutándose y el PHP soporta tanto 1.0 como 2.0, pero si compila el PHP con mSQL 1.0 sólo podrá acceder a bases de datos de esa versión y viceversa.

Vea también Directivas de Configuración de mSQL en el archivo de configuración.

Página de mSQL (http://www.hughes.com.au)

MySQL

`--with-mysql=`*`DIR`*

Habilita el soporte para MySQL. El parámetro es el directorio de instalación de MySQL y por defecto vale `/usr/local`. Este es el directorio de instalación de la distribución de MySQL.

Vea también Directivas de Configuración de MySQL en el archivo de configuración.

Página de MySQL (http://www.tcx.se)

iODBC

`--with-iodbc=`*`DIR`*

Incluye soporte para iODBC. Esta característica se desarrolló inicialmente para el iODBC Driver Manager, un gestor de controlador de ODBC de redistribución libre que ese ejecuta bajo varios sabores de UNIX. El parámetro es el directorio de instalación de iODBC y por defecto vale `/usr/local`.

Página de FreeODBC (http://users.ids.net/~bjepson/freeODBC/) o página de iODBC (http://www.iodbc.org)

OpenLink ODBC

`--with-openlink=`*`DIR`*

Incluye soporte para OpenLink ODBC. El parámetro es el directorio de instalación de OpenLink ODBC y por defecto vale `/usr/local/openlink`.

Página de OpenLink Software (http://www.openlinksw.com/)

Oracle

`--with-oracle=`*`DIR`*

Incluye soporte para Oracle. Se ha probado y debería funcionar al menos con las versiones de la 7.0 a la 7.3. El parámetro es el directorio ORACLE_HOME. No necesita especificar este parámetro si su entorno de Oracle ya está ajustado.

Página de Oracle (http://www.oracle.com)

PostgreSQL

`--with-pgsql=`*`DIR`*

Incluye soporte para PostgreSQL. El parámetro es el directorio base de la instalación de PostgreSQL y por defecto vale `/usr/local/pgsql`.

Vea también Directivas de Configuración de Postgres en el archivo de configuración.

Página de PostgreSQL (http://www.postgreSQL.org/)

Solid

`--with-solid=`*`DIR`*

Incluye soporte para Solid. El parámetro es el directorio de instalación y vale por defecto `/usr/local/solid`.

Página de Solid (http://www.solidtech.com)

Sybase

`--with-sybase=`*`DIR`*

Incluye soporte para Sybase. El parámetro es el directorio de instalación y vale por defecto `/home/sybase`.

Vea también Directivas de Configuración de Sybase en el archivo de configuración.

Página de Sybase (http://www.sybase.com)

Sybase-CT

`--with-sybase-ct=`*`DIR`*

Incluye soporte para Sybase-CT. El parámetro es el directorio de instalación de Sybase-CT y por defecto vale `/home/sybase`.

Vea también Directovas de Configuración de Sybase-CT en el archivo de configuración.

Velocis

`--with-velocis=`*`DIR`*

Incluye soporte para Velocis. El parámetro es el directorio de instalación de Velocis y vale por defecto `/usr/local/velocis`.

Página de Velocis (http://www.raima.com)

Una librería a medida de ODBC

```
--with-custom-odbc=DIR
```

Incluye soporte para una librería a medida arbitraria de ODBC. El parámetro es el directorio base y por defecto vale `/usr/local`.

Esta opción implica que se ha definido CUSTOM_ODBC_LIBS cuando se ejecutó el script de configuración. También deberá tener una cabecera odbc.h válida en algún lugar de su sendero (path) de inclusión. Si no tiene uno, créelo e incluya su cabecera específica desde ahí. Su cabecera puede requerir algunas definiciones extra, particularmente si es multiplataforma. Defínalas en CFLAGS.

Por ejemplo, puede usar Sybase SQL Anywhere bajo QNX como sigue: `CFLAGS=-DODBC_QNX LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc" ./configure --with-custom-odbc=/usr/lib/sqlany50`

ODBC Unificado

```
--disable-unified-odbc
```

Deshabilita el módulo de ODBC Unificado, que es un interfaz común a todas las bases de datos con interfaces basados en ODBC, tales como Solid y Adabas D. También funciona para librerías normales de ODBC. Ha sido probado con iODBC, Solid, Adabas D y Sybase SQL Anywhere. Requiere que uno (y sólo uno) de estos módulos o el módulo de Velocis esté habilitado, o que se especifique una librería a medida de ODBC. Esta opción sólo se puede aplicar si alguna de estas opciones es usada: --with-iodbc, --with-solid, --with-adabas, --with-velocis, o --with-custom-odbc.

Vea también Directivas de Configuración de ODBC Unificado en el archivo de configuración.

LDAP

```
--with-ldap=DIR
```

Incluye soporte para LDAP (Lightweight Directory Access Protocol - Protocolo Ligero de Acceso a Directorios). El parámetro es el directorio base de instalación de LDAP, y por defecto vale `/usr/local/ldap`.

Puede encontrar más información sobre LDAP en RFC1777 (ftp://ftp.isi.edu/in-notes/rfc1777.txt) y en RFC1778 (ftp://ftp.isi.edu/in-notes/rfc1778.txt).

Otras opciones de configuración

--with-mcrypt=*DIR*

```
--with-mcrypt
```

Incluye soporte para la librería mcrypt. Vea la documentación de mcrypt para más información. Si utiliza el argumento opcional *DIR*, el PHP buscará mcrypt.h en *DIR*/include.

--enable-sysvsem

```
--enable-sysvsem
```

Incluye soporte para semáforos Sys V (soportados por muchos derivados Unix). Vea la documentación sobre Semáforos y Memoria Compartida para más información.

--enable-sysvshm

```
--enable-sysvshm
```

Incluye soporte para la memoria compartida Sys V (soportada por muchos derivados Unix). Vea la documentación sobre Semáforos y Memoria Compartida para más información.

--with-xml

```
--with-xml
```

Incluye soporte para un parser XML no validador que utiliza la librería expat (http://www.jclark.com/xml/) de James Clark. Vea la referencia de funciones XML para más detalles.

--enable-maintainer-mode

```
--enable-maintainer-mode
```

Activa avisos extra de dependencias y del compilador utilizados por algunos de los desarrolladores del PHP.

--with-system-regex

```
--with-system-regex
```

Utiliza la librería de expresiones regulares del sistema en lugar de la incluída. Si está compilando PHP como módulo de servidor, debe utilizar la misma librería cuando genere el PHP y cuando lo enlace con el servidor. Active esto si la librería del sistema proporciona características especiales que pueda necesitar. Se recomienda utilizar la librería incluída siempre que sea posible.

--with-config-file-path

```
--with-config-file-path=DIR
```

El path utilizado para buscar el archivo de configuración cuando arranca el PHP.

--with-exec-dir

```
--with-exec-dir=DIR
```

Sólo permite ejecutar programas en DIR cuando está en modo seguro. Por defecto vale `/usr/local/bin`. Esta opción sólo fija el valor por defecto. Puede ser cambiado posteriormente mediante la directiva safe_mode_exec_dir en el fichero de configuración .

--enable-debug

```
--enable-debug
```

Habilita información de depuración adicional. Esto hace posible obtener información más detallada cuando hay problemas con el PHP. (Nótese que esto no tiene que ver con las facilidades de depuración o con la información disponible para los script PHP).

--enable-safe-mode

```
--enable-safe-mode
```

Habilita el "modo seguro" por defecto. Esto impone varias restricciones sobre lo que el PHP puede hacer, tales como abrir fichero sólo en el raiz de documentos. Lea el capítulo de Seguridad para más información. Los usuarios de CGI deberán siempre habilitar el modo seguro. Esta opción sólo fija el valor por defecto. Puede ser habilitado o deshabilitado posteriormente mediante la directiva safe_mode en el archivo de configuración.

--enable-track-vars

```
--enable-track-vars
```

Hace que el PHP lleve el control de dónde proceden las variables GET/POST/cookie usando las matrices HTTP_GET_VARS, HTTP_POST_VARS y HTTP_COOKIE_VARS. Esta opción sólo fija el valor por defecto. Puede ser habilitado o deshabilitado posteriormente mediante la directiva track_vars en el archivo de configuración.

--enable-magic-quotes

```
--enable-magic-quotes
```

Habilita las comillas mágicas por defecto. Esta opción sólo fija el valor por defecto. Puede ser habilitada o deshabilitada posteriormente mediante la directiva magic_quotes_runtime en el archivo de configuración. Vea también las directivas magic_quotes_gpc y magic_quotes_sybase.

--enable-debugger

```
--enable-debugger
```

Habilita el soporte de depuración interno del PHP. Esta característica aún está en estado experimental. Vea también las directivas de Configuración del Depurador en el archivo de configuración.

--enable-discard-path

```
--enable-discard-path
```

Si está habilitado, el ejecutable CGI del PHP se puede situar tranquilamente fuera del árbol de la web y la gente no podrá saltarse la seguridad del .htaccess. Lea la sección en el capítulo de seguridad sobre esta opción.

--enable-bcmath

```
--enable-bcmath
```

Habilita las funciones matemáticas de precisión arbitraria estilo **bc**. Vea también la opción bcmath.scale en el archivo de configuración.

--enable-force-cgi-redirect

```
--enable-force-cgi-redirect
```

Habilita la comprobación de seguridad para redirecciones internas del servidor. Deberá usar esta opción si está ejecutando la versión CGI bajo Apache.

Cuando se utiliza el PHP como un ejecutable CGI, siempre comprueba primero is está siendo utilizado bajo redirección (por ejemplo bajo Apache, usando directivas Action). Esto asegura que el ejecutable del PHP no se puede usar para saltarse los mecanismos estánder de autentificación del servidor web llamando al ejecutale directamente, como en `http://my.host/cgi-bin/php/secret/doc.html`. Este ejemplo accede al archivo `http://my.host/secret/doc.html` pero sin respetar ningún ajuste de seguridad del httpd para el directorio `/secret`.

No habilitando esta opción se deshabilita la comprobación y se permite el saltarse los ajustes de seguridad y autenticación del httpd. Haga esto sólo si el software de su servidor no puede indicar que se ha realizado una redirección segura y que todos sus archivos bajo la raiz de documentos y los directorios de los usuarios pueden ser accedidos por cualquiera.

Lea la sección en el capítulo de seguridad acerca de esta opción.

--disable-short-tags

```
--disable-short-tags
```

Deshabilita las etiquetas de PHP en formato corto `<? ?>`. Debe deshabilitar el formato corto si desea usar PHP con XML. Con el formato corto deshabilitado, la única etiqueta de código de PHP es `<?php ?>`. Esta opción sólo fija el valor por defecto. Puede ser habilitada o deshabilitada posteriormente mediante la directiva short_open_tag en el archivo de configuración.

--enable-url-includes

```
--enable-url-includes
```

Hace posible ejecutar código en otros servidores HTTP o FTP directamente desde el PHP usando include(). Vea también la opción include_path en el archivo de configuración.

--disable-syntax-hl

```
--disable-syntax-hl
```

Desconecta el resalte de sintáxis.

CPPFLAGS y LDFLAGS

Para hacer que la instalación de PHP busque los archivos de cabecera o de librería en distintos directorios, modifique las variables de entorno CPPFLAGS y LDFLAGS respectivamente. Si está utilizando un shell "sensible", podrá ejecutar **LDFLAGS=-L/my/lib/dir CPPFLAGS=-I/my/include/dir ./configure**

Construyendo

Cuando el PHP está configurado, ya está listo para construir el ejecutable CGI o la librería PERL. El comando **make** debería ocuparse de esto. Si fallara y no puede saber el motivo, vea la sección de Problemas.

Probando

Si ha construído el PHP como un programa CGI, puede probar su funcionamiento tecleando **make test**. Siempre es buena idea probar su construcción. Así puede atrapar pronto los problemas del PHP en su plataforma sin tener que batallar con ellos luego.

Comprobando la velocidad

Si ha construído el PHP como un programa CGI, puede comprobar la velocidad de su código escribiendo **make bench**. Nótese que se el modo seguro está habilitado por defecto, el test no podrá finalizar si se toma más de los 30 segundos disponibles. Esto se debe a que la función set_time_limit() no se puede usar en modo seguro. Use el ajuste de configuración max_execution_time para controlar este tiempo en sus propios script. **make bench** ignora el archivo de configuración.

Instalación en sistemas Windows 95/98/NT

Esta guía de instalación le ayudará a instalar y configurar el PHP en sus servidores web bajo Windows 9x/NT. Esta guía fue compilada por Bob Silva (mailto:bob_silva@mail.umesd.k12.or.us). La última revisión puede encontrarse en http://www.umesd.k12.or.us/php/win32install.html.

Esta guía proporciona soporte de instalacion para:

- Personal Web Server (se recomienda la última versión)
- Internet Information Server 3 ó 4
- Apache 1.3.x
- Omni HTTPd 2.0b1

Pasos Generales de Instalación

Los siguientes pasos deben realizarse en todas las instalaciones antes de las instrucciones específicas de cada servidor.

- Extraiga el archivo de distribución a un directorio de su elección. "C:\PHP3\" es un buen comienzo.
- Copie el archivo ’php3.ini-dist’ a su directorio ’%WINDOWS%’ y renómbrelo a ’php3.ini’. Su directorio ’%WINDOWS%’ es típicamente:
  c:\windows para Windows 95/98
  c:\winnt o c:\winnt40 para servidores NT
- Edite su archivo ’php3.ini’:
  - Necesitaá cambiar la opción ’extension_dir’ para que apunte a su php-install-dir, o a donde quiera que haya puesto sus archivos ’php3_*.dll’. P.ej.: c:\php3
  - Si está utilizando Omni Httpd, no siga el siguiente paso. Fije el ’doc_root’ para que apunte a la raiz web de sus servidores. P.ej.: c:\apache\htdocs o c:\webroot
  - Elija qué módulos desearía cargar cuando comience el PHP. Puede descomentar las líneas: ’extension=php3_*.dll’ para cargar estos módulos. Algunos módulos requieren que tenga instaladas en sus sistema librerías adicionales para que el módulo funcione correctamente. El FAQ (http://www.php.net/FAQ.php) de PHP tiene más información sobre dónde obtener librerías de soporte. También puede cargar un módulo dinámicamente en su script utilizando: **dl("php_*.dll");**
  - En el PWS y el IIS puede fijar el browscap.ini para que apunte a: ’c:\windows\system\inetsrv\browscap.ini’ bajo Windows 95/98 y a ’c:\winnt\system32\inetsrv\browscap.ini’ bajo NT Server.

Las DLL para las extensiones del PHP van precedidas de ’php3_’. Esto evita confusiones entre las extensiones del PHP y sus librerías de soporte.

Windows 95/98/NT y PWS/IIS 3

El método recomendado para configurar estos servidores es usar el archivo INF incluído con la distribución (php_iis_reg.inf). Quizás desee editar este archivo y asegurarse que las extensiones y directorios de instalación se ajustan a su configuración. O puede seguir los pasos que siguen para hacerlo de forma manual.

AVISO: Estos pasos conllevan el trabajar directamente con el registro de windows. Un error aquí puede dejar su sistema en un estado inestable. Le recomendamos encarecidamente que haga una copia de seguridad del registro con antelación. El equipo de Desarrollo del PHP no se hará responsable si se daña su registro.

- Ejecute Regedit.
- Navegue hasta: `HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap`.
- En el menú de edición elija: `New->String Value`.

- Escriba la extensión que desea usar para sus script PHP. P.ej.: `.php3`
- Haga doble click en el nuevo valor de cadena y escriba la ruta al `php.exe` en el campo del valor. P.ej.: `c:\php3\php.exe %s %s`. La parte '%s %s' son MUY importantes, pues el PHP no funcionará correctamente sin ella.
- Repita estos pasos para cada extensión que desee asociar con los scripts PHP.
- Ahora navegue hasta: `HKEY_CLASSES_ROOT`
- En el menú de edición elija: `New->Key`.
- Déle a la clave el nombre de la extensión que preparó en la sección anterior. P.ej.: `.php3`
- Marque la nueva clave y en el panel del lado derecho haga doble click en "default value" y escriba `phpfile`.
- Repita el último paso para cada extensión que haya preparado en la sección previa.
- Ahora cree otra `New->Key` bajo `HKEY_CLASSES_ROOT` y denomínela `phpfile`.
- Marque la nueva clave `phpfile` y haga doble click en el panel derecho sobre "default value" y escriba `PHP Script`.
- Pulse el botón derecho sobre la clave `phpfile` y seleccione `New->Key` y llámela `Shell`.
- Pulse el botón derecho sobre la clave `Shell` y elija `New->Key` y llámela `open`.
- Pulse el botón derecho sobre la clave `open` y elija `New->Key` y llámela `command`.
- Marque la nueva clave `command` y en el panel derecho haga doble click sobre "default value" y entre la ruta hasta el `php.exe`. P.ej.: `c:\php3\php.exe -q %1`. (no olvide el `%1`).
- Salga del Regedit.

Los usuarios de PWS e IIS3 tienen ahora un sistema completamente operativo. Los usuarios del IIS3 también pueden usar una curiosa herramienta (http://www.genusa.com/iis/iiscfg.html) de Steven Genusa para configurar sus mapeados de script.

Windows NT e IIS 4

Para instalar el PHP en un NT Server con IIS 4, siga estas instrucciones:

- En el Controlador de Servicios de Internet (MMC), elija el sitio Web o el directorio de comienzo de una aplicación.
- Abra las propiedades del directorio (haciendo click derecho y eligiendo propiedades) y luego pulse sobre la pestaña Carpeta Inicial, Directorio Virtual o Directorio.
- Pulse el botón Configuración y luego pulse sobre la pestaña Mapas de Aplicación.
- Pulse en Añadir, y en la caja Programa, escriba: `c:\path-to-php-dir\php.exe %s %s`. DEBE mantene los %s %s al final, pues el PHP no funcionará correctamente si se equivoca al hacerlo.
- En la caja Extensión, escriba la extensión de fichero que desea asociar a los script de PHP. Debe repetir los pasos 5 y 6 para cada extensión que desee asociar con los scripts PHP (`.php3` y `.phtml` son habituales).

- Ajuste la seguridad apropiada (esto se realiza en el Controlador de Servicio de Internet (ISM)), y si su NT Server usa el sistema de archivos NTFS, añada derechos de ejecución para I_USR_ al directorio que contenga el `php.exe`.

Windows 9x/NT y Apache 1.3.x

Debe editar sus archivos `srm.conf` o `httpd.conf` para configurar el Apache para que trabaje con el ejecutable CGI del PHP.

Aunque puede haber algunas variaciones al configurar PHP bajo Apache, esta es lo suficientemente simple para ser usada por el novato. Por favor, consulte la Documentación del Apache para saber de las subsiguientes directivas de configuración.

- `ScriptAlias /php3/ "c:/ruta-al-dir-del-php/"`
- `AddType application/x-httpd-php3 .php3`
- `AddType application/x-httpd-php3 .phtml`
- `Action application/x-httpd-php3 "/php3/php.exe"`

Para utilizar la capacidad de marcado del código fuente, cree simplemente un script de PHP y pegue este código en él: `<?php show_source("script_original_php.php3"); ?>`. Sustituya `script_original_php.php3` por el nombre del archivo del que desea visualizar el código fuente (esta es la única forma de hacerlo). *Nota:* Bajo Win-Apache todas las barras invertidas de una ruta tal como: "c:\directory\file.ext", deben ser convertidas a barras hacia adelante.

Omni HTTPd 2.0b1 para Windows

Esta ha resultado ser la configuración más sencilla:

Paso 1: Instale el servidor Omni
Paso 2: Pulse el botón derecho sobre el icono azul del OmniHTTPd que está en la barrita del sistema y elija `Propiedades`
Paso 3: Pulse sobre `Web Server Global Settings`
Paso 4: En la pestaña 'External', escriba: `virtual = .php3 | actual = c:\ruta-al-dir-del-php\php.exe`
Paso 5: En la pestaña `Mime`, escriba: `virtual = wwwserver/stdcgi | actual = .php3`
Paso 6: Pulse en `OK`

Repita los pasos 2 a 6 para cada extensión que desee asociar al PHP.

Módulos del PHP

Tabla 2-1. Módulos del PHP

php3_calendar.dll	Funciones de conversión de calendario

php3_crypt.dll	Funciones de criptografía
php3_dbase.dll	Funciones para DBase
php3_dbm.dll	Emulación GDBM con la librería Berkeley DB2
php3_filepro.dll	Acceso SÓLO LECTURA a bases de datos filepro
php3_gd.dll	Funciones de librería GD para manipular GIF
php3_hyperwave.dll	Funciones de HyperWave
php3_imap4r2.dll	Funciones de IMAP 4
php3_ldap.dll	Funciones de LDAP
php3_msql1.dll	Cliente de mSQL 1
php3_msql2.dll	Cliente de mSQL 2
php3_mssql.dll	Cliente de MSSQL client (requiere las librerías de MSSQL DB
php3_mysql.dll	Funciones de MySQL
php3_nsmail.dll	Funciones de correo de Netscape
php3_oci73.dll	Funciones de Oracle
php3_snmp.dll	Funciones get y walk de SNMP (¡sólo en NT!)
php3_zlib.dll	Funciones de ZLib

¿Problemas?

Lea las PMF (FAQ)

Algunos problemas son más comunes que otros. Los más comunes están listados en las PMF (Preguntas Más Frecuentes) del PHP, que están en http://www.php.net/FAQ.php

Informes de error

Si cree que ha encontrado un error en el PHP, por favor infórmenos. Los desarrolladores del PHP probablemente no tengan conocimiento del mismo, y salvo si informa del mismo, pocas probabilidades habrá de que lo solucionen. Puede informar de los errores usando el sistema de rastreo de errores en http://bugs.php.net/.

Otros problemas

Si aún se encuentra atascado, alguien de la lista de correos del PHP puede ser capaz de ayudarle. Deberá buscar primero en los archivos, por si acaso alguien ya ha respondido a otra persona que tuvo el mismo problema que usted. Los archivos están disponibles desde la página de soporte en http://www.php.net/.

Para suscribirse a la lista de correo de PHP, envíe un correo vacío a php-general-subscribe@lists.php.net (mailto:php-general-subscribe@lists.php.net). La dirección de la lista de correo es `php-general@lists.php.net`.

Si desea ayuda sobre la lista de correo, intente ser preciso y de los detalles necesarios sobre su entorno (qué sistema operativo, qué versión de PHP, qué servidor web, si está ejecutando el PHP como CGI o como módulo de servidor, etc.) y también código suficiente para que otros puedan reproducir y comprobar su problema.

Capítulo 3. Configuración

El archivo de configuración

El archivo de configuración (llamado `php3.ini` en PHP 3.0, y simplemente `php.ini` a partir del PHP 4.0) es leído cuando arranca el PHP. Para las versiones de PHP como módulo de servidor esto sólo ocurre una vez al arrancar el servidor web. Para la versión CGI, esto ocurre en cada llamada.

Cuando se utiliza PHP como módulo Apache, también puede cambiar los ajustes de configuración utilizando directivas en los archivos de configuración del Apache y en los .htaccess.

Con el PHP 3.0 hay directivas Apache que se corresponden a cada uno de los ajustes de configuración del `php3.ini`, con la excepción que su nombre va precedido de "php3_".

Con el PHP 4.0 sólo hay unas pocas directivas de Apache que le permiten cambiar los ajustes de configuración del PHP.

php_value *`nombre valor`*

Fija el valor de la variable especificada.

php_flag *`nombre on|off`*

Fija una opción de configuración de tipo Boolean.

php_admin_value *`nombre valor`*

Fija el valor de la variable especificada. Los ajustes de configuración de tipo "Admin" sólo se pueden fijar desde los archivos principales de configuración del Apache, y no desde los .htaccess.

php_admin_flag *`nombre on|off`*

Fija una opción de configuración de tipo Boolean.

Puede ver los ajustes de los valores de configuración en la salida de phpinfo(). También puede acceder a los valores individuales de los ajustes de configuración utilizando get_cfg_var().

Directivas Generales de Configuración

`asp_tags` boolean

Permite el uso de las etiquetas estilo ASP <% %> además de las habituales etiquetas <?php ?>. También se incluye el atajo para imprimir variables <%= $valor %>. Para más información, vea Escapando del HTML.

Nota: El soporte para etiquetas estilo ASP se añadió en la 3.0.4.

auto_append_file string

Especifica el nombre de un archivo que es troceado automáticamente después del archivo principal. El archivo se incluye como si fuese llamado mediante la función include(), así que se utiliza include_path.

El valor especial none desconecta la adición automática de archivos.

Nota: Si el script es terminado con exit(), *no* tendrá lugar la adición automática.

auto_prepend_file string

Especifica el nombre de un archivo que es troceado automáticamente antes del archivo principal. Specifies the name of a file that is automatically parsed before the main file. El archivo se incluye como si fuese llamado mediante la función include(), así que se utiliza include_path.

El valor especial none desconecta la adición automática de archivos.

cgi_ext string

display_errors boolean

Determina si los errores se visualizan en pantalla como parte de la salida en HTML o no.

doc_root string

"Directorio raiz" del PHP en el servidor. Sólo se usa si no está vacío. Si el PHP se configura con safe mode, no se sirven archivos fuera de este directorio.

engine boolean

Esta directiva sólo es realmente útil en la versión de PHP como módulo Apache. Se utiliza por sitios que desean habilitar la ejecución del PHP directorio por directorio o en base a cada servidor virtual. Poniendo **`php3_engine off`** en los sitios apropiados del archivo `httpd.conf`, se puede habilitar o deshabilitar el PHP.

error_log string

Nombre del fichero para registrar los errores de un script. Si se utiliza el valor especial `syslog`, los errores se envían al registro de errores del sistema. En UNIX se refiere a syslog(3) y en Windows NT al registro de eventos. El registro de errores del sistema no es soportado bajo Windows 95.

error_reporting integer

Fija el nivel de informe de errores. El parámetro es un entero que representa un campo de bits. Sume los valores de los niveles de informe de error que desea.

Tabla 3-1. Niveles de Informe de Errores

valor de bit	informe habilitado
1	errores normales
2	avisos normales
4	errores del troceador (parser)
8	avisos de estilo no críticos

El valor por defecto para esta directiva es 7 (se muestran los errores normales, avisos normales y errores de parser).

open_basedir string

Limita los archivos que se pueden abrir por el PHP al árbol de directorios especificado.

Cuando un script intenta abrir un archivo con, por ejemplo, fopen o gzopen, se comprueba su localización. Si el fichero está fuera del árbol de directorios especificado, PHP se negará a abrirlo. Todos los enlaces simbólicos son resueltos, de modo que no es posible evitar esta limitación usando uno de ellos.

El valor especial . indica que el directorio base será aquel en el que reside el script.

Bajo Windows, separe los directorios mediante punto y coma. En el resto de sistemas, sepárelos con dos puntos ":". Como módulo de Apache, los senderos para open_basedir de los directorios padre se heredan ahora automáticamente.

Nota: El soporte para directorios múltiples se añadió en la 3.0.7.

El valor por defecto es permitir abrir todos los archivos.

gpc_order string

Fija el order de troceo de variables GET/POST/COOKIE. El valor por defecto de esta directiva es "GPC". Fijándola, por ejemplo, a "GP", hará que el PHP ignore por completo las cookies y que sobreescriba las variables recibidas por GET con las que tengan el mismo nombre y vengan por POST.

ignore_user_abort string

Por defecto está a on. Si se cambia a off, los script terminarán tan pronto como intenten enviar algo después de que un cliente ha roto la conexión. ignore_user_abort().

include_path string

Especifica una lista de directorios en los que las funciones require(), include() y **fopen_with_path()** buscan los archivos. El formato es similar a la variable de entorno de sistema

PATH: una lista de directorios separados por dos puntos en UNIX o por punto y coma en Windows.

Ejemplo 3-1. include_path en UNIX

```
include_path=.:/home/httpd/php-lib
```

Ejemplo 3-2. include_path en Windows

```
include_path=".;c:\www\phplib"
```

El valor por defecto para esta directiva es . (sólo el directorio actual).

isapi_ext string

log_errors boolean

Dice si los mensajes de error de los script deben ser registrados o no en el registro del servidor. Esta opción, por tanto, es específica del mismo.

magic_quotes_gpc boolean

Fija el estado magic_quotes para operaciones GPC (Get/Post/Cookie). Si magic_quotes vale on, todas las ’ (comilla sencilla), " (comilla doble), \ (barra invertida) y los NUL son automáticamente marcados con una barra invertida. Si además magic_quotes_sybase vale on, la comilla sencilla es marcada con otra comilla sencilla en lugar de la barra invertida.

magic_quotes_runtime boolean

Si se habilita *magic_quotes_runtime*, muchas de las funciones que devuelven datos de algún tipo de fuente externa incluyendo bases de datos y archivos de texto devolverán las comillas marcadas con una barra invertida. Si también está activo *magic_quotes_sybase*, la comilla simple es marcada con una comilla simple en lugar de la barra invertida.

magic_quotes_sybase boolean

Si *magic_quotes_sybase* está a on, la comilla simple es marcada con una comilla simple en lugar de la barra invertida cuando están habilitados *magic_quotes_gpc* o *magic_quotes_runtime*.

max_execution_time integer

Fija el tiempo máximo en segundos que se le permite usar a un script antes de ser finalizado por el intérprete. Así se evita que scripts mal escritos puedan bloquear el servidor.

memory_limit integer

Fija el tamaño máximo de memoria en bytes que se permite reclamar a un script. Así se evita que script mal escritos se coman toda la memoria dispomible de un servidor.

nsapi_ext string

short_open_tag boolean

Indica si se debe permitir el formato corto (**<? ?>**) de la etiqueta de apertura del PHP. Si desea utilizar PHP en combinación con XML, deberá desactivar esta opción. Si está desactivada, deberá utilizar el formato largo de la etiqueta de apertura (**<?php ?>**).

sql.safe_mode boolean

track_errors boolean

Si está habilitada, el último mensaje de error estará siempre presente en la variable global $php_errormsg.

track_vars boolean

Si está activada, la información de entrada de GET, POST y de las cookies se puede encontrar en las matrices asociativas $HTTP_GET_VARS,$HTTP_POST_VARS y $HTTP_COOKIE_VARS respectivamente.

upload_tmp_dir string

El directorio temporal utilizado para almacenar archivos cuando se envían al servidor. Debe tener permiso de escritura para el usuario bajo el que corra el PHP.

user_dir string

El nombre base del directorio utilizado bajo el directorio inicial de un usuario para los archivos PHP. Por ejemplo: `paginas_html`.

warn_plus_overloading boolean

Si está activada, esta opción hace que el PHP muestre un aviso cuando el operador suma (+) se utiliza en cadenas. Así es más fácil encontrar scripts que necesitan ser reescritos utilizando en su lugar el concatenador de cadenas (.).

Directivas de Configuración de Correo

SMTP string

Nombre DNS o dirección IP del servidor de SMTP que el PHP bajo Windows deberá usar para enviar correo con la función mail().

sendmail_from string

La dirección del remitente (`"De:"`) para los correos enviados desde PHP bajo Windows.

sendmail_path string

Localización del programa **sendmail**. Generalmente `/usr/sbin/sendmail` o `/usr/lib/sendmail`. **configure** intenta localizarle este archivo lo mejor que puede y fijar un valor por defecto, pero en caso de fallo, lo puede usted fijar aquí.

Los sistemas que no usan sendmail deberán fijar esta directiva al nombre del programa alternativo que ofrezca su sistema de correo, si es que existe. Por ejemplo, los usuarios del Qmail (http://www.qmail.org/) pueden fijarlo normalmente a `/var/qmail/bin/sendmail`.

Directivas de Configuración de Modo Seguro

`safe_mode` boolean

Para activar el modo seguro del PHP. Lea el Capítulo de seguridad para más información.

`safe_mode_exec_dir` string

Si el PHP se utiliza en modo seguro, la función system() y el resto de funciones que ejecutan programas del sistema se niegan a ejecutar programas que no estén en este directorio.

Directivas de Configuración del Debugger

`debugger.host` string

Nombre DNS o dirección IP del servidor usado por el debugger.

`debugger.port` string

Número de puerto usado por el debugger.

`debugger.enabled` boolean

Indica si el debugger está habilitado o no.

Directivas de Carga de Extensiones

`enable_dl` boolean

Esta directiva sólo es útil en la versión del PHP como módulo del Apache. Puede habilitar o deshabilitar para un servidor virtual o para un directorio la carga dinámica de extensiones de PHP mediante dl().

La razón principal para deshabilitar la carga dinámica es la seguridad. Con la carga dinámica es posible ignorar las restricciones safe_mode y open_basedir.

El valor por defecto es permitir la carga dinámica, excepto cuando se usa el modo seguro. En modo seguro, siempre es imposible usar dl().

`extension_dir` string

En qué directorio debe buscar el PHP las extensiones cargables dinámicamente.

extension string

Qué extensiones dinámicas debe cargar el PHP cuando arranca.

Directivas de Configuración de MySQL

mysql.allow_persistent boolean

Si permitir o no conexiones MySQL persistentes.

mysql.default_host string

El servidor por defecto para utilizar cuando se conecte al servidor de bases de datos si no se especifica otro distinto.

mysql.default_user string

El nombre de usuario por defecto para utilizar cuando se conecta al servidor de base de datos si no se especifica otro.

mysql.default_password string

La clave por defecto para utilizar cuando se conecta al servidor de base de datos si no se especifica otro.

mysql.max_persistent integer

El número máximo de conexiones persistentes de MySQL por proceso.

mysql.max_links integer

El número máximo de conexiones de MySQL por proceso, incluyendo las persistentes.

Directivas de Configuración de mSQL

msql.allow_persistent boolean

Si se permiten o no conexiones persistentes de mSQL.

msql.max_persistent integer

El número máximo de conexiones persistentes mSQL por proceso.

msql.max_links integer

El número máximo de conexiones de mSQL por proceso, incluyendo las persistentes.

Directivas de Configuración de Postgres

pgsql.allow_persistent boolean

Si se permiten o no conexiones persistentes de Postgres.

pgsql.max_persistent integer

El número máximo de conexiones persistentes Postgres por proceso.

pgsql.max_links integer

El número máximo de conexiones de Postgres por proceso, incluyendo las persistentes.

SESAM Configuration Directives

sesam_oml string

Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. The BS2000 PLAM library must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

sesam_configfile string

Name of SESAM application configuration file. Required for using SESAM functions. The BS2000 file must be readable by the apache server's user id.

The application configuration file will usually contain a configuration like (see SESAM reference manual):

```
CNF=B
NAM=K
NOTYPE
```

sesam_messagecatalog string

Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive.

The message catalog must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

Directivas de Configuración de Sybase

`sybase.allow_persistent` boolean

Si se permiten o no conexiones persistentes de Sybase.

`sybase.max_persistent` integer

El número máximo de conexiones persistentes Sybase por proceso.

`sybase.max_links` integer

El número máximo de conexiones de Sybase por proceso, incluyendo las persistentes.

Directivas de Configuración de Sybase-CT

`sybct.allow_persistent` boolean

Si se permiten o no conexiones persistentes de Sybase-CT. El valor por defecto es on.

`sybct.max_persistent` integer

El número máximo de conexiones persistentes Sybase-CT por proceso. El valor por defecto es -1, que significa ilimitadas.

`sybct.max_links` integer

El número máximo de conexiones de Sybase-CT por proceso, incluyendo las persistentes. El valor por defecto es -1, que significa ilimitadas.

`sybct.min_server_severity` integer

Los mensajes de servidor con gravedad mayor o igual que *`sybct.min_server_severity`* serán reportados como avisos. Este valor también se puede cambiar desde un script usando la función sybase_min_server_severity(). El valor por defecto es 10, que reporta los errores de información con gravedad o mayores.

`sybct.min_client_severity` integer

Los mensajes de librería de cliente con gravedad mayor o igual que *`sybct.min_client_severity`* serán reportados como avisos. Este valor también se puede cambiar desde un script usando la función sybase_min_client_severity(). El valor por defecto es 10, que desconecta los avisos.

`sybct.login_timeout` integer

El número máximo de segundos de espera por un intento de conexión con éxito antes de indicar un fallo. Nótese que si se ha excedido *`max_execution_time`* cuando finaliza la espera de un intento de conexión, el script será finalizado antes de que se pueda tomar una acción en caso de fallo. El valor por defecto es 1 minuto.

sybct.timeout integer

El número máximo de segundos de espera por una operación de consulta o select_db con éxito antes de indicar un fallo. Nótese que si se ha excedido *max_execution_time* cuando finaliza la espera de un intento de conexión, el script será finalizado antes de que se pueda tomar una acción en caso de fallo. El valor por defecto es sin límite.

sybct.hostname string

El nombre de la máquina desde la que dice estarse conectando, para que se visualize con **sp_who()**. El valor por defecto es "none".

Directivas de Configuración de Informix

ifx.allow_persistent boolean

Si se permiten o no conexiones persistentes de Informix.

ifx.max_persistent integer

El número máximo de conexiones persistentes de Informix por proceso.

ifx.max_links integer

El número máximo de conexiones Informix por proceso, incluyendo las persistentes.

ifx.default_host string

El servidor por defecto al que conectarse si no se especifica uno en ifx_connect() o en ifx_pconnect().

ifx.default_user string

El id de usuario por defecto para utilizar si no se especifica uno en ifx_connect() o en ifx_pconnect().

ifx.default_password string

La clave por defecto para utilizar si no se especifica uno en ifx_connect() o en ifx_pconnect().

ifx.blobinfile boolean

Fíjelo a `TRUE` si desea recibir las columnas blob (objetos binarios grandes) en un archivo, y a `FALSE` si las desea en memoria. Puede cambiar el ajuste en tiempo de ejecución utilizando ifx_blobinfile_mode().

ifx.textasvarchar boolean

Fíjelo a `TRUE` si desea recibir las columnas TEXT como cadenas normales en las instrucciones select, y a `FALSE` si quiere usar parámetros de identificador de blobs. Puede cambiar el ajuste en tiempo de ejecución utilizando ifx_textasvarchar().

ifx.byteasvarchar boolean

Fíjelo a TRUE si desea devolver las columnas BYTE como cadenas normales en las instrucciones select, y a FALSE si quiere usar parámetros de identificador de blobs. Puede cambiar el ajuste en tiempo de ejecución utilizando ifx_byteasvarchar().

ifx.charasvarchar boolean

Fíjelo a TRUE si desea suprimir los espacios a la derecha de las columnas CHAR cuando las solicita.

ifx.nullformat boolean

Fíjelo a TRUE si desea que las columnas NULL (nulas) se devuelvan como la cadena literal "NULL", y a FALSE si desea que se devuelvan como la cadena vacía "". Puede cambiar el ajuste en tiempo de ejecución utilizando ifx_nullformat().

Directivas de Configuración de Matemática BC

bcmath.scale integer

Número de dígitos decimales para todas las funciones de bcmath.

Directivas de Configuración de Capacidades de los Navegadores

browscap string

Nombre del archivo de capacidades del navegador. Vea también get_browser().

Directivas Unificadas de Configuración de ODBC

uodbc.default_db string

Fuentes de datos ODBC a utilizar si no se especifica una en odbc_connect() o en odbc_pconnect().

uodbc.default_user string

Nombre de usuario si no se especifica uno en odbc_connect() o en odbc_pconnect().

uodbc.default_pw string

Clave para usar si no se especifica una en odbc_connect() o en odbc_pconnect().

uodbc.allow_persistent boolean

Si se permiten o no conexiones persistentes de ODBC.

uodbc.max_persistent integer

El número máximo de conexiones persistentes de ODBC por proceso.

uodbc.max_links integer

El número máximo de conexiones ODBC por proceso, incluyendo las persistentes.

Capítulo 4. Seguridad

PHP es un potente lenguaje y el interprete, tanto incluido en el servidor web como modulo o ejecutado como un binario CGI, puede acceder a ficheros, ejecutar comandos y abrir comunicaciones de red en el servidor. Todas estas caracteristicas hacen que lo que se ejecute en el servidor web sea inseguro por defecto. PHP ha sido disenado especificamente, para ser un lenguaje mas seguro para escribir programas CGI, que Perl o C y con la correcta seleccion de las opciones de configuración del tiempo de compilación y ejecucion se consigue la exacta combinación de libertad y seguridad que se necesita.

Ya que existen diferentes modos de utilizar PHP, existen multitud de opciones de configuración que permiten controlar su funcionamiento. Una gran selección de opciones garantiza que se pueda usar PHP para diferentes usos, pero tambien significa que existen combinaciones de estas opciones y configuraciones del servidor que producen instalaciones inseguras. Este capitulo explica las diferentes combinaciones de opciones de configuración y las situaciones donde pueden ser usadas de manera segura.

Binarios CGI

Posibles ataques

Usando PHP como un binario CGI es una opción para instalaciones que por cualquier causa no quieren integrar PHP como modulo en el software servidor (p.ej: Apache), o usaran PHP con diferentes clases de CGI wrappers para crear entornos chroot y setuid seguros para los scripts. Esta configuración implica generalmente el instalar el binario ejecutable de PHP en el directorio cgi-bin del servidor web. El documento del CERT CA-96.11 (http://www.cert.org/advisories/CA-96.11.interpreters_in_cgi_bin_dir.html) recomienda no instalar interpretes en cgi-bin. Aunque el binario PHP puede ser usado como interprete independiente, PHP esta diseñado para prevenir los ataques que esta configuración hace posible.

- Accediendo a ficheros del sistema: `http://my.host/cgi-bin/php?/etc/passwd`

  La información introducida despues del signo de interrogación (?) es transferida como argumento de la línea de comando al intérprete por el interfaz del CGI. Normalmente los interpretes abren y ejecutan el fichero especificado como el primer argumento en la línea de comando.

  Cuando se ejecuta como un CGI script, PHP rechaza interpretar los argumentos de la línea de comando.

- Accediendo cualquier documento web en el servidor: `http://my.host/cgi-bin/php/secret/doc.html`

  La información con el camino (path) de la URL despues del nombre del binario PHP, `/secret/doc.html` es usada convencionalmente para especificar el nombre del fichero que sera abierto e interpretado por el programa CGI. Normalmente, algunas directivas del servidor web (Apache:Action) son usadas para redireccionar peticiones de documentos como `http://my.host/secret/script.php3` al interprete PHP. Con esta configuración, el servidor web comprueba primero los permisos de acceso al directorio `/secret`, y despues crea la petición redireccionada `http://my.host/cgi-bin/php/secret/script.php3`. Desafortunadamente, si la petición es hecha de esta forma en un principio, el servidor web no comprueba los permisos de

acceso del fichero `/secret/script.php3`, sino solamente del fichero `/cgi-bin/php`. De esta manera cualquier usuario que pueda acceder `/cgi-bin/php` tambien puede acceder a cualquier documento protegido en el servidor web.

En PHP, a la hora de compilar, la opción de configuracion --enable-force-cgi-redirect y las directivas de configuracion a la hora de ejecutar doc_root y user_dir pueden ser usadas para prevenir este ataque, si el arbol de documentos del servidor tiene cualquier directorio con acceso restringido. Ver mas adelante la explicacion de las diferentes combinaciones.

Caso 1: solamente se sirven ficheros publicos

Si tu servidor no contiene informacion que este protegida con clave o acceso de control de IPs, no se necesitan estas opciones de configuracion. Si tu servidor web no permite realizar redireccionamientos, o el servidor no tiene modo de comunicar al binario PHP que la peticion es una peticion segura redireccionada, podeis especificar la opcion --disable-force-cgi-redirect en el script de configuracion. De todas maneras, teneis que aseguraros que vuestros scripts PHP no confíen en la manera al llamar al script, ni de forma directa `http://my.host/cgi-bin/php/dir/script.php3` o por redireccion `http://my.host/dir/script.php3`.

Redireccionamiento puede ser configurado en Apache usando las directivas AddHandler y Action (ver mas abajo).

Caso 2: usando --enable-force-cgi-redirect

Esta opcion a la hora de compilar previene que alguien llame PHP directamente con una url como la siguiente `http://my.host/cgi-bin/php/secretdir/script.php3`. PHP solamente analizara en este modo si ha pasado por una regla de redireccionamiento en el servidor.

Normalmente la redireccion en la configuracion de Apache es hecha con la siguientes directivas:

```
Action php3-script /cgi-bin/php
AddHandler php3-script .php3
```

Esta opcion ha sido solo comprobada con el servidor web Apache, y depende de Apache para fijar la variable de entorno CGI no estandar REDIRECT_STATUS en las peticiones de redireccionamiento. Si tu servidor web no soporta ningun modo para informar si una peticion es directa o redireccionada, no podeis usar esta opcion y debereis usar alguno de los otros modos de ejecucion de la version CGI documentados aqui.

Caso 3: Usando doc_root or user_dir

Incluir contenidos activos, como script y ejecutables, en el directorio de documentos del servidor web, es algunas veces considerada una practica insegura. Si por algun fallo de configuracion, los scripts no son ejecutados pero mostrados como documentos HTML, cualquiera podra conseguir codigo registrado o informacion de seguridad, como p.ej: claves de acceso. Por ello, muchos administradores prefieren

utilizar otra estructura de directorios que contenga solamente los scripts, los cuales seran solamente accesibles via PHP CGI, y por ello siempre seran interpretados y no mostrados.

Habra que tener en cuenta que si el metodo que asegura que las peticiones no son redireccionadas, como hemos descrito en la seccion anterior, no esta disponible, sera necesario configurar un script doc_root que sea diferente del "web document root".

Podeis definir el script PHP "document root" con la directiva de configuracion doc_root en el fichero de configuracion, o definir la variable de entorno PHP_DOCUMENT_ROOT. Si esta definida, la version CGI de PHP siempre obtendra el nombre del fichero a abrir con *`doc_root`* y el camino (path) utilizado en la peticion, asi podeis estar seguros que ningun script sera ejecutado fuera de este directorio (excepto para *`user_dir`*, ver a continuacion)

Otra opcion que se puede usar aqui es user_dir. Cuando user_dir no esta definido, lo unico que controla la apertura del fichero es *`doc_root`*. Si intentamos abrir una url tal como esta `http://my.host/~user/doc.php3` no se abrira un fichero en el directorio de usuarios, en su lugar se abrira un fichero llamado `~user/doc.php3` en el directorio doc_root. (si, un directorio que empieza por tilde [`~`]).

Si user_dir esta definido por ejemplo como `public_php`, una peticion tal como `http://my.host/~user/doc.php3`, abrira un fichero llamado `doc.php3` en el directorio llamado `public_php` del directorio "home" del usuario. Si el directorio del usuario es `/home/user`, el fichero ejecutado sera `/home/user/public_php/doc.php3`.

La expansion de *`user_dir`* ocurre sin tener en cuenta la configuracion de *`doc_root`*, de este modo se puede controlar los accesos al directorio principal (document root) y al directorio de usuario separadamente.

Caso 4: Analizador PHP fuera del arbol web.

Una opcion muy segura es poner el analizador binario PHP, en algun lugar fuera del arbol de ficheros web. Por ejemplo en `/usr/local/bin`. La unica pega real de esta opcion es que habra que poner una linea similar a:

```
#!/usr/local/bin/php
```

como primera linea en cualquier fichero que contenga codigo PHP. Tambien sera necesario asignar al fichero permisos de ejecucion. De esta manera, es tratado de la misma manera que cualquier otro CGI script escrito en Perl o sh o otro lenguaje utilizado para scripts y que utilicen el mecanismo `#!` para ejecutarse.

Para conseguir que PHP maneje correctamente con esta configuracion, la informacion de PATH_INFO y PATH_TRANSLATED, el analizador PHP deberia ser compilado con la opcion de configuracion --enable-discard-path.

Modulo Apache

Cuando PHP es usado como modulo Apache, hereda los permisos de usuario de Apache (normalmente "nobody")

Parte II. Referencia del Lenguaje

Capítulo 5. Síntaxis básica

Saliendo de HTML

Para interpretar un archivo, php símplemente interpreta el texto del archivo hasta que encuentra uno de los carácteres especiales que delimitan el inicio de código PHP. El intérprete ejecuta entonces todo el código que encuentra, hasta que encuentra una etiqueta de fin de código, que le dice al intérprete que siga ignorando el código siguiente. Este mecanismo permite embeber código PHP dentro de HTML: todo lo que está fuera de las etiquetas PHP se deja tal como está, mientras que el resto se interpreta como código.

Hay cuatro conjuntos de etiquetas que pueden ser usadas para denotar bloques de código PHP. De estas cuatro, sólo 2 (<?php. . .?> y <script language="php">. . .</script>) están siempre disponibles; el resto pueden ser configuradas en el fichero de `php.ini` para ser o no aceptadas por el intérprete. Mientras que el formato corto de etiquetas (short-form tags) y el estilo ASP (ASP-style tags) pueden ser convenientes, no son portables como la versión de formato largo de etiquetas. Además, si se pretende embeber código PHP en XML o XHTML, será obligatorio el uso del formato <?php. . .?> para la compatibilidad con XML.

Las etiquetas soportadas por PHP son:

Ejemplo 5-1. Formas de escapar de HTML

```
1.  <?php echo("si quieres servir documentos XHTML o XML, haz como aqu&iacute;\n"); ?>

2.  <? echo ("esta es la más simple, una instrucci&oacute;n de procesado SGML \n"); ?>
    <?= expression ?> Esto es una abreviatura de "<? echo expression ?>"

3.  <script language="php">
        echo ("muchos editores (como FrontPage) no
              aceptan instrucciones de procesado");
    </script>

4.  <% echo ("Opcionalmente, puedes usar las etiquetas ASP"); %>
    <%= $variable; # Esto es una abreviatura de "<% echo . . ." %>
```

El método primero, <?php. . .?>, es el más conveniente, ya que permite el uso de PHP en código XML como XHTML.

El método segundo no siempre está disponible. El formato corto de etiquetas está disponible con la función **short_tags()** (sólo PHP 3), activando el parámetro del fichero de configuración de PHP short_open_tag, o compilando PHP con la opción --enable-short-tags del comando **configure**. Aunque esté activa por defecto en php.ini-dist, se desaconseja el uso del formato de etiquetas corto.

El método cuarto sólo está disponible si se han activado las etiquetas ASP en el fichero de configuración: asp_tags.

> **Nota:** El soporte de etiquetas ASP se añadió en la versión 3.0.4.

> **Nota:** No se debe usar el formato corto de etiquetas cuando se desarrollen aplicaciones o librerías con intención de redistribuirlas, o cuando se desarrolle para servidores que no están bajo nuestro control, porque puede ser que el formato corto de etiquetas no esté soportado en el servidor. Para generar código portable y redistribuíble, asegúrate de no usar el formato corto de etiquetas.

La etiqueta de fin de bloque incluirá tras ella la siguiente línea si hay alguna presente. Además, la etiqueta de fin de bloque lleva implícito el punto y coma; no necesitas por lo tanto añadir el punto y coma final de la última línea del bloque PHP.

PHP permite estructurar bloques como:

Ejemplo 5-2. Métodos avanzados de escape

```
<?php
if ($expression) {
    ?>
    <strong>This is true.</strong>
    <?php
} else {
    ?>
    <strong>This is false.</strong>
    <?php
}
?>
```

Este ejemplo realiza lo esperado, ya que cuando PHP encuentra las etiquetas ?> de fin de bloque, empieza a escribir lo que encuentra tal cual hasta que encuentra otra etiqueta de inicio de bloque. El ejemplo anterior es, por supuesto, inventado. Para escribir bloques grandes de texto generamente es más eficiente separalos del código PHP que enviar todo el texto mediante las funciones echo(), print() o similares.

Separación de instrucciones

Las separación de instrucciones se hace de la misma manera que en C o Perl - terminando cada declaración con un punto y coma.

La etiqueta de fin de bloque (?>) implica el fin de la declaración, por lo tanto lo siguiente es equivalente:

```
<?php
    echo "This is a test";
?>

<?php echo "This is a test" ?>
```

Comentarios

PHP soporta el estilo de comentarios de ’C’, ’C++’ y de la interfaz de comandos de Unix. Por ejemplo:

```
<?php
    echo "This is a test"; // This is a one-line c++ style comment
    /* This is a multi line comment
       yet another line of comment */
    echo "This is yet another test";
    echo "One Final Test"; # This is shell-style style comment
?>
```

Los estilos de comentarios de una linea actualmente sólo comentan hasta el final de la linea o el bloque actual de código PHP, lo primero que ocurra.

```
<h1>This is an <?php # echo "simple";?> example.</h1>
<p>The header above will say ’This is an example’.
```

Hay que tener cuidado con no anidar comentarios de estilo ’C’, algo que puede ocurrir al comentar bloques largos de código.

```
<?php
 /*
    echo "This is a test"; /* This comment will cause a problem */
 */
?>
```

Los estilos de comentarios de una linea actualmente sólo comentan hasta el final de la linea o del bloque actual de código PHP, lo primero que ocurra. Esto implica que el código HTML tras // ?> SER??? impreso: ?> sale del modo PHP, retornando al modo HTML, el comentario // no le influye.

Capítulo 6. Types

PHP soporta los siguientes tipos:

- array
- números en punto flotante
- entero
- objeto
- cadena

El tipo de una variable normalmente no lo indica el programador; en su lugar, lo decide PHP en tiempo de ejecución dependiendo del contexto en el que se utilice esa variable.

Si se quisiese obligar a que una variable se convierta a un tipo concreto, se podría forzar la variable o usar la función settype() para ello.

Nótese que una variable se puede comportar de formas diferentes en ciertas situaciones, dependiendo de qué tipo sea en ese momento. Para más información, vea la sección Conversión de Tipos.

Enteros

Los enteros se puede especificar usando una de las siguientes sintaxis:

```
$a = 1234; # número decimal
$a = -123; # un número negativo
$a = 0123; # número octal (equivalente al 83 decimal)
$a = 0x12; # número hexadecimal (equivalente al 18 decimal)
```

Números en punto flotante

Los números en punto flotante ("double") se pueden especificar utilizando cualquiera de las siguientes sintaxis:

```
$a = 1.234; $a = 1.2e3;
```

Cadenas

Las cadenas de caracteres se pueden especificar usando uno de dos tipos de delimitadores.

Si la cadena está encerrada entre dobles comillas ("), las variables que estén dentro de la cadena serán expandidas (sujetas a ciertas limitaciones de interpretación). Como en C y en Perl, el carácter de barra invertida ("\") se puede usar para especificar caracteres especiales:

Tabla 6-1. Caracteres protegidos

secuencia	significado
`\n`	Nueva línea
`\r`	Retorno de carro
`\t`	Tabulación horizontal
`\\`	Barra invertida
`\$`	Signo del dólar
`\"`	Comillas dobles
`\[0-7]{1,3}`	la secuencia de caracteres que coincida con la expresión regular es un carácter en notación octal
`\x[0-9A-Fa-f]{1,2}`	la secuencia de caracteres que coincida con la expresión regular es un carácter en notación hexadecimal

Se puede proteger cualquier otro carácter, pero se producirá una advertencia en el nivel de depuración más alto.

La segunda forma de delimitar una cadena de caracteres usa el carácter de comilla simple ("'"). Cuando una cadena va encerrada entre comillas simples, los únicos caracteres de escape que serán comprendidos son "\\" y "\'". Esto es por convenio, así que se pueden tener comillas simples y barras invertidas en una cadena entre comillas simples. Las variables *no* se expandirán dentro de una cadena entre comillas simples.

Otra forma de delimitar cadenas es usando la sintaxis de documento incrustado ("<<<"). Se debe proporcionar un identificador después de `<<<`, después la cadena, y después el mismo identificador para cerrar el entrecomillado.

Ejemplo 6-1. He aquí un ejemplo de entrecomillado de cadenas con sintaxis de documento incrustado

```
$str = <<<EOD
Ejemplo de cadena
Expandiendo múltiples líneas
usando sintaxis de documento incrustado.
EOD;
```

Nota: La sintaxis de documento incristado fue añadida en PHP 4.

Las cadenas se pueden concatenar usando el operador '.' (punto). Nótese que el operador '+' (suma) no sirve para esto. Por favor mire Operadores de cadena para más información.

Se puede acceder a los caracteres dentro de una cadena tratándola como un array de caracteres indexado numéricamente, usando una sintaxis similar a la de C. Vea un ejemplo más abajo.

Ejemplo 6-2. Algumos ejemplos de cadenas

```
<?php
/* Asignando una cadena. */
$str = "Esto es una cadena";

/* Añadiendo a la cadena. */
$str = $str . " con algo más de texto";

/* Otra forma de añadir, incluye un carácter de nueva línea protegido. */
$str .= " Y un carácter de nueva línea al final.\n";

/* Esta cadena terminará siendo '<p>Número: 9</p>' */
$num = 9;
$str = "<p>Número: $num</p>";

/* Esta será '<p>Número: $num</p>' */
$num = 9;
$str = '<p>Número: $num</p>';

/* Obtener el primer carácter de una cadena  */
$str = 'Esto es una prueba.';
$first = $str[0];

/* Obtener el último carácter de una cadena. */
$str = 'Esto es aún una prueba.';
$last = $str[strlen($str)-1];
?>
```

Conversión de cadenas

Cuando una cadena se evalúa como un valor numérico, el valor resultante y el tipo se determinan como sigue.

La cadena se evaluará como un doble si contiene cualquiera de los caracteres '.', 'e', o 'E'. En caso contrario, se evaluará como un entero.

El valor viene dado por la porción inicial de la cadena. Si la cadena comienza con datos de valor numérico, este será el valor usado. En caso contrario, el valor será 0 (cero). Los datos numéricos válidos son un signo opcional, seguido por uno o más dígitos (que opcionalmente contengan un punto decimal), seguidos por un exponente opcional. El exponente es una 'e' o una 'E' seguidos por uno o más dígitos.

Cuando la primera expresión es una cadena, el tipo de la variable dependerá de la segunda expresión.

```
$foo = 1 + "10.5";              // $foo es doble (11.5)
$foo = 1 + "-1.3e3";            // $foo es doble (-1299)
$foo = 1 + "bob-1.3e3";         // $foo es entero (1)
$foo = 1 + "bob3";              // $foo es entero (1)
$foo = 1 + "10 Cerditos";     // $foo es entero (11)
$foo = 1 + "10 Cerditos"; // $foo es entero (11)
$foo = "10.0 cerdos " + 1;        // $foo es entero (11)
$foo = "10.0 cerdos " + 1.0;      // $foo es double (11)
```

Para más información sobre esta conversión, mire en la página del manual de Unix strtod(3).

Si quisiera probar cualquiera de los ejemplos de esta sección, puede cortar y pegar los ejemplos e insertar la siguiente línea para ver por sí mismo lo que va ocurriendo:

```
echo "\$foo==$foo; el tipo es " . gettype( $foo ) . "<br>\n";
```

Arrays

Los arrays actualmente actúan tanto como tablas hash (arrays asociativos) como arrays indexados (vectores).

Arrays unidimensionales

PHP soporta tanto arrays escalares como asociativos. De hecho, no hay diferencias entre los dos. Se puede crear una array usando las funciones list() o array(), o se puede asignar el valor de cada elemento del array de manera explícita.

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;
```

También se puede crear un array simplemente añadiendo valores al array. Cuando se asigna un valor a una variable array usando corchetes vacíos, el valor se añadirá al final del array.

```
$a[] = "hola"; // $a[2] == "hola"
$a[] = "mundo"; // $a[3] == "mundo"
```

Los arrays se pueden ordenar usando las funciones asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort(), y uksort() dependiendo del tipo de ordenación que se desee.

Se puede contar el número de elementos de un array usando la función count().

Se puede recorrer un array usando las funciones next() y prev(). Otra forma habitual de recorrer un array es usando la función each().

Arrays Multidimensionales

Los arrays multidimensionales son bastante simples actualmente. Para cada dimensión del array, se puede añadir otro valor [clave] al final:

```
$a[1]      = $f;           # ejemplos de una sola dimensión
$a["foo"]  = $f;

$a[1][0]     = $f;         # bidimensional
$a["foo"][2] = $f;         # (se pueden mezclar índices numéricos y asociativos)
$a[3]["bar"] = $f;         # (se pueden mezclar índices numéricos y asociativos)

$a["foo"][4]["bar"][0] = $f;   # tetradimensional!
```

En PHP3 no es posible referirse a arrays multidimensionales directamente dentro de cadenas. Por ejemplo, lo siguiente no tendrá el resultado deseado:

```
$a[3]['bar'] = 'Bob';
echo "Esto no va a funcionar: $a[3][bar]";
```

En PHP3, lo anterior tendrá la salida `Esto no va a funcionar: Array[bar]`. De todas formas, el operador de concatenación de cadenas se puede usar para solucionar esto:

```
$a[3]['bar'] = 'Bob';
echo "Esto no va a funcionar: " . $a[3][bar];
```

En PHP4, sin embargo, todo el problema se puede circunvenir encerrando la referencia al array (dentro de la cadena) entre llaves:

```
$a[3]['bar'] = 'Bob';
echo "Esto va a funcionar: {$a[3][bar]}";
```

Se pueden "rellenar" arrays multidimensionales de muchas formas, pero la más difícil de comprender es cómo usar el comando array() para arrays asociativos. Estos dos trozos de código rellenarán el array unidimensional de la misma manera:

```
# Ejemplo 1:

$a["color"] = "rojo";
$a["sabor"] = "dulce";
$a["forma"] = "redondeada";
$a["nombre"] = "manzana";
$a[3] = 4;

# Example 2:
$a = array(
     "color" => "rojo",
     "sabor" => "dulce",
     "forma" => "redondeada",
     "nombre"  => "manzana",
     3       => 4
);
```

La función array() se puede anidar para arrays multidimensionales:

```
<?
$a = array(
     "manzana"  => array(
          "color"  => "rojo",
          "sabor"  => "dulce",
          "forma"  => "redondeada"
     ),
     "naranja"  => array(
          "color"  => "naranja",
          "sabor"  => "ácido",
          "forma"  => "redondeada"
     ),
     "plátano"  => array(
          "color"  => "amarillo",
          "sabor"  => "paste-y",
          "forma"  => "aplatanada"
     )
);

echo $a["manzana"]["sabor"];    # devolverá "dulce"
?>
```

Objetos

Inicialización de Objetos

Para inicializar un objeto, se usa la sentencia new para instanciar el objeto a una variable.

```
class foo {
    function do_foo () {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
```

Type juggling

PHP no requiere (o soporta) la declaración explícita del tipo en la declaración de variables; el tipo de una variable se determina por el contexto en el que se usa esa variable. Esto quiere decir que si se asigna un valor de cadena a la variable *var*, *var* se convierte en una cadena. Si después se asigna un valor entero a la variable *var*, se convierte en una variable entera.

Un ejemplo de conversión de tipo automática en PHP3 es el operador suma '+'. Si cualquiera de los operandos es un doble, entonces todos los operandos se evalúan como dobles, y el resultado será un doble. En caso contrario, los operandos se interpretarán como enteros, y el resultado será también un entero. Nótese que esto NO cambia los tipos de los operandos propiamente dichos; el único cambio está en cómo se evalúan los operandos.

```
$foo = "0";  // $foo es una cadena (ASCII 48)
$foo++;      // $foo es la cadena "1" (ASCII 49)
$foo += 1;   // $foo ahora es un entero (2)
$foo = $foo + 1.3;  // $foo ahora es un doble (3.3)
$foo = 5 + "10 Cerditos Pequeñitos"; // $foo es entero (15)
$foo = 5 + "10 Cerditos";     // $foo es entero (15)
```

Si los últimos dos ejemplos anteriores parecen confusos, vea Conversión de cadenas.

Si se desea obligar a que una variable sea evaluada con un tipo concreto, mire la sección Forzado de tipos. Si se desea cambiar el tipo de una variable, vea la función settype().

Si quisiese probar cualquiera de los ejemplos de esta sección, puede cortar y pegar los ejemplos e insertar la siguiente línea para ver por sí mismo lo que va ocurriendo:

```
echo "\$foo==$foo; el tipo es " . gettype( $foo ) . "<br>\n";
```

Nota: La posibilidad de una conversión automática a array no está definida actualmente.

```
$a = 1;       // $a es un entero
$a[0] = "f";  // $a se convierte en un array, en el que $a[0] vale "f"
```

Aunque el ejemplo anterior puede parecer que claramente debería resultar en que $a se convierta en un array, el primer elemento del cual es 'f', consideremos esto:

```
$a = "1";     // $a es una cadena
$a[0] = "f";  // ¿Qué pasa con los índices de las cadenas? ¿Qué ocurre?
```

Dado que PHP soporta indexación en las cadenas vía offsets usando la misma sintaxis que la indexación de arrays, el ejemplo anterior nos conduce a un problema: ¿debería convertirse $a en un array cuyo primer elemento sea "f", o debería convertirse "f" en el primer carácter de la cadena $a?

Por esta razón, tanto en PHP 3.0.12 como en PHP 4.0b3-RC4, el resultado de esta conversión automática se considera que no está definido. Los parches se están discutiendo, de todas formas.

Forzado de tipos

El forzado de tipos en PHP funciona como en C: el nombre del tipo deseado se escribe entre paréntesis antes de la variable a la que se pretende forzar.

```
$foo = 10;   // $foo es un entero
$bar = (double) $foo;   // $bar es un doble
```

Los forzados de tipo permitidos son:

- (int), (integer) - fuerza a entero (integer)
- (real), (double), (float) - fuerza a doble (double)
- (string) - fuerza a cadena (string)
- (array) - fuerza a array (array)
- (object) - fuerza a objeto (object)

Nótese que las tabulaciones y espacios se permiten dentro de los paréntesis, así que los siguientes ejemplos son funcionalmente equivalentes:

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

Puede no ser obvio que ocurrirá cuando se fuerce entre ciertos tipos. Por ejemplo, lo siguiente debería ser tenido en cuenta.

Cuando se fuerza el cambio de un escalar o una variable de cadena a un array, la variable se convertirá en el primer elemento del array:

```
$var = 'ciao';
$arr = (array) $var;
echo $arr[0];  // produce la salida 'ciao'
```

Cuando se fuerza el tipo de una variable escalar o de una cadena a un objeto, la variable se convertirá en un atributo del objeto; el nombre del atributo será 'scalar':

```
$var = 'ciao';
$obj = (object) $var;
echo $obj->scalar;  // produce la salida 'ciao'
```

Capítulo 7. Variables

Conceptos Básicos

En PHP las variables se representan como un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var"; // produce la salida "Bob, Joe"
```

En PHP3, las variables siempre se asignan por valor. Esto significa que cuando se asigna una expresión a una variable, el valor íntegro de la expresión original se copia en la variable de destino. Esto quiere decir que, por ejemplo, después e asignar el valor de una variable a otra, los cambios que se efectúen a una de esas variables no afectará a la otra. Para más información sobre este tipo de asignación, vea Expresiones.

PHP4 ofrece otra forma de asignar valores a las variables: *asignar por referencia*. Esto significa que la nueva variable simplemente referencia (en otras palabras, "se convierte en un alias de" o "apunta a") la variable original. Los cambios a la nueva variable afectan a la original, y viceversa. Esto también significa que no se produce una copia de valores; por tanto, la asignación ocurre más rápidamente. De cualquier forma, cualquier incremento de velocidad se notará sólo en los bucles críticos cuando se asignen grandes arrays u objetos.

Para asignar por referencia, simplemente se antepone un ampersand (&) al comienzo de la variable cuyo valor se está asignando (la variable fuente). Por ejemplo, el siguiente trozo de código produce la salida 'Mi nombre es Bob' dos veces:

```
<?php
$foo = 'Bob';              // Asigna el valor 'Bob' a $foo
$bar = &$foo;              // Referencia $foo vía $bar.
$bar = "Mi nombre es $bar";  // Modifica $bar...
echo $foo;                 // $foo también se modifica.
echo $bar;
?>
```

Algo importante a tener en cuenta es que sólo las variables con nombre pueden ser asignadas por referencia.

```
<?php
$foo = 25;
$bar = &$foo;      // Esta es una asignación válida.
$bar = &(24 * 7);  // Inválida; referencia una expresión sin nombre.

function test() {
   return 25;
}

$bar = &test();    // Inválida.
```

```
?>
```

Variables predefinidas

PHP proporciona una gran cantidad de variables predefinidas a cualquier script que se ejecute. De todas formas, muchas de esas variables no pueden estar completamente documentadas ya que dependen de sobre qué servidor se esté ejecutando, la versión y configuración de dicho servidor, y otros factores. Algunas de estas variables no estarán disponibles cuando se ejecute PHP desde la línea de comandos.

A pesar de estos factores, aquí tenemos una lista de variables predefinidas disponibles en una instalación por defecto de PHP 3 corriendo como modulo de un Apache (http://www.apache.org/) 1.3.6 con su configuración también por defecto.

Para una lista de variables predefinidas (y muchas más información útil), por favor, vea (y use) phpinfo().

> **Nota:** Esta lista no es exhaustiva ni pretende serlo. Simplemente es una guía de qué tipo de variables predefinidas se puede esperar tener disponibles en un script.

Variables de Apache

Estas variables son creadas por el servidor web Apache (http://www.apache.org/). Si se está utilizando otro servidor web, no hay garantía de que proporcione las mismas variables; pueden faltar algunas, o proporcionar otras no listadas aquí. Dicho esto, también están presentes las variables de la especificación CGI 1.1 (http://hoohoo.ncsa.uiuc.edu/cgi/env.html), por lo que también se deben tener en cuenta.

Tenga en cuenta que unas pocas, como mucho, de estas variables van a estar disponibles (o simplemente tener sentido) si se ejecuta PHP desde la línea de comandos.

GATEWAY_INTERFACE

Qué revisión de la especificación CGI está usando el servidor; por ejemplo ’CGI/1.1’.

SERVER_NAME

El nombre del equipo servidor en el que se está ejecutando el script. Si el script se está ejecutando en un servidor virtual, este será el valor definido para dicho servidor virtual.

SERVER_SOFTWARE

Una cadena de identificación del servidor, que aparece en las cabeceras al responderse a las peticiones.

SERVER_PROTOCOL

Nombre y revisión del protocolo a través del que se solicitó la página; p.ej. ’HTTP/1.0’;

REQUEST_METHOD

Qué método de petición se usó para acceder a la página; p.ej. 'GET', 'HEAD', 'POST', 'PUT'.

QUERY_STRING

La cadena de la petición, si la hubo, mediante la que se accedió a la página.

DOCUMENT_ROOT

El directorio raíz del documento bajo el que se ejecuta el script, tal y como está definido en el fichero de configuración del servidor.

HTTP_ACCEPT

Los contenidos de la cabecera `Accept:` de la petición actual, si hay alguna.

HTTP_ACCEPT_CHARSET

Los contenidos de la cabecera `Accept-Charset:` de la petición actual, si hay alguna. Por ejemplo: 'iso-8859-1,*,utf-8'.

HTTP_ENCODING

Los contenidos de la cabecera `Accept-Encoding:` de la petición actual, si la hay. Por ejemplo: 'gzip'.

HTTP_ACCEPT_LANGUAGE

Los contenidos de la cabecera `Accept-Language:` de la petición actual, si hay alguna. Por ejemplo: 'en'.

HTTP_CONNECTION

Los contenidos de la cabecera `Connection:` de la petición actual, si hay alguna. Por ejemplo: 'Keep-Alive'.

HTTP_HOST

Los contenidos de la cabecera `Host:` de la petición actual, si hay alguna.

HTTP_REFERER

La dirección de la página (si la hay) desde la que el navegador saltó a la página actual. Esto lo establece el navegador del usuario; no todos los navegadores lo hacen.

HTTP_USER_AGENT

Los contenidos de la cabecera `User_Agent:` de la petición actual, si hay alguna. Indica el navegador que se está utilizando para ver la página actual; p.ej. `Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)`. Entre otras cosas, se puede usar este valor con get_browser() para adaptar la funcionalidad de la página a las posibilidades del navegador del usuario.

REMOTE_ADDR

La dirección IP desde la que el usuario está viendo la página actual.

REMOTE_PORT

El puerto que se está utilizando en la máquina del usuario para comunicarse con el servidor web.

SCRIPT_FILENAME

La vía de acceso absoluta del script que se está ejecutando.

SERVER_ADMIN

El valor que se haya dado a la directiva SERVER_ADMIN (en Apache) en el fichero de configuración del servidor web. Si el script se está ejecutando en un servidor virtual, será el valor definido para dicho servidor virtual.

SERVER_PORT

El puerto del equipo servidor que está usando el servidor web para la comunicación. Para configuraciones por defecto, será '80'; al usar SSL, por ejemplo, cambiará al puerto que se haya definido como seguro para HTTP.

SERVER_SIGNATURE

Una cadena que contiene la versión del servidor y el nombre del servidor virtual que es añadida a las páginas generadas por el servidor, si está característica está activa.

PATH_TRANSLATED

Vía de acceso basada en el sistema de ficheros- (no el directorio raíz del documento-) del script en cuestión, después de que el servidor haya hecho la conversión virtual-a-real.

SCRIPT_NAME

Contiene la vía de acceso del script actual. Es útil para páginas que necesitan apuntar a sí mismas.

REQUEST_URI

La URI que se dió para acceder a esta página; por ejemplo, '/index.html'.

Variables de entorno

Estas variables se importan en el espacio de nombres global de PHP desde el entorno en el que se esté ejecutando el intérprete PHP. Muchas son proporcionadas por el intérprete de comandos en el que se está ejecutando PHP, y dado que a sistemas diferentes les gusta ejecutar diferentes tipos de intérpretes de comandos, es imposible hacer una lista definitiva. Por favor, mire la documentación de su intérprete de comandos para ver una lista de las variables de entorno definidas.

Otras variables de entorno son las de CGI, que están ahí sin importar si PHP se está ejecutando como un módulo del servidor o como un intérprete CGI.

Variables de PHP

Estas variables son creadas por el propio PHP.

argv

Array de argumentos pasados al script. Cuando el script se ejecuta desde la línea de comandos, esto da un acceso, al estilo de C, a los parámetros pasados en línea de comandos. Cuando se le llama mediante el método GET, contendrá la cadena de la petición.

argc

Contiene el número de parámetros de la línea de comandos pasados al script (si se ejecuta desde la línea de comandos).

PHP_SELF

El nombre del fichero que contiene el script que se esta ejecutando, relativo al directorio raíz de los documentos. Si PHP se está ejecutando como intérprete de línea de comandos, esta variable no está disponible.

HTTP_COOKIE_VARS

Un array asociativo de variables pasadas al script actual mediante cookies HTTP. Sólo está disponible si el seguimiento de variables ha sido activado mediante la directiva de configuración track_vars o la directiva `<?php_track_vars?>`.

HTTP_GET_VARS

Un array asociativo de variables pasadas al script actual mediante el método HTTP GET. Sólo está disponible si --variable tracking-- ha sido activado mediante la directiva de configuración track_vars o la directiva `<?php_track_vars?>`.

HTTP_POST_VARS

Un array asociativo de variables pasadas al script actual mediante el método HTTP POST. Sólo está disponible si --variable tracking-- ha sido activado mediante la directiva de configuración track_vars o la directiva `<?php_track_vars?>`.

Ambito de las variables

El ámbito de una variable es el contexto dentro del que la variable está definida. La mayor parte de las variables PHP sólo tienen un ámbito simple. Este ámbito simple también abarca los ficheros incluidos y los requeridos. Por ejemplo:

```
$a = 1;
include "b.inc";
```

Aquí, la variable $a dentro del script incluido b.inc. De todas formas, dentro de las funciones definidas por el usuario aparece un ámbito local a la función. Cualquier variables que se use dentro de una función está, por defecto, limitada al ámbito local de la función. Por ejemplo:

```
$a = 1; /* ámbito global */

Function Test () {
    echo $a; /* referencia a una variable de ámbito local */
}

Test ();
```

Este script no producirá salida, ya que la orden echo utiliza una versión local de la variable $a, a la que no se ha asignado ningún valor en su ámbito. Puede que usted note que hay una pequeña diferencia con el lenguaje C, en el que las variables globales están disponibles automáticamente dentro de la función a menos que sean expresamente sobreescritas por una definición local. Esto puede causar algunos problemas, ya que la gente puede cambiar variables globales inadvertidamente. En PHP, las variables globales deben ser declaradas globales dentro de la función si van a ser utilizadas dentro de dicha función. Veamos un ejemplo:

```
$a = 1;
$b = 2;

Function Sum () {
    global $a, $b;

    $b = $a + $b;
}

Sum ();
echo $b;
```

El script anterior producirá la salida "3". Al declarar $a y $b globales dentro de la función, todas las referencias a tales variables se referirán a la versión global. No hay límite al número de variables globales que se pueden manipular dentro de una función.

Un segundo método para acceder a las variables desde un ámbito global es usando el array $GLOBALS propio de PHP3. El ejemplo anterior se puede reescribir así:

```
$a = 1;
$b = 2;

Function Sum () {
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Sum ();
echo $b;
```

El array $GLOBALS es un array asociativo con el nombre de la variable global como clave y los contenidos de dicha variable como el valor del elemento del array.

Otra característica importante del ámbito de las variables es la variable *static*. Una variable estática existe sólo en el ámbito local de la función, pero no pierde su valor cuando la ejecución del programa abandona este ámbito. Consideremos el siguiente ejemplo:

```
Function Test () {
    $a = 0;
    echo $a;
    $a++;
}
```

Esta función tiene poca utilidad ya que cada vez que es llamada asigna a $a el valor 0 y representa un "0". La sentencia $a++, que incrementa la variable, no sirve para nada, ya que en cuanto la función termina la variable $a desaparece. Para hacer una función útil para contar, que no pierda la pista del valor actual del conteo, la variable $a debe declararse como estática:

```
Function Test () {
    static $a = 0;
    echo $a;
    $a++;
}
```

Ahora, cada vez que se llame a la función Test(), se representará el valor de $a y se incrementará.

Las variables estáticas también proporcionan una forma de manejar funciones recursivas. Una función recursiva es la que se llama a sí misma. Se debe tener cuidado al escribir una función recursiva, ya que puede ocurrir que se llame a sí misma indefinidamente. Hay que asegurarse de implementar una forma adecuada de terminar la recursión. La siguiente función cuenta recursivamente hasta 10, usando la variable estática $count para saber cuándo parar:

```
Function Test () {
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        Test ();
    }
    $count--;
}
```

Variables variables

A veces es conveniente tener nombres de variables variables. Dicho de otro modo, son nombres de variables que se pueden establecer y usar dinámicamente. Una variable normal se establece con una sentencia como:

```
$a = "hello";
```

Una variable variable toma el valor de una variable y lo trata como el nombre de una variable. En el ejemplo anterior, *hello*, se puede usar como el nombre de una variable utilizando dos signos de dólar. p.ej.

```
$$a = "world";
```

En este momento se han definido y almacenado dos variables en el árbol de símbolos de PHP: $a, que contiene "hello", y $hello, que contiene "world". Es más, esta sentencia:

```
echo "$a ${$a}";
```

produce el mismo resultado que:

```
echo "$a $hello";
```

p.ej. ambas producen el resultado: *hello world*.

Para usar variables variables con arrays, hay que resolver un problema de ambigüedad. Si se escribe $$a[1] el intérprete necesita saber si nos referimos a utilizar $a[1] como una variable, o si se pretendía utilizar $$a como variable y el índice [1] como índice de dicha variable. La sintaxis para resolver esta ambiguedad es: ${$a[1]} para el primer caso y ${$a}[1] para el segundo.

Variables externas a PHP

Formularios HTML (GET y POST)

Cuando se envía un formulario a un script PHP, las variables de dicho formulario pasan a estar automáticamente disponibles en el script gracias a PHP. Por ejemplo, consideremos el siguiente formulario:

Ejemplo 7-1. Variables de formulario simples

```
<form action="foo.php3" method="post">
    Name: <input type="text" name="name"><br>
    <input type="submit">
</form>
```

Cuando es enviado, PHP creará la variable `$name`, que contendrá lo que sea que se introdujo en el campo *Name:* del formulario.

PHP también maneja arrays en el contexto de variables de formularios, pero sólo en una dimensión. Se puede, por ejemplo, agrupar juntas variables relacionadas, o usar esta característica para recuperar valores de un campo select input múltiple:

Ejemplo 7-2. Variables de formulario más complejas

```
<form action="array.php" method="post">
    Name: <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="beer[]">
        <option value="warthog">Warthog
        <option value="guinness">Guinness
        <option value="stuttgarter">Stuttgarter Schwabenbräu
        </select>
    <input type="submit">
</form>
```

Si la posibilidad de PHP de track_vars está activada, ya sea mediante la opción de configuración track_vars o mediante la directiva `<?php_track_vars?>`, las variables enviadas con los métodos POST o GET también se encontrarán en los arrays asociativos globales $HTTP_POST_VARS y $HTTP_GET_VARS.

IMAGE SUBMIT variable names

Cuando se envía un formulario, es posible usar una imagen en vez del botón submit estándar con una etiqueta como:

```
<input type=image src="image.gif" name="sub">
```

Cuando el usuario hace click en cualquier parte de la imagen, el formulario que la acompaña se transmitirá al servidor con dos variables adicionales, sub_x y sub_y. Estas contienen las coordenadas del click del usuario dentro de la imagen. Los más experimentados puede notar que los nombres de variable enviados por el navegador contienen un guión en vez de un subrayado (guión bajo), pero PHP convierte el guión en subrayado automáticamente.

Cookies HTTP

PHP soporta cookies de HTTP de forma transparente tal y como están definidas en en las Netscape's Spec (http://www.netscape.com/newsref/std/cookie_spec.html). Las cookies son un mecanismo para almacenar datos en el navegador y así rastrear o identificar a usuarios que vuelven. Se pueden crear cookies usando la función **SetCookie()**. Las cookies son parte de la cabecera HTTP, así que se debe llamar a la función SetCookie antes de que se envíe cualquier salida al navegador. Es la misma

restricción que para la función header(). Cualquier cookie que se reciba procedente del cliente será convertida automáticamente en una variable de PHP como con los datos en los métodos GET y POST.

Si se quieren asignar múltiples valores a una sola cookie, basta con añadir *[]* al nombre de la. Por ejemplo:

```
SetCookie ("MyCookie[]", "Testing", time()+3600);
```

Nótese que una cookie reemplazará a una cookie anterior que tuviese el mismo nombre en el navegador a menos que el camino (path) o el dominio fuesen diferentes. Así, para una aplicación de carro de la compra se podría querer mantener un contador e ir pasándolo. P.ej.

Ejemplo 7-3. SetCookie Example

```
$Count++;
SetCookie ("Count", $Count, time()+3600);
SetCookie ("Cart[$Count]", $item, time()+3600);
```

Variables de entorno

PHP hace accesibles las variables de entorno automáticamente tratándolas como variables normales.

```
echo $HOME;  /* Shows the HOME environment variable, if set. */
```

Dado que la información que llega vía mecanismos GET, POST y Cookie crean automáticamente variables de PHP, algunas veces es mejor leer variables del entorno explícitamente para asegurarse de que se está trabajando con la versión correcta. La función getenv() se puede usar para ello. También se puede asignar un valor a una variable de entorno con la función putenv().

Puntos en los nombres de variables de entrada

Típicamente, PHP no altera los nombres de las variables cuando se pasan a un script. De todas formas, hay que notar que el punto no es un carácter válido en el nombre de una variable PHP. Por esta razón, mire esto:

```
$varname.ext;  /* nombre de variable no válido */
```

Lo que el intérprete ve es el nombre de una variable $varname, seguido por el operador de concatenación, y seguido por la prueba (es decir, una cadena sin entrecomillar que no coincide con ninguna palabra clave o reservada conocida) 'ext'. Obviamente, no se pretendía que fuese este el resultado.

Por esta razón, es importante hacer notar que PHP reemplazará automáticamente cualquier punto en los nombres de variables de entrada por guiones bajos (subrayados).

Determinando los tipos de variables

Dado que PHP determina los tipos de las variables y los convierte (generalmente) según necesita, no siempre resulta obvio de qué tipo es una variable dada en un momento concreto. PHP incluye varias funciones que descubren de qué tipo es una variable. Son gettype(), is_long(), is_double(), is_string(), is_array(), y is_object().

Capítulo 8. Constantes

Una constante es un identificador para expresar un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script. (Las constantes especiales `__FILE__` y `__LINE__` son una excepción a esto, ya que actualmente no lo soin). Una constante es sensible a mayúsculas por defecto. Por convención, los identificadores de constantes suelen declararse en mayúsculas

El nombre de una constante sigue las mismas reglas que cualquier etiqueta en PHP. Un nombre de constante válido empieza con una letra o un caracter de subrayado, seguido por cualquier número de letras, números, o subrayados. Se podrían expresar mediante la siguiente expresión regular: `[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*`

> **Nota:** Para nuestros propósitos , entenderemos como letra los carácteres a-z, A-Z, y los ASCII del 127 hasta el 255 (0x7f-0xff).

El alcanze de una constante es global, Es decir, es posible acceder a ellas sin preocuparse por el ámbito de alcance.

Sintaxis

Se puede definir una constante usando la función define(). Una vez definida, no puede ser modificada ni eliminada .

Solo se puede definir como constantes valores escalares (boolean, integer, float y string).

Para obtener el valor de una constante solo es necesario especificar su nombre. A diferencia de las variables, *no* se tiene que especificar el prefijo `$`. Tambien se puede utilizar la función constant(), para obtener el valor de una constante, en el caso de que queramos expresarla de forma dinámica Usa la función get_defined_constants() parar obtener una lista de todas las constantes definidas.

> **Nota:** Las contantes y las variables (globales) se encuentran en un espacio de nombres distinto. Esto implica que por ejemplo `TRUE` y `$TRUE` son diferentes.

Si usas una constante todavia no definida, PHP asume que estás refiriéndote al nombre de la constante en si. Se lanzará un aviso si esto sucede. Usa la función defined() para comprobar la existencia de dicha constante.

Estas son las diferencias entre constantes y variables:

- Las constantes no son precedidas por un símbolo de dolar (`$`)
- Las contantes solo pueden ser definidas usando la **función()** define , nunca por simple asignación
- Las constantes pueden ser definidas y accedidas sin tener en cuenta las reglas de alcanze del ámbito.
- Las constantes no pueden ser redefinidas o eliminadas despues de establecerse; y
- Las constantes solo puede albergar valores escalares

Ejemplo 8-1. Definiendo constantes

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.
?>
```

Constantes predefinidas

PHP ofrece un largo número de constantes predefinidas a cualquier script en ejecución. Muchas de estas constantes, sin embargo, son creadas por diferentes extensiones, y solo estarán presentes si dichas extensiones están disponibles, bien por carga dinámica o porque has sido compiladas.

Se puede encontrar una lista de constantes predefinidas en la seccion Constantes predefinidas.

Capítulo 9. Expresiones

Las expresiones son la piedra angular de PHP. En PHP, casi cualquier cosa que escribes es una expresión. La forma más simple y ajustada de definir una expresión es "cualquier cosa que tiene un valor".

Las formas más básicas de expresiones son las constantes y las variables. Cuando escribes "$a = 5", estás asignando ’5’ a $a. ’5’, obviamente, tiene el valor 5 o, en otras palabras ’5’ es una expresión con el valor 5 (en este caso, ’5’ es una constante entera).

Después de esta asignación, esperarás que el valor de $a sea 5 también, de manera que si escribes $b = $a, esperas que se comporte igual que si escribieses $b = 5. En otras palabras, $a es una expresión también con el valor 5. Si todo va bien, eso es exactamente lo que pasará.

Las funciones son un ejemplo algo más complejo de expresiones. Por ejemplo, considera la siguiente función:

```
function foo () {
    return 5;
}
```

Suponiendo que estés familiarizado con el concepto de funciones (si no lo estás échale un vistazo al capítulo sobre funciones), asumirás que teclear `$c = foo()` es esencialmente lo mismo que escribir `$c = 5`, y has acertado. Las funciones son expresiones que valen el valor que retornan. Como foo() devuelve 5, el valor de la expresión ’foo()’ es 5. Normalmente las funciones no devuelven un valor fijo, sino que suele ser calculado.

Desde luego, los valores en PHP no se limitan a enteros, y lo más normal es que no lo sean. PHP soporta tres tipos escalares: enteros, punto flotante y cadenas (los tipos escalares son aquellos cuyos valores no pueden ’dividirse’ en partes menores, no como los arrays, por ejemplo). PHP también soporta dos tipos compuestos (no escalares): arrays y objetos. Se puede asignar cada uno de estos tipos de valor a variables o bien retornarse de funciones, sin ningún tipo de limitación.

Hasta aquí, los usuarios de PHP/FI 2 no deberían haber notado ningún cambio. Sin embargo, PHP lleva las expresiones mucho más allá, al igual que otros lenguajes. PHP es un lenguaje orientado a expresiones, en el sentido de que casi todo es una expresión. Considera el ejemplo anterior ’$a = 5’. Es sencillo ver que hay dos valores involucrados, el valor de la constante entera ’5’, y el valor de $a que está siendo actualizado también a 5. Pero la verdad es que hay un valor adicional implicado aquí, y es el valor de la propia asignación. La asignación misma se evalúa al valor asignado, en este caso 5. En la práctica, quiere decir que ’$a = 5’, independientemente de lo que hace, es una expresión con el valor 5. De esta manera, escribir algo como ’$b = ($a = 5)’ es como escribir ’$a = 5; $b = 5;’ (un punto y coma marca el final de una instrucción). Como las asignaciones se evalúan de derecha a izquierda, puedes escribir también ’$b = $a = 5’.

Otro buen ejemplo de orientación a expresiones es el pre y post incremento y decremento. Los usuarios de PHP/FI 2 y los de otros muchos lenguajes les sonará la notación variable++ y variable--. Esto son las operaciones de incremento y decremento. En PHP/FI 2, la instrucción ’$a++’ no tiene valor (no es una expresión), y no puedes asignarla o usarla de ningún otro modo. PHP mejora las características del incremento/decremento haciéndolos también expresiones, como en C. En PHP, como en C, hay dos tipos de incremento - pre-incremento y post-incremento. Ambos, en esencia, incrementan la variable y el efecto en la variable es idéntico. La diferencia radica en el valor de la propia expresion incremento. El preincremento , escrito ’++$variable’, se evalúa al valor incrementado (PHP incrementa la variable antes de leer su valor, de ahí el nombre ’preincremento’). El postincremento, escrito ’$variable++’, se evalúa al

valor original de $variable antes de realizar el incremento (PHP incrementa la variable después de leer su valor, de ahí el nombre 'postincremento').

Un tipo muy corriente de expresiones son las expresiones de comparación. Estas expresiones se evalúan a 0 o 1, significando FALSO (`FALSE`) o CIERTO (`TRUE`), respectivamente. PHP soporta > (mayor que), >= (mayor o igual que), == (igual que), != (distinto), < (menor que) y <= (menor o igual que). Estas expresiones se usan frecuentemente dentro de la ejecución condicional como la instrucción `if`.

El último tipo de expresiones que trataremos, es la combinación operador-asignación. Ya sabes que si quieres incrementar $a en 1, basta con escribir '$a++' o ++$a'. Pero qué pasa si quieres añadir más de 1, por ejemplo 3? Podrías escribir '$a++' múltiples veces, pero no es una forma de hacerlo ni eficiente ni cómoda. Una práctica mucho más corriente es escribir '$a = $a + 3'. '$a + 3' se evalúa al valor de $a más 3, y se asigna de nuevo a $a, lo que resulta en incrementar $a en 3. En PHP, como en otros lenguajes como C, puedes escribir esto de una forma más concisa, que con el tiempo será más clara y también fácil de entender. Añadir 3 al valor actual de $a se puede escribir como '$a += 3'. Esto quiere decir exactamente "toma el valor de $a, súmale 3, y asígnalo otra vez a $a". Además de ser más corto y claro, también resulta en una ejecución más rápida. El valor de '$a += 3', como el valor de una asignación normal y corriente, es el valor asignado. Ten en cuenta que NO es 3, sino el valor combinado de $a más 3 (ése es el valor asignado a $a). Cualquier operación binaria puede ser usada en forma de operador-asignación, por ejemplo '$a -= 5' (restar 5 del valor de $a), '$b *= 7' (multiplicar el valor de $b por 5), etc.

Hay otra expresión que puede parecer extraña si no la has visto en otros lenguaes, el operador condicional ternario:

```
$first ? $second : $third
```

Si el valor de la primera subexpresión es verdadero (distinto de cero), entonces se evalúa la segunda subexpresión, si no, se evalúa la tercera y ése es el valor.

El siguiente ejemplo te ayudará a comprender un poco mejor el pre y post incremento y las expresiones en general:

```
function double($i) {
    return $i*2;
}
$b = $a = 5;        /* asignar el valor cinco a las variables $a y $b */
$c = $a++;          /* postincremento, asignar el valor original de $a (5) a $c */
$e = $d = ++$b;     /* preincremento, asignar el valor incrementado de $b (6) a
                       $d y a $e */

/* en este punto, tanto $d como $e son iguales a 6 */

$f = double($d++);  /* asignar el doble del valor de $d antes
                       del incremento, 2*6 = 12 a $f */
$g = double(++$e);  /* asignar el doble del valor de $e después
                       del incremento, 2*7 = 14 a $g */
$h = $g += 10;      /* primero, $g es incrementado en 10 y termina valiendo 24.
                       después el valor de la asignación (24) se asigna a $h,
                       y $h también acaba valiendo 24. */
```

Al principio del capítulo hemos dicho que describiríamos los distintos tipos de instrucciones y, como prometimos, las expresiones pueden ser instrucciones. Sin embargo, no todas las expresiones son instrucciones. En este caso, una instrucción tiene la forma ’expr’ ’;’, es decir, una expresión seguida de un punto y coma. En ’$b=$a=5;’, $a=5 es una expresión válida, pero no es una instrucción en sí misma. Por otro lado ’$b=$a=5:’ sí es una instrucción válida.

Una última cosa que vale la pena mencionar, es el valor booleano de las expresiones. En muchas ocasiones, principalmente en condicionales y bucles, no estás interesado en el valor exacto de la expresión, sino únicamente si es CIERTA (`TRUE`) o FALSA (`FALSE`) (PHP no tiene un tipo booleano específico). El valor de verdad de las expresiones en PHP se calcula de forma similar a perl. Cualquier valor numérico distinto de cero es CIERTO (`TRUE`), cero es FALSO (`FALSE`). Fíjate en que los valores negativos son distinto de cero y considerados CIERTO (`TRUE`)! La cadena vacía y la cadena "0" son FALSO (`FALSE`); todas las demás cadenas son `TRUE`. Con los tipos no escalares (arrays y objetos) - si el valor no contiene elementos se considera FALSO (`FALSE`), en caso contrario se considera CIERTO (`TRUE`).

PHP te brinda una completa y potente implementación de expresiones, y documentarla enteramente está más allá del objetivo de ete manual. Los ejemplos anteriores, deberían darte una buena idea de qué son las expresiones y cómo construir expresiones útiles. A lo largo del resto del manual, escribiremos *`expr`* para indicar una expresión PHP válida.

Capítulo 10. Operadores

Operadores Aritméticos

¿Recuerdas la aritmética básica del colegio? Pues estos operadores funcionan exactamente igual.

Tabla 10-1. Operadores Aritméticos

ejemplo	nombre	resultado
$a + $b	Adición	Suma de $a y $b.
$a - $b	Substracción	Diferencia entre $a y $b.
$a * $b	Multiplicación	Producto de $a and $b.
$a / $b	División	Cociente de $a entre $b.
$a % $b	Módulo	Resto de $a dividido entre $b.

Operadores de Asignación

El operador básico de asignación es "=". A primera vista podrías pensar que es el operador de comparación "igual que". Pero no. Realmente significa que el operando de la izquierda toma el valor de la expresión a la derecha, (esto es, "toma el valor de").

El valor de una expresión de asignación es el propio valor asignado. Esto es, el valor de "$a = 3" es 3. Esto permite hacer cosas curiosas como

```
$a = ($b = 4) + 5; // ahora $a es igual a 9, y $b vale 4.
```

Además del operador básico de asignación, existen los "operadores combinados" para todas las operaciones aritméticas y de cadenas que sean binarias. Este operador combinado te permite, de una sola vez, usar una variable en una expresión y luego establecer el valor de esa variable al resultado de la expresión. Por ejemplo:

```
$a = 3;
$a += 5; // establece $a a 8, como si hubiésemos escrito: $a = $a + 5;
$b = "Hola ";
$b .= "Ahí!"; // establece $b a "Hola Ahí!", igual que si hiciésemos $b = $b . "Ahí!";
```

Fíjate en que la asignación realiza una nueva copia de la variable original (asignación por valor), por lo que cambios a la variable original no afectan a la copia. Esto puede tener interés si necesitas copiar algo como un array con muchos elementos dentro de un bucle que se repita muchas veces (cada vez se realizará una nueva copia del array). PHP4 soporta asignación por referencia, usando la sintaxis `$var = &$othervar;`, pero esto no es posible en PHP3. 'Asignación por referencia' quiere decir que ambas variables acabarán apuntando al mismo dato y que nada es realmente copiado.

Operadores Bit a bit

Los operadores bit a bit te permiten activar o desactivar bits individuales de un entero.

Tabla 10-2. Operadores Bit a bit

ejemplo	nombre	resultado
$a & $b	Y	Se activan los bits que están activos tanto en $a como $b.
$a | $b	O	Se activan los bits que están activos en $a o que lo están en $b.
$a ^ $b	Xor ("o exclusiva")	Se activan los bits que están activos en $a o en $b pero no en ambos a la vez.
~ $a	No	Se activan los bits que no están activos en $a.
$a << $b	Desplazamiento a la izquierda	Desplaza los bits de $a, $b posiciones hacia la izquierda (por aritmética binaria, cada posición desplazada equivale a multiplicar por dos el valor de $a)
$a >> $b	Desplazamiento a la derecha	Desplaza los bits de $a, $b posiciones hacia la derecha (por aritmética binaria, cada posición desplazada equivale a dividir entre dos el valor de $a)

Operadores de Comparación

Los operadores de comparación, como su nombre indica, permiten comparar dos valores.

Tabla 10-3. Operadores de Comparación

ejemplo	nombre	resultado
$a == $b	Igualdad	Cierto si $a es igual a $b.
$a === $b	Identidad	Cierto si $a es igual a $b y si son del mismo tipo (sólo PHP4)
$a != $b	Desigualdad	Cierto si $a no es igual a $b.
$a < $b	Menor que	Cierto si $a es estrictamente menor que $b.
$a > $b	Mayor que	Cierto si $a es estrictamente mayor que $b.

ejemplo	nombre	resultado
$a <= $b	Menor o igual que	Cierto si $a es menor o igual que $b.
$a >= $b	Mayor o igual que	Cierto si $a es mayor o igual que $b.

Otro operador condicional es el operador "?:" (o ternario), que funciona como en C y otros muchos lenguajes.

```
(expr1) ? (expr2) : (expr3);
```

La expresión toma el valor *`expr2`* si *`expr1`* se evalúa a cierto, y *`expr3`* si *`expr1`* se evalúa a falso.

Operador de ejecución

PHP soporta un operador de ejecución: el apóstrofe invertido (“). ¡Fíjate que no son apostrofes normales! PHP intentará ejecutar la instrucción contenida dentro de los apóstrofes invertidos como si fuera un comando del shell; y su salida devuelta como el valor de esta expresión (i.e., no tiene por qué ser simplemente volcada como salida; puede asignarse a una variable).

```
$output = `ls -al`;
echo "<pre>$output</pre>";
```

Ver también system(), passthru(), exec(), popen(), y escapeshellcmd().

Operadores de Incremento/decremento

PHP soporta los operadores de predecremento y post incremento al estilo de C.

Tabla 10-4. Operadores de Incremento/decremento

ejemplo	nombre	efecto
++$a	Preincremento	Incrementa $a en uno y después devuelve $a.
$a++	Postincremento	Devuelve $a y después incrementa $a en uno.
--$a	Predecremento	Decrementa $a en uno y después devuelve $a.
$a--	Postdecremento	Devuelve $a y después decrementa $a en uno.

He aquí un listado de ejemplo:

```
<?php
echo "<h3>Postincremento</h3>";
$a = 5;
echo "Debería ser 5: " . $a++ . "<br>\n";
echo "Debería ser 6: " . $a . "<br>\n";

echo "<h3>Preincremento</h3>";
$a = 5;
echo "Debería ser 6: " . ++$a . "<br>\n";
echo "Debería ser 6: " . $a . "<br>\n";

echo "<h3>Postdecremento</h3>";
$a = 5;
echo "Debería ser 5: " . $a-- . "<br>\n";
echo "Debería ser 4: " . $a . "<br>\n";

echo "<h3>Predecremento</h3>";
$a = 5;
echo "Debería ser 4: " . --$a . "<br>\n";
echo "Debería ser 4: " . $a . "<br>\n";
?>
```

Operadores Lógicos

Tabla 10-5. Operadores Lógicos

ejemplo	nombre	resultado
$a and $b	Y	Cierto si tanto $a como $b son ciertos.
$a or $b	O	Cierto si $a o $b son ciertos.
$a xor $b	O exclusiva	Cierto si $a es cierto o $b es cierto, pero no ambos a la vez.
! $a	Negación	Cierto si $a no es cierto.
$a && $b	Y	Cierto si tanto $a como $b son ciertos.
$a || $b	O	Cierto si $a o $b son ciertos.

La razón de las dos variaciones de "y" y "o" es que operan con distinta precedencia (ver Precedencia de Operadores.)

Precedencia de Operadores

La precedencia de operadores especifica cómo se agrupan las expresiones. Por ejemplo, en la expresión `1 + 5 * 3`, la respuesta es 16 y no 18 porque el operador de multiplicación ("*") tiene una mayor precedencia que el de adición ("+").

La siguiente tabla lista la precedencia de operadores, indicándose primero los de menor precedencia.

Tabla 10-6. Precedencia de Operadores

Asociatividad	Operadores
izquierda	,
izquierda	or
izquierda	xor
izquierda	and
derecha	print
izquierda	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
izquierda	? :
izquierda	||
izquierda	&&
izquierda	|
izquierda	^
izquierda	&
no asociativo	== != ===
no asociativo	< <= > >=
izquierda	<< >>
izquierda	+ - .
izquierda	* / %
derecha	! ~ ++ -- (int) (double) (string) (array) (object) @
derecha	[
no asociativo	new

Operadores de Cadenas

Hay dos operadores de cadenas. El primero es el operador de concatenación (’.’), que devuelve el resultado de concatenar sus operandos izquierdo y derecho. El segundo es el operador de concatenación y asignación (’.=’). Consulta Operadores de Asignación para más información.

```
$a = "Hola ";
$b = $a . "Mundo!"; // ahora $b contiene "Hola Mundo!"
```

```
$a = "Hola ";
$a .= "Mundo!"; // ahora $a contiene "Hola Mundo!"
```

Capítulo 11. Estructuras de Control

Todo archivo de comandos PHP se compone de una serie de sentencias. Una sentencia puede ser una asignación, una llamada a función, un bucle, una sentencia condicional e incluso una sentencia que no haga nada (una sentencia vacía). Las sentencias normalmente acaban con punto y coma. Además, las sentencias se pueden agrupar en grupos de sentencias encapsulando un grupo de sentencias con llaves. Un grupo de sentencias es también una sentencia. En este capítulo se describen los diferentes tipos de sentencias.

if

La construcción `if` es una de las más importantes características de muchos lenguajes, incluido PHP. Permite la ejecución condicional de fragmentos de código. PHP caracteriza una estructura `if` que es similar a la de C:

```
if (expr)
    sentencia
```

Como se describe en la sección sobre expresiones, expr se evalúa a su valor condicional. Si *`expr`* se evalúa como `TRUE`, PHP ejecutará la sentencia, y si se evalúa como `FALSE` - la ignorará.

El siguiente ejemplo mostraría `a es mayor que b` si *`$a`* fuera mayor que *`$b`*:

```
if ($a > $b)
    print "a es mayor que b";
```

A menudo, se desea tener más de una sentencia ejecutada de forma condicional. Por supuesto, no hay necesidad de encerrar cada sentencia con una cláusula `if`. En vez de eso, se pueden agrupar varias sentencias en un grupo de sentencias. Por ejemplo, este código mostraría `a es mayor que b` si *`$a`* fuera mayor que *`$b`*, y entonces asignaría el valor de *`$a`* a *`$b`*:

```
if ($a > $b) {
    print "a es mayor que b";
    $b = $a;
}
```

Las sentencias if se pueden anidar indefinidamente dentro de otras sentencias `if`, lo cual proporciona una flexibilidad completa para ejecuciones condicionales en las diferentes partes de tu programa.

else

A menudo queremos ejecutar una sentencia si se cumple una cierta condicion, y una sentencia distinta si la condición no se cumple. Esto es para lo que sirve `else`. `else` extiende una sentencia `if` para ejecutar una sentencia en caso de que la expresión en la sentencia `if` se evalúe como `FALSE`. Por ejemplo, el siguiente código mostraría `a es mayor que b` si *$a* fuera mayor que *$b*, y `a NO es mayor que b` en cualquier otro caso:

```
if ($a > $b) {
    print "a es mayor que b";
} else {
    print "a NO es mayor que b";
}
```

La sentencia `else` se ejecuta solamente si la expresión `if` se evalúa como `FALSE`, y si hubiera alguna expresión `elseif` - sólo si se evaluaron también a `FALSE` (Ver elseif).

elseif

`elseif`, como su nombre sugiere, es una combinación de `if` y `else`. Como `else`, extiende una sentencia `if` para ejecutar una sentencia diferente en caso de que la expresión `if` original se evalúa como `FALSE`. No obstante, a diferencia de `else`, ejecutará esa expresión alternativa solamente si la expresión condicional `elseif` se evalúa como `TRUE`. Por ejemplo, el siguiente código mostraría `a es mayor que b`, `a es igual a b` o `a es menor que b`:

```
if ($a > $b) {
    print "a es mayor que b";
} elseif ($a == $b) {
    print "a es igual que b";
} else {
    print "a es mayor que b";
}
```

Puede haber varios `elseif`s dentro de la misma sentencia `if`. La primera expresión `elseif` (si hay alguna) que se evalúe como `TRUE` se ejecutaría. En PHP, también se puede escribir 'else if' (con dos palabras) y el comportamiento sería idéntico al de un 'elseif' (una sola palabra). El significado sintáctico es ligeramente distinto (si estas familiarizado con C, es el mismo comportamiento) pero la línea básica es que ambos resultarían tener exactamente el mismo comportamiento.

La sentencia `elseif` se ejecuta sólo si la expresión `if` precedente y cualquier expresión `elseif` precedente se evalúan como `FALSE`, y la expresión `elseif` actual se evalúa como `TRUE`.

Sintaxis Alternativa de Estructuras de Control

PHP ofrece una sintaxis altenativa para alguna de sus estructuras de control; a saber, `if`, `while`, `for`, y `switch`. En cada caso, la forma básica de la sintaxis alternativa es cambiar abrir-llave por dos puntos (:) y cerrar-llave por `endif;`, `endwhile;`, `endfor;`, or `endswitch;`, respectivamente.

```
<?php if ($a==5): ?>
A es igual a 5
<?php endif; ?>
```

En el ejemplo de arriba, el bloque HTML "A = 5" se anida dentro de una sentencia `if` escrita en la sintaxis alternativa. El bloque HTML se mostraría solamente si $a fuera igual a 5.

La sintaxis alternativa se aplica a `else` y también a `elseif`. La siguiente es una estructura `if` con `elseif` y `else` en el formato alternativo:

```
if ($a == 5):
    print "a es igual a 5";
    print "...";
elseif ($a == 6):
    print "a es igual a 6";
    print "!!!";
else:
    print "a no es ni 5 ni 6";
endif;
```

Mirar también while, for, e if para más ejemplos.

`while`

Los bucles `while` son los tipos de bucle más simples en PHP. Se comportan como su contrapartida en C. La forma básica de una sentencia `while` es:

```
while (expr) sentencia
```

El significado de una sentencia `while` es simple. Le dice a PHP que ejecute la(s) sentencia(s) anidada(s) repetidamente, mientras la expresión `while` se evalúe como `TRUE`. El valor de la expresión es comprobado cada vez al principio del bucle, así que incluso si este valor cambia durante la ejecución de la(s) sentencia(s) anidada(s), la ejecución no parará hasta el fin de la iteración (cada vez que PHP ejecuta las sentencias en el bucle es una iteración). A veces, si la expresión `while` se evalúa como `FALSE` desde el principio de todo, la(s) sentencia(s) anidada(s) no se ejecutarán ni siquiera una vez.

Como con la sentencia `if`, se pueden agrupar multiples sentencias dentro del mismo bucle `while` encerrando un grupo de sentencias con llaves, o usando la sintaxis alternativa:

```
while (expr): sentencia ... endwhile;
```

Los siguientes ejemplos son idénticos, y ambos imprimen números del 1 al 10:

```
/* ejemplo 1 */

$i = 1;
while ($i <= 10) {
    print $i++;  /* el valor impreso sería
                    $i antes del incremento
                    (post-incremento) */
}

/* ejemplo 2 */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

do..while

Los bucles `do..while` son muy similares a los bucles `while`, excepto que las condiciones se comprueban al final de cada iteración en vez de al principio. La principal diferencia frente a los bucles regulares `while` es que se garantiza la ejecución de la primera iteración de un bucle `do..while` (la condición se comprueba sólo al final de la iteración), mientras que puede no ser necesariamente ejecutada con un bucle `while` regular (la condición se comprueba al principio de cada iteración, si esta se evalúa como `FALSE` desde el principio la ejecución del bucle finalizará inmediatamente).

Hay una sola sintaxis para los bucles `do..while`:

```
$i = 0;
do {
    print $i;
} while ($i>0);
```

El bucle de arriba se ejecutaría exactamente una sola vez, después de la primera iteración, cuando la condición se comprueba, se evalúa como `FALSE` ($i no es más grande que 0) y la ejecución del bucle finaliza.

Los usuarios avanzados de C pueden estar familiarizados con un uso distinto del bucle `do..while`, para permitir parar la ejecución en medio de los bloques de código, encapsulandolos con `do..while`(0), y usando la sentencia `break`. El siguiente fragmento de código demuestra esto:

```
do {
    if ($i < 5) {
        print "i no es lo suficientemente grande";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i es correcto";
    ...procesa i...
} while(0);
```

No se preocupes si no entiende esto completamente o en absoluto. Se pueden codificar archivos de comandos e incluso archivos de comandos potentes sin usar esta ’propiedad’.

for

Los bucles `for` son los bucles más complejos en PHP. Se comportan como su contrapartida en C. La sintaxis de un bucle `for` es:

```
for (expr1; expr2; expr3) sentencia
```

La primera expresión (*`expr1`*) se evalúa (ejecuta) incondicionalmente una vez al principio del bucle.

Al comienzo de cada iteración, se evalúa *`expr2`* . Si se evalúa como `TRUE`, el bucle continúa y las sentencias anidadas se ejecutan. Si se evalúa como `FALSE`, la ejecución del bucle finaliza.

Al final de cada iteración, se evalúa (ejecuta) *`expr3`*.

Cada una de las expresiones puede estar vacía. Que *`expr2`* esté vacía significa que el bucle debería correr indefinidamente (PHP implicitamente lo considera como `TRUE`, al igual que C). Esto puede que no sea tan inútil como se podría pensar, puesto que a menudo se quiere salir de un bucle usando una sentencia `break` condicional en vez de usar la condición de `for`.

Considera los siguientes ejemplos. Todos ellos muestran números del 1 al 10:

```
/* ejemplo 1 */
```

```
for ($i = 1; $i <= 10; $i++) {
    print $i;
}

/* ejemplo 2 */

for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}

/* ejemplo 3 */

$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}

/* ejemplo 4 */

for ($i = 1; $i <= 10; print $i, $i++) ;
```

Por supuesto, el primer ejemplo parece ser el mas elegante (o quizás el cuarto), pero uno puede descubrir que ser capaz de usar expresiones vacías en bucles `for` resulta útil en muchas ocasiones.

PHP también soporta la "sintaxis de dos puntos" alternativa para bucles `for`.

```
for (expr1; expr2; expr3): sentencia; ...; endfor;
```

Otros lenguajes poseen una sentencia `foreach` para traducir un array o una tabla hash. PHP3 no posee tal construcción; PHP4 sí (ver foreach). En PHP3, se puede combinar while con las funciones list() y each() para conseguir el mismo efecto. Mirar la documentación de estas funciones para ver un ejemplo.

foreach

PHP4 (PHP3 no) incluye una construcción `foreach`, tal como perl y algunos otros lenguajes. Esto simplemente da un modo fácil de iterar sobre arrays. Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

```
foreach(expresion_array as $value) sentencia
foreach(expresion_array as $key => $value) sentencia
```

La primera forma recorre el array dado por `expresion_array`. En cada iteración, el valor del elemento actual se asigna a `$value` y el puntero interno del array se avanza en una unidad (así en el siguiente paso, se estará mirando el elemento siguiente).

La segunda manera hace lo mismo, salvo que la clave del elemento actual será asignada a la variable `$key` en cada iteración.

> **Nota:** Cuando `foreach` comienza su primera ejecución, el puntero interno a la lista (array) se reinicia automáticamente al primer elemento del array. Esto significa que no se necesita llamar a reset() antes de un bucle `foreach`.

> **Nota:** Hay que tener en cuanta que`foreach` con una copia de la lista (array) especificada y no la lista en si, por ello el puntero de la lista no es modificado como en la construcción each.

Puede haber observado que las siguientes son funcionalidades idénticas:

```
reset( $arr );
while( list( , $value ) = each( $arr ) ) {
   echo "Valor: $value<br>\n";
}

foreach( $arr as $value ) {
   echo "Valor: $value<br>\n";
}
```

Las siguientes también son funcionalidades idénticas:

```
reset( $arr );
while( list( $key, $value ) = each( $arr ) ) {
   echo "Key: $key; Valor: $value<br>\n";
}
```

```
foreach( $arr as $key => $value ) {
   echo "Key: $key; Valor: $value<br>\n";
}
```

Algunos ejemplos más para demostrar su uso:

```
/* foreach ejemplo 1: sólo valor*/
$a = array(1, 2, 3, 17);

foreach($a as $v) {
   print "Valor actual de \$a: $v.\n";
}

/* foreach ejemplo 2: valor (con clave impresa para ilustrar) */
$a = array(1, 2, 3, 17);

$i = 0; /* sólo para propósitos demostrativos */

foreach($a as $v) {
   print "\$a[$i] => $k.\n";
}

/* foreach ejemplo 3: clave y valor */
$a = array(
   "uno" => 1,
   "dos" => 2,
   "tres" => 3,
   "diecisiete" => 17
);

foreach($a as $k => $v) {
   print "\$a[$k] => $v.\n";
}
```

break

`break` escapa de la estructuras de control iterante (bucle) actuales `for`, `while`, o `switch`.

`break` accepta un parámetro opcional, el cual determina cuantas estructuras de control hay que escapar.

```
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list (, $val) = each ($arr)) {
    if ($val == 'stop') {
        break;    /* You could also write 'break 1;' here. */
```

```
    }
    echo "$val<br>\n";
}

/* Using the optional argument. */

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "At 5<br>\n";
        break 1;  /* Exit only the switch. */
    case 10:
        echo "At 10; quitting<br>\n";
        break 2;  /* Exit the switch and the while. */
    default:
        break;
    }
}
```

continue

`continue` se usa dentro de la estructura del bucle para saltar el resto de la iteración actual del bucle y continuar la ejecución al comienzo de la siguiente iteración.

`continue` accepta un parámetro opcional, el cual determina cuantos niveles (bluces) hay que saltar antes de continuar con la ejecución.

```
 while (list($key,$value) = each($arr)) {
     if ($key % 2) { // salta los miembros impares
         continue;
     }
     do_something_odd ($value);
 }
$i = 0;
while ($i++ < 5) {
    echo "Outer<br>\n";
    while (1) {
        echo "  Middle<br>\n";
        while (1) {
            echo "  Inner<br>\n";
            continue 3;
        }
        echo "This never gets output.<br>\n";
    }
    echo "Neither does this.<br>\n";
}
```

switch

La sentencia `switch` es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, se quiere comparar la misma variable (o expresión) con nuchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para ello sirve la sentencia `switch`.

Los siguientes dos ejemplos son dos modos distintos de escribir la misma cosa, uno usa una serie de sentencias `if`, y el otro usa la sentencia `switch`:

```
if ($i == 0) {
    print "i es igual a 0";
}
if ($i == 1) {
    print "i es igual a 1";
}
if ($i == 2) {
    print "i es igual a 2";
}

switch ($i) {
    case 0:
        print "i es igual a 0";
        break;
    case 1:
        print "i es igual a 1";
        break;
    case 2:
        print "i es igual a 2";
        break;
}
```

Es importante entender cómo se ejecuta la sentencia `switch` para evitar errores. La sentencia `switch` ejecuta línea por línea (realmente, sentencia a sentencia). Al comienzo, no se ejecuta código. Sólo cuando se encuentra una sentencia `case` con un valor que coincide con el valor de la expresión `switch` PHP comienza a ejecutar las sentencias. PHP continúa ejecutando las sentencias hasta el final del bloque `switch`, o la primera vez que vea una sentencia `break`. Si no se escribe una sentencia `break` al final de una lista de sentencias case, PHP seguirá ejecutando las sentencias del siguiente case. Por ejemplo:

```
switch ($i) {
    case 0:
        print "i es igual a 0";
    case 1:
        print "i es igual a 1";
```

```
    case 2:
        print "i es igual a 2";
}
```

Aquí, si $i es igual a 0, ¡PHP ejecutaría todas las sentecias print! Si $i es igual a 1, PHP ejecutaría las últimas dos sentencias print y sólo si $i es igual a 2, se obtendría la conducta 'esperada' y solamente se mostraría 'i es igual a 2'. Así, es importante no olvidar las sentencias `break` (incluso aunque pueda querer evitar escribirlas intencionadamente en ciertas circunstancias).

En una sentencia `switch`, la condición se evalúa sólo una vez y el resultado se compara a cada sentencia `case`. En una sentencia `elseif`, la condición se evalúa otra vez. Si tu condición es más complicada que una comparación simple y/o está en un bucle estrecho, un `switch` puede ser más rápido.

La lista de sentencias de un case puede también estar vacía, lo cual simplemente pasa el control a la lista de sentencias del siguiente case.

```
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i es menor que 3, pero no negativo";
        break;
    case 3:
        print "i es 3";
}
```

Un case especial es el default case. Este case coincide con todo lo que no coincidan los otros case. Por ejemplo:

```
switch ($i) {
    case 0:
        print "i es igual a 0";
        break;
    case 1:
        print "i es igual a 1";
        break;
    case 2:
        print "i es igual a 2";
        break;
    default:
        print "i no es igual a 0, 1 o 2";
}
```

La expresión `case` puede ser cualquier expresión que se evalúe a un tipo simple, es decir, números enteros o de punto flotante y cadenas de texto. No se pueden usar aquí ni arrays ni objetos a menos que se conviertan a un tipo simple.

La sintaxis alternativa para las estructuras de control está también soportada con switch. Para más información, ver Sintaxis alternativa para estructuras de control.

```
switch ($i):
     case 0:
         print "i es igual 0";
         break;
     case 1:
         print "i es igual a 1";
         break;
     case 2:
         print "i es igual a 2";
         break;
     default:
         print "i no es igual a 0, 1 o 2";
 endswitch;
```

require()

La sentencia require() se sustituye a sí misma con el archivo especificado, tal y como funciona la directiva `#include` de C.

Un punto importante sobre su funcionamiento es que cuando un archivo se incluye con include() o se requiere con require()), el intérprete sale del modo PHP y entra en modo HTML al principio del archivo referenciado, y vuelve de nuevo al modo PHP al final. Por esta razón, cualquier código dentro del archivo referenciado que debiera ser ejecutado como código PHP debe ser encerrado dentro de etiquetas válidas de comienzo y fin de PHP.

require() no es en realidad una función de PHP; es más una construcción del lenguaje. Está sujeta a algunas reglas distintas de las de funciones. Por ejemplo, require() no esta sujeto a ninguna estructura de control contenedora. Por otro lado, no devuelve ningún valor; intentar leer un valor de retorno de una llamada a un require() resulta en un error del intérprete.

A diferencia de include(), require() *siempre* leerá el archivo referenciado, *incluso si la línea en que está no se ejecuta nunca*. Si se quiere incluir condicionalmente un archivo, se usa include(). La sentencia conditional no afecta a require(). No obstante, si la línea en la cual aparece el require() no se ejecuta, tampoco se ejecutará el código del archivo referenciado.

De forma similar, las estructuras de bucle no afectan la conducta de require(). Aunque el código contenido en el archivo referenciado está todavía sujeto al bucle, el propio require() sólo ocurre una vez.

Esto significa que no se puede poner una sentencia require() dentro de una estructura de bucle y esperar que incluya el contenido de un archivo distinto en cada iteración. Para hacer esto, usa una sentencia include().

```
require( 'header.inc' );
```

When a file is require()ed, the code it contains inherits the variable scope of the line on which the require() occurs. Any variables available at that line in the calling file will be available within the called file. If the require() occurs inside a function within the calling file, then all of the code contained in the called file will behave as though it had been defined inside that function.

If the require()ed file is called via HTTP using the fopen wrappers, and if the target server interprets the target file as PHP code, variables may be passed to the require()ed file using an URL request string as used with HTTP GET. This is not strictly speaking the same thing as require()ing the file and having it inherit the parent file's variable scope; the script is actually being run on the remote server and the result is then being included into the local script.

```
/* This example assumes that someserver is configured to parse .php
 * files and not .txt files. Also, 'works' here means that the variables
 * $varone and $vartwo are available within the require()ed file. */

/* Won't work; file.txt wasn't handled by someserver. */
require ("http://someserver/file.txt?varone=1&vartwo=2");

/* Won't work; looks for a file named 'file.php?varone=1&vartwo=2'
 * on the local filesystem. */
require ("file.php?varone=1&vartwo=2");

/* Works. */
require ("http://someserver/file.php?varone=1&vartwo=2");

$varone = 1;
$vartwo = 2;
require ("file.txt");  /* Works. */
require ("file.php");  /* Works. */
```

En PHP3, es posible ejecutar una sentencia `return` dentro de un archivo referenciado con require(), en tanto en cuanto esa sentencia aparezca en el ámbito global del archivo requerido (require()). No puede aparecer dentro de ningún bloque (lo que siginifica dentro de llaves({})). En PHP4, no obstante, esta capacidad ha sido desestimada. Si se necesita esta funcionalidad, véase include().

Ver tambien include(), require_once(), include_once(), readfile(), y virtual().

include()

La sentencia include() incluye y evalúa el archivo especificado.

Si "URL fopen wrappers" esta activada en PHP (como está en la configuración inicial), se puede especificar el fichero que se va a incluir usando una URL en vez de un fichero local (con su Path) Ver Ficheros remotos y fopen() para más información.

Un punto importante sobre su funcionamiento es que cuando un archivo se incluye con include() o se requiere con require(), el intérprete sale del modo PHP y entra en modo HTML al principio del archivo referenciado, y vuelve de nuevo al modo PHP al final. Por esta razón, cualquier código dentro del archivo referenciado que debiera ser ejecutado como código PHP debe ser encerrado dentro de etiquetas válidas de comienzo y fin de PHP.

Esto sucede cada vez que se encuentra la sentencia include(), así que se puede usar una sentencia include() dentro de una estructura de bucle para incluir un número de archivos diferentes.

```
$archivos = array ('primero.inc', 'segundo.inc', 'tercero.inc');
for ($i = 0; $i < count($archivos); $i++) {
    include $archivos[$i];
}
```

include() difiere de require() en que la sentencia include se re-evalúa cada vez que se encuentra (y sólo cuando está siendo ejecutada), mientras que la sentencia require() se reemplaza por el archivo referenciado cuando se encuentra por primera vez, se vaya a evaluar el contenido del archivo o no (por ejemplo, si está dentro de una sentencia if cuya condición evaluada es falsa).

Debido a que include() es una construcción especial del lenguaje, se debe encerrar dentro de un bloque de sentencias si está dentro de un bloque condicional.

```
/* Esto es ERRÓNEO y no funcionará como se desea. */

if ($condicion)
    include($archivo);
else
    include($otro);

/* Esto es CORRECTO. */

if ($condicion) {
    include($archivo);
} else {
    include($otro);
}
```

En ambos, PHP3 y PHP4, es posible ejecutar una sentencia `return` dentro de un archivo incluido con include(), para terminar el procesado de ese archivo y volver al archivo de comandos que lo llamó. Existen algunas diferencias en el modo en que esto funciona, no obstante. La primera es que en PHP3, `return` no puede aparecer dentro de un bloque a menos que sea un bloque de función, en el cual `return` se aplica a esa función y no al archivo completo. En PHP4, no obstante, esta restricción no existe. También, PHP4 permite devolver valores desde archivos incluidos con include(). Se puede capturar el valor de la llamada a include() como se haría con una función normal. Esto genera un error de intérprete en PHP3.

Ejemplo 11-1. include() en PHP3 y PHP4

Asumamos la existencia del siguiente archivo (llamado `test.inc`) en el mismo directorio que el archivo principal:

```
<?php
echo "Antes del return <br>\n";
if ( 1 ) {
   return 27;
}
echo "Después del return <br>\n";
?>
```

Asumamos que el archivo principal (`main.html`) contiene lo siguiente:

```
<?php
$retval = include( 'test.inc' );
echo "El archivo devolvió: '$retval'<br>\n";
?>
```

Cuando se llama a `main.html` en PHP3, generará un error del intérprete en la linea 2; no se puede capturar el valor de un include() en PHP3. En PHP4, no obstante, el resultado será:

```
Antes del return
El archivo devolvió: '27'
```

Ahora, asumamos que se ha modificado `main.html` para que contenga lo siguiente:

```
<?php
include( 'test.inc' );
echo "De vuelta en main.html<br>\n";
?>
```

En PHP4, la salida será:

```
Antes del return
De vuelta en main.html
```

No obstante, PHP3 dará la siguiente salida:

```
Antes del return
27De vuelta en main.html

Parse error: parse error in /home/torben/public_html/phptest/main.html on line 5
```

El error del intérprete es resultado del hecho de que la sentencia `return` está encerrada en un bloque de no-función dentro de `test.inc`. Cuando el return se mueve fuera del bloque, la salida es:

```
Antes del return
27De vuelta en main.html
```

El '27' espúreo se debe al hecho de que PHP3 no soporta devolver valores con `return` desde archivos como ese.

When a file is include()ed, the code it contains inherits the variable scope of the line on which the include() occurs. Any variables available at that line in the calling file will be available within the called file. If the include() occurs inside a function within the calling file, then all of the code contained in the called file will behave as though it had been defined inside that function.

If the include()ed file is called via HTTP using the fopen wrappers, and if the target server interprets the target file as PHP code, variables may be passed to the include()ed file using an URL request string as used with HTTP GET. This is not strictly speaking the same thing as include()ing the file and having it inherit the parent file's variable scope; the script is actually being run on the remote server and the result is then being included into the local script.

```
/* This example assumes that someserver is configured to parse .php
 * files and not .txt files. Also, 'works' here means that the variables
 * $varone and $vartwo are available within the include()ed file. */

/* Won't work; file.txt wasn't handled by someserver. */
include ("http://someserver/file.txt?varone=1&vartwo=2");

/* Won't work; looks for a file named 'file.php?varone=1&vartwo=2'
 * on the local filesystem. */
include ("file.php?varone=1&vartwo=2");

/* Works. */
include ("http://someserver/file.php?varone=1&vartwo=2");

$varone = 1;
$vartwo = 2;
include ("file.txt");  /* Works. */
include ("file.php");  /* Works. */
```

See also require(), require_once(), include_once(), readfile(), and virtual().

require_once()

The require_once() statement replaces itself with the specified file, much like the C preprocessor's `#include` works, and in that respect is similar to the require() statement. The main difference is that in an inclusion chain, the use of require_once() will assure that the code is added to your script only once, and avoid clashes with variable values or function names that can happen.

For example, if you create the following 2 include files `utils.inc` and `foolib.inc`

Ejemplo 11-2. utils.inc

```
<?php
define(PHPVERSION, floor(phpversion()));
echo "GLOBALS ARE NICE\n";
```

```
function goodTea() {
        return "Oolong tea tastes good!";
}
?>
```

Ejemplo 11-3. foolib.inc

```
<?php
require ("utils.inc");
function showVar($var) {
        if (PHPVERSION == 4) {
                print_r($var);
        } else {
                dump_var($var);
        }
}

// bunch of other functions ...
?>
```

And then you write a script `cause_error_require.php`

Ejemplo 11-4. cause_error_require.php

```
<?php
require("foolib.inc");
/* the following will generate an error */
require("utils.inc");
$foo = array("1",array("complex","quaternion"));
echo "this is requiring utils.inc again which is also\n";
echo "required in foolib.inc\n";
echo "Running goodTea: ".goodTea()."\n";
echo "Printing foo: \n";
showVar($foo);
?>
```

When you try running the latter one, the resulting ouptut will be (using PHP 4.01pl2):

```
GLOBALS ARE NICE
GLOBALS ARE NICE

Fatal error:  Cannot redeclare causeerror() in utils.inc on line 5
```

By modifying `foolib.inc` and `cause_errror_require.php` to use require_once() instead of require() and renaming the last one to `avoid_error_require_once.php`, we have:

Ejemplo 11-5. foolib.inc (fixed)

```
...
require_once("utils.inc");
function showVar($var) {
...
```

Ejemplo 11-6. avoid_error_require_once.php

```
...
require_once("foolib.inc");
require_once("utils.inc");
$foo = array("1",array("complex","quaternion"));
...
```

And when running the latter, the output will be (using PHP 4.0.1pl2):

```
GLOBALS ARE NICE
this is requiring globals.inc again which is also
required in foolib.inc
Running goodTea: Oolong tea tastes good!
Printing foo:
Array
(
    [0] => 1
    [1] => Array
        (
            [0] => complex
            [1] => quaternion
        )

)
```

Also note that, analogous to the behavior of the `#include` of the C preprocessor, this statement acts at "compile time", e.g. when the script is parsed and before it is executed, and should not be used for parts of the script that need to be inserted dynamically during its execution. You should use include_once() or include() for that purpose.

For more examples on using require_once() and include_once(), look at the PEAR code included in the latest PHP source code distributions.

See also: require(), include(), include_once(), get_required_files(), get_included_files(), readfile(), and virtual().

include_once()

The include_once() statement includes and evaluates the specified file during the execution of the script. This is a behavior similar to the include() statement, with the important difference that if the code from a file has already been included, it will not be included again.

As mentioned in the require_once() description, the include_once() should be used in the cases in which the same file might be included and evaluated more than once during a particular execution of a script, and you want to be sure that it is included exactly once to avoid problems with function redefinitions, variable value reassignments, etc.

For more examples on using require_once() and include_once(), look at the PEAR code included in the latest PHP source code distributions.

See also: require(), include(), require_once(), get_required_files(), get_included_files(), readfile(), and virtual().

Capítulo 12. Funciones

Funciones definidas por el usuario

Una función se define con la siguiente sintaxis:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
    echo "Función de ejemplo.\n";
    return $retval;
}
```

Cualquier instrucción válida de PHP puede aparecer en el cuerpo de la función, incluso otras funiones y definiciones de clases.

En PHP3, las funciones deben definirse antes de que se referencien. En PHP4 no existe tal requerimiento.

PHP no soporta la sobrecarga de funciones, y tampoco es posible redefinir u ocultar funciones previamente declaradas.

PHP3 no soporta un número variable de parámetros, aunque sí soporta parámetros por defecto (ver Valores por defecto de de los parámetros para más información). PHP4 soporta ambos: ver Listas de longitud variable de parámetros y las referencias de las funciones func_num_args(), func_get_arg(), y func_get_args() para más información.

Parámetros de las funciones

La información puede suministrarse a las funciones mediante la lista de parámetros, una lista de variables y/o constantes separadas por comas.

PHP soporta pasar parámetros por valor (el comportamiento por defecto), por referencia, y parámetros por defecto. Listas de longitud variable de parámetros sólo están soportadas en PHP4 y posteriores; ver Listas de longitud variable de parámetros y la referencia de las funciones func_num_args(), func_get_arg(), y func_get_args() para más información. Un efecto similar puede conseguirse en PHP3 pasando un array de parámetros a la función:

```
function takes_array($input) {
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Pasar parámetros por referencia

Por defecto, los parámetros de una función se pasan por valor (de manera que si cambias el valor del argumento dentro de la función, no se ve modificado fuera de ella). Si deseas permitir a una función modificar sus parámetros, debes pasarlos por referencia.

Si quieres que un parámetro de una función siempre se pase por referencia, puedes anteponer un ampersand (&) al nombre del parámetro en la definición de la función:

```
function add_some_extra(&$string) {
    $string .= ' y algo más.';
}
$str = 'Esto es una cadena, ';
add_some_extra($str);
echo $str;    // Saca 'Esto es una cadena, y algo más.'
```

Si deseas pasar una variable por referencia a una función que no toma el parámetro por referencia por defecto, puedes anteponer un ampersand al nombre del parámetro en la llamada a la función:

```
function foo ($bar) {
    $bar .= ' y algo más.';
}
$str = 'Esto es una cadena, ';
foo ($str);
echo $str;    // Saca 'Esto es una cadena, '
foo (&$str);
echo $str;    // Saca 'Esto es una cadena, y algo más.'
```

Parámetros por defecto

Una función puede definir valores por defecto para los parámetros escalares estilo C++:

```
function makecoffee ($type = "cappucino") {
    return "Hacer una taza de $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
```

La salida del fragmento anterior es:

```
Hacer una taza de cappucino.
Hacer una taza de espresso.
```

El valor por defecto tiene que ser una expresión constante, y no una variable o miembro de una clase.

En PHP 4.0 también es posible especificar `unset` como parámetro por defecto. Esto significa que el argumento no tomará ningún valor en absoluto si el valor no es suministrado.

Destacar que cuando se usan parámetros por defecto, estos tienen que estar a la derecha de cualquier parámetro sin valor por defecto; de otra manera las cosas no funcionarán de la forma esperada. Considera el siguiente fragmento de código:

```
function makeyogurt ($type = "acidophilus", $flavour) {
    return "Haciendo un bol de $type $flavour.\n";
}

echo makeyogurt ("mora");   // No funcionará de la manera esperada
```

La salida del ejemplo anterior es:

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Haciendo un bol de mora.
```

Y ahora, compáralo con:

```
function makeyogurt ($flavour, $type = "acidophilus") {
    return "Haciendo un bol de $type $flavour.\n";
}

echo makeyogurt ("mora");   // funciona como se esperaba
```

La salida de este ejemplo es:

```
Haciendo un bol de acidophilus mora.
```

Lista de longitud variable de parámetros

PHP4 soporta las listas de longitud variable de parámetros en las funciones definidas por el usuario. Es realmente fácil, usando las funciones func_num_args(), func_get_arg(), y func_get_args().

No necesita de ninguna sintaxis especial, y las listas de parámetros pueden ser escritas en la llamada a la función y se comportarán de la manera esperada.

Devolver valores

Los valores se retornan usando la instrucción opcional return. Puede devolverse cualquier tipo de valor, incluyendo listas y objetos.

```
function square ($num) {
    return $num * $num;
}
echo square (4);   // saca '16'.
```

No puedes devolver múltiples valores desde una función, pero un efecto similar se puede conseguir devolviendo una lista.

```
function small_numbers() {
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

`old_function`

La instrucción `old_function` permite declarar una función usando una sintaxis idéntica a la de PHP/FI2 (excepto que debes reemplazar 'function' por 'old_function').

Es una característica obsoleta, y debería ser usada únicamente por el conversor PHP/FI2->PHP3.

Aviso

Las funciones declaradas como `old_function` no pueden llamarse desde el código interno de PHP. Entre otras cosas, esto significa que no puedes usarlas en funciones como usort(), array_walk(), y register_shutdown_function(). Puedes solventar esta limitación escribiendo un "wrapper" (en PHP3 normal) que a su vez llame a la función declarada como `old_function`.

Funciones variable

PHP soporta el concepto de funciones variable, esto significa que si una variable tiene unos paréntesis añadidos al final, PHP buscará una función con el mismo nombre que la evaluación de la variable, e intentará ejecutarla. Entre otras cosas, esto te permite implementar retrollamadas (callbacks), tablas de funciones y demás.

Ejemplo 12-1. Ejemplo de función variable

```
<?php
function foo() {
    echo "Dentro de foo()<br>\n";
}

function bar( $arg = ” ) {
    echo "Dentro de bar(); el parámetro fue ’$arg’.<br>\n";
}

$func = ’foo’;
$func();
$func = ’bar’;
$func( ’test’ );
?>
```

Capítulo 13. Clases y Objetos

class

Una clase es una colección de variables y de funciones que acceden a esas variables. Una clase se define con la siguiente sintaxis:

```
<?php
class Cart {
    var $items;  // Items en nuestro carro de la compra

    // Añadir $num artículos de tipo $artnr al carro

    function add_item ($artnr, $num) {
        $this->items[$artnr] += $num;
    }

    // Sacar $num artículos del tipo $artnr del carro

    function remove_item ($artnr, $num) {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

El ejemplo define una clase llamada Cart que consiste en un array asociativo de artículos en el carro y dos funciones para meter y sacar ítems del carro

Las clases son tipos, es decir, son plantillas para variables. Tienes que crear una variable del tipo deseado con el operador new.

```
 $cart = new Cart;
 $cart->add_item("10", 1);
```

Este ejemplo crea un objeto $cart de clase Cart. La función add_item() de ese objeto se llama para añadir un item del artículo número 10 al carro.

Las Clases pueden ser extensiones de otras clases. Las clases extendidas o derivadas tienen todas las variables y funciones de la clase base y lo que les añadas al extender la definición. La herencia múltiple no está soportada.

```
class Named_Cart extends Cart {
    var $owner;

    function set_owner ($name) {
        $this->owner = $name;
```

```
    }
}
```

Ese ejemplo define una clase Named_Cart (carro con nombre o dueño) que tiene todas las variables y funciones de Cart, y además añade la variable $owner y una función adicional set_owner(). Un carro con nombre se crea de la forma habitual y, una vez hecho, puedes acceder al propietario del carro. En los carros con nombre también puedes acceder a las funciones normales del carro:

```
$ncart = new Named_Cart;    // Creamos un carro con nombre
$ncart->set_owner ("kris"); // Nombramos el carro
print $ncart->owner;        // Imprimimos el nombre del propietario
$ncart->add_item ("10", 1); // Funcionalidad heredada de Cart
```

Entre funciones de una clase, la variable $this hace referencia al propio objeto. Tienes que usar $this->loquesea para acceder a una variable o función llamada loquesea del objeto actual.

Los constructores son funciones de una clase que se llaman automáticamente al crear una nueva instancia (objeto) de una clase. Una función se convierte en constructor cuando tiene el mismo nombre que la clase.

```
class Auto_Cart extends Cart {
    function Auto_Cart () {
        $this->add_item ("10", 1);
    }
}
```

Este ejemplo define una clase Auto_Cart que es un Cart junto con un constructor que inicializa el carro con un item del tipo de artículo "10" cada vez que se crea un nuevo Auto_Cart con "new". Los constructores también pueden recibir parámetros y estos parámetros pueden ser opcionales, lo que los hace más útiles.

```
class Constructor_Cart extends Cart {
    function Constructor_Cart ($item = "10", $num = 1) {
        $this->add_item ($item, $num);
    }
}

// Compramos las mismas cosas aburridas de siempre

$default_cart   = new Constructor_Cart;

// Compramos las cosas interesantes

$different_cart = new Constructor_Cart ("20", 17);
```

Atención

Para las clases derivadas, el constructor de la clase padre no es llamado automáticamente cuando se llama al constructor de la clase derivada.

Capítulo 14. References Explained

What References Are

References in PHP are a means to access the same variable content by different names. They are not like C pointers, they are symbol table aliases. Note that in PHP, variable name and variable content are different, so the same content can have different names. The most close analogy is with Unix filenames and files - variable names are directory entries, while variable contents is the file itself. References can be thought of as hardlinking in Unix filesystem.

What References Do

PHP references allow you to make two variables to refer to the same content. Meaning, when you do:

```
$a =& $b
```

it means that `$a` and `$b` point to the same variable.

> **Nota:** `$a` and `$b` are completely equal here, that's not `$a` is pointing to `$b` or vice versa, that's `$a` and `$b` pointing to the same place.

The same syntax can be used with functions, that return references, and with `new` operator (in PHP 4.0.4 and later):

```
$bar =& new fooclass();
$foo =& find_var ($bar);
```

> **Nota:** Not using the `&` operator causes a copy of the object to be made. If you use `$this` in the class it will operate on the current instance of the class. The assignment without `&` will copy the instance (i.e. the object) and `$this` will operate on the copy, which is not always what is desired. Usually you want to have a single instance to work with, due to performance and memory consumption issues.
>
> While you can use the `@` operator to *mute* any errors in the constructor when using it as `@new`, this does not work when using the `&new` statement. This is a limitation of the Zend Engine and will therefore result in a parser error.

The second thing references do is to pass variables by-reference. This is done by making a local variable in a function and a variable in the calling scope reference to the same content. Example:

```
function foo (&$var)
{
    $var++;
}

$a=5;
foo ($a);
```

will make `$a` to be 6. This happens because in the function `foo` the variable `$var` refers to the same content as `$a`. See also more detailed explanations about passing by reference.

The third thing reference can do is return by reference.

What References Are Not

As said before, references aren't pointers. That means, the following construct won't do what you expect:

```
function foo (&$var)
{
    $var =& $GLOBALS["baz"];
}
foo($bar);
```

What happens is that `$var` in `foo` will be bound with `$bar` in caller, but then it will be re-bound with `$GLOBALS["baz"]`. There's no way to bind `$bar` in the calling scope to something else using the reference mechanism, since `$bar` is not available in the function `foo` (it is represented by `$var`, but `$var` has only variable contents and not name-to-value binding in the calling symbol table).

Passing by Reference

You can pass variable to function by reference, so that function could modify its arguments. The syntax is as follows:

```
function foo (&$var)
{
    $var++;
}
```

```
$a=5;
foo ($a);
// $a is 6 here
```

Note that there's no reference sign on function call - only on function definition. Function definition alone is enough to correctly pass the argument by reference.

Following things can be passed by reference:

- Variable, i.e. `foo($a)`
- New statement, i.e. `foo(new foobar())`
- Reference, returned from a function, i.e.:

```
function &bar()
{
    $a = 5;
    return $a;
}
foo(bar());
```

See also explanations about returning by reference.

Any other expression should not be passed by reference, as the result is undefined. For example, the following examples of passing by reference are invalid:

```
function bar() // Note the missing &
{
    $a = 5;
    return $a;
}
foo(bar());

foo($a = 5) // Expression, not variable
foo(5) // Constant, not variable
```

These requirements are for PHP 4.0.4 and later.

Returning References

Returning by-reference is useful when you want to use a function to find which variable a reference should be bound to. When returning references, use this syntax:

```
function &find_var ($param)
{
    ...code...
    return $found_var;
}

$foo =& find_var ($bar);
$foo->x = 2;
```

In this example, the property of the object returned by the `find_var` function would be set, not the copy, as it would be without using reference syntax.

> **Nota:** Unlike parameter passing, here you have to use `&` in both places - to indicate that you return by-reference, not a copy as usual, and to indicate that reference binding, rather than usual assignment, should be done for `$foo`.

Unsetting References

When you unset the reference, you just break the binding between variable name and variable content. This does not mean that variable content will be destroyed. For example:

```
$a = 1;
$b =& $a;
unset ($a);
```

won't unset `$b`, just `$a`.

Again, it might be useful to think about this as analogous to Unix **unlink** call.

Spotting References

Many syntax constructs in PHP are implemented via referencing mechanisms, so everything told above about reference binding also apply to these constructs. Some constructs, like passing and returning by-reference, are mentioned above. Other constructs that use references are:

`global` References

When you declare variable as **global $var** you are in fact creating reference to a global variable. That means, this is the same as:

```
$var =& $GLOBALS["var"];
```

That means, for example, that unsetting `$var` won't unset global variable.

`$this`

In an object method, `$this` is always reference to the caller object.

Parte III. Características

Capítulo 15. Manejando errores

Existen diferentes tipos de errores y advertencias en PHP. Son los siguientes:

Tabla 15-1. Tipos de error PHP

Valor	Constante	Descripción	Nota
1	E_ERROR	errores fatales en tiempo de ejecución	
2	E_WARNING	advertencias en tiempo de ejecución (no errores fatales)	
4	E_PARSE	errores fatales en tiempo de compilación	
8	E_NOTICE	avisos en tiempo de ejecución (no tan importante como una advertencia)	
16	E_CORE_ERROR	errores fatales que ocurren durante el proceso inicial de arranque de PHP	Solo en PHP 4
32	E_CORE_WARNING	advertencias fatales que ocurren durante el proceso inicial de arranque de PHP	Solo en PHP 4
64	E_COMPILE_ERROR	errores fatales en tiempo de compilación	solo en PHP 4
128	E_COMPILE_WARNING	advertencias en tiempo de compilación (no errores fatales)	Solo en PHP 4
256	E_USER_ERROR	mensaje de error generado por el usuario	Solo en PHP 4
512	E_USER_WARNING	mensaje de advertencia generado por el usuario	solo en PHP 4
1024	E_USER_NOTICE	mensaje de aviso generado por el usuario	Solo en PHP 4
	E_ALL	todos los anteriores, all of the above, según lo soportado	

Los valores indicados arriba (tanto num$eacute;ricos como simbólicos) son usados para crear una máscara de bits (bitmask) que especifica de los errores que hay que informar. Podeis usar los operadores bitwise para combinar estos valores ó aplicar una máscara a ciertos tipos de errores. Tener en cuenta que que solamente '|', '~', '!', y '&' seran interpretados dentro de `php.ini` y que ningún operador bitwise será interpretado dentro de `php3.ini`.

En PHP 4, el valor por defecto de error_reporting es `E_ALL & ~E_NOTICE`, esto siginifica que todos los errores y advertencias que no pertenecen al nivel E_NOTICE, serán presentados cuando ocurran. En PHP 3, el valor por defecto es `(E_ERROR | E_WARNING | E_PARSE)`, teniendo el mismo significado. Tener en cuenta que al no soportar constantes en el fichero de configuración de PHP `php3.ini`, el valor de error_reporting debe ser numérico, por lo tanto es `7`.

La configuración inicial puede cambiarse, en el fichero ini con la directiva error_reporting, en el fichero `httpd.conf` de Apache con la directiva php_error_reporting (php3_error_reporting en PHP 3) y finalmente en tiempo de ejecución desde el script PHP que se este ejecutando, usando la función error_reporting().

> **Aviso**
>
> Cuando actualiceis código o servidores de PHP 3 a PHP 4, deberiais comprobar los valores y llamadas a error_reporting() para no deshabilitar los nuevos tipos de mensajes de error, especialmente E_COMPILE_ERROR. Si esto ocurriese podriais obtener documentos vacios sin ningún tipo de mensaje de error ó donde buscar el fallo.

Todas las expresioness PHP pueden también ser llamadas con el prefijo "@", el cual desactiva el aviso de errores para esa expresión en particular. Si ocurre un error en una expresión en tal situación y la característica track_errors está habilitada, podrás encontrar el mensaje de error en la variable global `$php_errormsg`.

> **Nota:** El operador de control de errores @ no desactivará mensajes producidos por errores en el parseador.

> **Aviso**
>
> Actualmente el operador de control de errores @ incluso desactivará mensajes producidos por errores críticos que terminarán la ejecución del script. Entre otras cosas, esto significa que si utilizais @ para suprimir mensajes de error de alguna función, tanto si no está disponible como si contiene algún error, el script quedará interumpido sin ningún tipo de indicación de porque.

A continuación tenemos un ejemplo de como manejar errores en PHP. Definimos una función de manejo de errores, la cual registra el error en un fichero (usando el formato XML) y manda un e-mail al programador si un error crítico ocurre.

Ejemplo 15-1. Usando el manejo de errores en un script

```
<?php
// Los errores los manejamos nosotros
error_reporting(0);

// funcion de manejos de errores definida por el usuario
function userErrorHandler ($errno, $errmsg, $filename, $linenum, $vars) {
```

```
    // timestamp for the error entry
    $dt = date("Y-m-d H:i:s (T)");

    // Define una array con los valores de errores
    // en realidad solamente deberiamos de tener
    // en cuenta los valores 2,8,256,512 y 1024

    $errortype = array (
                1   =>  "Error",
                2   =>  "Warning",
                4   =>  "Parsing Error",
                8   =>  "Notice",
                16  =>  "Core Error",
                32  =>  "Core Warning",
                64  =>  "Compile Error",
                128 =>  "Compile Warning",
                256 =>  "User Error",
                512 =>  "User Warning",
                1024=>  "User Notice"
                );

    // errores a tener en cuenta
    $user_errors = array(E_USER_ERROR, E_USER_WARNING, E_USER_NOTICE);

    $err = "<errorentry>\n";
    $err .= "\t<datetime>".$dt."</datetime>\n";
    $err .= "\t<errornum>".$errno."</errornum>\n";
    $err .= "\t<errortype>".$errortype[$errno]."</errortype>\n";
    $err .= "\t<errormsg>".$errmsg."</errormsg>\n";
    $err .= "\t<scriptname>".$filename."</scriptname>\n";
    $err .= "\t<scriptlinenum>".$linenum."</scriptlinenum>\n";

    if (in_array($errno, $user_errors))
        $err .= "\t<vartrace>".wddx_serialize_value($vars,"Variables")."</vartrace>\n";
    $err .= "</errorentry>\n\n";

    // Para comprobar
    // echo $err;

    // grabar en el fichero de errores y mandar un e-mail si ocurre un error critico de usua
    error_log($err, 3, "/usr/local/php4/error.log");
    if ($errno == E_USER_ERROR)
        mail("phpdev@example.com","Critical User Error",$err);
}


function distance ($vect1, $vect2) {
    if (!is_array($vect1) || !is_array($vect2)) {
        trigger_error("Incorrect parameters, arrays expected", E_USER_ERROR);
        return NULL;
    }

    if (count($vect1) != count($vect2)) {
```

```
        trigger_error("Vectors need to be of the same size", E_USER_ERROR);
        return NULL;
    }

    for ($i=0; $i<count($vect1); $i++) {
        $c1 = $vect1[$i]; $c2 = $vect2[$i];
        $d = 0.0;
        if (!is_numeric($c1)) {
            trigger_error("Coordinate $i in vector 1 is not a number, using zero",
                          E_USER_WARNING);
            $c1 = 0.0;
        }
        if (!is_numeric($c2)) {
            trigger_error("Coordinate $i in vector 2 is not a number, using zero",
                          E_USER_WARNING);
            $c2 = 0.0;
        }
        $d += $c2*$c2 - $c1*$c1;
    }
    return sqrt($d);
}

$old_error_handler = set_error_handler("userErrorHandler");

//constante no definida, genera una advertencia
$t = I_AM_NOT_DEFINED;

// definimos algunos vectores
$a = array(2,3,"foo");
$b = array(5.5, 4.3, -1.6);
$c = array (1,-3);

// genera un error de usuario
$t1 = distance($c,$b)."\n";

// genera otro error de usuario
$t2 = distance($b,"i am not an array")."\n";

// genera una advertencia
$t3 = distance($a,$b)."\n";

?>
```

Este es un ejemplo simple que muestra como utilizar las funciones de manejo de errores y registro.

Consultar tambien error_reporting(), error_log(), set_error_handler(), restore_error_handler(), trigger_error(), user_error()

Capítulo 16. Creando y manipulando imágenes

PHP no está limitado a crear solo salidas de HTML. Puede ser usado también para crear y manipular ficheros de imágenes en diferentes formatos, incluyendo gif, png, jpg, wbmp, y xpm. PHP puede incluso mandar flujos de imágenes directamente al navegador. Necesitais compilar PHP con la biblioteca de funciones de imágenes GD para esta tarea. GD y PHP puede que necesiten otras bibliotecas, dependiendo del formato de imagen con el que querais trabajar. GD dejo de soportar imagenes Gif en la versión 1.6.

Ejemplo 16-1. Creación de PNGs con PHP

```
<?php
    Header("Content-type: image/png");
    $string=implode($argv," ");
    $im = ImageCreateFromPng("images/button1.png");
    $orange = ImageColorAllocate($im, 220, 210, 60);
    $px = (imagesx($im)-7.5*strlen($string))/2;
    ImageString($im,3,$px,9,$string,$orange);
    ImagePng($im);
    ImageDestroy($im);
?>
```

Este ejemplo será llamado desde una página con una línea como esta: <imgsrc="button.php?text"> Este script de arriba button.php toma esta cadena "text" la situa sobre la imagen base, en este caso es "images/button1.png" y muestra la imagen resultante. Esta es una forma muy conveniente para evitar tener que dibujar un nuevo botón cada vez que quiera cambiar el texto del mismo. Con este método los botones son generados dinámicamente.

Capítulo 17. Autentificación HTTP con PHP

Las caracteríticas de autentificación HTTP en PHP solo estan disponibles cuando se está ejecutando como un módulo en Apache y hasta ahora no lo estan en la versión CGI. En un script PHP como módulo de Apache, se puede usar la función header() para enviar un mensaje de "Autentificación requerida" al navegador cliente haciendo que muestre una ventana de entrada emergente con nombre de usuario y contraseña. Una vez que el usuario ha rellenado el nombre y la contraseña, la URL que contiene el script PHP vuelve a ser llamada con las variables $PHP_AUTH_USER, $PHP_AUTH_PW y $PHP_AUTH_TYPE rellenas con el nombre de usuario, la contraseña y el tipo de autentificación respectivamente. Sólo autentificación "Básica" esta soportada en este momento. Consulte la función header() para más información.

Un fragmento de script de ejmplo que fuerce la autentificación del cliente en una página sería como el siguiente:

Ejemplo 17-1. Ejemplo de autentificación HTTP

```
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header("WWW-Authenticate: Basic realm=\"My Realm\"");
    header("HTTP/1.0 401 Unauthorized");
    echo "Text to send if user hits Cancel button\n";
    exit;
  } else {
    echo "<p>Hello {$_SERVER['PHP_AUTH_USER']}.</p>";
    echo "<p>You entered {$_SERVER['$PHP_AUTH_PW']} as your password.</p>";
  }
?>
```

> **Nota:** Por favor tener cuidado cuando esteis programando las lines de cabecera HTTP. Para garantizar la maxima compatibilidad con todos los clientes, la palabra clave "Basic" debe de ser escrita con "B" mayúscula, la cadena de texto debe estar incluida entre comillas dobles (no simples) y un espacio debe preceder el código "401" en la linea de cabecera "HTTP/1.0 401"

En vez de, sencillamente, mostrar $PHP_AUTH_USER y $PHP_AUTH_PW, seguramente querais comprobar la validez del nombre de usuario y la contraseña. Tal vez enviando una consulta a una base de datos o buscando el usuario en un fichero dbm.

Vigilar aquí los navegadores Interner Explorer con bugs. Parecen muy quisquillosos con el orden de las cabeceras. Enviar la cabecera *WWW-Autentificación* antes que la cabecera HTTP/1.0 401 parece ser el truco por ahora.

Para prevenir que alguien escriba un script que revele la contraseña de una página que ha sido autentificada a través de algún mecanismo externo tradicional, las variables PHP_AUTH no serán rellenadas si algún tipo de autentificación externo ha sido activado para una página en particular. En este caso, la variable $REMOTE_USER puede ser usada para identificar al usuario autentificado externamente.

> **Configuration Note:** PHP usa la directiva `AuthType` para determinar si una autentificación externa esta en uso. Recordar no usar esta directiva cuando querais usar la autentificación de PHP (si no todo intentento de autentificación fallará)

Nota, a pesar de todo, lo ya dicho no proteje de que alguien que controle una URL no autentificada robe contraseñas de URLs autentificadas en el mismo servidor.

Tanto Netscape como Internet Explorer borrarán la caché de la ventana de autentificación en el navegador local después de recibir una respuesta 401 del servidor. Esto puede usarse, de forma efectiva, para "desconectar" a un usuario, forzandole a reintroducir su nombre y contraseña. Algunas personas usan esto para "hacer caducar" entradas, o para proveer un botón de "desconectar".

Ejemplo 17-2. Ejemplo de autentificación HTTP forzando una reentrada

```
<?php
  function authenticate() {
    header( "WWW-Authenticate: Basic realm=\"Test Authentication System\"");
    header( "HTTP/1.0 401 Unauthorized");
    echo "You must enter a valid login ID and password to access this resource\n"
;
    exit;
  }

  if (!isset($_SERVER['PHP_AUTH_USER']) || ($SeenBefore == 1 && $OldAuth == $_SER
VER['$PHP_AUTH_USER']))) {
   authenticate();
  }
  else {
   echo "<p>Welcome: {$_SERVER['$PHP_AUTH_USER']}<br>";
   echo "Old: {$_REQUEST['$OldAuth']}";
   echo "<form action='{$_SERVER['$PHP_SELF']}' METHOD='POST'>\n";
   echo "<input type='hidden' name='SeenBefore' value='1'>\n";
   echo "<input type='hidden' name='OldAuth' value='{$_SERVER['$PHP_AUTH_USER']}'
>\n";
   echo "<input type='submit' value='Re Authenticate'>\n";
   echo "</form></p>\n";
  }
?>
```

Este comportamiento no es requerido por el estándar de autentificación básica de HTTP, por lo que nunca debe depender de esto. Pruebas con Lynx han demostrado que Lynx no borra las credenciales de autentificación con una respuesta 401 del servidor, por lo que pulsando atrás y después adelante abriría el recurso de nuevo (siempre que los requerimientos de contraseña no hayan cambiado).

Además tener en cuanta que esto no funciona usando el servidor IIS de Microsoft y la versión CGI de PHP debido a una limitación del IIS

Capítulo 18. Cookies

PHP soporta transparentemente cookies HTTP. Las Cookies son un mecanismo que sirve para almacenar datos en el navegador del usuario remoto, para así poder identificar al usuario cuando vuelva. Se pueden poner cookies usando la función **setcookies()**. Las Cookies son parte de la cabecera HTTP, por tanto la función setcookie() debe ser llamada antes de que se produzca cualquier salida al navegador. Esta limitación es la misma a la de la función header(). Se pueden usar las funciones de almacenamiento intermedio del resultado para retrasar el resultado del script hasta que hayas decidido mandar o no una cookie o cabecera.

Cualquier cookie enviada a ti desde el cliente, automáticamente se convertirá en una variable PHP igual como ocurre con los métodos de datos GET y POST, dependiendo de las variables de configuración `register_globals` y `variables_order`. Si deseas asignar multiples valores a una cookie simple, añade simplemente *[]* a el nombre de la cookie.

En PHP 4.1.0 y posteriores, la array auto-global `$_COOKIE` será siempre actualizada con cualquier cookie mandada por el cliente. `$HTTP_COOKIE_VARS` es tambien actualizada en versiones anteriores de PHP cuando la variable de configuración `track_vars` esté activada.

Para más detalles ver la función setcookie().

Capítulo 19. Manejo de envío de ficheros

Envío de archivos con el método POST

PHP es capaz de recibir envíos de archivo de cualquier navegador que cumpla la norma RFC-1867 (entre los que se incluyen Netscape Navigator 3 o posterior, Microsoft Internet Explorer 3 con un parche o posterior sin éste). Ésta característica permite que los usuarios envien archivos de texto y binarios. Mediante la autentificación y funciones de manejo de archivos de PHP, es posible un control total de quién puede enviar archivos y que se hace con éstos una vez recibidos.

Es importante destacar que PHP también soporta el método PUT para envío de archivos tal y como lo utiliza Netscape Composer y el cliente Amaya de W3C. Consulte Soporte del método PUT para más detalles.

Una página de envío de archivos se puede crear mediante un formulario parecido a éste:

Ejemplo 19-1. Formulario de envío de archivo

```
<form enctype="multipart/form-data" action="_URL_" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <input name="userfile" type="file">
<input type="submit" value="Send File">
</form>
```

La _URL_ debe tener como destino un script PHP. El input oculto MAX_FILE_SIZE debe encontrarse antes del input de tipo "file" para indicar el tamaño máximo de archivo que se puede enviar en bytes

> **Aviso**
>
> MAX_FILE_SIZE debe ser consultado por el navegador; aun así es sencillo saltarse este máximo por lo tanto no se debe presuponer que el navegador siempre lo respetará. En contrapartida, la configuracion de PHP relativa al tamaño maximo no puede ser obviada.

Las variables definidas para los archivos enviados varian en función de la versión y configuración de PHP que se utilice. Las variables de las que hablamos a continuación serán definidas en la página destino despues de una recepción de fichero correcta. Cuando track_vars este activado, el array $HTTP_POST_FILES/$_FILES se inicializará. Por ultimo, las variables relacionadas seran inicializadas como globales cuando register_globals esté habilitado. Cabe señalar que el uso de las variables globales no esta recomendado en ningún caso.

> **Nota:** track_vars esta activado siempre desde PHP 4.0.3. A partir de PHP 4.1.0 , $_FILES puede ser utilizado alternativamente a `$HTTP_POST_FILES`. `$_FILES` es siempre global asi que `global` no debe ser usado con $_FILES en el ámbito de función.

`$HTTP_POST_FILES`/`$_FILES` contienen la información sobre el fichero recibido.

A continuación se describe el contenido de `$HTTP_POST_FILES`. Se ha tomado el nombre 'userfile' para el fichero recibido tal y como se usaba en el script de ejemplo anterior:

`$HTTP_POST_FILES['userfile']['name']`

El nombre original del fichero en la máquina cliente.

`$HTTP_POST_FILES['userfile']['type']`

El tipo mime del fichero (si el navegador lo proporciona). Un ejemplo podría ser `"image/gif"`.

`$HTTP_POST_FILES['userfile']['size']`

El tamaño en bytes del fichero recibido.

`$HTTP_POST_FILES['userfile']['tmp_name']`

El nombre del fichero temporal que se utiliza para almacenar en el servidor el archivo recibido.

Nota: A partir de PHP 4.1.0 se puede utilizar el variable corta `$_FILES`. PHP 3 no soporta `$HTTP_POST_FILES`.

Cuando register_globals se activa en el `php.ini` las siguientes variables son accesibles. Se ha tomado el nombre 'userfile' para el fichero recibido tal y como se usaba en el script de ejemplo del principio:

- `$userfile` - El nombre del fichero temporal que se utiliza para almacenar en el servidor el archivo recibido.
- `$userfile_name` - El nombre original del fichero en la máquina cliente.
- `$userfile_size` - El tamaño en bytes del fichero recibido.
- `$userfile_type` - El tipo mime del fichero (si el navegador lo proporciona). Un ejemplo podría ser "image/gif".

Se puede ver que "`$userfile`" (en las variables anteriores) toma el valor del atributo "name" que contenga el campo <input> de tipo "file" del formulario de envio. En el ejemplo anterior, elegimos llamarlo "userfile".

Nota: `register_globals = On` se desaconseja por razones de seguridad y rendimiento.

Por defecto, los ficheros serán almacenados en el directorio temporal por defecto del servidor a no ser que se especifique otra localizacion con la directiva upload_tmp_dir en `php.ini`. El directorio temporal por defecto del servidor puede ser modificado cambiando el valor de la variable de entorno TMPDIR en el contexto en que se ejecuta PHP La configuración de las variables de entorno no se puede realizar en PHP a través de la función putenv(). Esta variable de entorio puede ser utilizada también para asegurarnos que otras operaciones con archivos recibidos están funcionando correctamente.

Ejemplo 19-2. Verificando los archivos recibidos

Los siguientes ejemplos son validos para versiones de PHP 4 superiores a la 4.0.2. Veanse las funciones is_uploaded_file() y move_uploaded_file().

```
<?php
// In PHP 4.1.0 or later, $_FILES should be used instead of $HTTP_POST_FILES.
if (is_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'])) {
    copy($HTTP_POST_FILES['userfile']['tmp_name'], "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack. Filename: " . $HTTP_POST_FILES['userfile']['name'];
}
/* ...or... */
move_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'], "/place/to/put/uploaded/file");
?>
```

El script PHP que recibe el fichero, debe implementar la lógica necesaria para determinar que debe ser realizado con el fichero. Se puede utilizar, por ejemplo, la variable `$HTTP_POST_FILES['userfile']['size']` para descartar los ficheros demasiado chicos o demasiado grandes; por otro lado, se puede usar la variable `$HTTP_POST_FILES['userfile']['type']` para descartar los que no se ajusten a algun criterio de tipo. Cualquiera que sea la logica que utilicemos, se debe borrar o mover el archivo del directorio temporal.

El archivo será borrado del directorio temporal al final de la petición si no se ha movido o renombrado.

Errores comunes

A `MAX_FILE_SIZE` no se le puede dar un valor mayor que el valor que se haya especificado en la directivaupload_max_filesize. Por defecto se tiene un límite de 2 MegaBytes.

Si se ha activado el límite de memoria, se deben especificar un valor alto para memory_limit. En cualquier caso, se debe asegurar un valor suficientemente grande para memory_limit.

Si max_execution_time tiene un valor muy pequeño, la ejecución del script puede exceder este valor. De esta forma, se debe asegurar un valor suficientemente grande para `max_execution_time`.

Si post_max_size< tiene un valor muy pequeño, los ficheros mas grandes a este valor, no podrán ser enviados. Por ello, se debe asegurar un valor suficientemente grande para `post_max_size`.

No verificar que ficheros se estan manipulando puede tener como consecuencia que los usuarios puedan acceder a información sensible en otros directorios.

Cabe señalar que el httpd de CERN parece cortar todo a partir del primer espacio en blanco en el "content-type" de la cabecera mime que obtiene del cliente. Si este es el caso, con el httpd de CERN no se soporta la funcionalidad de envío de ficheros.

Envío de multiples ficheros

Se pueden enviar multiples ficheros usando diferentes nombres (`name`) para los `input`.

Así mismo, es posible enviar varios archivos simultaneamente y tener organizada en arrays la información. Para hacer esto, se utiliza la misma sintáxis que cuando tenemos multiples "selects" o "checkboxes" en el formulario HTML:

> **Nota:** El soporte para envío multiple de ficheros fue añadido en la versión 3.0.10.

Ejemplo 19-3. Envío de multiples ficheros

```
<form action="file-upload.php" method="post" enctype="multipart/form-data">
  Send these files:<br>
  <input name="userfile[]" type="file"><br>
  <input name="userfile[]" type="file"><br>
  <input type="submit" value="Send files">
</form>
```

Cuando el formulario del ejemplo es enviado, los arrays `$HTTP_POST_FILES['userfile']`, `$HTTP_POST_FILES['userfile']['name']` y `$HTTP_POST_FILES['userfile']['size']` son inicializados. Así mismo pasa con $_FILES en PHP 4.1.0 o superiores y $HTTP_POST_VARS en PHP 3. Cuando `register_globals` esta activa, las variables globales para los archivos recibidos también son inicializadas. Cada uno de estos arrays tendrá en los índices numericos correspondientes los valores para cada fichero recibido.

Por ejemplo, si tomamos como nombres de archivo enviados `/home/test/review.html` y `/home/test/xwp.out`. Tendríamos en `$HTTP_POST_FILES['userfile']['name'][0]` el valor de `review.html`, y en `$HTTP_POST_FILES['userfile']['name'][1]` tendríamos `xwp.out`; analogamente, `$HTTP_POST_FILES['userfile']['size'][0]` contendría el tamaño del fichero `review.html`, y asi sucesivamente...

`$HTTP_POST_FILES['userfile']['name'][0]`,
`$HTTP_POST_FILES['userfile']['tmp_name'][0]`,
`$HTTP_POST_FILES['userfile']['size'][0]` y
`$HTTP_POST_FILES['userfile']['type'][0]` tambien son asignadas.

Soporte del método PUT

PHP soporta el metodo HTTP PUT que usan aplicaciones como Netscape Composer y Amaya del W3C. Las peticiones PUT son más sencillas que el método POST. Un ejemplo:

```
PUT /path/filename.html HTTP/1.1
```

Esto normalmente significaría que el cliente remoto quiere salvar el contenido como: /path/filename.html en tu árbol web. Lógicamente no una buena idea que la gente pueda escribir en tu árbol web. Para manipular esta petición debes decirle al servidor que esta petición sea atendida por un script PHP. En Apache, por ejemplo, se utiliza para esto la directiva *Script* en los alguno de los archivos de configuración del servidor. Un sitio típico de uso es dentro del bloque <Directory> o quizás en el bloque <Virtualhost>. Una linia así deberia hacer ésta función:

```
Script PUT /put.php
```

Ésto le dice a Apache que envíe todas peticiones PUT para URIs que contengan esta linia al script put.php. Se asume que PHP se encuentra activo y con la extensión .php enlazada a él.

Dentro del script put.php3 se podría implementar algo así:

```
<?php copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

Esto copiaría el archivo a la localización requerida por el cliente remoto. Aqui se pueden ejecutar funciones de autentificación de usuario o cualquier otro tipo de chequeo. El archivo se guarda en el archivo temporal del sistema servidor de la misma manera que el Método POST. Cuando la petición finaliza, el archivo temporal es eliminado. En consequencia el script dede proceder al trato de éste inmediatamente, ya sea para copiarlo, renombrarlo, etc. El archivo se encuentra en la variable $PHP_PUT_FILENAME, y el destino sugerido por el cliente en la variable $REQUEST_URI (puede variar en servidores web que no sean Apache). No es necesario hacer caso al destino sugerido por el cliente. Por ejemplo se podrían copiar los archivos enviados a directorios especialmente designados para esta tarea.

Capítulo 20. Usando archivos remotos

Siempre que el soporte para la "envoltura URL fopen" esté habilitado cuando se configura PHP (lo cual ocurre a menos que se pasa explícitamente la opción `--disable-url-fopen-wrapper` a configure (para versiones hasta la 4.0.3), ó configurar como "off" el parámetro *`allow_url_fopen`* en `php.ini` (para las nuevas versiones)) se pueden usar URLs HTTP y FTP con la mayoría de las funciones que toman un archivo como parámetro, incluyendo las sentencias require() e include().

> **Nota:** La versión actual de `PHP` para Windows no soporta el acceso remoto a ficheros en las siguientes funciones: include(), include_once(), require() y require_once().

Por ejemplo, se puede usar este para abrir un archivo en un servidor web remoto, analizar en la salida la información que se quiera, y entonces, usar la información en una consulta a base de datos, o simplemente para sacarlas en un estilo que coincida con el resto de su sitio web.

Ejemplo 20-1. Obtener el título de una página remota

```
<?php
$file = fopen ("http://www.example.com/", "r");
if (!$file) {
    echo "<p>Unable to open remote file.\n";
    exit;
}
while (!feof ($file)) {
    $line = fgets ($file, 1024);
    /* This only works if the title and its tags are on one line */
    if (eregi ("<title>(.*)</title>", $line, $out)) {
        $title = $out[1];
        break;
    }
}
fclose($file);
?>
```

También se puede escribir a archivos en un FTP siempre que se conecte como un usuario con los correctos derechos de acceso, y el archivo no exista ya.Para conectar como un usuario distinto de 'anonymous', se necesita especificar el nombre de usuario (y posiblemente contraseña) dentro de la URL, tales como 'ftp://usuario:clave@ftp.ejemplo.com/camino/a/archivo'. (Se puede usar la misma clase de sintaxis para acceder a archivos via HTTP cuando se requería una autenticació de same sort of syntax to access files via HTTP when they require Basic authentication.)

Ejemplo 20-2. Almacenando datos en un servidor remoto

```
<?php
$file = fopen ("ftp://ftp.example.com/incoming/outputfile", "w");
if (!$file) {
    echo "<p>Unable to open remote file for writing.\n";
    exit;
}
/* Write the data here. */
fputs ($file, $HTTP_SERVER_VARS['HTTP_USER_AGENT'] . "\n");
fclose ($file);
?>
```

Nota: Podeis captar la idea en el ejemplo anterior de como escribir en un registro remoto, pero como ya hemos comentado antes, solamente se puede escribir a un fichero nuevo usando la "envoltura URL fopen" Para registros distribuidos, consultar la función syslog().

Capítulo 21. Manejando conexiones

Nota: Todo lo siguiente se aplica a partir de la versión 3.0.7 y posterior.

Internamente en PHP se mantiene el estado de la conexión. Hay 3 posibles estados:

- 0 - NORMAL
- 1 - ABORTED (Abortado)
- 2 - TIMEOUT (Fuera de tiempo)

Cuando un script PHP se está ejecutando se activa el estado NORMAL. Si el cliente remoto se desconecta, se pasa al estado ABORTED. Esto suele ocurrir cuando el usuario pulsa en el botón STOP del navegador. Si se alcanza el límite de tiempo impuesto por PHP (ver **set_time_limit()**), se pasa al estado TIMEOUT.

Puedes decidir si quieres que la desconexión de un cliente cause que tu script sea abortado. Algunas veces es cómodo que tus scripts se ejecuten por completo, incluso si no existe ya un navegador remoto que reciba la salida. El comportamiento por defecto es sin embargo, que tu script se aborte cuando el cliente remoto se desconecta. Este comportamiento puede ser configurado vía la directiva ignore_user_abort en el fichero php3.ini, o también con la función ignore_user_abort(). Si no le espeficicas al PHP que cuando un usuario aborte lo ignore, tu script terminará su ejecución. La única excepción es si tienes registrada un función de desconexión usando la función register_shutdown_function(). Con una función de desconexión, cuando un usuario remoto pulsa en el botón STOP, la próxima vez que tu script intenta mostrar algo, PHP detecta que la conexión ha sido abortada y se llama a la función de desconexión. Esta función de desconexión también se llama al final de la ejecución de tu script cuando se ha ejecutado normalmente, de manera que si quieres hacer algo diferente en caso de que un cliente se haya desconectado, puedes usar la función connection_aborted(). Esta función devuelve `TRUE` si la conexión fue abortada.

Vuestro script también se puede terminar por un temporizador interno. El timeout por defecto es de 30 segundos. Se puede cambiar usando la directiva max_execution_time en el fichero `php.ini` o la correspondiente directiva php_max_execution_time en la configuración del servidor de páginas Apache, como también con la función set_time_limit(). Cuando el temporizador expira, el script se aborta como en el caso de la desconexión del cliente, de manera que si se ha definido una función de desconexión, esta se llamará. Dentro de esta función de desconexión, puedes comprobar si fue el timeout el que causó que se llamara a la función de desconexión, llamando a la función connection_timeout(). Esta función devolverá verdadero si el timeout causa que se llame a la función de desconexión.

Hay que destacar que ambos, el estado ABORTED y el TIMEOUT, se pueden activar al mismo tiempo. Esto es posible si le dices a PHP que ignore las desconexiones intencionadas de los usuarios. PHP aún notará el hecho de que el usuario puede haberse desconectado, pero el script continuará ejecutándose. Si se alcanza el tiempo límite de ejecución será abortado y, si se ha definido una función de desconexión, esta será llamada. En este punto, encontrarás que las funciones connection_timeout() y connection_aborted() devuelven verdadero. Puedes comprobar ambos estados de una manera simple usando la función **connection_status()**. Esta función devuelve un campo de bit de los estados activos. De este modo, si ambos estados están activos devolvería por ejemplo un valor 3.

Capítulo 22. Conexiones persistentes a bases de datos

Las conexiones persistentes son enlaces SQL que no se cierran cuando termina la ejecución del archivo de comandos. Cuando se pide una conexión persistente, PHP comprueba si hay ya una conexión persistente idéntica (que permanecía abierta desde antes) - y si existe, la usa. Si no existe, crea un enlace. Una conexión ’idéntica’ es una conexión que se abrió hacia el mismo "host", con el mismo nombre de usuario y la misma contraseña (donde sea aplicable).

> **Nota:** Existen otras extensiones que proporcionan conexiones persistentes, tal como la extensión IMAP

La gente que no está familiarizada con el modo como trabajan y distribuyen la carga los servidores "web" puede confundir que significa conexiones persistentes. En particular, *no* te dan la habilidad de abrir ’sesiones de usuario’ en el mismo enlace SQL, *no* dan la habilidad de construir una transacción de forma eficiente, y no hacen un montón de otras cosas. De hecho, para ser extremadamente claros sobre el tema las conexiones persistentes no te dan *ninguna* functionalidad que no fuera posible con sus hermanas no-persistentes.

¿Por qué?

Esto tiene que ver con el modo como funcionan los servidores "web". Hay tres modos en que un servidor "web" puede utilizar PHP para generar páginas web.

El primer método es usar PHP como una capa CGI. Cuando corre de este modo, se crea y destruye una instancia del intérprete PHP por cada página solicitada (para una página PHP) a tu servidor. Debido a que se destruye después de cada petición, cualquier recurso que adquiera (como un enlace a un servidor de base de datos SQL) se cierra cuando es destruido. En este caso, no se gana nada si se intentan usar conexiones persistentes, ya que simplemente no persisten.

El segundo, y más popular, método es correr PHP como un módulo en un servidor web multiproceso, lo cual actualmente sólo incluye Apache. Un servidor multiproceso tiene típicamente un proceso (el padre) que coordina un conjunto de procesos (sus hijos) que realmente hacen el trabajo se servir las páginas web. Cuando entra cada petición de un cliente, es entregada a uno de los hijos que no esté ya sirviendo a otro cliente. Esto significa que cuando el mismo cliente hace una segunda petción al servidor, puede ser atendido por un proceso hijo distinto del de la primera vez. Lo que una conexión persistente hace por ti en este caso es hacerlo de tal modo que cada proceso hijo sólo necesita conectar a tu SQL server la primera vez que sirve una página que hace uso de una conexión así. Cuando otra página solicita una conexión a SQL server, puede reutilizar la conexión que el hijo estableció previamente.

El último método es usar PHP como un "plug-in" para un servidor web multihilo. En la actualidad es solamente teórico -- PHP no funciona aún como "plug-in" para ningún servidor web multihilo. Actualmente PHP 4 soporta ISAPI, WSAPI y NSAPI (en Windows), lo cual permite a PHP ser utilizado como "plug-in" para servidores web multihilo como Netscape FastTrack, Internet Information Server (IIS) de Microsoft, y O’Reilly’s WebSite Pro. El comportamiento es exactamente el mismo que para el modelo de multiprocesador descrito anteriormente. Tener en cuanta que el soporte para SAPI no está disponible en PHP 3.

Si las conexiones persistentes no aportan ninguna funcionalidad añadida, ¿para qué son buenas?

La respuesta aqui es extremadamente simple -- eficiencia. Las conexiones persistentes son buenas si las cabeceras de control para crear un enlace a tu servidor SQL es alta. Que estas cabeceras sean o no realmente altas depende de muchos factores. Como, qué clase de base de datos es, si esta o no situada en el mismo ordenador que el servidor web, cómo está de cargada la máquina donde se encuentre el

servidor SQL, y otras así. El hecho fundamental es que si la cabecera de conexión es alta, las conexiones persistentes te ayudan considerablemente . Ellas hacen que el proceso hijo simplemente conecte solamente una vez durante todo su intervalo de vida, en vez de cada vez que procesa una pagina que requiere conectar al servidor SQL. Esto significa que por cada hijo que abrió una conexión persistente tendrá su propia conexión persistente al servidor. Por ejemplo, si tienes 20 procesos hijos distintos que corran un archivo de comandos que cree una conexión persistente a tu servidor SQL, tendrías 20 conexiones diferentes a ti servidor SQL, una por cada hijo.

No obstante, hay que tener en cuenta que esto puede tener desventajas si estais utilizando una base de datos con límites de conexión, por debajo del numero de procesos hijo con conexiones persistentes. Si tu base de datos tiene un límite de 16 conexiones simultaneas y en el curso de una sesión de servidor, 17 procesos hijo intentan conectarse, uno de ellos no podrá hacerlo. Si existen errores en los scripts, que no permitan terminar la conexion (p.ej.bucles infinitos), una base de datos con solo 32 conexiones puede ser rápidamente hundida. Comprobar la documentacion de vuestra base de datos para obtener información sobre que hacer con conexiones abandonadas ó libres.

Aviso

Un par de advertencias más a tener en cuenta cuando utiliceis conexiones persistentes. La primera, si utilizais bloqueos en una tabla desde una conexión persistente y por cualquier causa el script no puede desbloquear la tabla, todos los scripts posteriores que usen esta conexión, quedarán bloqueados indefinídamente y se requerirá que, ó bien el servidor httpd ó la base de datos sean arrancados de nuevo. La segunda, cuando utiliceis transacciones, un bloqueo por transacción será heredado por el próximo script usando la conexión, si la ejecución del primer script termina antes que el bloqueo. en ambos caso podeis utilizar register_shutdown_function() para registrar una funcion simple de limpieza que desbloquee las tablas ó deshaga la transacción. Lo mejor para evitar problemas es no usar conexiones persistentes en scripts que usen bloqueos de tablas ó transacciones (para todolo demás pueden ser usadas sin problemas)

Un resumen importante. Las conexiones persistentes fueron diseñadas para tener una equivalencia uno-a-uno con las conexiones normales. Eso significa que deberíais *siempre* ser capaz de reemplazar las conexiones persistentes por conexiones no persistentes y no cambiará, el modo como se comporta el archivo de comandos. *Puede* cambiar la eficiencia del archivo de comandos (y probablemete lo hará), ¡pero no su comportamiento!

Capítulo 23. Modo Seguro (Safe Mode)

El Modo Seguro de PHP es un intento para resolver el problema de la seguridad en un servidor compartido. Tratar de resolver este problema al nivel de PHP es arquitectónicamente incorrecto, pero ya que las alternativas en un servidor web y a niveles de sistemas operativos no son tan realistas, mucha gente, especialmente la de proveedores de Internet (ISP), usa el Modo Seguro por ahora.

Tabla 23-1. Las directivas de Configuración que controlan el Modo Seguro son:

Directiva	Valor por Omisión
safe_mode	`Off`
safe_mode_gid	`0`
safe_mode_include_dir	`""`
safe_mode_exec_dir	`1`
open_basedir	`""`
`safe_mode_allowed_env_vars`	`PHP_`
`safe_mode_protected_env_vars`	`LD_LIBRARY_PATH`
`disable_functions`	`""`

Cuando safe_mode está en On, el PHP verifica si el dueño del script actual coincide con el dueño del fichero a ser operado por una función de fichero. Por ejemplo:

```
-rw-rw-r--    1 rasmus   rasmus       33 Jul  1 19:20 script.php
-rw-r--r--    1 root     root       1116 May 26 18:01 /etc/passwd
```

Corriendo este script.php

```
<?php
 readfile('/etc/passwd');
?>
```

resulta in este error cuando Modo Seguro está habilitado:

```
Warning: SAFE MODE Restriction in effect. The script whose uid is 500 is not
allowed to access /etc/passwd owned by uid 0 in /docroot/script.php on line 2
```

Sin embargo, pueden haber ambientes donde una estricta verificación del `UID` no es apropiada, y una relajada verificación del `GID` es suficiente. Esto es soportado por medio del switch safe_mode_gid.

Seteándolo a `On` hace la verificación relajada `GID`, seteándolo a `Off` (el valor por omisión) hace la verificación del `UID`.

Si en vez del safe_mode, Ud. setea un directorio open_basedir, entonces todas las operaciones de fichero estarán limitadas a los ficheros bajo ese directorio especificado. Por ejemplo (ejemplo de httpd.conf de Apache):

```
<Directory /docroot>
  php_admin_value open_basedir /docroot
</Directory>
```

Si Ud. corre el mismo script.php con este seteo open_basedir, entonces este es el resultado:

```
Warning: open_basedir restriction in effect. File is in wrong directory in
/docroot/script.php on line 2
```

Ud. también puede inhabilitar funciones individuales. Note que la directiva disable_functions no puede ser usada fuera del fichero `php.ini` lo que significa que Ud. no puede inhabilitar funciones en los principios per-virtualhost o per-directory en su fichero httpd.conf. Si agregamos esto a nuestro fichero `php.ini`:

```
disable_functions readfile,system
```

Entonces obtenemos esta salida:

```
Warning: readfile() has been disabled for security reasons in
/docroot/script.php on line 2
```

Funciones restringidas/inhabilitadas por Modo Seguro

Esta es una lista probablemente incompleta y posiblemente incorrecta de las funciones limitadas por safe mode.

Tabla 23-2. Funciones limitadas por Modo Seguro

Función	Limitaciones
dbmopen()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
dbase_open()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
filepro()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
filepro_rowcount()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
filepro_retrieve()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
ifx_*()	restricciones sql_safe_mode, (!= safe mode)
ingres_*()	restricciones sql_safe_mode, (!= safe mode)
mysql_*()	restricciones sql_safe_mode, (!= safe mode)
pg_loimport()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
posix_mkfifo()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
putenv()	Obecede las ini-directivas safe_mode_protected_env_vars y safe_mode_allowed_env_vars. Vea también la documentación de putenv()
move_uploaded_file()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
chdir()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.

Función	**Limitaciones**
dl()	Esta función no está activada en safe-mode (modo-seguro)
backtick operator	Esta función no está activada en safe-mode (modo-seguro)
shell_exec() (equivalencia funcional de backticks)	Esta función no está activada en safe-mode (modo-seguro)
exec()	Ud. puede correr sólo ejecutables dentro delsafe_mode_exec_dir. Por razones prácticas, no está actualmente permitido tener componentes . . en la ruta del fichero ejecutable.
system()	Ud. puede correr sólo ejecutatables dentro delsafe_mode_exec_dir. Por razones prácticas, no está actualmente permitido tener componentes . . en la ruta del fichero ejecutable.
passthru()	Ud. puede correr sólo ejecutatables dentro delsafe_mode_exec_dir. Por razones prácticas, no está actualmente permitido tener componentes . . en la ruta del fichero ejecutable.
popen()	Ud. puede correr sólo ejecutatables dentro delsafe_mode_exec_dir. Por razones prácticas, no está actualmente permitido tener componentes . . en la ruta del fichero ejecutable.
mkdir()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
rmdir()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
rename()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.

Función	**Limitaciones**
unlink()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
copy()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. (en *source* y *target*)
chgrp()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
chown()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.
chmod()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. Además, Ud. no puede setear los bits de SUID, SGID y sticky
touch()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.

Función	**Limitaciones**
symlink()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. (Nota: sólo el target es comprobado)
link()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. (Nota: sólo the target es comprobado)
getallheaders()	En Modo Seguro, las cabeceras que empiezan con 'authorization' (insensitivo al tipo de letra) no serán retornadas. Advertencia: esto está roto por la implementación de aol-server de getallheaders()!
header()	En Modo Seguro, el UID del script está agregado a la parte `realm` de la cabecera `WWW-Authenticate` si Ud. setea esta cabecera (usado por HTTP Authentication).
highlight_file(), show_source()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. (Nota: sólo afectado desde PHP 4.2.1)
parse_ini_file()	**Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado. (Nota: sólo afectado desde PHP 4.2.1)
Cualquier función que usa `php4/main/fopen_wrappers.c`	??

Capítulo 24. Using PHP from the command line

Since version 4.3, PHP supports a new SAPI type (Server Application Programming Interface) named CLI which means *Command Line Interface*. As the name implies, this SAPI type main focus is on developing shell (or desktop as well) applications with PHP. There are quite some differences between the CLI SAPI and other SAPIs which are further explained throughout this chapter.

The CLI SAPI was released for the first time with PHP 4.2.0, but was still experimental back then and had to be explicitely enabled with --enable-cli when running ./configure. Since PHP 4.3.0 the CLI SAPI is no longer experimental and is therefore **always** built and installed as the php (called php.exe on Windows) binary.

Remarkable differences of the CLI SAPI compared to other SAPIs:

- Unlikely the CGI SAPI, no headers are written to the output.

  Though the CGI SAPI provies a way to suppress HTTP headers, there's not equivalent switch to enable them in the CLI SAPI.

- The are certain php.ini directives which are overriden by the CLI SAPI because the do not make sense in shell environments:

Tabla 24-1. Overriden php.ini directives

Directive	CLI SAPI default value	Comment
html_errors	FALSE	It can be quite hard to read the error message in your shell when it's cluttered with all those meaningless HTML tags, therefore this directive defaults to FALSE.
implicit_flush	TRUE	It is desired that any output coming from print(), echo() and friends is immidiately written to the output and not cached in any buffer. You still can use output buffering if you want to defer or manipulate standard output.
max_execution_time	0 (unlimited)	Due to endless possibilities of using PHP in shell environments, the maximum execution time has been set to unlimited. Whereas applications written for the web are executed within splits of a seconds, shell application tend to have a much longer execution time.

Directive	CLI SAPI default value	Comment
register_argc_argv	TRUE	The global PHP variables $argc (number of arguments passed to the application) and $argv (array of the actual arguments) are always registered and filled in with the appropriate values when using the CLI SAPI.

Nota: These directives cannot be initialzied with another value from the configuration file php.ini or a custom one (if specified). This is a limitation because those default values are applied after all configuration files have been parsed. However, their value can be changed during runtime (which does not make sense for all of those directives, e.g. register_argc_argv).

- To ease working in the shell environment, the following constants are defined:

Tabla 24-2. CLI specific Constants

Constant	Description
STDIN	An already opened stream to stdin. This saves opening it with `$stdin = fopen('php://stdin', 'r');`
STDOUT	An already opened stream to stdout. This saves opening it with `$stdout = fopen('php://stdout', 'w');`
STDERR	An already opened stream to stdout. This saves opening it with `$stderr = fopen('php://stderr', 'w');`

Given the above, you don't need to open e.g. a stream for stderr yourself but simply use the constant instead of the stream resource:

```
php -r 'fwrite(STDERR, "stderr\n");'
```

You do not need to explicitly close these streams, this is automatically done by PHP.

- The CLI SAPI does **not** change the current directory to the directory of the executed script !

Example showing the difference to the CGI SAPI:

```
<?php
    /* Our simple test application */
```

```
    echo getcwd(), "\n";
?>
```

When using the CGI version, the output is

```
$ pwd
/tmp

$ php-cgi -f another_directory/test.php
/tmp/another_directory
```

This clearly shows that PHP changes its current directory to the one of the executed script.

Using the CLI SAPI yields:

```
$ pwd
/tmp

$ php -f another_directory/test.php
/tmp
```

This allows greater flexibility when writing shell tools in PHP.

> **Nota:** The CGI SAPI supports the CLI SAPI behaviour by means of the -C switch when ran from the command line.

The list of command line options provided by the PHP binary can be queried anytime by running PHP with the -h switch:

```
Usage: php [options] [-f] <file> [args...]
       php [options] -r <code> [args...]
       php [options] [-- args...]
  -s               Display colour syntax highlighted source.
  -w               Display source with stripped comments and whitespace.
  -f <file>        Parse <file>.
  -v               Version number
  -c <path>|<file> Look for php.ini file in this directory
  -a               Run interactively
  -d foo[=bar]     Define INI entry foo with value 'bar'
  -e               Generate extended information for debugger/profiler
  -z <file>        Load Zend extension <file>.
```

```
-l               Syntax check only (lint)
-m               Show compiled in modules
-i               PHP information
-r <code>        Run PHP <code> without using script tags <?..?>
-h               This help

args...          Arguments passed to script. Use -- args when first argument
                 starts with - or script is read from stdin
```

The `CLI SAPI` has three different ways of getting the `PHP` code you want to execute:

1. Telling `PHP` to execute a certain file.

   ```
   php my_script.php

   php -f my_script.php
   ```

   Both ways (using the `-f` switch or not) execute the given file `my_script.php`. You can choose any file to execute, your `PHP` scripts do not have to end with the `.php` extension but can give them any name or extension you want them to have.

2. Pass the `PHP` code to execute directly on the command line.

   ```
   php -r 'print_r(get_defined_constants());'
   ```

   Special care has to be taken in regards of shell variable substitution and quoting usage.

   > **Nota:** Read the example carefully, thera are no beginning or ending tags! The `-r` switch simply does not need them. Using them will lead to a parser error.

3. Provide the `PHP` code to execute via standard input (`stdin`).

   This gives the powerful ability to dynamically create `PHP` code and feed it to the binary, as shown in this (fictional) example:

   ```
   $ some_application | some_filter | php | sort -u >final_output.txt
   ```

You cannot combine any of the three ways to execute code.

Like every shell application, the `PHP` binary accepts a number of arguments but also your `PHP` script can receive them. The number of arguments which can be passed to your script is not limited by `PHP` (the shell has a certain size limit in numbers of characters which can be passed; usually you won't hit this limit). The arguments passed to your script are available in the global array `$argv`. The zero index always contains the script name (which is `-` in case the `PHP` code is coming from either standard input or from the command line switch `-r`). The second registered global variable is `$argc` which contains the number of elements in the `$argv` array (**not** the number of arguments passed to the script).

As long as the arguments you want to pass to your script do not start with the `-` character, there's nothing special to watch out for. Passing an argument to your script which starts with a `-` will cause trouble because `PHP` itself thinks it has to handle it. To prevent this use the argument list separator `--`. After the argument has been parsed by `PHP`, every argument following it is passed untoched/unparsed to your script.

```
# This will not execute the given code but will show the PHP usage
$ php -r 'var_dump($argv);' -h
Usage: php [options] [-f] <file> [args...]
[...]

# This will pass the '-h' argument to your script and prevent PHP from showing it's usage
$ php -r 'var_dump($argv);' -- -h
array(2) {
  [0]=>
  string(1) "-"
  [1]=>
  string(2) "-h"
}
```

However, there's another way of using `PHP` for shell scripting. You can write a script where the first line starts with `#!/usr/bin/php` and then following the normal `PHP` code included within the `PHP` starting and end tags and set the execution attributes of the file appropriately. This way it can be executed like a normal shell or perl script:

```
#!/usr/bin/php
<?php
    var_dump($argv);
?>
```

Assuming this file is named `test` in the current directory, we can now do the following:

```
$ chmod 755 test
$ ./test -h -- foo
```

```
array(4) {
  [0]=>
  string(6) "./test"
  [1]=>
  string(2) "-h"
  [2]=>
  string(2) "--"
  [3]=>
  string(3) "foo"
}
```

As you see no care has to be taken when passing parameters to your script which start with `-`.

Tabla 24-3. Command line options

Option	Description
-s	Display colour syntax highlighted source. This option uses the internal mechanism to parse the file and produces a `HTML` highlighted version of it and writes it to standard output. Note that all it does it to generate a block of `<code> [...] </code>` `HTML` tags, no `HTML` headers. **Nota:** This option does not work together with the `-r` option.
-w	Display source with stripped comments and whitespace. **Nota:** This option does not work together with the `-r` option.
-f	Parses and executed the given filename to the `-f` option. This switch is optional and can be left out. Only providing the filename to execute is sufficient.

-v	Writes the PHP, PHP SAPI, and Zend version to standard output, e.g.
	
	`$ php -v`
	`PHP 4.3.0-dev (cli), Copyright (c) 1997-2002 The PHP Group`
	`Zend Engine v1.2.1, Copyright (c) 1998-2002 Zend Technologies`

Option	Description

-c	With this option one can either specify a directory where to look for `php.ini` or you can specify a custom `INI` file directly (which does not need to be named `php.ini`), e.g.:
	
	`$ php -c /custom/directory/ my_script.php`
	
	`$ php -c /custom/directory/custom-file.ini my_script`

-a	Runs PHP interactively.

-d	This option allows to set a custom value for any of the configuration directives allowed in `php.ini`. The syntax is:
	
	`-d configuration_directive[=value]`
	
	Examples:
	
	`# Ommiting the value part will set the given configuration directive to "1"`
	`$ php -d max_execution_time -r '$foo = ini_get("max_`
	`string(1) "1"`
	
	`# Passing an empty value part will set the configuration directive to ""`
	`php -d max_execution_time= -r '$foo = ini_get("max_e`
	`string(0) ""`
	
	`# The configuration directive will be set to anything passed after the '=' character`
	`$  php -d max_execution_time=20 -r '$foo = ini_get("m`
	`string(2) "20"`
	`$  php -d max_execution_time=doesntmakesense -r '$foo = ini_get("max_execution_time"); var_dump($f`
	`string(15) "doesntmakesense"`

-e	Generate extended information for debugger/profiler.

Option	Description
-z	Load Zend extension. If only a filename is given, PHP tries to load this extension from the current default library path on your system (usually specified `/etc/ld.so.conf` on Linux systems). Passing a filename with an absolute path information will not use the systems library search path. A relative filename with a directory information will tell `PHP` only to try to load the extension relative to the current directory.
-l	This option provides a convenient way to only perform a syntax check on the given `PHP` code. On succes, the text `No syntax errors detected in <filename>` is written to standard output and the shell return code is `0`. On failure, the text `Errors parsing <filename>` in addition to the internal parser error message is written to standard output and the shell return code is set to `255`. This option won't find fatal errors (like undefined functions). Use `-f` if you would like to test for fatal errors too. **Nota:** This option does not work together with the `-r` option.

-m	Using this option, PHP prints out the built in (and loaded) PHP and Zend modules:
	
	`$ php -m`
	`[PHP Modules]`
	`xml`
	`tokenizer`
	`standard`
	`session`
	`posix`
	`pcre`
	`overload`
	`mysql`
	`mbstring`
	`ctype`
	
	`[Zend Modules]`

Option	Description
-i	This command line option calls phpinfo(), and prints out the results. If `PHP` is not working well, it is advisable to make a `php -i` and see if any error messages are printed out before or in place of the information tables. Beware that the output is in `HTML` and therefore quite huge.

Option	Description

-r	This option allows execution of PHP right from within the command line. The PHP start and end tags (<?php and ?>) are **not needed** and will cause a parser errors.
	Nota: Care has to be taken when using this form of PHP to not collide with command line variable substitution done by the shell.
	Example showing a parser error
	`$ php -r "$foo = get_defined_constants();"`
	`Command line code(1) : Parse error - parse error, unexpected '='`
	The problem here is that the sh/bash performs variable substitution even when using double quotes ". Since the variable $foo is unlikely to be defined, it expands to nothing which results in being the code passed to PHP for executin in fact reads:
	`$ php -r " = get_defined_constants();"`
	The correct way would be to use single quotes '. variables in strings quoted with single quotes are not expanded by sh/bash.
	`$ php -r '$foo = get_defined_constants(); var_dump($fo`
	`array(370) {`
	`["E_ERROR"]=>`
	`int(1)`
	`["E_WARNING"]=>`
	`int(2)`
	`["E_PARSE"]=>`
	`int(4)`
	`["E_NOTICE"]=>`
	`int(8)`
	`["E_CORE_ERROR"]=>`
	`[...]`
	If you are using a shell different from sh/bash, you might experience further issues. Feel free to open a bug report or send a mail to phpdoc@lists.php.net. One still can easily run intro troubles when trying to get shell variables into the code or using backslashes for escaping. You've been warned.

Option	Description
-h	With this option, you can get information about the actual list of command line options and some one line descriptions about what they do.

The PHP executable can be used to run PHP scripts absolutely independent from the web server. If you are on a Unix system, you should add a special first line to your PHP script, and make it executable, so the system will know, what program should run the script. On a Windows platform you can associate `php.exe` with the double click option of the `.php` files, or you can make a batch file to run the script through PHP. The first line added to the script to work on Unix won't hurt on Windows, so you can write cross platform programs this way. A simple example of writing a command line PHP program can be found below.

Ejemplo 24-1. Script intended to be run from command line (script.php)

```
#!/usr/bin/php
<?php

if ($argc != 2 || in_array($argv[1], array('--help', '-help', '-h', '-?'))) {
?>

This is a command line PHP script with one option.

  Usage:
  <?php echo $argv[0]; ?> <option>

  <option> can be some word you would like
  to print out. With the --help, -help, -h,
  or -? options, you can get this help.

<?php
} else {
    echo $argv[1];
}
?>
```

In the script above, we used the special first line to indicate, that this file should be run by PHP. We work with a CLI version here, so there will be no HTTP header printouts. There are two variables you can use while writing command line applications with PHP: `$argc` and `$argv`. The first is the number of arguments plus one (the name of the script running). The second is an array containing the arguments, starting with the script name as number zero (`$argv[0]`).

In the program above we checked if there are less or more than one arguments. Also if the argument was `--help`, `-help`, `-h` or `-?`, we printed out the help message, printing the script name dynamically. If we received some other argument we echoed that out.

If you would like to run the above script on Unix, you need to make it executable, and simply call it as `script.php echothis` or `script.php -h`. On Windows, you can make a batch file for this task:

Ejemplo 24-2. Batch file to run a command line PHP script (script.bat)

```
@c:\php\php.exe script.php %1 %2 %3 %4
```

Assuming, you named the above program as `script.php`, and you have your `php.exe` in `c:\php\php.exe` this batch file will run it for you with your added options: `script.bat echothis` or `script.bat -h`.

See also the Readline extension documentation for more functions you can use to enhance your command line applications in PHP.

Parte IV. Referencia de las Funciones

I. Funciones específicas de Apache

Estas funciones están disponibles solamente si ejecutamos PHP como módulo de Apache.

apache_child_terminate (PHP 4 >= 4.0.5)

Terminate apache process after this request

bool **apache_child_terminate** (void) \linebreak

apache_child_terminate() will register the Apache process executing the current PHP request for termination once execution of PHP code it is completed. It may be used to terminate a process after a script with high memory consumption has been run as memory will usually only be freed internally but not given back to the operating system.

Nota: The availability of this feature is controlled by the `php.ini` directive `apache.child_terminate`, which is set to `off` by default.

This feature is also not available on multithreaded versions of apache like the win32 version.

See also exit().

apache_lookup_uri (PHP 3>= 3.0.4, PHP 4)

Efectua una petición parcial a la URI especificada y devuelve toda la información sobre ella.

class **apache_lookup_uri** (string filename) \linebreak

Esta función efectua una llamada parcial a URI. Esta llamada no hace sino obtener toda la información importante sobre el recurso pedido y la devuelve en un tipo clase .Las propiedades de esa clase son:
status
the_request
status_line
method
content_type
handler
uri
filename
path_info
args
boundary
no_cache
no_local_copy
allowed
send_bodyct
bytes_sent
byterange
clength
unparsed_uri
mtime
request_time

Nota: Nota: apache_lookup_uri solo funciona cuando el PHP está instalado como módule del Apache.

apache_note (PHP 3>= 3.0.2, PHP 4)

Recibe y establece los valores de una petición en una tabla de notas del Apache

string **apache_note** (string note_name [, string note_value]) \linebreak

apache_note() es una función específica del Apache que recibe y establece valores de la petición en una `tabla de notas`. Si se llama con un solo parámetro,devuelve el valor de `note_name`. Si se llama con dos parámetros, establece el valor de `note_value` en `note_value` y devuelve el valor que había en `note_name`.

apache_setenv (PHP 4 >= 4.2.0)

Set an Apache subprocess_env variable

int **apache_setenv** (string variable, string value [, bool walk_to_top]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ascii2ebcdic (PHP 3>= 3.0.17)

Translate string from ASCII to EBCDIC

int **ascii2ebcdic** (string ascii_str) \linebreak

ascii2ebcdic() is an Apache-specific function which is available only on EBCDIC based operating systems (OS/390, BS2000). It translates the ASCII encoded string *`ascii_str`* to its equivalent EBCDIC representation (binary safe), and returns the result.

See also the reverse function ebcdic2ascii()

ebcdic2ascii (PHP 3>= 3.0.17)

Translate string from EBCDIC to ASCII

int **ebcdic2ascii** (string ebcdic_str) \linebreak

ebcdic2ascii() is an Apache-specific function which is available only on EBCDIC based operating systems (OS/390, BS2000). It translates the EBCDIC encoded string *`ebcdic_str`* to its equivalent ASCII representation (binary safe), and returns the result.

See also the reverse function ascii2ebcdic()

getallheaders (PHP 3, PHP 4)

Recibe todas las cabeceras de una petición HTTP

array **getallheaders** (void) \linebreak

Esta función devuelve asociados en un vector todas las cabeceras de la actual petición HTTP.

> **Nota:** También puedes obtener los valores de las variables de los CGIs mediante variables de entorno, que funcionan, esté o no el PHP funcionando como módulo del Apache. Utiliza phpinfo()para ver una lista de todas las variables de entorno definidas de esta forma.

Ejemplo 1. ObtenerTodaslasCabeceras() Ejemplo

```
$cabeceras = getallheaders();
while (list($cabecera, $valor) = each($cabeceras)) {
    echo "$cabecera: $valor<br>\n";
}
```

Este ejemplo visualiza todas las cabeceras de la petición actual.

> **Nota: ObtenerTodaslasCabeceras()** actualmente solo funcionará si el PHP es cargado como módulo del Apache .

virtual (PHP 3, PHP 4)

Ejecuta una sub-petición al Apache

int **virtual** (string filename) \linebreak

virtual() es una función específica del Apache que es equivalente a <!--#include virtual...--> en mod_include. Esto ejecuta una sup-petición al Apache .Esto, es util para incluir CGI-scripts o páginas .shtml o cualquier tipo de fichero que puedas procesar mediante el Apache. Los CGI-scripts deberán generar cabeceras válidas. Esto, implica como mínimo un include() ó un require(); La función **virtual()** no puede ser usada para incluir un documento que sea por si mismo un documento PHP.

II. Funciones de matrices

Introducción

Estas funciones permiten trabajar y manipular matrices (arrays) de diferentes maneras. Las matrices se utilizan para guardar, manejar y operar grupos de variables.

Matrices simples y multi-dimensionales están soportadas y pueden ser creadas por el usuario u otras funciones. Existen funciones específicas de manejo de bases de datos que actualizan matrices con el resultado devuelto por la base de datos, numerosas otras funciones devuelven matrices como resultado.

Consultar la sección del manual Matrices si quereis una explicación detallada de como las matrices están implementadas en PHP.

Requerimientos

Estas funciones están disponibles como parte del módulo estandar, el cual está siempre disponible.

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

Esta extensión no define ningún tipo de recurso.

Constantes predefinidas

`CASE_UPPER` y `CASE_LOWER` son usadas con la función array_change_key_case(). Son usadas

respectivamente para cambiar una cadena literal de mayúsculas a minúsculas.

Ver tambien

Ver tambien is_array(), explode(), implode(), split(), y join().

array_change_key_case (PHP 4 >= 4.2.0)

Returns an array with all string keys lowercased or uppercased

array **array_change_key_case** (array input [, int case]) \linebreak

array_change_key_case() changes the keys in the *input* array to be all lowercase or uppercase. The change depends on the last optional *case* parameter. You can pass two constants there, CASE_UPPER and CASE_LOWER. The default is CASE_LOWER. The function will leave number indices as is.

Ejemplo 1. array_change_key_case() example

```
$input_array = array("FirSt" => 1, "SecOnd" => 4);
print_r(array_change_key_case($input_array, CASE_UPPER));
```

The printout of the above program will be:

```
Array
(
    [FIRST] => 1
    [SECOND] => 2
)
```

array_chunk (PHP 4 >= 4.2.0)

Split an array into chunks

array **array_chunk** (array input, int size [, bool preserve_keys]) \linebreak

array_chunk() splits the array into several arrays with *size* values in them. You may also have an array with less values at the end. You get the arrays as members of a multidimensional array indexed with numbers starting from zero.

By setting the optional *preserve_keys* parameter to TRUE, you can force PHP to preserve the original keys from the input array. If you specify FALSE new number indices will be used in each resulting array with indices starting from zero. The default is FALSE.

Ejemplo 1. array_chunk() example

```
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
print_r(array_chunk($input_array, 2, TRUE));
```

The printout of the above program will be:

```
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [0] => c
            [1] => d
        )

    [2] => Array
        (
            [0] => e
        )

)
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [2] => c
            [3] => d
        )

    [2] => Array
        (
            [4] => e
        )

)
```

array_count_values (PHP 4)

Cuenta todos los valores de una matriz

array **array_count_values** (array entrada) \linebreak

array_count_values() devuelve una matriz usando los valores de la matriz *`entrada`* como claves y su frecuencia de aparición en la *`entrada`* como valores.

Ejemplo 1. Ejemplo de array_count_values()

```
$matriz = array(1, "hola", 1, "mundo", "hola");
array_count_values($matriz); // devuelve array(1=>2, "hola"=>2, "mundo"=>1)
```

Nota: Esta función fue añadida en el PHP 4.0.

array_diff (PHP 4 >= 4.0.1)

Computes the difference of arrays

array **array_diff** (array array1, array array2 [, array ...]) \linebreak

array_diff() returns an array containing all the values of *`array1`* that are not present in any of the other arguments. Note that keys are preserved.

Ejemplo 1. array_diff() example

```
$array1 = array ("a" => "green", "red", "blue", "red");
$array2 = array ("b" => "green", "yellow", "red");
$result = array_diff ($array1, $array2);
```

This makes `$result` have `array ("blue");`. Multiple occurrences in $array1 are all treated the same way.

Nota: Two elements are considered equal if and only if `(string) $elem1 === (string) $elem2`. In words: when the string representation is the same.

Aviso

This was broken in PHP 4.0.4!

See also array_intersect().

array_fill (PHP 4 >= 4.2.0)

Fill an array with values

array **array_fill** (int start_index, int num, mixed value) \linebreak

array_fill() fills an array with *num* entries of the value of the *value* parameter, keys starting at the *start_index* parameter.

Ejemplo 1. array_fill() example

```
$a = array_fill(5, 6, 'banana');
```

$a now has the following entries using print_r():

```
Array
(
    [5]  => banana
    [6]  => banana
    [7]  => banana
    [8]  => banana
    [9]  => banana
    [10] => banana
)
```

array_filter (PHP 4 >= 4.0.6)

Filters elements of an array using a callback function

array **array_filter** (array input [, mixed callback]) \linebreak

array_filter() returns an array containing all the elements of *input* filtered according a callback function. If the *input* is an associative array the keys are preserved.

Ejemplo 1. array_filter() example

```
function odd($var) {
    return ($var % 2 == 1);
}

function even($var) {
    return ($var % 2 == 0);
}

$array1 = array ("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$array2 = array (6, 7, 8, 9, 10, 11, 12);

echo "Odd :\n";
print_r(array_filter($array1, "odd"));
echo "Even:\n";
print_r(array_filter($array2, "even"));
```

The printout of the program above will be:

```
Odd :
Array
(
    [a] => 1
    [c] => 3
    [e] => 5
)
Even:
Array
(
    [0] => 6
    [2] => 8
    [4] => 10
    [6] => 12
)
```

Nota: En lugar de un nombre de función, se pueden proporcionar una matriz que contenga una referencia a un objeto o el nombre de un método.

Users may not change the array itself from the callback function. e.g. Add/delete an element, unset the array that **array_filter()** is applied to. If the array is changed, the behavior of this function is undefined.

See also array_map() and array_reduce().

array_flip (PHP 4)

Intercambia los valores de una matriz

array **array_flip** (array trans) \linebreak

array_flip() devuelve una matriz con los valores intercambiados.

Ejemplo 1. Ejemplo de array_flip()

```
$trans = array_flip ($trans);
$original = strtr ($str, $trans);
```

Nota: Esta función fue añadida en el PHP 4.0.

array_intersect (PHP 4 >= 4.0.1)

Computes the intersection of arrays

array **array_intersect** (array array1, array array2 [, array ...]) \linebreak

array_intersect() returns an array containing all the values of *`array1`* that are present in all the arguments. Note that keys are preserved.

Ejemplo 1. array_intersect() example

```
$array1 = array ("a" => "green", "red", "blue");
$array2 = array ("b" => "green", "yellow", "red");
$result = array_intersect ($array1, $array2);
```

This makes `$result` have

```
Array
(
```

```
    [a] => green
    [0] => red
)
```

Nota: Two elements are considered equal if and only if `(string) $elem1 === (string) $elem2`. In words: when the string representation is the same.

Aviso

This was broken in PHP 4.0.4!

See also array_diff().

array_key_exists (PHP 4 >= 4.1.0)

Checks if the given key or index exists in the array

bool **array_key_exists** (mixed key, array search) \linebreak

array_key_exists() returns `TRUE` if the given *`key`* is set in the array. *`key`* can be any value possible for an array index.

Ejemplo 1. array_key_exists() example

```
$search_array = array("first" => 1, "second" => 4);
if (array_key_exists("first", $search_array)) {
    echo "The 'first' element is in the array";
}
```

Nota: The name of this function is **key_exists()** in PHP version 4.0.6.

See also isset().

array_keys (PHP 4)

Devuelve todas las claves de una matriz

array **array_keys** (array entrada [, mixed val_a_buscar]) \linebreak

array_keys() devuelve las claves, numéricas y de cadena, de la matriz *entrada*.

Si se especifica el parámetro opcional *val_a_buscar*, sólo se devuelven las claves para dicho valor. De otro modo, se devuelven todas las claves de la *entrada*.

Ejemplo 1. Ejemplo de array_keys()

```
$matriz = array(0 => 100, "color" => "rojo");
array_keys ($matriz);       // devuelve array (0, "color")

$matriz = array(1, 100, 2, 100);
array_keys ($matriz, 100);  // devuelve array (0, 2)
```

Vea también: array_values().

Nota: Esta función fue añadida en el PHP 4.0.

array_map (PHP 4 >= 4.0.6)

Applies the callback to the elements of the given arrays

array **array_map** (mixed callback, array arr1 [, array arr2...]) \linebreak

array_map() returns an array containing all the elements of *arr1* after applying the callback function to each one. The number of parameters that the callback function accepts should match the number of arrays passed to the **array_map()**

Ejemplo 1. array_map() example

```
function cube($n) {
    return $n*$n*$n;
}

$a = array(1, 2, 3, 4, 5);
$b = array_map("cube", $a);
print_r($b);
```

This makes `$b` have:

```
Array
(
    [0] => 1
    [1] => 8
    [2] => 27
    [3] => 64
    [4] => 125
)
```

Ejemplo 2. array_map() - using more arrays

```
function show_Spanish($n, $m) {
    return "The number $n is called $m in Spanish";
}

function map_Spanish($n, $m) {
    return array ($n => $m);
}

$a = array(1, 2, 3, 4, 5);
$b = array("uno", "dos", "tres", "cuatro", "cinco");

$c = array_map("show_Spanish", $a, $b);
print_r($c);

$d = array_map("map_Spanish", $a , $b);
print_r($d);
```

This results:

```
// printout of $c
Array
(
    [0] => The number 1 is called uno in Spanish
    [1] => The number 2 is called dos in Spanish
    [2] => The number 3 is called tres in Spanish
    [3] => The number 4 is called cuatro in Spanish
    [4] => The number 5 is called cinco in Spanish
)
```

```
// printout of $d
Array
(
    [0] => Array
        (
            [1] => uno
        )

    [1] => Array
        (
            [2] => dos
        )

    [2] => Array
        (
            [3] => tres
        )

    [3] => Array
        (
            [4] => cuatro
        )

    [4] => Array
        (
            [5] => cinco
        )

)
```

Usually when using two or more arrays, they should be of equal length because the callback function is applied in parallel to the corresponding elements. If the arrays are of unequal length, the shortest one will be extended with empty elements.

An interesting use of this function is to construct an array of arrays, which can be easily performed by using `NULL` as the name of the callback function

Ejemplo 3. Creating an array of arrays

```
$a = array(1, 2, 3, 4, 5);
$b = array("one", "two", "three", "four", "five");
$c = array("uno", "dos", "tres", "cuatro", "cinco");

$d = array_map(null, $a, $b, $c);
print_r($d);
```

The printout of the program above will be:

```
Array
(
    [0] => Array
        (
            [0] => 1
            [1] => one
            [2] => uno
        )

    [1] => Array
        (
            [0] => 2
            [1] => two
            [2] => dos
        )

    [2] => Array
        (
            [0] => 3
            [1] => three
            [2] => tres
        )

    [3] => Array
        (
            [0] => 4
            [1] => four
            [2] => cuatro
        )

    [4] => Array
        (
            [0] => 5
            [1] => five
            [2] => cinco
        )

)
```

See also array_filter() and array_reduce().

array_merge_recursive (PHP 4 >= 4.0.1)

Merge two or more arrays recursively

array **array_merge_recursive** (array array1, array array2 [, array ...]) \linebreak

array_merge_recursive() merges the elements of two or more arrays together so that the values of one are appended to the end of the previous one. It returns the resulting array.

If the input arrays have the same string keys, then the values for these keys are merged together into an array, and this is done recursively, so that if one of the values is an array itself, the function will merge it with a corresponding entry in another array too. If, however, the arrays have the same numeric key, the later value will not overwrite the original value, but will be appended.

Ejemplo 1. array_merge_recursive() example

```
$ar1 = array ("color" => array ("favorite" => "red"), 5);
$ar2 = array (10, "color" => array ("favorite" => "green", "blue"));
$result = array_merge_recursive ($ar1, $ar2);
```

The `$result` will be:

```
Array
(
    [color] => Array
        (
            [favorite] => Array
                (
                    [0] => red
                    [1] => green
                )

            [0] => blue
        )

    [0] => 5
    [1] => 10
)
```

See also array_merge().

array_merge (PHP 4)

Combina dos o más matrices

array **array_merge** (array matriz1, array matriz2 [, ...]) \linebreak

array_merge() combina los elementos de dos o más matrices conjuntamente de modo que los valores de una son agregados al final de los valores de la anterior. Devuelve la matriz resultante.

Si las matrices de entrada tienen las mismas claves de cadena, el último valor para cada clave reemplazará el valor previo de la misma. Si, por el contrario, las matrices tienen la misma clave numérica, esto no pasa y los valores son simplemente agregados.

Ejemplo 1. Ejemplo de array_merge()

```
$matriz1 = array ("color" => "rojo", 2, 4);
$matriz2 = array ("a", "b", "color" => "verde", "forma" => "trapezoide");
array_merge ($matriz1, $matriz2);
```

La matriz resultante sería array("color" => "verde", 2, 4, "a", "b", "forma" => "trapezoide").

Nota: Esta función fue añadida en el PHP 4.0.

array_multisort (PHP 4)

Sort multiple or multi-dimensional arrays

bool **array_multisort** (array ar1 [, mixed arg [, mixed ... [, array ...]]]) \linebreak

array_multisort() can be used to sort several arrays at once or a multi-dimensional array according by one of more dimensions. It maintains key association when sorting.

The input arrays are treated as columns of a table to be sorted by rows - this resembles the functionality of SQL ORDER BY clause. The first array is the primary one to sort by. The rows (values) in that array that compare the same are sorted by the next input array, and so on.

The argument structure of this function is a bit unusual, but flexible. The very first argument has to be an array. Subsequently, each argument can be either an array or a sorting flag from the following lists.

Sorting order flags:

- SORT_ASC - sort in ascending order
- SORT_DESC - sort in descending order

Sorting type flags:

- SORT_REGULAR - compare items normally
- SORT_NUMERIC - compare items numerically
- SORT_STRING - compare items as strings

No two sorting flags of the same type can be specified after each array. The sorting flags specified after an array argument apply only to that array - they are reset to default SORT_ASC and SORT_REGULAR after before each new array argument.

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

Ejemplo 1. Sorting multiple arrays

```
$ar1 = array ("10", 100, 100, "a");
$ar2 = array (1, 3, "2", 1);
array_multisort ($ar1, $ar2);
```

In this example, after sorting, the first array will contain 10, "a", 100, 100. The second array will contain 1, 1, "2", 3. The entries in the second array corresponding to the identical entries in the first array (100 and 100) were sorted as well.

Ejemplo 2. Sorting multi-dimensional array

```
$ar = array (array ("10", 100, 100, "a"), array (1, 3, "2", 1));
array_multisort ($ar[0], SORT_ASC, SORT_STRING,
                 $ar[1], SORT_NUMERIC, SORT_DESC);
```

In this example, after sorting, the first array will contain 10, 100, 100, "a" (it was sorted as strings in ascending order), and the second one will contain 1, 3, "2", 1 (sorted as numbers, in descending order).

array_pad (PHP 4)

Rellena una matriz con un valor hasta el tamaño especificado

array **array_pad** (array entrada, int tama_relleno, mixed valor_relleno) \linebreak

array_pad() Devuelve una copia de la *entrada* rellenada hasta el tamaño *tama_relleno* con el valor *valor_relleno*. Si *tama_relleno* es positivo, entonces la matriz es rellenada por la derecha, y si es negativo, por la izquierda. Si el valor absoluto de *tama_relleno* es menor o igual que el tamaño de la *entrada* no se produce relleno alguno.

Ejemplo 1. Ejemplo de array_pad()

```
$entrada = array (12, 10, 9);

$resultado = array_pad ($entrada, 5, 0);
// el resultado es array (12, 10, 9, 0, 0)

$resultado = array_pad ($entrada, -7, -1);
// el resultado es array (-1, -1, -1, -1, 12, 10, 9)

$resultado = array_pad ($entrada, 2, "no");
// no rellenado
```

array_pop (PHP 4)

Extrae el último elemento de la matriz

mixed **array_pop** (array matriz) \linebreak

array_pop() extrae y devuelve el último valor de la *matriz*, acortando la *matriz* en un elemento.

Ejemplo 1. Ejemplo de array_pop()

```
$pila = array ("naranja", "manzana", "frambuesa");
$fruta = array_pop ($pila);
```

Tras esto, $pila contiene sólo 2 elementos: "naranja" y "manzana", y $fruta contiene "frambuesa".

Vea también: array_push(), array_shift(), y array_unshift().

Nota: Esta función fue añadida en el PHP 4.0.

array_push (PHP 4)

Inserta uno o más elementos al final de la matriz

int **array_push** (array matriz, mixed var [, ...]) \linebreak

array_push() considera a la *matriz* como una pila, e inserta las variables que se le pasan al final de la *matriz*. La longitud de la *matriz* se incrementa en el número de variables insertadas. Tiene el mismo efecto que ejecutar:

```
$matriz[] = $var;
```

para cada *var*.

Devuelve el nuevo número de elementos de la matriz.

Ejemplo 1. Ejemplo de array_push()

```
$pila = array (1, 2);
array_push($pila, "+", 3);
```

Este ejemplo dejará $pila conteniendo 4 elementos: 1, 2, "+", y 3.

Vea también: array_pop(), array_shift(), y array_unshift().

Nota: Esta función fue añadida en el PHP 4.0.

array_rand (PHP 4)

Pick one or more random entries out of an array

mixed **array_rand** (array input [, int num_req]) \linebreak

array_rand() is rather useful when you want to pick one or more random entries out of an array. It takes an *input* array and an optional argument *num_req* which specifies how many entries you want to pick - if not specified, it defaults to 1.

If you are picking only one entry, **array_rand()** returns the key for a random entry. Otherwise, it returns an array of keys for the random entries. This is done so that you can pick random keys as well as values out of the array.

Don’t forget to call srand() to seed the random number generator.

Ejemplo 1. array_rand() example

```
srand ((float) microtime() * 10000000);
$input = array ("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$rand_keys = array_rand ($input, 2);
print $input[$rand_keys[0]]."\n";
print $input[$rand_keys[1]]."\n";
```

array_reduce (PHP 4 >= 4.0.5)

Iteratively reduce the array to a single value using a callback function

mixed **array_reduce** (array input, mixed callback [, int initial]) \linebreak

array_reduce() applies iteratively the *`callback`* function to the elements of the array *`input`*, so as to reduce the array to a single value. If the optional *`initial`* is available, it will be used at the beginning of the process, or as a final result in case the array is empty.

Ejemplo 1. array_reduce() example

```
function rsum($v, $w) {
    $v += $w;
    return $v;
}

function rmul($v, $w) {
    $v *= $w;
    return $v;
}

$a = array(1, 2, 3, 4, 5);
$x = array();
$b = array_reduce($a, "rsum");
$c = array_reduce($a, "rmul", 10);
$d = array_reduce($x, "rsum", 1);
```

This will result in `$b` containing `15`, `$c` containing `1200` (= 1*2*3*4*5*10), and `$d` containing `1`.

See also array_filter() and array_map().

array_reverse (PHP 4)

Devuelve una matriz con los elementos en orden inverso

array **array_reverse** (array matriz) \linebreak

array_reverse() toma la *matriz* de entrada y devuelve una nueva matriz con los elementos en orden inverso.

Ejemplo 1. Ejemplo de array_reverse()

```
$entrada = array ("php", 4.0, array ("verde", "rojo"));
$resultado = array_reverse ($entrada);
```

Esto hace que $resultado contenga array (array ("verde", "rojo"), 4.0, "php").

Nota: Esta función fue añadida en PHP 4.0 Beta 3.

array_search (PHP 4 >= 4.0.5)

Searches the array for a given value and returns the corresponding key if successful

mixed **array_search** (mixed needle, array haystack [, bool strict]) \linebreak

Searches *haystack* for *needle* and returns the key if it is found in the array, FALSE otherwise.

Nota: Prior to PHP 4.2.0, **array_search()** returns NULL on failure instead of FALSE.

If the optional third parameter *strict* is set to TRUE then the **array_search()** will also check the types of the *needle* in the *haystack*.

Aviso

Esta función puede devolver el Boolean FALSE, pero también puede devolver un valor no-Boolean que será evaluado FALSE, como por ejemplo 0 o "". Por favor, lea la sección Booleans para más información. Utilice el operador === para comprobar el valor devuelto por esta función.

See also array_keys() and in_array().

array_shift (PHP 4)

Extrae un elemento del comienzo de la matriz

mixed **array_shift** (array matriz) \linebreak

array_shift() extrae el primer valor de la *matriz* y lo devuele, acortando la *matriz* en un elemnto y moviendo todo hacia arriba.

Ejemplo 1. Ejemplo de array_shift()

```
$args = array ("-v", "-f");
$opcion = array_shift ($args);
```

Esto da como resultado que $args tenga como elemento restante "-f" y que $opcion valga "-v".

Vea también: array_unshift(), array_push(), y array_pop().

Nota: Esta función fue añadida en el PHP 4.0.

array_slice (PHP 4)

Extrae una porción de la matriz

array **array_slice** (array matriz, int desplazamiento [, int tamano]) \linebreak

array_slice() devuelve una secuencia de elementos de la *matriz* especificada por los parámetros *desplazamiento* y *tamano*.

Si el *desplazamiento* es positivo, la secuencia comenzará en dicha posición de la *matriz*. Si el *desplazamiento* es negativo, la secuencia comenzará en esa posición desde el final de la *matriz*.

Si se especifica el *tamano* y éste es positivo, la secuencia contendrá tantos elementos como se diga en él. Si fuese negativo, la secuencia se detendrá a tantos elementos del final de la matriz. Si se omite, la secuencia contendrá todos los elementos desde el *desplazamiento* hasta el final de la *matriz*.

Ejemplo 1. Ejemplo de array_slice() examples

```
$entrada = array ("a", "b", "c", "d", "e");

$salida = array_slice ($entrada, 2);      // devuelve "c", "d", y "e"
$salida = array_slice ($entrada, 2, -1);  // devuelve "c", "d"
$salida = array_slice ($entrada, -2, 1);  // devuelve "d"
$salida = array_slice ($entrada, 0, 3);   // devuelve "a", "b", y "c"
```

Vea también: array_splice().

Nota: Esta función fue añadida en el PHP 4.0.

array_splice (PHP 4)

Suprime una porción de la matriz y la sustituye por otra cosa

array **array_splice** (array entrada, int desplazamiento [, int tamano [, array sustitucion]]) \linebreak

array_splice() suprime los elementos designados por el *desplazamiento* y el *tamano* de la matriz *entrada*, y los sustituye con los elementos de la matriz de *sustitucion* si se especifica.

Si el *desplazamiento* es positivo, el comienzo de la parte suprimida sería en esa posición desde el comienzo de la matriz de *entrada*. Si el *desplazamiento* es negativo, se cuenta la posición desde el final de la matriz de *entrada*.

Si se omite *tamano*, se suprime todo desde el *desplazamiento* hasta el final de la matriz. Si se especifica el *tamano* y es positivo, se suprimirán tantos elementos como se especifica. Si fuera negativo, el final de la porción eliminada estará a tantos elementos del final de la matriz. Truco: para eliminar todo desde el *desplazamiento* hasta el final de la matriz cuando también se especifica *sustitucion*, utilice `count($entrada)` como *tamano*.

Si se especifia la matriz de *sustitucion*, entonces los elementos suprimidos son reemplazados con los elementos de dicha matriz. Si los valores de *desplazamiento* y *tamano* son tales que nada es borrado, los elementos de la matriz *sustitucion* se insertarán en la posición indicada por el *desplazamiento*. Truco: si sólo se va a sustituir algo por un elemento nada más, no hace falta poner `array()` alrededor del mismo, salvo que dicho elemento sea una matriz en sí mismo.

Las siguientes funciones son equivalentes:

```
array_push($entrada, $x, $y)     array_splice($entrada, count($entrada), 0, ar-
ray($x, $y))
array_pop($entrada)              array_splice($entrada, -1)
array_shift($entrada)            array_splice($entrada, 0, 1)
array_unshift($entrada, $x, $y)  array_splice($entrada, 0, 0, array($x, $y))
$a[$x] = $y                    array_splice($entrada, $x, 1, $y)
```

Devuelve una matriz que tiene los elementos eliminados

Ejemplo 1. Ejemplos de array_splice()

```
$entrada = array("rojo", "verde", "azul", "amarillo");

array_splice($entrada, 2);      // $entrada vale ahora array("rojo", "verde")
array_splice($entrada, 1, -1);  // $entrada vale ahora array("rojo", "amarillo")
array_splice($entrada, 1, count($entrada), "naranja");
                                // $entrada vale ahora array("rojo", "naranja")
array_splice($entrada, -1, 1, array("negro", "marrón"));
                                // $entrada vale ahora array("rojo", "verde",
                                //          "azul", "negro", "marrón")
```

Vea también: array_slice().

Nota: Esta función fue añadida en el PHP 4.0.

array_sum (PHP 4 >= 4.0.4)

Calculate the sum of values in an array.

mixed **array_sum** (array array) \linebreak

array_sum() returns the sum of values in an array as an integer or float.

Ejemplo 1. array_sum() examples

```
$a = array(2, 4, 6, 8);
echo "sum(a) = ".array_sum($a)."\n";

$b = array("a"=>1.2,"b"=>2.3,"c"=>3.4);
echo "sum(b) = ".array_sum($b)."\n";
```

The printout of the program above will be:

```
sum(a) = 20
sum(b) = 6.9
```

Nota: PHP versions prior to 4.0.6 modified the passed array itself and converted strings to numbers (which most of the time converted them to zero, depending on their value).

array_unique (PHP 4 >= 4.0.1)

Removes duplicate values from an array

array **array_unique** (array array) \linebreak

array_unique() takes input *array* and returns a new array without duplicate values.

Note that keys are preserved. **array_unique()** sorts the values treated as string at first, then will keep the first key encountered for every value, and ignore all following keys. It does not mean that the key of the first related value from the unsorted *array* will be kept.

Nota: Two elements are considered equal if and only if `(string) $elem1 === (string) $elem2`. In words: when the string representation is the same.

The first element will be used.

Aviso

This was broken in PHP 4.0.4!

Ejemplo 1. array_unique() example

```
$input = array ("a" => "green", "red", "b" => "green", "blue", "red");
$result = array_unique ($input);
print_r($result);
```

This will output:

```
Array
(
    [b] => green
    [1] => blue
    [2] => red
)
```

Ejemplo 2. array_unique() and types

```
$input = array (4,"4","3",4,3,"3");
$result = array_unique ($input);
var_dump($result);
```

The printout of the program above will be (PHP 4.0.6):

```
array(2) {
  [3]=>
  int(4)
  [4]=>
  int(3)
}
```

array_unshift (PHP 4)

Introduce uno o más elementos al principio de la matriz

int **array_unshift** (array matriz, mixed var [, ...]) \linebreak

array_unshift() añade los elementos que se le pasan al principio de la *matriz*. Nótese que la lista de elementos es añadida como un todo, de modo que los elementos añadidos mantienen su orden.

Devuelve el número de elementos en la *matriz*.

Ejemplo 1. Ejemplo de array_unshift()

```
$cola = array("p1", "p3");
array_unshift($cola, "p4", "p5", "p6");
```

Esto hará que $cola contenga 5 elementos: "p4", "p5", "p6", "p1", y "p3".

Vea también: array_shift(), array_push(), y array_pop().

Nota: Esta función fue añadida en el PHP 4.0.

array_values (PHP 4)

Devuelve todos los valores de una matriz

array **array_values** (array entrada) \linebreak

array_values() devuelve todos los valores de la matriz *entrada*.

Ejemplo 1. Ejemplo de array_values()

```
$matriz = array("talla" => "XL", "color" => "dorado");
array_values($matriz);    // devuelve array("XL", "dorado")
```

Nota: Esta función fue añadida en el PHP 4.0.

array_walk (PHP 3>= 3.0.3, PHP 4)

Aplica una función del usuario a cada elemento de una matriz.

int **array_walk** (array matriz, string func, mixed datosvarios) \linebreak

Aplica la función llamada *func* a cada elemento de la *matriz*. La función *func* recibirá el valor de la matriz como primer parámetro y la clave como segundo. Si se proporciona el parámetro *datosvarios* será pasado como tercer parámetro a la función de usuario.

Si *func* necesita más de dos o 3 argumentos, dependiendo de *datosvarios*, se generará un aviso cada vez que **array_walk()** llama a *func*. Estos avisos pueden suprimirse si se pone '@' antes de la llamada a **array_walk()**, o usando la función error_reporting().

Nota: Si *func* precisa trabajar con los valores reales de la matriz, especifique que el valor del primer parámetro de *func* debe pasarse por referencia. Desde ese instante, los cambios realizados sobre dichos elementos también serán realizados en la propia matriz.

Nota: El pasar la clave y los datos de usuario a *func* fue una característica añadida en PHP 4.0.

En PHP 4 se debe llamar reset() las veces necesarias, pues **array_walk()** no reajusta la matriz por defecto.

Ejemplo 1. Ejemplo de array_walk()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana");

function test_alterar (&$item1, $clave, $prefix) {
   $item1 = "$prefix: $item1";
}

function test_ver ($item2, $clave) {
   echo "$clave. $item2<br>\n";
}

array_walk ($frutas, 'test_ver');
reset ($frutas);
array_walk ($frutas, 'test_alterar', 'fruta');
reset ($frutas);
array_walk ($frutas, 'test_ver');
```

Vea también: each() y list().

array (unknown)

Crear una matriz

array **array** (mixed ...) \linebreak

Devuelve una matriz con los parámetros que se le pasan. A dichos parámetros se les puede dar un índice usando el operador =>.

> **Nota: array()** es una construcción del lenguaje que se utiliza para representar matrices literales, no una función regular.

El siguiente ejemplo demuestra cómo crear una matriz bidimensional, cómo especificar claves para matrices asociativas, y cómo especificar índices no consecutivos en matrices normales.

Ejemplo 1. Ejemplo de array()

```
$frutas = array (
    "frutas"  => array("a"=>"naranja", "b"=>"plátano", "c"=>"manzana"),
    "números" => array(1, 2, 3, 4, 5, 6),
    "hoyos"   => array("primero", 5 => "segundo", "tercero")
);
```

Vea también: list().

arsort (PHP 3, PHP 4)

Ordena una matriz en orden inverso y mantiene la asociación de índices

void **arsort** (array matriz) \linebreak

Esta función ordena una matriz de modo que los índices mantengan su correlación con los elementos de la misma a los que están asociados. Esto se utiliza principalmente para ordenar matrices asociativas en las que el orden de los elementos es importante.

Ejemplo 1. Ejemplo de arsort()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana");
arsort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostraría: `frutas[b] = plátano frutas[a] = naranja frutas[c] = manzana frutas[d] = limón` Las frutas han sido ordenadas en orden alfabético inverso y los índices asociados con cada elemento se han mantenido.

Vea también: asort(), rsort(), ksort(), y sort().

asort (PHP 3, PHP 4)

Ordena una matriz y mantiene la asociación de índices

void **asort** (array matriz) \linebreak

Esta función ordena una matriz de modo que los índices mantengan su correlación con los elementos de la misma a los que están asociados. Esto se utiliza principalmente para ordenar matrices asociativas en las que el orden de los elementos es importante.

Ejemplo 1. Ejemplo de asort()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana");
asort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostrará: `frutas[d] = limón frutas[a] = naranja frutas[c] = manzana frutas[d] = plátano` Las frutas han sido ordenadas en orden alfabético y los índices asociados con cada elemento se han mantenido.

Vea también: arsort(), rsort(), ksort(), y sort().

compact (PHP 4)

Crea una matriz que contiene variables y sus valores

array **compact** (string nombrevar | array nombrevars [, ...]) \linebreak

compact() toma un número variable de parámetros. Cada uno puede ser tanto una cadena que contiene el nombre de la variable, como una matriz de nombres de variable. La matriz puede contener otras matrices de nombres de variable en su interior; **compact()** los procesa recursivamente.

Para cada uno de estos, **compact()** busca una variable con dicho nombre en la tabla de símbolos y la añade a la matriz de salida de modo que el nombre de la variable es la clave y el contenido de ésta es el valor para dicha clave. Para resumir, hace lo contrario de extract(). Devuelve la matriz de salida con las variables añadidas a la misma.

Ejemplo 1. Ejemplo de compact()

```
$ciudad = "San Francisco";
$estado = "CA";
$evento = "SIGGRAPH";

$location_vars = array ("ciudad", "estado");

$resultado = compact ("evento", $location_vars);
```

Tras esto, $resultado valdrá array ("evento" => "SIGGRAPH", "ciudad" => "San Francisco", "estado" => "CA").

Vea también: extract().

Nota: Esta función fue añadida en el PHP 4.0.

count (PHP 3, PHP 4)

Cuenta los elementos de una variable

int **count** (mixed var) \linebreak

Devuelve el número de elementos en *var*, que típicamente es una matriz (porque cualquier otra cosa tendría sólo un elemento).

Devuele 1 si la variable no es una matriz.

Devuelve 0 si la variable no tiene valor.

> **Aviso**
>
> **count()** puede devolver 0 para una variable sin valor, pero también puede devolver 0 para una variable ya inicializada pero con una matriz vacía. Utilice isset() para comprobar si una variable está inicializada.

Vea también: sizeof(), isset(), y is_array().

current (PHP 3, PHP 4)

Devuelve el elemento actual de una matriz

mixed **current** (array matriz) \linebreak

Cada matriz tiene un puntero interno al elemento "actual", que se inicializa al primer elemento insertado en la misma.

La función **current()** simplemente devuelve el elemento de la tabla al que apunta el puntero interno. No mueve el puntero de ninguna manera. Si el puntero interno apunta fuera del final de la lista de elementos, **current()** devuelve `FALSE`.

> **Aviso**
>
> Si la matriz contiene elementos vacíos (0 ó "", la cadena vacía) esta función devolverá `FALSE` también para dichos elementos. Esto hace imposible determinar si se está realmente al final de la lista en tales matrices usando **current()**. Para recorrer adecuadamente una matriz que pueda contener elementos vacíos, utilice la función each().

Vea también: end(), next(), prev() y reset().

each (PHP 3, PHP 4)

Devuelve el siguiente par clave/valor de una matriz

array **each** (array matriz) \linebreak

Devuelve el par clave/valor actual para la *matriz* y avanza el cursor de la misma. Esta pareja se devuele en una matriz de 4 elementos, con las claves *0*, *1*, *key*, y *value*. Los elementos *0* y *key* contienen el nombre de clave del elemento de la matriz, y *1* y *value* contienen los datos.

Si el puntero interno para la matriz apunta pasado el final del contenido de la matriz, **each()** devuelve FALSE.

Ejemplo 1. Ejemplos de each()

```
$chorrada = array ("bob", "fred", "jussi", "jouni", "egon", "marliese");
$tonteria = each ($chorrada);
```

$tonteria contiene ahora los siguientes pares clave/valor:

- 0 => 0
- 1 => 'bob'
- key => 0
- value => 'bob'

```
$chorrada = array ("Robert" => "Bob", "Seppo" => "Sepi");
$tonteria = each ($chorrada);
```

$tonteria contiene ahora los siguientes pares clave/valor:

- 0 => 'Robert'
- 1 => 'Bob'
- key => 'Robert'
- value => 'Bob'

each() se usa normalmente de forma conjunta a list() para recorrer una matriz; por ejemplo, $HTTP_POST_VARS:

Ejemplo 2. Recorriendo $HTTP_POST_VARS con each()

```
echo "Valores enviados con el método POST:<br>";
reset ($HTTP_POST_VARS);
while (list ($clave, $val) = each ($HTTP_POST_VARS)) {
    echo "$clave => $val<br>";
}
```

Cuando se ha ejecutado **each()**, el cursor de la matriz quedará en el siguiente elemento de la misma, o en el último si llega al final de ésta.

Vea también: key(), list(), current(), reset(), next(), y prev().

end (PHP 3, PHP 4)

Mueve el puntero interno de una tabla al último elemento

end (array matriz) \linebreak

end() avanza el puntero interno de la *matriz* al último elemento.

Vea también: current(), each(), **end()**, next(), y reset().

extract (PHP 3>= 3.0.7, PHP 4)

Importa variables a la tabla de símbolos desde una matriz

void **extract** (array matriz_vars [, int tipo_extraccion [, string prefijo]]) \linebreak

Esta función se utiliza para importar variables desde una matriz a la tabla de símbolos actual. Toma la matriz asoiativa *matriz_vars* y trata las claves como nombres de variable y los valores como los valores de éstas. Para cada par clave/valor creará una variable en la tabla de símbolos actual, sujeto a los parámetros *tipo_extraccion* y *prefijo*.

extract() controla las colisiones con las variables que ya existen. La forma de tratar éstas se determina por el *tipo_extraccion*. Puede tener únicamente uno de los siguientes valores:

EXTR_OVERWRITE

Si hay colisión, sobreescribe la variable existente.

EXTR_SKIP

Si hay colisión, no sobreescribas la variable existente.

EXTR_PREFIX_SAME

Si hay una colisión, añade el *prefijo* a la nueva variable.

EXTR_PREFIX_ALL

Añade el *prefijo* a todas las variables.

Si no se especifica *tipo_extraccion*, se asume que vale EXTR_OVERWRITE.

Nótese que el *prefijo* sólo se necesita si *tipo_extraccion* vale EXTR_PREFIX_SAME o EXTR_PREFIX_ALL.

extract() comprueba si cada clave es un nombre válido de variable, y sólo lo importa si lo es.

> **Nota:** N.T.: En el caso español, no valdría "año" como nombre variable (pero sí como clave en una matriz cualquiera).

Un uso posible para extract sería importar en la tabla de símbolos las variables contenidas en la matriz asociativa que devuelve wddx_deserialize().

Ejemplo 1. Ejemplo de extract()

```
<php?

/* Suponemos que $matriz_var es una matriz devuelta por
   wddx_deserialize */

$tamano = "grande";
$matriz_var = array ("color" => "azul",
                     "tamano"  => "media",
                     "forma" => "esfera");
extract ($matriz_var, EXTR_PREFIX_SAME, "wddx");

print "$color, $tamano, $forma, $wddx_tamano\n";

?>
```

El programa anterior producirá:

```
azul, grande, esfera, media
```

La variable $tamano no fue sobreescrita porque especificamos EXTR_PREFIX_SAME, que provocó la creación de $wddx_tamano. Si se hubiera especificado EXTR_SKIP, $wddx_tamano ni siquiera habría sido creada. EXTR_OVERWRITE habría provocado que $tamano tuviera el valor "media", y EXTR_PREFIX_ALL habría provocado que aparecieran nuevas variables llamadas $wddx_color, $wddx_tamano, y $wddx_forma.

in_array (PHP 4)

Devuelve `TRUE` si un valor está en una matriz

bool **in_array** (mixed aguja, array pajar) \linebreak

Busca la *aguja* en el *pajar*, y devuelve TRUE si se encuentra y FALSE en caso contrario.

Ejemplo 1. Ejemplo de in_array()

```
$os = array ("Mac", "NT", "Irix", "Linux");
if (in_array ("Irix", $os))
    print "Encontrado Irix";
```

Nota: Esta función fue añadida en el PHP 4.0.

key (PHP 3, PHP 4)

Obtiene una clave de una matriz asociativa

mixed **key** (array matriz) \linebreak

key() devuelve el elemento índice de la posición actual en la matriz.

Vea también: current(), next()

krsort (PHP 3>= 3.0.13, PHP 4)

Ordena una matriz por clave en orden inverso

int **krsort** (array matriz) \linebreak

Ordena una matriz por clave en orden inverso, manteniendo las correlaciones clave a dato. Esto es útil principalmente en matrices asociativas.

Ejemplo 1. Ejemplo de krsort()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana");
krsort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostrará: `frutas[d] = limón frutas[c] = manzana frutas[b] = plátano frutas[a] = naranja`

Vea también: asort(), arsort(), ksort() sort(), y rsort().

ksort (PHP 3, PHP 4)

Ordena una matriz por clave

int **ksort** (array matriz) \linebreak

Ordena una matriz por clave, manteniendo las correlaciones clave a dato. Esto es útil principalmente en matrices asociativas.

Ejemplo 1. Ejemplo de ksort()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana");
ksort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostrará: `frutas[a] = naranja frutas[b] = plátano frutas[c] = manzana frutas[d] = limón`

Vea también: asort(), arsort(), sort(), y rsort().

list (unknown)

Asigna variables como si fueran una matriz

void **list** (mixed ...) \linebreak

Como array(), esta no es realmente una función, sino una construcción del lenguaje. **list()** se usa para asignar una lista de variables en una sola operación.

Ejemplo 1. Ejemplo de list()

```
<table>
 <tr>
  <th>Nombre empleado</th>
  <th>Sueldo</th>
 </tr>

<?php

$resultado = mysql($conn, "SELECT id, nombre, salario FROM empleados");
while (list($id, $nombre, $salario) = mysql_fetch_row($resultado)) {
    print(" <tr>\n".
          "  <td><a href=\"info.php3?id=$id\">$nombre</a></td>\n".
```

```
            "  <td>$salario</td>\n".
            " </tr>\n");
}

?>

</table>
```

Vea también: each(), array().

natcasesort (PHP 4)

Sort an array using a case insensitive "natural order" algorithm

void **natcasesort** (array array) \linebreak

This function implements a sort algorithm that orders alphanumeric strings in the way a human being would. This is described as a "natural ordering".

natcasesort() is a case insensitive version of natsort(). See natsort() for an example of the difference between this algorithm and the regular computer string sorting algorithms.

For more information see: Martin Pool’s Natural Order String Comparison (http://naturalordersort.org/) page.

See also sort(), natsort(), strnatcmp(), and strnatcasecmp().

natsort (PHP 4)

Sort an array using a "natural order" algorithm

void **natsort** (array array) \linebreak

This function implements a sort algorithm that orders alphanumeric strings in the way a human being would. This is described as a "natural ordering". An example of the difference between this algorithm and the regular computer string sorting algorithms (used in sort()) can be seen below:

Ejemplo 1. natsort() example

```
$array1 = $array2 = array ("img12.png", "img10.png", "img2.png", "img1.png");

sort($array1);
echo "Standard sorting\n";
print_r($array1);
```

```
natsort($array2);
echo "\nNatural order sorting\n";
print_r($array2);
```

The code above will generate the following output:

```
Standard sorting
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Natural order sorting
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

For more information see: Martin Pool's Natural Order String Comparison (http://naturalordersort.org/) page.

See also natcasesort(), strnatcmp(), and strnatcasecmp().

next (PHP 3, PHP 4)

Avanza el puntero interno de una matriz

mixed **next** (array matriz) \linebreak

Devuelve el elemento de la matriz que ocupa el lugar siguiente al apuntado por el puntero interno, o `FALSE` si no hay más elementos.

next() se comporta como current(), con una diferencia. Avanza el puntero interno de la matriz en una posición antes de devolver el elemento. Eso significa que devuelve el siguiente elemento de la matriz y que avanza el puntero interno en uno. Si al avanzar se pasa del final de la lista de elementos, **next()** devuelve `FALSE`.

> **Aviso**
>
> Si la matriz contiene elementos vacíos, esta función también devolverá `FALSE` para dichos elementos. Para recorrer adecuadamente una matriz que pueda contener elementos vacíos, vea la función each().

Vea también: current(), end() prev() y reset()

pos (PHP 3, PHP 4)

Obtiene el elemento actual de una matriz

mixed **pos** (array matriz) \linebreak

Este es un alias para current().

Vea también: end(), next(), prev() y reset().

prev (PHP 3, PHP 4)

Rebobina el puntero interno de una matriz

mixed **prev** (array matriz) \linebreak

Devuelve el elemento de la matriz que está en la posición anterior a la que apuntaba previamente el puntero interno, o `FALSE` si no hay más elementos.

> **Aviso**
>
> Si la matriz contiene elementos vacíos, esta función también devolverá `FALSE` para dichos elementos. Para recorrer adecuadamente una matriz que puede contener elementos vacíos, vea la función each().

prev() se comporta igual que next(), excepto que rebobina el puntero interno una posición en lugar de avanzarlo.

Vea también: current(), end() next() y reset()

rango (unknown)

Crea una matriz que contiene un rango de enteros

array **rango** (int bajo, int alto) \linebreak

rango() devuelve una matriz de enteros desde *bajo* hasta *alto*, ambos inclusive.

Vea un ejemplo de su uso en la función shuffle().

reset (PHP 3, PHP 4)

Fija el puntero interno de una matriz a su primer elemento

mixed **reset** (array matriz) \linebreak

reset() rebobina el puntero interno de la *matriz* a su primer elemento.

reset() devuelve el valor del primer elemento de la matriz.

Vea también: current(), each(), next(), prev(), y **reset()**.

rsort (PHP 3, PHP 4)

Ordena una matriz en orden inverso

void **rsort** (array matriz) \linebreak

Esta función ordena una matriz en orden inverso (mayor a menor).

Ejemplo 1. Ejemplo de rsort()

```
$frutas = array ("limón", "naranja", "plátano", "manzana");
rsort ($frutas);
for (reset ($frutas); list ($clave, $valor) = each ($frutas); ) {
    echo "frutas[$clave] = ", $valor, "\n";
}
```

Este ejemplo mostrará: `frutas[0] = plátano frutas[1] = naranja frutas[2] = manzana frutas[3] = limón` Las frutas han sido ordenadas en orden alfabético inverso.

Vea también: arsort(), asort(), ksort(), sort(), y usort().

shuffle (PHP 3>= 3.0.8, PHP 4)

Mezcla una matriz

void **shuffle** (array matriz) \linebreak

Esta función mezcla (cambia aleatoriamente el orden de los elementos de) una matriz.

Ejemplo 1. Ejemplo de shuffle()

```
$numeros = range (1,20);
srand (time());
shuffle ($numeros);
while (list(, $numero) = each ($numeros)) {
    echo "$numero ";
}
```

Vea también: arsort(), asort(), ksort(), rsort(), sort() y usort().

sizeof (PHP 3, PHP 4)

Obtiene el número de elementos de una matriz

int **sizeof** (array matriz) \linebreak

Devueve el número de elementos de la matriz.

Vea también: count()

sort (PHP 3, PHP 4)

Ordena una matriz

void **sort** (array matriz) \linebreak

Esta función ordena una matriz. Los elementos estarán ordenados de menor a mayor cuando la función termine.

Ejemplo 1. Ejemplo de sort()

```
$frutas = array ("limón", "naranja", "plátano", "manzana");
sort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostrará: `frutas[0] = limón frutas[1] = manzana frutas[2] = naranja frutas[3] = plátano` Las frutas han sido ordenadas en orden alfabético.

Vea también: arsort(), asort(), ksort(), rsort(), y usort().

uasort (PHP 3>= 3.0.4, PHP 4)

Ordena una matriz mediante una función de comparación definida por el usuario y mantiene la asociación de índices

void **uasort** (array matriz, function func_comparar) \linebreak

Esta función ordena una matriz de modo que los índices de la misma mantengan su correlación con los elementos a los que están asociados. Esto se utiliza principalmente para ordenar matrices asociativas en las que el orden de los elementos es importante. La función de comparación viene definida por el usuario.

uksort (PHP 3>= 3.0.4, PHP 4)

Ordena una matriz por claves mediante una función definida por el usuario

void **uksort** (array matriz, function func_comparar) \linebreak

Esta función ordenará las claves de una matriz utilizando una función de comparación suministrada por el usuario. Si la matriz a ordenar necesita utilizar un criterio poco trivial, esta es la función que deberá usar.

Ejemplo 1. Ejemplo de uksort()

```
function micomparar ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}
$a = array (4 => "cuatro", 3 => "tres", 20 => "veinte", 10 => "diez");
uksort ($a, micomparar);
while (list ($clave, $valor) = each ($a)) {
    echo "$clave: $valor\n";
}
```

Este ejemplo mostrará: `20: veinte 10: diez 4: cuatro 3: tres`

Vea también: arsort(), asort(), uasort(), ksort(), rsort(), y sort().

usort (PHP 3>= 3.0.3, PHP 4)

Ordena una matriz por valores mediante una funcion definida por el usuario

void **usort** (array matriz, function func_comparar) \linebreak

Esta función ordenará una matriz por sus valores utilizando una función suministrada por el usuario. Si la matriz que desea ordenar necesita utilizar un criterio poco trivial, esta es la función que deberá usar.

La función de comparación deberá devolver un entero menor, igual, o mayor que cero, si el primer argumento se considera respectivamente menor que, igual que, o mayor que el segundo. Si dos miembros resultan ser iguales, su orden en la matriz ordenada será cualquiera.

Ejemplo 1. Ejemplo de usort()

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}
$a = array (3, 2, 5, 6, 1);
usort ($a, cmp);
while (list ($clave, $valor) = each ($a)) {
    echo "$clave: $valor\n";
}
```

Este ejemplo mostrará: `0: 6 1: 5 2: 3 3: 2 4: 1`

Nota: Obviamente en este caso trivial la función rsort() habría sido más apropiada.

Aviso

La función quicksort subyacente en ciertas librerías de C (tales como las de Solaris) pueden hacer que el PHP falle si la función de comparación no devuelve valores consistentes.

Vea también: arsort(), asort(), ksort(), rsort() y sort().

III. Funciones Aspell [deprecated]

Introducción

La función **aspell()** permite comprobar la ortografía de una palabra y ofrece alternativas a la misma.

Requerimientos

aspell funciona solamente con versiones muy antiguas (hasta la .27.* mas ó menos) de la biblioteca aspell. Ni este módulo ni las versiones de la biblioteca aspell se soportan actualmente. Si quereis utilizar capacidades de comprobación ortográfica en php, usar pspell. Utiliza la biblioteca pspell y funciona con versiones recientes de aspell.

Instalación

Necesitais la biblioteca aspell disponible en: http://aspell.sourceforge.net/.

Ver tambien

Ver tambien pspell.

aspell_check-raw (PHP 3>= 3.0.7, PHP 4)

Comprueba una palabra sin cambiarla ó intentar arreglarla [deprecated]

boolean **aspell_check_raw** (int dictionary_link, string word) \linebreak

aspell_check_raw() comprueba la ortografía de una palabra,sin cambiarla ni intentar arreglarla esté bien o mal. Si está bien, devuelve cierto (`TRUE`), si no lo está, devuelve falso(`FALSE`).

Ejemplo 1. aspell_check_raw

```
$aspell_link = aspell_new("english");

if (aspell_check_raw($aspell_link, "test")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

aspell_check (PHP 3>= 3.0.7, PHP 4)

Comprueba una palabra[deprecated]

boolean **aspell_check** (int dictionary_link, string word) \linebreak

aspell_check() comprueba la ortografía de una palabra, y devuelve cierto(`TRUE`) si la ortografía es correcta ,falso (`FALSE`) si no lo es .

Ejemplo 1. aspell_check

```
$aspell_link = aspell_new("english");

if (aspell_check($aspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

aspell_new (PHP 3>= 3.0.7, PHP 4)

Lee un nuevo diccionario [deprecated]

int **aspell_new** (string master, string personal) \linebreak

aspell_new() Abre un nuevo diccionario devolviendo el identificador de este para ser utilizado en otras funciones ortográficas.

Ejemplo 1. Nuevo_diccionario

```
$aspell_link = aspell_new("english");
```

aspell_suggest (PHP 3>= 3.0.7, PHP 4)

Sugiere la ortografía para una palabra [deprecated]

array **aspell_suggest** (int dictionary_link, string word) \linebreak

aspell_suggest() devuelve una matriz con posibles correciones ortográficas para la palabra dada.

Ejemplo 1. aspell_suggest

```
$aspell_link = aspell_new("english");

if (!aspell_check($aspell_link, "test")) {
    $suggestions = aspell_suggest($aspell_link, "test");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br>\n";
    }
}
```

IV. Funciones matemáticas de precisión arbitraria BCMath

Introducción

Para operaciones matemáticas de precisión arbitraria, PHP tiene disponible la Calculadora Binaria que soporta números de cualquier tamaño y precisión, representados como cadenas de texto.

Requerimientos

Desde PHP 4.0.4, libbcmath se encuentra incorporada en PHP. No se necesitan bibliotecas externas para esta extensión.

Instalación

En PHP 4, estas funciones están disponibles solamente si PHP ha sido configurado con `--enable-bcmath` en PHP 3, estas funciones están disponibles solamente si PHP no ha sido configurado con `--disable-bcmath`.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

Esta extensión no define ningún tipo de recurso.

Constantes predefinidas

Esta extensión no define ninguna constante.

bcadd (PHP 3, PHP 4)

Suma dos números de precisión arbitriaria.

string **bcadd** (string operando izq, string operando der [, int escala]) \linebreak

Suma el *operando izq* con el *operando der* y devuelve la suma en una cadena de texto. El parámetro opcional *escala* se usa para fijar el número de dígitos tras el punto decimal que aparecerán en el resultado.

Vea también bcsub().

bccomp (PHP 3, PHP 4)

Compara dos números de precisión arbitraria.

int **bccomp** (string operando izq, string operando der [, int escala]) \linebreak

Compara el *operando izq* con el *operando der* y devuelve el resultado como un entero. El parámetro opcional *escala* se usa para fijar el número de dígitos tras el punto decimal que se utilizarán en la comparación. El valor devuelto es 0 si los dos operandos son iguales. Si el *operando izq* es mayor que el *operando der* el valor devuelto es +1 y si el *operando izq* es menor que el *operando der* el valor devuelto es -1.

bcdiv (PHP 3, PHP 4)

Divide dos números de precisión arbitraria.

string **bcdiv** (string operando izq, string operando der [, int escala]) \linebreak

Divide el *operando izq* por el *operando der* y devuelve el resultado. El parámetro opcional *escala* fija el número de dígitos tras el punto decimal a usar en el resultado.

Ver también bcmul().

bcmod (PHP 3, PHP 4)

Obtiene el módulo de un número de precisión arbitraria.

string **bcmod** (string operando izq, string modulo) \linebreak

Obtiene el módulo del *operando izq* usando *modulo*.

Ver también bcdiv().

bcmul (PHP 3, PHP 4)

Multiplica dos números de precisión arbitraria.

string **bcmul** (string operando izq, string operando der [, int escala]) \linebreak

Multiplica el *operando izq* por el *operando der* y devuelve el resultado. El parámetro opcional *escala* fija el número de dígitos tras el punto decimal del resultado.

Ver también bcdiv().

bcpow (PHP 3, PHP 4)

Eleva un número de precisión arbitraria a otro.

string **bcpow** (string x, string y [, int escala]) \linebreak

Eleva *x* a la potencia de *y*. El parámetro opcional *escala* se puede usar para fijar el número de dígitos tras el punto decimal del resultado.

Ver también bcsqrt().

bcscale (PHP 3, PHP 4)

Fija el parámetro de escala por defecto para todas las funciones matemáticas bc.

string **bcscale** (int escala) \linebreak

Esta función fija el parámetro de escala por defecto para las subsiguientes funciones matemáticas bc que no especifican dicho parámetro explícitamente.

bcsqrt (PHP 3, PHP 4)

Obtiene la raíz cuadrada de un número de precisión arbitraria.

string **bcsqrt** (string operando, int escala) \linebreak

Devuelve la raíz cuadrada del *operando*. El parámetro opcional *escala* fija el número de dígitos tras el punto decimal del resultado.

Ver también bcpow().

bcsub (PHP 3, PHP 4)

Resta un número de precisión arbitraria de otro.

string **bcsub** (string operando izq, string operando der [, int escala]) \linebreak

Resta el *`operando der`* del *`operando izq`* y devuelve el resultado en una cadena. El parámetro opcional *`escala`* se utiliza para fijar el número de dígitos tras el punto decimal del resultado.

Ver también bcadd().

V. Funciones de compresión Bzip2

Introducción

Las funciones bzip2 son usadas para leer y escribir de forma transparente, ficheros comprimidos bzip2 (.bz2)

Requerimientos

Este módulo usa las funciones de la biblioteca bzip2 (http://sources.redhat.com/bzip2/) de Julian Seward.

Instalación

El soporte Bzip2 en PHP no está activado por defecto. Necesitais usar la opción de configuración --with-bz2 cuando vayais a compilar PHP, si quereis soporte bzip2. Este módulo requiere bzip2/libbzip2 versión >= 1.0.x.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

Esta extensión define un tipo de recurso: un puntero de fichero que identifica el fichero bz2 con el que se va a trabajar.

Constantes predefinidas

Esta extensión no define ninguna constante.

Ejemplos

Este ejemplo abre un fichero temporal, escribe una cadena literal en el y presenta el contenido de dicho

fichero.

Ejemplo 1. Ejemplo simple de bzip2

```
<?php

$filename = "/tmp/testfile.bz2";
$str = "This is a test string.\n";

// open file for writing
$bz = bzopen($filename, "w");

// write string to file
bzwrite($bz, $str);

// close file
bzclose($bz);

// open file for reading
$bz = bzopen($filename, "r");

// read 10 characters
print bzread($bz, 10);

// output until end of the file (or the next 1024 char) and close it.
print bzread($bz);

bzclose($bz);

?>
```

bzclose (PHP 4 >= 4.0.4)

Close a bzip2 file pointer

int **bzclose** (resource bz) \linebreak

Closes the bzip2 file referenced by the pointer *bz*.

Devuelve TRUE si todo fue bien, FALSE en caso de fallo.

The file pointer must be valid, and must point to a file successfully opened by bzopen().

See also bzopen().

bzcompress (PHP 4 >= 4.0.4)

Compress a string into bzip2 encoded data

string **bzcompress** (string source [, int blocksize [, int workfactor]]) \linebreak

bzcompress() compresses the *source* string and returns it as bzip2 encoded data.

The optional parameter *blocksize* specifies the blocksize used during compression and should be a number from 1 to 9 with 9 giving the best compression, but using more resources to do so. *blocksize* defaults to 4.

The optional parameter *workfactor* controls how the compression phase behaves when presented with worst case, highly repetitive, input data. The value can be between 0 and 250 with 0 being a special case and 30 being the default value. Regardless of the *workfactor*, the generated output is the same.

Ejemplo 1. bzcompress() Example

```
<?php
$str = "sample data";
$bzstr = bzcompress($str, 9);
print( $bzstr );
?>
```

See also bzdecompress().

bzdecompress (PHP 4 >= 4.0.4)

Decompresses bzip2 encoded data

string **bzdecompress** (string source [, int small]) \linebreak

bzdecompress() decompresses the *source* string containing bzip2 encoded data and returns it. If the optional parameter *small* is TRUE, an alternative decompression algorithm will be used which uses less memory (the maximum memory requirement drops to around 2300K) but works at roughly half the speed. See the bzip2 documentation (http://sources.redhat.com/bzip2/) for more information about this feature.

Ejemplo 1. bzdecompress()

```
<?php
$start_str = "This is not an honest face?";
$bzstr = bzcompress($start_str);

print( "Compressed String: " );
print( $bzstr );
print( "\n<br>\n" );

$str = bzdecompress($bzstr);
print( "Decompressed String: " );
print( $str );
print( "\n<br>\n" );
?>
```

See also bzcompress().

bzerrno (PHP 4 >= 4.0.4)

Returns a bzip2 error number

int **bzerrno** (resource bz) \linebreak

Returns the error number of any bzip2 error returned by the file pointer *bz*.

See also bzerror() and bzerrstr().

bzerror (PHP 4 >= 4.0.4)

Returns the bzip2 error number and error string in an array

array **bzerror** (resource bz) \linebreak

Returns the error number and error string, in an associative array, of any bzip2 error returned by the file pointer *bz*.

Ejemplo 1. bzerror() Example

```
<?php
$error = bzerror($bz);

echo $error["errno"];
echo $error["errstr"];
?>
```

See also bzerrno() and bzerrstr().

bzerrstr (PHP 4 >= 4.0.4)

Returns a bzip2 error string

string **bzerrstr** (resource bz) \linebreak

Returns the error string of any bzip2 error returned by the file pointer *bz*.

See also bzerrno() and bzerror().

bzflush (PHP 4 >= 4.0.4)

Force a write of all buffered data

int **bzflush** (resource bz) \linebreak

Forces a write of all buffered bzip2 data for the file pointer *bz*.

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

See also bzread() and bzwrite().

bzopen (PHP 4 >= 4.0.4)

Open a bzip2 compressed file

resource **bzopen** (string filename, string mode) \linebreak

Opens a bzip2 (.bz2) file for reading or writing. *`filename`* is the name of the file to open. *`mode`* is similar to the fopen() function ('r' for read, 'w' for write, etc.).

If the open fails, the function returns `FALSE`, otherwise it returns a pointer to the newly opened file.

Ejemplo 1. bzopen() Example

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$decompressed_file = bzread($bz, filesize("/tmp/foo.bz2"));
bzclose($bz);

print( "The contents of /tmp/foo.bz2 are: " );
print( "\n<br>\n" );
print( $decompressed_file );
?>
```

See also bzclose().

bzread (PHP 4 >= 4.0.4)

Binary safe bzip2 file read

string **bzread** (resource bz [, int length]) \linebreak

bzread() reads up to *`length`* bytes from the bzip2 file pointer referenced by *`bz`*. Reading stops when *`length`* (uncompressed) bytes have been read or EOF is reached, whichever comes first. If the optional parameter *`length`* is not specified, **bzread()** will read 1024 (uncompressed) bytes at a time.

Ejemplo 1. bzread() Example

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$str = bzread($bz, 2048);
print( $str );
?>
```

See also bzwrite() and bzopen().

bzwrite (PHP 4 >= 4.0.4)

Binary safe bzip2 file write

int **bzwrite** (resource bz, string data [, int length]) \linebreak

bzwrite() writes the contents of the string *data* to the bzip2 file stream pointed to by *bz*. If the optional *length* argument is given, writing will stop after length (uncompressed) bytes have been written or the end of string is reached, whichever comes first.

Ejemplo 1. bzwrite() Example

```
<?php
$str = "uncompressed data";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
bzclose($bz);
?>
```

See also bzread() and bzopen().

VI. Funciones de calendario

Introducción

La extensión calendar pone a disposición una serie de funciones para simplificar la conversión entre los distintos formatos de calendario. El intermediario ó estándar en que se basa es la Cuenta de Días Juliana. La Cuenta de Días Juliana es una cuenta que comienza mucho antes que lo que mucha gente podría necesitar contar (como alrededor del 4000 AC). Para convertir entre sistemas de calendario, primero deberá convertir a la Cuenta de Días Juliana y luego al sistema de su elección. ¡La Cuenta de Días es muy diferente del Calendario Juliano! Para más informaci??? sobre la Cuenta de Días Juliana visitar http://serendipity.magnet.ch/hermetic/cal_stud/jdn.htm. Para más información sobre sistemas de calendario, visitar http://genealogy.org/~scottlee/cal-overview.html. En estas instrucciones se han incluído extractos entrecomillados de dicha página.

Instalación

Para que estas funciones funcionen, hay que compilar PHP con la opción `--enable-calendar`.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

Esta extensión no define ningún tipo de recurso.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión

ha sido o bien compilada dentro de PHP o grabada dinamicamente en tiempo de ejecución.

`CAL_GREGORIAN` (entero)

`CAL_JULIAN` (entero)

`CAL_JEWISH` (entero)

`CAL_FRENCH` (entero)

`CAL_NUM_CALS` (entero)

`CAL_DOW_DAYNO` (entero)

`CAL_DOW_SHORT` (entero)

`CAL_DOW_LONG` (entero)

`CAL_MONTH_GREGORIAN_SHORT` (entero)

`CAL_MONTH_GREGORIAN_LONG` (entero)

`CAL_MONTH_JULIAN_SHORT` (entero)

`CAL_MONTH_JULIAN_LONG` (entero)

`CAL_MONTH_JEWISH` (entero)

`CAL_MONTH_FRENCH` (entero)

Las siguientes constantes se pueden utilizar desde `PHP` 4.3.0 :

`CAL_EASTER_DEFAULT` (entero)

`CAL_EASTER_ROMAN` (entero)

`CAL_EASTER_ALWAYS_GREGORIAN` (entero)

cal_days_in_month (PHP 4 >= 4.1.0)

Devuelve el número de dias en un mes para un determinado año y calendario

int **cal_days_in_month** (int calendario, int mes, int año) \linebreak

Esta función devuelve el numero de dias en el *mes* del *año* para el calendario especificado *calendario*.

Ver también jdtounix().

cal_from_jd (PHP 4 >= 4.1.0)

Convierte de Cuenta de Días Juliana a un calendario soportado y devuelve información adicional.

array **cal_from_jd** (int jd, int calendario) \linebreak

cal_info (PHP 4 >= 4.1.0)

Devuelve información sobre un calendario den particular.

array **cal_info** (int calendario) \linebreak

cal_to_jd (PHP 4 >= 4.1.0)

Convierte de un calendario soportado a Cuenta de Días Juliana.

int **cal_to_jd** (int calendario, int mes, int dia, int año) \linebreak

easter_date (PHP 3>= 3.0.9, PHP 4)

devuelve la marca de tiempo UNIX para la medianoche de Pascua de un año dado

int **easter_date** ([int anno]) \linebreak

Devuelve la marca de tiempo UNIX que corresponde a la medianoche de Pascua del año dado.

A partir de PHP 4.3.0, el parametro *anno* es opcional y si se omite, usa por defecto el año en curso según "localtime".

Aviso: Esta función generará un aviso si el año está fuera del rango para las marcas de tiempo del UNIX (es decir, antes de 1970 o después del 2037).

Ejemplo 1. ejemplo de easter_date()

```
echo date ("M-d-Y", easter_date(1999));        /* "Apr-04-1999" */
echo date ("M-d-Y", easter_date(2000));        /* "Apr-23-2000" */
echo date ("M-d-Y", easter_date(2001));        /* "Apr-15-2001" */
```

La fecha del Día de Pascua fue definida por el Concilio de Nicea en el 325 D.C. como el domingo tras la primera luna llena que cayera en ó después del equinoccio de Primavera. El equinoccio se supone que siempre cae en el 21 de marzo, de modo que el cálculo se reduce a determinar la fecha de la luna llena y la del domingo siguiente. El algoritmo usado aquí fue introducido en el año 532 por Dionisio Exiguo. Bajo el Calendario Juliano (para años anteriores al 1753), se usa un ciclo simple de 19 años para calcular las fases de la luna. Bajo el Calendario Gregoriano (años posteriores al 1753, diseñado por Clavio y Lilio, e introducido por el Papa Gregorio XIII en Octubre de 1582, y en Gran Bretaña y sus colonias en septiembre de 1752) se añaden dos factores de corrección para hacer el ciclo más preciso.

(El código se basa en un programa en C de Simon Kershaw, <webmaster@ely.anglican.org>)

Ver easter_days() para calcular la Pascua antes del 1970 o después del 2037.

easter_days (PHP 3>= 3.0.9, PHP 4)

Obtiene el número de días tras el 21 de marzo en que cae la Pascua en un año dado

int **easter_days** ([int anno [, int metodo]]) \linebreak

Devuelve el número de días tras el 21 de marzo en que cae la Pascua en un año dado. Si no se especifica año, se asume el actual.

A partir de PHP 4.3.0, el parametro *anno* es opcional y si se omite, usa por defecto el año en curso según "localtime".

El parámetro *metodo* fue introducido en la version PHP 4.3.0 y permite calcular fechas de pascua basadas en el Calendario Gregoriano durante los años 1582 - 1752 si se le da el valor CAL_EASTER_ROMAN. Ver las constantes de calendario para más información sobre estas constantes.

Esta función se puede usar en lugar de easter_date() para calcular la Pascua para años que se salen del rango de las marcas de fecha del UNIX (o sea, antes del 1970 o después del 2037).

Ejemplo 1. ejemplo de easter_date()

```
echo easter_days (1999);        /* 14, i.e. April 4   */
echo easter_days (1492);        /* 32, i.e. April 22  */
echo easter_days (1913);        /*  2, i.e. March 23  */
```

La fecha del Día de Pascua fue definida por el Concilio de Nicea en el 325 D.C. como el domingo tras la primera luna llena que cayera en o después del equinoccio de Primavera. El equinoccio se supone que siempre cae en el 21 de marzo, de modo que el cálculo se reduce a determinar la fecha de la luna llena y la del domingo siguiente. El algoritmo usado aquí fue introducido en el año 532 por Dionisio Exiguo. Bajo el Calendario Juliano (para años anteriores al 1753), se usa un ciclo simple de 19 años para calcular las fases de la luna. Bajo el Calendario Gregoriano (años posteriores al 1753, diseñado por Clavio y Lilio, e introducido por el Papa Gregorio XIII en Octubre de 1582, y en Gran Bretaña y sus colonias en septiembre de 1752) se añaden dos factores de corrección para hacer el ciclo más preciso.

(El código se basa en un programa en C de Simon Kershaw, <webmaster@ely.anglican.org>)

Vea también easter_date().

FrenchToJD (PHP 3, PHP 4)

Convierte del Calendario Republicano Francés a la Cuenta de Días Juliana

int **frenchtojd** (int mes, int dia, int anno) \linebreak

Convierte una fecha del Calendario Republicano Francés a la Cuenta de Días Juliana.

Estas rutinas sólo convierten fechas entre los años 1 y 14 (fechas Gregorianas del 22 de septiembre de 1792 al 22 de septiembre de 1806). Esto cubre ampliamente el periodo en el que estuvo en uso este calendario.

GregorianToJD (PHP 3, PHP 4)

Convierte de fecha Gregoriana a la Cuenta de Días Juliana

int **gregoriantojd** (int mes, int dia, int anno) \linebreak

El rango válido para el Calendario Gregoriano es desde el 4714 A.C. hasta el 9999 D.C.

Aunque este programa puede manejar fechas tan lejanas como el 4714 A.C., usarlo no tendría sentido. El calendario Gregoriano fue instituído el 15 de octubre de 1582 (o el 5 de octubre de 1582 en el calendario Juliano). Algunos países no lo aceptaron hasta mucho después. Por ejemplo, Gran Bretaña se convirtió en 1752, la URSS en 1918 y Grecia en 1923. Muchos países europeos usaron el calendario Juliano antes que el Gregoriano.

Ejemplo 1. Funciones de calendario

```
<?php
$jd = GregorianToJD (10,11,1970);
echo "$jd\n";
```

```
$gregorian = JDToGregorian ($jd);
echo "$gregorian\n";
?>
```

JDDayOfWeek (PHP 3, PHP 4)

Devuelve el día de la semana

mixed **jddayofweek** (int diajuliano, int modo) \linebreak

Devuelve el día de la semana. Dependiendo del modo, devuelve un entero ó una cadena.

Tabla 1. Modos para el día de la semana

Modo	Significado
0	devuelve el día de la semana como entero (0=domingo, 1=lunes, etc)
1	devuelve una cadena con el día de la semana (inglés, gregoriano)
2	devuelve una cadena con el día de la semana abreviado (inglés, gregoriano)

JDMonthName (PHP 3, PHP 4)

Devuelve el nombre de un mes

string **jdmonthname** (int diajuliano, int modo) \linebreak

Devuelve una cadena que contiene el nombre del mes. *modo* le dice a esta función a qué calendario debe convertir la Cuenta de Días Juliana, y qué tipo de nombres de mes debe devolver.

Tabla 1. Modos de calendario

Modo	Significado	Valores
0	Gregoriano - abreviado	Jan, Feb, Mar, Apr
1	Gregoriano	January, February,
2	Juliano - abreviado	Jan, Feb, Mar, Apr
3	Juliano	January, February,
4	Judío	Tishri, Heshvan, K

Modo	Significado	Valores
5	Republicano Francés	Vendemiaire, Brur

JDToFrench (PHP 3, PHP 4)

Convierte de Cuenta de Días al Calendario Republicano Francés

string **jdtofrench** (int diajuliano) \linebreak

Convierte una Cuenta de Días Juliana al Calendario Republicano Francés.

JDToGregorian (PHP 3, PHP 4)

Convierte de Cuenta de Días a fecha Gregoriana

string **jdtogregorian** (int diajuliano) \linebreak

Convierte de Cuenta de Días Juliana a una cadena que contiene la fecha Gregoriana en formato "mes/día/año"

JDToJewish (PHP 3, PHP 4)

Convierte de Cuenta de Días Juliana a Calendario Judío

string **jdtojewish** (int diajuliano) \linebreak

Convierte una Cuenta de Días Juliana al Calendario Judío.

JDToJulian (PHP 3, PHP 4)

Convierte de Cuenta de Días Juliana a Calendario Juliano

string **jdtojulian** (int diajuliano) \linebreak

Convierte una Cuenta de Días Juliana a una cadena que contiene la fecha del Calendario Juliano en formato "mes/día/año".

jdtounix (PHP 4)

Convierte un dia Juliano a UNIX timestamp

int **jdtounix** (int jday) \linebreak

Esta funció devuelve el "UNIX timestamp" correspondiante a el dia Juliano definido en *jday* ó falso (FALSE) si *jday* no se encuentra en la ??? UNIX (años entre 1970 y 2037 ó 2440588 <= *jday* <= 2465342). El tiempo devuelto es localtime (y no GMT).

Ver también unixtojd().

JewishToJD (PHP 3, PHP 4)

Convierte del Calendario Judío a la Cuenta de Días Juliana

int **jewishtojd** (int mes, int dia, int anno) \linebreak

Aunque este programa puede manejar fechas tan lejanas como el año 1 (3761 A.C.), usarlo no tendría sentido. El Calendario Judío ha estado en uso miles de años, pero en los días primeros no había una fórmula que calculara el comienzo de un mes. Un mes comenzaba cuando se veía por primera vez la luna nueva.

JulianToJD (PHP 3, PHP 4)

Convierte de Calendario Juliano a Cuenta de Días Juliana

int **juliantojd** (int mes, int dia, int anno) \linebreak

Rango válido para el Calendario Juliano: del 4713 A.C al 9999 D.C.

Aunque este programa puede manejar fechas tan lejanas como el 4713 A.C., usarlo no tendría sentido. El calendario se creó en el 46 A.C., pero sus detalles no se estabilizaron hasta al menos el 8 D.C., y quizás no lo hiciera hasta el siglo IV. Además, el comienzo de un año variaba de una a otra cultura: no todas aceptaban enero como el primer mes.

Atención

Recordar que el actual sistema de calendario en uso en todo el mundo es el calendario Gregoriano. gregoriantojd() puede ser usada para convertir los dias del calendario Gregoriano a Cuenta de Días Juliana.

unixtojd (PHP 4)

Convierte de UNIX timestamp a dia Juliano

int **unixtojd** ([int timestamp]) \linebreak

Devuelve el dia Juliano correspondiente a un UNIX `timestamp` (segundos desde 01.01.1970), ó al dia actual si no se especifica `timestamp`

Ver tambienjdtounix().

VII. Funciones del API de CCVS

Introducción

Estas funciones interaccionan con el API de CCVS, permitiendo trabajar con CCVS directamente desde un script PHP. CCVS es la solución de RedHat (http://www.redhat.com/) para el intermediario en el procesamiento de tarjetas de crédito. Permite conectar directamente con las centrales de las tarjetas desde una máquina *nix con un módem.

> **Nota:** CCVS ha sido discontinuado por Red Hat y no existen planes de ofrecer nuevas funcionalidades ó contratos de ayuda. Los que necesiten usar esta funcionalidad pueden probar MCVE by Main Street Softworks (http://www.mcve.com/). Es similar en diseño y tiene documentación para su uso con PHP

Instalación

Para activar el soporte de CCVS en PHP hay que tener instalado CCVS en vuestro sistema. Seguidamente es necesario configurar PHP con la opción `--with-ccvs`. Si se usa esta opcion sin especificar el directorio donde CCVS está instalado, PHP intentará encontrar CCVS en la localización por defecto (/usr/local/ccvs). Si CCVS está instalado en una localización no estándar, ejecutar configure con: `--with-ccvs=$ccvs_path`, donde $ccvs_path es el directorio donde CCVS esta instalado. Tener en cuenta que el soporte de CCVS en PHP necesita que $ccvs_path/lib y $ccvs_path/include existan, que cv_api.h se encuentre en el directorio include y que libccvs.a se encuentre en el directorio lib.

Adicionalmente se necesita un proceso ccvs ejecutandose en el sistema para las configuraciones que se ejecuten desde PHP. Los procesos PHP deben ejecutarse bajo el mismo usuario que use CCVS (p.ej. Si ccvs usa el usuario 'ccvs', PHP debe ejecutarse como 'ccvs' tambien).

Ver tambien

Información adicional sobre CCVS se puede encontrar en http://www.redhat.com/products/ccvs. Red Hat casi no mantiene la documentación de CCVS, pero todavia es de gran ayuda, se puede encontrar en http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html (http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html).

ccvs_add (PHP 4 >= 4.0.2)

Añadir datos a una transacción

cadena **ccvs_add** (cadena sesión, cadena factura, cadena argtype, cadena argval) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_auth (PHP 4 >= 4.0.2)

Realiza un test de una autorización a crédito en una transacción

cadena **ccvs_auth** (cadena sesión, cadena factura) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_command (PHP 4 >= 4.0.2)

Ejecuta un comando que es peculiar para un protocolo concreto, y que no está disponible en el API general de CCVS

cadena **ccvs_command** (cadena sesión, cadena tipo, cadena argval) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_count (PHP 4 >= 4.0.2)

Encuentra cuantas transacciones de un tipo dado están almacenadas en el sistema

entero **ccvs_count** (cadena sesión, cadena tipo) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_delete (PHP 4 >= 4.0.2)

Borra una transacción

cadena **ccvs_delete** (cadena sesi???, cadena factura) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_done (PHP 4 >= 4.0.2)

Finaliza el motor de CCVS y hace una limpieza

cadena **ccvs_done** (cadena sesió) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_init (PHP 4 >= 4.0.2)

Inicializa un CCVS para usarlo

cadena **ccvs_init** (cadena nombre) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_lookup (PHP 4 >= 4.0.2)

Busca un item de un tipo en particular en la base de datos #

cadena **ccvs_lookup** (cadena sesión, cadena factura, entero inum) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_new (PHP 4 >= 4.0.2)

Crea una nueva, transacción en blanco

cadena **ccvs_new** (cadena sesión, cadena cadena) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_report (PHP 4 >= 4.0.2)

Devuelve el estado del proceso de comunicación en background

cadena **ccvs_report** (cadena sesión, cadena tipo) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_return (PHP 4 >= 4.0.2)

Transfiere fondos del comerciante al titular de la tarjeta

cadena **ccvs_return** (cadena sesión, cadena factura) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_reverse (PHP 4 >= 4.0.2)

Realiza una revocación completa en una autorización ya procesada

cadena **ccvs_reverse** (cadena sesión, cadena factura) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_sale (PHP 4 >= 4.0.2)

Transfiere fondos del titular de la tarjeta al comerciante

cadena **ccvs_sale** (cadena sesión, cadena factura) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_status (PHP 4 >= 4.0.2)

Chequear el estado de una factura

cadena **ccvs_status** (cadena sesión, cadena factura) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_textvalue (PHP 4 >= 4.0.2)

Obtiene el valor de retorno de texto para una llamada anterior a una función

cadena **ccvs_textvalue** (cadena sesión) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ccvs_void (PHP 4 >= 4.0.2)

Realizar una revocación completa en una transacción completada

cadena **ccvs_void** (cadena sesión, cadena factura) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

VIII. soporte de las funciones COM para Windows

Estas funciones solo están disponibles en la versión para Windows de PHP. Estas funciones han sido añadidas en PHP4.

COM (unknown)

COM class

Synopsis

```
$obj = new COM("server.object")
```

The COM class provides a framework to integrate (D)COM components into your php scripts.

string **COM::COM** (string module_name [, string server_name [, int codepage]]) \linebreak

COM class constructor. Parameters:

module_name

name or class-id of the requested component.

server_name

name of the DCOM server from which the component should be fetched. If `NULL`, `localhost` is assumed. To allow DCOM `com.allow_dcom` has to be set to `TRUE` in `php.ini`.

codepage

specifies the codepage that is used to convert php-strings to unicode-strings and vice versa. Possible values are `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7` and `CP_UTF8`.

Ejemplo 1. COM example (1)

```
// starting word
$word = new COM("word.application") or die("Unable to instanciate Word");
print "Loaded Word, version {$word->Version}\n";

//bring it to front
$word->Visible = 1;

//open an empty document
$word->Documents->Add();

//do some weird stuff
$word->Selection->TypeText("This is a test...");
$word->Documents[1]->SaveAs("Useless test.doc");

//closing word
$word->Quit();
```

```
//free the object
$word->Release();
$word = null;
```

Ejemplo 2. COM example (2)

```
$conn = new COM("ADODB.Connection") or die("Cannot start ADO");
$conn->Open("Provider=SQLOLEDB; Data Source=localhost;
Initial Catalog=database; User ID=user; Password=password");

$rs = $conn->Execute("SELECT * FROM sometable");    // Recordset

$num_columns = $rs->Fields->Count();
echo $num_columns . "\n";

for ($i=0; $i < $num_columns; $i++)
{
    $fld[$i] = $rs->Fields($i);
}

$rowcount = 0;
while (!$rs->EOF)
{
    for ($i=0; $i < $num_columns; $i++)
    {
        echo $fld[$i]->value . "\t";
    }
    echo "\n";
    $rowcount++;            // increments rowcount
    $rs->MoveNext();
}

$rs->Close();
$conn->Close();

$rs->Release();
$conn->Release();

$rs = null;
$conn = null;
```

VARIANT (unknown)

VARIANT class

Synopsis

```
$vVar = new VARIANT($var)
```

A simple container to wrap variables into VARIANT structures.

string **VARIANT::VARIANT** ([mixed value [, int type [, int codepage]]]) \linebreak

VARIANT class constructor. Parameters:

value

initial value. if omitted an VT_EMPTY object is created.

type

specifies the content type of the VARIANT object. Possible values are `VT_UI1`, `VT_UI2`, `VT_UI4`, `VT_I1`, `VT_I2`, `VT_I4`, `VT_R4`, `VT_R8`, `VT_INT`, `VT_UINT`, `VT_BOOL`, `VT_ERROR`, `VT_CY`, `VT_DATE`, `VT_BSTR`, `VT_DECIMAL`, `VT_UNKNOWN`, `VT_DISPATCH` and `VT_VARIANT`. These values are mutual exclusive, but they can be combined with `VT_BYREF` to specify being a value. If omitted, the type of *`value`* is used. Consult the msdn library for additional information.

codepage

specifies the codepage that is used to convert php-strings to unicode-strings and vice versa. Possible values are `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7` and `CP_UTF8`.

com_addref (PHP 4 >= 4.1.0)

Increases the components reference counter.

void **com_addref** (void) \linebreak

Increases the components reference counter.

com_get (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

???

mixed **com_get** (resource object, string property) \linebreak

com_invoke (PHP 3>= 3.0.3)

???

mixed **com_invoke** (resource object, string function_name [, mixed function parameters, ...]) \linebreak

com_isenum (PHP 4 >= 4.1.0)

Grabs an IEnumVariant

void **com_isenum** (object com_module) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

com_load_typelib (PHP 4 >= 4.1.0)

Loads a Typelib

void **com_load_typelib** (string typelib_name [, int case_insensitive]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

com_load (PHP 3>= 3.0.3)

???

string **com_load** (string module name [, string server name]) \linebreak

com_propget (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

???

mixed **com_propget** (resource object, string property) \linebreak

com_propput (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

???

void **com_propput** (resource object, string property, mixed value) \linebreak

com_propset (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

???

void **com_propset** (resource object, string property, mixed value) \linebreak

Esta función es un alias para com_propput().

com_release (PHP 4 >= 4.1.0)

Decreases the components reference counter.

void **com_release** (void) \linebreak

Decreases the components reference counter.

com_set (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

???

void **com_set** (resource object, string property, mixed value) \linebreak

Esta función es un alias para **com_set()**.

IX. Funciones de Clases/Objectos

Introducción

Estas funciones permiten obtener informacion sobre clases y objetos. Se puede obtener el nombre de la clase a la que pertenece un objeto, asi como las propiedades de sus miembros y métodos. Usando estas funciones se puede obtener no solo lo comentado en la frase anterior, tambien se puede obtener la familia del objeto (p.ej. que clase está extendiendo la clase a la que pertenece el objeto)

Ejemplos

En este ejemplo, definimos primero una clase base y una extensión de esta clase. La clase base define un vegetal genérico, si es comestible y su color. La subclase `Spinach` añade un metodo para cocinarlo y otro para saber si esta cocinado.

Ejemplo 1. classes.inc

```
<?php

// base class with member properties and methods
class Vegetable {

    var $edible;
    var $color;

    function Vegetable( $edible, $color="green" ) {
        $this->edible = $edible;
        $this->color = $color;
    }

    function is_edible() {
        return $this->edible;
    }

    function what_color() {
        return $this->color;
    }

} // end of class Vegetable

// extends the base class
class Spinach extends Vegetable {

    var $cooked = false;

    function Spinach() {
```

```
        $this->Vegetable( true, "green" );
    }

    function cook_it() {
        $this->cooked = true;
    }

    function is_cooked() {
        return $this->cooked;
    }

} // end of class Spinach

?>
```

Creamos 2 objetos de estas clases e imprimimos información sobre ellos, incluyendo la jerarquia de clases a la que pertenecen. También definimos algunas funciones, especialmente para imprimir las variables de una manera ordenada.

Ejemplo 2. test_script.php

```
<pre>
<?php

include "classes.inc";

// utility functions

function print_vars($obj) {
    $arr = get_object_vars($obj);
    while (list($prop, $val) = each($arr))
        echo "\t$prop = $val\n";
}

function print_methods($obj) {
    $arr = get_class_methods(get_class($obj));
    foreach ($arr as $method)
        echo "\tfunction $method()\n";
}

function class_parentage($obj, $class) {
    global $$obj;
    if (is_subclass_of($$obj, $class)) {
        echo "Object $obj belongs to class ".get_class($$obj);
        echo " a subclass of $class\n";
    } else {
        echo "Object $obj does not belong to a subclass of $class\n";
    }
```

```
}

// instantiate 2 objects

$veggie = new Vegetable(true,"blue");
$leafy = new Spinach();

// print out information about objects
echo "veggie: CLASS ".get_class($veggie)."\n";
echo "leafy: CLASS ".get_class($leafy);
echo ", PARENT ".get_parent_class($leafy)."\n";

// show veggie properties
echo "\nveggie: Properties\n";
print_vars($veggie);

// and leafy methods
echo "\nleafy: Methods\n";
print_methods($leafy);

echo "\nParentage:\n";
class_parentage("leafy", "Spinach");
class_parentage("leafy", "Vegetable");
?>
</pre>
```

One important thing to note in the example above is that the object `$leafy` is an instance of the class `Spinach` which is a subclass of `Vegetable`, therefore the last part of the script above will output:

```
       [...]
Parentage:
Object leafy does not belong to a subclass of Spinach
Object leafy belongs to class spinach a subclass of Vegetable
```

call_user_method_array (PHP 4 >= 4.0.5)

Call a user method given with an array of parameters [deprecated]

mixed **call_user_method_array** (string method_name, object obj [, array paramarr]) \linebreak

> **Aviso**
>
> The **call_user_method_array()** function is deprecated as of PHP 4.1.0, use the call_user_func_array() variety with the `array(&$obj, "method_name")` syntax instead.

Calls the method referred by *`method_name`* from the user defined *`obj`* object, using the parameters in *`paramarr`*.

See also: call_user_func_array(), call_user_func(), call_user_method().

Nota: This function was added to the CVS code after release of PHP 4.0.4pl1

call_user_method (PHP 3>= 3.0.3, PHP 4)

Call a user method on an specific object [deprecated]

mixed **call_user_method** (string method_name, object obj [, mixed parameter [, mixed ...]]) \linebreak

> **Aviso**
>
> The **call_user_method()** function is deprecated as of PHP 4.1.0, use the call_user_func() variety with the `array(&$obj, "method_name")` syntax instead.

Calls the method referred by *`method_name`* from the user defined *`obj`* object. An example of usage is below, where we define a class, instantiate an object and use **call_user_method()** to call indirectly its `print_info` method.

```
<?php
class Country {
    var $NAME;
    var $TLD;

    function Country($name, $tld) {
        $this->NAME = $name;
        $this->TLD = $tld;
    }
```

```
    function print_info($prestr="") {
        echo $prestr."Country: ".$this->NAME."\n";
        echo $prestr."Top Level Domain: ".$this->TLD."\n";
    }
}

$cntry = new Country("Peru","pe");

echo "* Calling the object method directly\n";
$cntry->print_info();

echo "\n* Calling the same method indirectly\n";
call_user_method ("print_info", $cntry, "\t");
?>
```

See also call_user_func_array(), call_user_func(), and call_user_method_array().

class_exists (PHP 4)

Checks if the class has been defined

bool **class_exists** (string class_name) \linebreak

This function returns `TRUE` if the class given by *`class_name`* has been defined, `FALSE` otherwise.

get_class_methods (PHP 4)

Devuelve un vector (matriz unidimensional) con los nombres de los métodos de la clase en question.

vector **get_class_methods** (string class_name) \linebreak

Esta función devuelve un vector con los nombres de los métodos definidos en la clase especificada como *`class_name`*.

Nota: A partir de PHP 4.0.6, se puede especificar el objeto a sí mismo en vez de *`class_name`*. Por ejemplo:

```
$class_methods = get_class_methods($my_class); // see below the full example
```

Ejemplo 1. get_class_methods() ejemplo

```
<?php

class myclass {
    // constructor
    function myclass() {
        return(TRUE);
    }

    // method 1
    function myfunc1() {
        return(TRUE);
    }

    // method 2
    function myfunc2() {
        return(TRUE);
    }
}

$my_object = new myclass();

$class_methods = get_class_methods(get_class($my_object));

foreach ($class_methods as $method_name) {
    echo "$method_name\n";
}

?>
```

Producira:

```
myclass
myfunc1
myfunc2
```

Ver también get_class_vars() y get_object_vars().

get_class_vars (PHP 4)

Devuelve un vector con las propiedades (inicializadas por defecto) de la clase

array **get_class_vars** (string class_name) \linebreak

Esta función devuelve un vector con las propiedades que han sido inicializadas por defecto en la clase. Los elementos de este vector están estan organizados de la forma *varname => value*.

> **Nota:** Las variables de la clase que no estén inicializadas, no será presentadas por **get_class_vars()**.

Ejemplo 1. get_class_vars() ejemplo

```
<?php

class myclass {

    var $var1; // this has no default value...
    var $var2 = "xyz";
    var $var3 = 100;

    // constructor
    function myclass() {
        return(TRUE);
    }
}

$my_class = new myclass();

$class_vars = get_class_vars(get_class($my_class));

foreach ($class_vars as $name => $value) {
    echo "$name : $value\n";
}

?>
```

Producira:

```
var2 : xyz
var3 : 100
```

Ver también get_class_methods(), get_object_vars()

get_class (PHP 4)

Returns the name of the class of an object

string **get_class** (object obj) \linebreak

This function returns the name of the class of which the object *obj* is an instance. Returns FALSE if *obj* is not an object.

> **Nota: get_class()** returns a user defined class name in lowercase. A class defined in a PHP extension is returned in its original notation.

See also get_parent_class(), gettype(), and is_subclass_of().

get_declared_classes (PHP 4)

Returns an array with the name of the defined classes

array **get_declared_classes** (void) \linebreak

This function returns an array of the names of the declared classes in the current script.

> **Nota:** In PHP 4.0.1pl2, three extra classes are returned at the beginning of the array: `stdClass` (defined in `Zend/zend.c`), `OverloadedTestClass` (defined in `ext/standard/basic_functions.c`) and `Directory` (defined in `ext/standard/dir.c`).
>
> Also note that depending on what libraries you have compiled into PHP, additional classes could be present. This means that you will not be able to define your own classes using these names. There is a list of predefined classes in the Predefined Classes section of the appendices.

get_object_vars (PHP 4)

Devuelve un vector de propiedades del objecto

array **get_class_vars** (object obj) \linebreak

Esta función devuelve un vector con las propiedades definidas en el objecto especificado como *obj*. Las variables declaradas en la clase a la que pertenece *obj*, que no les ha sido asignado un valor, no serán devueltas en el vector.

Ejemplo 1. Uso de get_object_vars()

```
<?php
class Point2D {
    var $x, $y;
    var $label;

    function Point2D($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }

    function setLabel($label) {
        $this->label = $label;
    }

    function getPoint() {
        return array("x" => $this->x,
                     "y" => $this->y,
                     "label" => $this->label);
    }
}

// "$label" is declared but not defined
$p1 = new Point2D(1.233, 3.445);
print_r(get_object_vars($p1));

$p1->setLabel("point #1");
print_r(get_object_vars($p1));

?>
```

El resultado de este programa es:

```
 Array
 (
     [x] => 1.233
     [y] => 3.445
 )

 Array
 (
     [x] => 1.233
     [y] => 3.445
     [label] => point #1
```

)

Ver tambien get_class_methods() y get_class_vars()!

get_parent_class (PHP 4)

Retrieves the parent class name for object or class

string **get_parent_class** (mixed obj) \linebreak

If *obj* is an object, returns the name of the parent class of the class of which *obj* is an instance.

If *obj* is a string, returns the name of the parent class of the class with that name. This functionality was added in PHP 4.0.5.

See also get_class() and is_subclass_of()

is_a (PHP 4 >= 4.2.0)

Returns TRUE if the object is of this class or has this class as one of its parents

bool **is_a** (object object, string class_name) \linebreak

This function returns TRUE if the object is of this class or has this class as one of its parents, FALSE otherwise.

See also get_class(), get_parent_class(), and is_subclass_of().

is_subclass_of (PHP 4)

Returns TRUE if the object has this class as one of its parents

bool **is_subclass_of** (object object, string class_name) \linebreak

This function returns TRUE if the object *object*, belongs to a class which is a subclass of *class_name*, FALSE otherwise.

See also get_class(), get_parent_class() and is_a().

method_exists (PHP 4)

Comprueba que el metódo de clase existe

bool **method_exists** (object object, string method_name) \linebreak

Esta función devuelve verdadero (`TRUE`) si el metódo referido por *`method_name`* ha sido definido en el objecto *`object`*, en cualquier otro caso devuelve falso (`FALSE`)

X. Funciones de ClibPDF

ClibPDF Le permite crear documentos PDF con PHP. Está disponible en FastIO (http://www.fastio.com) pero no es software libre. Debería leer la licencia antes de comenzar a utilizar ClibPDF. Si usted no puede cumplir el acuerdo de la licencia considere el utilizar la pdflib de Thomas Merz, que tambien es muy potente. La funcionalidad y la API de ClibPDF son similares a la pdflib de Thomas Merz pero, de acuerdo con FastIO, ClibPDF es mas rápida y crea documentos mas pequeños. Esto puede haber cambiado con la nueva versión 2.0 de pdflib. Un simple banco de pruebas (el ejemplo pdfclock.c de pdflib 2.0 trasformado en un script php) en realidad no muestra ninguna diferencia en velocidad. Por tanto, pruebe las dos y vea cual hace el mejor trabajo para usted.

Esta documentación debería ser leída junto con el manual de ClibPDF ya que este explica la librería con mucho mas detalle.

Muchas funciones en le ClibPDF nativa y el módulo PHP, así como en pdflib, tienen el mismo nombre. Todas las funciones excepto cpdf_open() toman el manejador del documento com el primer parámetro. Actualmente este manejador no se usa internamente desde que ClibPDF no soporta la creación de varios documentos PDF al mismo tiempo. Realmente, ni debería intentarlo, los resultados son impredecibles. No puedo supervisar cuales son las consecuencias en un sistema multihilo. De acuerdo con el autor de ClibPDF, esto cambiará en alguno de las próximas veriones (la versión actual, cuando eto fue escrito es 1.10). Si usted necesita esta capacidad, use el módulo pdflib.

> **Nota:** La función cpdf_set_font() ha cambiado desde que PHP3 soporta fuentes asiáticas. El parámetro que codifica ya no es un entero sino una cadena.

Una gran ventaja de ClibPDF sobre pdflib es la posibilidad de crear el documento PDF completamente en memoria sin usar ficheros temporales. Esto también proporciona la capaciad de pasar coordenadas en una unidad de longitud predefinida. Esta es una cualidad útil pero puede ser simulada con pdf_translate().

La mayoría de las funciones son fáciles de usar. La parte mas difícil es, probablemente, crear un documento PDF muy simple. El siguiente ejemplo debería ayudarle a comenzar. En él se crea un documento con una página. La página contiene el texto "Times-Roman" con una fuente de 30pt. El texto está subrayado.

Ejemplo 1. Ejemplo simple de ClibPDF

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_moveto($cpdf, 50, 740);
cpdf_lineto($cpdf, 330, 740);
cpdf_stroke($cpdf);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
```

```
?>
```

La distribución de pdflib contiene un ejemplo mas comlejo que crea una serie de páginas con un reloj analógico. Aquí está ese ejemplo convertido en PHP usando la extensión ClibPDF:

Ejemplo 2. Ejemplo con pdfclock de la distribución pdflib 2.0

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 40;

$pdf = cpdf_open(0);
cpdf_set_creator($pdf, "pdf_clock.php3");
cpdf_set_title($pdf, "Reloj Analógico");

while($pagecount-- > 0) {
  cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius + $mar-
gin), 1.0);

  cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* limpiar */

  cpdf_translate($pdf, $radius + $margin, $radius + $margin);
  cpdf_save($pdf);
  cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

  /*  cambio de minuto */
  cpdf_setlinewidth($pdf, 2.0);
  for ($alpha = 0; $alpha < 360; $alpha += 6)
    {
    cpdf_rotate($pdf, 6.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin/3, 0.0);
    cpdf_stroke($pdf);
    }

  cpdf_restore($pdf);
  cpdf_save($pdf);

  /* cambios de 5 minutos */
  cpdf_setlinewidth($pdf, 3.0);
  for ($alpha = 0; $alpha < 360; $alpha += 30)
  {
    cpdf_rotate($pdf, 30.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin, 0.0);
    cpdf_stroke($pdf);
  }

  $ltime = getdate();

  /* dibujar la aguja de las horas */
```

```
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['minutos']/60.0) + $ltime['horas'] - 3.0) * 30.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius/2, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* dibujar el minutero */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['segundos']/60.0) + $ltime['minutos'] - 15.0) * 6.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius * 0.8, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* dibujar la seguna mano */
  cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
  cpdf_setlinewidth($pdf, 2);
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['segundos'] - 15.0) * 6.0));
  cpdf_moveto($pdf, -$radius/5, 0.0);
  cpdf_lineto($pdf, $radius, 0.0);
  cpdf_stroke($pdf);
  cpdf_restore($pdf);

  /* dibujar un pequeño círculo en el centro */
  cpdf_circle($pdf, 0, 0, $radius/30);
  cpdf_fill($pdf);

  cpdf_restore($pdf);

  cpdf_finalize_page($pdf, $pagecount+1);
}

cpdf_finalize($pdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($pdf);
cpdf_close($pdf);
?>
```

cpdf_add_annotation (PHP 3>= 3.0.12, PHP 4)

Añade una anotación

void **cpdf_add_annotation** (int pdf document, double llx, double lly, double urx, double ury, string title, string content, int mode) \linebreak

La función **cpdf_add_annotation()** añade una nota con la esquina inferior izquierda en (*llx*, *lly*) y la esquina superior derecha en (*urx*, *ury*).

El últomo parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

cpdf_add_outline (PHP 3>= 3.0.9, PHP 4)

Añade una marca en la página actual

void **cpdf_add_outline** (int pdf document, string text) \linebreak

La función **cpdf_add_outline()** añade una marca con el texto *text* que apunta a la página actual.

Ejemplo 1. Añadiendo un contorno de página

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Página 1");
// ...
// Algún dibujo
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

cpdf_arc (PHP 3>= 3.0.8, PHP 4)

Dibuja un arco

void **cpdf_arc** (int pdf document, double x-koor, double y-koor, double radius, double start, double end, int mode) \linebreak

La función **cpdf_arc()** dibuja un arco con el centro rn el punto (*x-koor*, *y-koor*) y radio *radius*, empezando en el ángulo *start* y terminando en el ángulo *end*.

El último parámetro opcional especifica el tamaño de la unidad. Si es 0 o se omite, se usa la unidad especificada por defecto. De otro modo las coordenadas son medidas en puntos postscript,despreciando la unidad actual.

Vea también cpdf_circle().

cpdf_begin_text (PHP 3>= 3.0.8, PHP 4)

Inicializa una sección de texto

void **cpdf_begin_text** (int pdf document) \linebreak

La función **cpdf_begin_text()** comienza una sección de texto. Debe ser terminada con cpdf_end_text().

Ejemplo 1. Salida de texto

```
<?php cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Algún texto");
cpdf_end_text($pdf) ?>
```

Vea también cpdf_end_text().

cpdf_circle (PHP 3>= 3.0.8, PHP 4)

Dibuja un círculo

void **cpdf_circle** (int pdf document, double x-koor, double y-koor, double radius, int mode) \linebreak

La función **cpdf_circle()** dibuja un círculo con centro en el punto (*x-koor*, *y-koor*) y radio *radius*.

El último parámetro opcional define el tamaño de la unidad. Si es 0 o se omite, se usa el valor por defecto para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también cpdf_arc().

cpdf_clip (PHP 3>= 3.0.8, PHP 4)

Ajusta al camino actual

void **cpdf_clip** (int pdf document) \linebreak

La función **cpdf_clip()** ajusta todos los dibujos al camino actual.

cpdf_close (PHP 3>= 3.0.8, PHP 4)

Cierra un documento PDF

void **cpdf_close** (int pdf document) \linebreak

La función **cpdf_close()** cierra un documento PDF. Esta debería ser la última operación incluso después de cpdf_finalize(), cpdf_output_buffer() y cpdf_save_to_file().

Vea también cpdf_open().

cpdf_closepath_fill_stroke (PHP 3>= 3.0.8, PHP 4)

Cierra, llena y traza el camino actual

void **cpdf_closepath_fill_stroke** (int pdf document) \linebreak

La función **cpdf_closepath_fill_stroke()** cierra, llena el interior del caminoa catual con el color actual de relleno y dibuja el camino actual.

Vea también cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_closepath_stroke (PHP 3>= 3.0.8, PHP 4)

Cierra el camino y dibuja una línea a lo largo del camino

void **cpdf_closepath_stroke** (int pdf document) \linebreak

La función **cpdf_closepath_stroke()** es una combinación de cpdf_closepath() y cpdf_stroke(). Después limpia el camino.

Vea también cpdf_closepath(), cpdf_stroke().

cpdf_closepath (PHP 3>= 3.0.8, PHP 4)

Cierra el camino

void **cpdf_closepath** (int pdf document) \linebreak

La función **cpdf_closepath()** cierra el camino actual.

cpdf_continue_text (PHP 3>= 3.0.8, PHP 4)

Pone texto en la línea siguiente

void **cpdf_continue_text** (int pdf document, string text) \linebreak

La función **cpdf_continue_text()** pone la cadena *text* en la línea siguiente.

Vea también cpdf_show_xy(), cpdf_text(), cpdf_set_leading(), cpdf_set_text_pos().

cpdf_curveto (PHP 3>= 3.0.8, PHP 4)

Dibuja una curva

void **cpdf_curveto** (int pdf document, double x1, double y1, double x2, double y2, double x3, double y3, int mode) \linebreak

La función **cpdf_curveto()** dibuja una curva Bezier desde el punto actual al punto (*x3*, *y3*) usando (*x1*, *y1*) y (*x2*, *y2*) como puntos de control.

El último parámetro opcional especifica el tamaño de la unidad. Si es 0 o se omite, se usa la unidad especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad en curso.

Vea también cpdf_moveto(), cpdf_rmoveto(), cpdf_rlineto(), cpdf_lineto().

cpdf_end_text (PHP 3>= 3.0.8, PHP 4)

Finaliza una sección de texto

void **cpdf_end_text** (int pdf document) \linebreak

La función **cpdf_end_text()** finaliza unasección de texto que fue inicializada con cpdf_begin_text().

Ejemplo 1. Salida de texto

```
<?php cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Algún texto");
cpdf_end_text($pdf) ?>
```

Vea también cpdf_begin_text().

cpdf_fill_stroke (PHP 3>= 3.0.8, PHP 4)

LLena y traza el camino actual

void **cpdf_fill_stroke** (int pdf document) \linebreak

La función **cpdf_fill_stroke()** llena el interior del camino actual con el color de relleno actual y dibuja el camino actual.

Vea también cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_fill (PHP 3>= 3.0.8, PHP 4)

LLena el camino actual

void **cpdf_fill** (int pdf document) \linebreak

La función **cpdf_fill()** llena el interior del camino actual con el color alctual de relleno.

Vea también cpdf_closepath(), cpdf_stroke(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_finalize_page (PHP 3>= 3.0.10, PHP 4)

Finaliza una página

void **cpdf_finalize_page** (int pdf document, int page number) \linebreak

La función **cpdf_finalize_page()** finaliza una página con número de página *`page number`*. Esta función es sólo para ahorrar memoria. Una página terminada ocupa menos memoria pero no puede volver a ser modificada.

Vea también cpdf_page_init().

cpdf_finalize (PHP 3>= 3.0.8, PHP 4)

Finaliza un documento

void **cpdf_finalize** (int pdf document) \linebreak

La función **cpdf_finalize()** finaliza un documento. Aún se tiene que llamar a cpdf_close().

Vea también cpdf_close().

cpdf_global_set_document_limits (PHP 4)

Sets document limits for any pdf document

void **cpdf_global_set_document_limits** (int maxpages, int maxfonts, int maximages, int maxannotations, int maxobjects) \linebreak

La función **cpdf_global_set_document_limits()** define varios límites del documento. Esta función debe ser llamada antes de cpdf_open() para que haga efecto. Ello define los límites de cualquier documento abierto con anterioridad.

Vea también cpdf_open().

cpdf_import_jpeg (PHP 3>= 3.0.9, PHP 4)

Abre una imagen JPEG

int **cpdf_open_jpeg** (int pdf document, string file name, double x-koor, double y-koor, double angle, double width, double height, double x-scale, double y-scale, int mode) \linebreak

La función **cpdf_import_jpeg()** abre una imagen almacenada en el fichero de nombre *file name*. El formato de la imagen debe ser JPEG. La imagen es situada en la página actual en la posición (*x-koor*, *y-koor*). La imagen es rotada *angle* grados.

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también cpdf_place_inline_image(),

cpdf_lineto (PHP 3>= 3.0.8, PHP 4)

Dibuja una línea

void **cpdf_lineto** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

La función **cpdf_lineto()** dibuja una línea desde el punto actual al punto con coordenadas (*x-koor*, *y-koor*).

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa el valor especificado para la página por defecto. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto().

cpdf_moveto (PHP 3>= 3.0.8, PHP 4)

Define el punto actual

void **cpdf_moveto** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

La función**cpdf_moveto()** pone el punto actual en las coordenadas *x-koor* y *y-koor*.

El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, la unidad por defecto será la especificada para la página. De otro modo las coordenadas se medirán en puntos postscript despreciando la unidad en curso.

cpdf_newpath (PHP 3>= 3.0.9, PHP 4)

Starts a new path

void **cpdf_newpath** (int pdf document) \linebreak

The **cpdf_newpath()** starts a new path on the document given by the *pdf document* parameter.

cpdf_open (PHP 3>= 3.0.8, PHP 4)

Abre un nuevo documento PDF

int **cpdf_open** (int compression, string filename) \linebreak

LA función **cpdf_open()** abre un documento PDF nuevo. El primer parámetro activa la compresión del documento si no es igual a 0. El segundo parámetro, opcional, es el fichero en el que el documento es escrito. Si es omitido, el documento es creado en memoria y puede ser escrito en un fichero mediante la función cpdf_save_to_file() o escrito por la salida estándar con cpdf_output_buffer().

> **Nota:** El valor de retorno será necesario en nuevas versiones de ClibPDF como el primer parámetro en todas las demás funciones que escriben en el documento PDF.
>
> La librería ClibPDF toma el nombre de fichero "-" como sinónimo de stdout (salida estándar). Si se compila PHP como módulo de apache esto no funcionará porque la manera en que ClibPDF direcciona a la salida estándar no funciona con apache. Usted puede solucionar este problema evitando el enobre de fichero y usando cpdf_output_buffer() para la salida de documentos PDF.

Vea también cpdf_close(), cpdf_output_buffer().

cpdf_output_buffer (PHP 3>= 3.0.9, PHP 4)

Pone el documento PDF en el buffer de memoria

void **cpdf_output_buffer** (int pdf document) \linebreak

La función **cpdf_output_buffer()** muestra el documento PDF por la salida estándar. El documento debe ser creado en memoria, que es el caso de la función cpdf_open() cuando ha sido llamada sin parámetros.

Vea también cpdf_open().

cpdf_page_init (PHP 3>= 3.0.8, PHP 4)

Comienza una nueva página

void **cpdf_page_init** (int pdf document, int page number, int orientation, double height, double width, double unit) \linebreak

La función **cpdf_page_init()** crea una nueva página de altura *height* y profundidad *width*. La página tiene el número *page number* y orientación *orientation*. *orientation* puede ser 0 para retrato y 1 para paisaje. El último parámetro opcional *unit* define la unidad del sistema de coordenadas. El valor debería ser el número de puntos postscript por unidad. Como el valor de una pulgada el igual a 72 puntos, un valor de 72 sería la unidad para una pulgada. Por defecto es 72.

Vea también cpdf_set_current_page().

cpdf_place_inline_image (PHP 3>= 3.0.9, PHP 4)

Situa una imagen en la página

void **cpdf_place_inline_image** (int pdf document, int image, double x-koor, double y-koor, double angle, double width, double height, int mode) \linebreak

La función **cpdf_place_inline_image()** situa una imagen creada con las funciones de imagenes de PHP en la posición de la página (*x-koor*, *y-koor*). La imagen puede ser escalada al mismo tiempo.

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas son medidas en puntos postscript, descartando la unidad actual.

Vea también cpdf_import_jpeg(),

cpdf_rect (PHP 3>= 3.0.8, PHP 4)

Dibuja un rectángulo

void **cpdf_rect** (int pdf document, double x-koor, double y-koor, double width, double height, int mode) \linebreak

La función **cpdf_rect()** dibuja un rectángulo con su esquina inferior izquierda en el punto (*x-koor*, *y-koor*). La anchura es *widgth*. La altura es *height*.

El último parámetro opcional define el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

cpdf_restore (PHP 3>= 3.0.8, PHP 4)

Restaura un entorno formalmente salvado

void **cpdf_restore** (int pdf document) \linebreak

La función **cpdf_restore()** restaura el entorno salvado con cpdf_save(). Funciona como el comando grestore de postscript. Muy útil si se quiere trasladar o rotar un objeto sin afectar ortros objetos.

Ejemplo 1. Salvar/Restaurar

```
<?php cpdf_save($pdf);
// hacer todo tipo de rotaciones, transformaciones, ...
cpdf_restore($pdf) ?>
```

Vea también cpdf_save().

cpdf_rlineto (PHP 3>= 3.0.9, PHP 4)

Dibuja una línea

void **cpdf_rlineto** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

La función **cpdf_rlineto()** dibuja una línea desde el punto actual al punto relativo con coordenadas (*x-koor*, *y-koor*).

El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa el valor por defecto para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto().

cpdf_rmoveto (PHP 3>= 3.0.9, PHP 4)

Define el punto actual

void **cpdf_rmoveto** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

La función **cpdf_rmoveto()** pone el punto actual relativo a las coordenadas *x-koor* y *y-koor*.

El último parámetro opciona determina la loingitud de la unidad. Si es 0 o se omite, la unidad por defecto será la especificada para la página. De otro modo las coordenadas se medirán en puntos postscript, despreciando la unidad en curso.

Vea también cpdf_moveto().

cpdf_rotate_text (PHP 3>= 3.0.9, PHP 4)

Sets text rotation angle

void **cpdf_rotate_text** (int pdfdoc, float angle) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cpdf_rotate (PHP 3>= 3.0.8, PHP 4)

Define la rotación

void **cpdf_rotate** (int pdf document, double angle) \linebreak

La función **cpdf_rotate()** define la rotación en *angle* grados.

cpdf_save_to_file (PHP 3>= 3.0.8, PHP 4)

Escribe el documento PDF en un fichero

void **cpdf_save_to_file** (int pdf document, string filename) \linebreak

La función **cpdf_save_to_file()** guarda el documento PDF en un fichero si este documeto ha sido creado en memoria. Esta función no es necesaria si el documento PDF ha sido abierto mediante la especificación de un nombre de fichero en la función cpdf_open().

Vea también cpdf_output_buffer(), cpdf_open().

cpdf_save (PHP 3>= 3.0.8, PHP 4)

Salva el entorno actual

void **cpdf_save** (int pdf document) \linebreak

La función **cpdf_save()** salva el entorno actual. Funciona como el comando gsave de postscript. Muy útil si se quiere trasladar o trotar un objeto sin afetar a los demás.

Vea también cpdf_restore().

cpdf_scale (PHP 3>= 3.0.8, PHP 4)

Define la escala

void **cpdf_scale** (int pdf document, double x-scale, double y-scale) \linebreak

La función **cpdf_scale()** define el factor de escala en los dos sentidos.

cpdf_set_action_url (PHP 3>= 3.0.9, PHP 4)

Sets hyperlink

void **cpdf_set_action_url** (int pdfdoc, float xll, float yll, float xur, float xur, string url [, int mode]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cpdf_set_char_spacing (PHP 3>= 3.0.8, PHP 4)

Determina el espacio entre caracteres

void **cpdf_set_char_spacing** (int pdf document, double space) \linebreak

LA función **cpdf_set_char_spacing()** define el espacio entre caracteres.

Vea también cpdf_set_word_spacing(), cpdf_set_leading().

cpdf_set_creator (PHP 3>= 3.0.8, PHP 4)

Define el campo creator en el documento PDF

void **cpdf_set_creator** (string creator) \linebreak

La función **cpdf_set_creator()** define el creador de un documento PDF.

Vea también cpdf_set_subject(), cpdf_set_title(), cpdf_set_keywords().

cpdf_set_current_page (PHP 3>= 3.0.9, PHP 4)

Define la página actual

void **cpdf_set_current_page** (int pdf document, int page number) \linebreak

La función **cpdf_set_current_page()** define la página en la que se van a realizar todas las operaciones. Uno puede cambiar entre páginas a menos que una página ha sido finalizada con cpdf_finalize_page().

Vea también cpdf_finalize_page().

cpdf_set_font_directories (PHP 4 >= 4.0.6)

Sets directories to search when using external fonts

void **cpdf_set_font_directories** (int pdfdoc, string pfmdir, string pfbdir) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cpdf_set_font_map_file (PHP 4 >= 4.0.6)

Sets fontname to filename translation map when using external fonts

void **cpdf_set_font_map_file** (int pdfdoc, string filename) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cpdf_set_font (PHP 3>= 3.0.8, PHP 4)

Selecciona la fuente y el tamaño actual

void **cpdf_set_font** (int pdf document, string font name, double size, string encoding) \linebreak

La función **cpdf_set_font()** define la fuente actual, el tamaño y la codificación. Actualmente solo son soportadas las fuentes estándar de postscript. El último parámetro *encoding* puede tomar los siguientes valores: "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding", y "NULL". "NULL" es para el cifrado incluído en la fuente. Para mas información vea el manual de ClibPDF, especialmente para cómo soportar las fuentes asiáticas.

cpdf_set_horiz_scaling (PHP 3>= 3.0.8, PHP 4)

Define la escala horizontal del texto

void **cpdf_set_horiz_scaling** (int pdf document, double scale) \linebreak

La función **cpdf_set_horiz_scaling()** define la escala horizontal al *scale* por ciento.

cpdf_set_keywords (PHP 3>= 3.0.8, PHP 4)

Pone el valor del campo 'keywords'(palabras clave) de un documento PDF

void **cpdf_set_keywords** (string keywords) \linebreak

La función **cpdf_set_keywords()** define las palabras clave de un documento PDF.

Vea también cpdf_set_title(), cpdf_set_creator(), cpdf_set_subject().

cpdf_set_leading (PHP 3>= 3.0.8, PHP 4)

Define la distancias entre las líneas de texto

void **cpdf_set leading** (int pdf document, double distance) \linebreak

La función **cpdf_set_leading()** define la distancia entre las líneas de texto. Esto se usará si el texto es la salida de cpdf_continue_text().

Vea también cpdf_continue_text().

cpdf_set_page_animation (PHP 3>= 3.0.9, PHP 4)

Define la separación entre páginas

void **cpdf_set_page_animation** (int pdf document, int transition, double duration) \linebreak

La función **cpdf_set_page_animation()** define la transición entre páginas que se siguen.

El valor de *transition* puede ser

0 para ninguno,
1 para dos líneas que se barren a través de la pantalla, revelen la página,
2 para múltiples líneas,
3 para que una caja revele la página,
4 para una única línea,
5 para que la página naterior se disipe para revelar la pagina,
6 para que el efecto de disolución se mueva de un extremop de la página al otro,
7 para que la página antígua simplemente sea reemplazada por la nueva página (default)

El valor de *duration* es el número de segundos entre las páginas que se pasan.

cpdf_set_subject (PHP 3>= 3.0.8, PHP 4)

Define el valor del campo subjet de un documento PDF

void **cpdf_set_subject** (string subject) \linebreak

La función **cpdf_set_subject()** define el asunto de un documento PDF

Vea también cpdf_set_title(), cpdf_set_creator(), cpdf_set_keywords().

cpdf_set_text_matrix (PHP 3>= 3.0.8, PHP 4)

Define la matriz de texto

void **cpdf_set_text_matrix** (int pdf document, array matrix) \linebreak

La función **cpdf_set_text_matrix()** define una matriz que describe una transformación aplicada a la fuente actual de texto.

cpdf_set_text_pos (PHP 3>= 3.0.8, PHP 4)

Define la posición del texto

void **cpdf_set_text_pos** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

La función **cpdf_set_text_pos()** define la posición del texto para la siguiente llamada a cpdf_show().

El último parámetro opcional *mode* determina la longitud de la unidad. Si es 0 o se omite, se usa el valor por defecto para la página. De otro modo, las coordenadas son medidas en puntos postscript, despreciando la unidad actual.

Vea también cpdf_show(), cpdf_text().

cpdf_set_text_rendering (PHP 3>= 3.0.8, PHP 4)

Determina cómo es presentado el texto

void **cpdf_set_text_rendering** (int pdf document, int mode) \linebreak

La función **cpdf_set_text_rendering()** determina cómo es presentado el texto. Los posibles valores para *mode* son 0=llenar texto, 1=poner texto, 2=llenar y poner texto, 3=invisible, 4=llenar texto y añadirlo al camino de corte, 5=poner texto y añadirlo al camino de corte, 6=llenar y poner texto y añadirlo al camino de corte, 7=añadirlo al camino de corte

cpdf_set_text_rise (PHP 3>= 3.0.8, PHP 4)

Define la elevación del texto

void **cpdf_set_text_rise** (int pdf document, double value) \linebreak

La función **cpdf_set_text_rise()** define la elevación del texto a *value* unidades.

cpdf_set_title (PHP 3>= 3.0.8, PHP 4)

Define el campo title de un documento PDF

void **cpdf_set_title** (string title) \linebreak

La función **cpdf_set_title()** define el título de un documento PDF

Vea también cpdf_set_subject(), cpdf_set_creator(), cpdf_set_keywords().

cpdf_set_viewer_preferences (PHP 3>= 3.0.9, PHP 4)

How to show the document in the viewer

void **cpdf_set_viewer_preferences** (int pdfdoc, array preferences) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cpdf_set_word_spacing (PHP 3>= 3.0.8, PHP 4)

Define el espacio entre palabras

void **cpdf_set_word_spacing** (int pdf document, double space) \linebreak

La función **cpdf_set_word_spacing()** especifica el espacio entre palabras.

Vea también cpdf_set_char_spacing(), cpdf_set_leading().

cpdf_setdash (PHP 3>= 3.0.8, PHP 4)

Defina el patrón de la raya

void **cpdf_setdash** (int pdf document, double white, double black) \linebreak

La función **cpdf_setdash()** define el patrón de la raya `white` unidades blancas y `black` unidades negras. Si los dos son 0 se pone una línea sólida.

cpdf_setflat (PHP 3>= 3.0.8, PHP 4)

Define la monotonía

void **cpdf_setflat** (int pdf document, double value) \linebreak

La función **cpdf_setflat()** pone la monotonía a un valor de entre 0 y 100.

cpdf_setgray_fill (PHP 3>= 3.0.8, PHP 4)

Pone el color de relleno al valor gris

void **cpdf_setgray_fill** (int pdf document, double value) \linebreak

La función **cpdf_setgray_fill()** define el valor de gris actual para rellelanr un camino.

Vea también cpdf_setrgbcolor_fill().

cpdf_setgray_stroke (PHP 3>= 3.0.8, PHP 4)

Define el color para dibujar al valor gris

void **cpdf_setgray_stroke** (int pdf document, double gray value) \linebreak

La función **cpdf_setgray_stroke()** pone el color de dibujo actual al valor de gris dado.

Vea también cpdf_setrgbcolor_stroke().

cpdf_setgray (PHP 3>= 3.0.8, PHP 4)

Pone el color de relleno y dibujo a gris

void **cpdf_setgray** (int pdf document, double gray value) \linebreak

La función cpdf_setgray_stroke() pone el color de relleno y dibujo al color gris dado.

Vea también cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_setlinecap (PHP 3>= 3.0.8, PHP 4)

Define el parámetro linecap

void **cpdf_setlinecap** (int pdf document, int value) \linebreak

La función **cpdf_setlinecap()** define el parámetro linecap entre los valores 0 y 2. 0 = empalmar al final, 1 = redondear, 2 = esquina proyectada

cpdf_setlinejoin (PHP 3>= 3.0.8, PHP 4)

Define el parámetro linejoin

void **cpdf_setlinejoin** (int pdf document, long value) \linebreak

La función **cpdf_setlinejoin()** define el parámetro entre un valor de 0 y 2. 0 = ingletes, 1 = redondear, 2 = ángulo oblícuo

cpdf_setlinewidth (PHP 3>= 3.0.8, PHP 4)

Define la profundidad de la línea

void **cpdf_setlinewidth** (int pdf document, double width) \linebreak

La función **cpdf_setlinewidth()** define la preofundidad de la línea a *`width`*.

cpdf_setmiterlimit (PHP 3>= 3.0.8, PHP 4)

Define el límite del inglete

void **cpdf_setmiterlimit** (int pdf document, double value) \linebreak

La función **cpdf_setmiterlimit()** define el límite del inglete a un valor mayor o igual a 1.

cpdf_setrgbcolor_fill (PHP 3>= 3.0.8, PHP 4)

Pone el color de relleno a l valor de clor rgb

void **cpdf_setrgbcolor_fill** (int pdf document, double red value, double green value, double blue value) \linebreak

La función **cpdf_setrgbcolor_fill()** pone el color rgb actual para rellenar un camino.

Vea también cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor().

cpdf_setrgbcolor_stroke (PHP 3>= 3.0.8, PHP 4)

Pone el color de dibujo al valor de color rgb

void **cpdf_setrgbcolor_stroke** (int pdf document, double red value, double green value, double blue value) \linebreak

La función **cpdf_setrgbcolor_stroke()** pone el color de dibujo actual al valor de color rgb dado.

Vea también cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_setrgbcolor (PHP 3>= 3.0.8, PHP 4)

Pone el color de relleno y dibujo al valor de color rgb

void **cpdf_setrgbcolor** (int pdf document, double red value, double green value, double blue value) \linebreak

La función cpdf_setrgbcolor_stroke() pone el color de relleno y dibujo actual al color rgb dado.

Vea también cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_show_xy (PHP 3>= 3.0.8, PHP 4)

Muestra texto en la posición

void **cpdf_show_xy** (int pdf document, string text, double x-koor, double y-koor, int mode) \linebreak

La función **cpdf_show_xy()** muestra la cadena *text* en la posición con coordenadas (*x-coor*, *y-coor*). El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas son medidas en puntos postscript, despreciando la unidad actual.

Nota: La función **cpdf_show_xy()** es idéntica a cpdf_text() sin el parámetro opcional.

Vea también cpdf_text().

cpdf_show (PHP 3>= 3.0.8, PHP 4)

Muestra el texto en la posición actual

void **cpdf_show** (int pdf document, string text) \linebreak

La función **cpdf_show()** muestra la cadena *text* en la posixción actual.

Vea también cpdf_text(), cpdf_begin_text(), cpdf_end_text().

cpdf_stringwidth (PHP 3>= 3.0.8, PHP 4)

Devuelve la anchura del texto en la fuente actual

double **cpdf_stringwidth** (int pdf document, string text) \linebreak

La función **cpdf_stringwidth()** devuelve la anchura de la cadena *text*. Requiere haber definido antes una fuente.

Vea también cpdf_set_font().

cpdf_stroke (PHP 3>= 3.0.8, PHP 4)

Dibuja una línea a lo largo del camino

void **cpdf_stroke** (int pdf document) \linebreak

La función **cpdf_stroke()** dibuja una línea a lo largo del camino actual.

Vea también cpdf_closepath(), cpdf_closepath_stroke().

cpdf_text (PHP 3>= 3.0.8, PHP 4)

Muestra texto conparámetros

void **cpdf_text** (int pdf document, string text, double x-koor, double y-koor, int mode, double orientation, int alignmode) \linebreak

La función **cpdf_text()** muestra la cadena *text* en la posición de coordenadas (*x-coor*, *y-coor*). El parámero opcional determina la longitud de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas son medidas en puntos postscript despreciando la unidad actual. El parámetro opcional *orientation* es la rotación del texto en grados. El parámetro opcional *alignmode* determina cómo está alineado el texto. Vea la documentación de ClibPDF para los posibles valores.

Vea también cpdf_show_xy().

cpdf_translate (PHP 3>= 3.0.8, PHP 4)

Define el sistema de origen de coordenadas

void **cpdf_translate** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

La función **cpdf_translate()** define el sistema origen de coordenadas en el punto (*x-coor*, *y-coor*).

El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada en la página. De otro modo las coordenadas son medidas en puntos postscript, depreciando la unidad actual.

XI. Crack functions

Introducción

These functions allow you to use the CrackLib library to test the ’strength’ of a password. The ’strength’ of a password is tested by that checks length, use of upper and lower case and checked against the specified CrackLib dictionary. CrackLib will also give helpful diagnostic messages that will help ’strengthen’ the password.

Requerimientos

More information regarding CrackLib along with the library can be found at http://www.users.dircon.co.uk/~crypto/.

Instalación

In order to use these functions, you must compile PHP with Crack support by using the `--with-crack[=DIR]` option.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

Esta extensión no define ningún tipo de recurso.

Constantes predefinidas

Esta extensión no define ninguna constante.

Ejemplos

This example shows how to open a CrackLib dictionary, test a given password, retrieve any diagnostic

messages, and close the dictionary.

Ejemplo 1. CrackLib example

```
<?php
// Open CrackLib Dictionary
$dictionary = crack_opendict('/usr/local/lib/pw_dict')
     or die('Unable to open CrackLib dictionary');

// Perform password check
$check = crack_check($dictionary, 'gx9A2s0x');

// Retrieve messages
$diag = crack_getlastmessage();
echo $diag; // 'strong password'

// Close dictionary
crack_closedict($dictionary);
?>
```

Nota: If crack_check() returns TRUE, crack_getlastmessage() will return 'strong password'.

crack_check (PHP 4 >= 4.0.5)

Performs an obscure check with the given password

bool **crack_check** ([resource dictionary, string password]) \linebreak

Returns `TRUE` if *`password`* is strong, or `FALSE` otherwise.

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

crack_check() performs an obscure check with the given *`password`* on the specified *`dictionary`* . If *`dictionary`* is not specified, the last opened dictionary is used.

crack_closedict (PHP 4 >= 4.0.5)

Closes an open CrackLib dictionary

bool **crack_closedict** ([resource dictionary]) \linebreak

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

crack_closedict() closes the specified *`dictionary`* identifier. If *`dictionary`* is not specified, the current dictionary is closed.

crack_getlastmessage (PHP 4 >= 4.0.5)

Returns the message from the last obscure check

string **crack_getlastmessage** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

crack_getlastmessage() returns the message from the last obscure check.

crack_opendict (PHP 4 >= 4.0.5)

Opens a new CrackLib dictionary

resource **crack_opendict** (string dictionary) \linebreak

Returns a dictionary resource identifier on success, or `FALSE` on failure.

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

crack_opendict() opens the specified CrackLib *`dictionary`* for use with crack_check().

> **Nota:** Only one dictionary may be open at a time.

See also: crack_check(), and crack_closedict().

XII. CURL, Client URL Library Functions

PHP supports libcurl, a library, created by Daniel Stenberg, that allows you to connect and communicate to many different types of servers with many different types of protocols. libcurl currently supports the http, https, ftp, gopher, telnet, dict, file, and ldap protocols. libcurl also supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading (this can also be done with PHP's ftp extension), HTTP form based upload, proxies, cookies and user+password authentication.

In order to use the CURL functions you need to install the CURL (http://curl.haxx.se/) package. PHP requires that you use CURL 7.0.2-beta or higher. PHP will not work with any version of CURL below version 7.0.2-beta.

To use PHP's CURL support you must also compile PHP `--with-curl[=DIR]` where DIR is the location of the directory containing the lib and include directories. In the "include" directory there should be a folder named "curl" which should contain the easy.h and curl.h files. There should be a file named "libcurl.a" located in the "lib" directory.

These functions have been added in PHP 4.0.2.

Once you've compiled PHP with CURL support, you can begin using the curl functions. The basic idea behind the CURL functions is that you initialize a CURL session using the curl_init(), then you can set all your options for the transfer via the curl_exec() and then you finish off your session using the curl_close(). Here is an example that uses the CURL functions to fetch the PHP homepage into a file:

Ejemplo 1. Using PHP's CURL module to fetch the PHP homepage

```
<?php

$ch = curl_init ("http://www.php.net/");
$fp = fopen ("php_homepage.txt", "w");

curl_setopt ($ch, CURLOPT_INFILE, $fp);
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);
curl_close ($ch);
fclose ($fp);
?>
```

curl_close (PHP 4 >= 4.0.2)

Close a CURL session

void **curl_close** (int ch) \linebreak

This functions closes a CURL session and frees all ressources. The CURL handle, *ch*, is also deleted.

curl_errno (PHP 4 >= 4.0.3)

Return an integer containing the last error number

int **curl_errno** (resource ch) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

curl_error (PHP 4 >= 4.0.3)

Return a string containing the last error for the current session

string **curl_error** (resource ch) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

curl_exec (PHP 4 >= 4.0.2)

Perform a CURL session

bool **curl_exec** (int ch) \linebreak

This function is should be called after you initialize a CURL session and all the options for the session are set. Its purpose is simply to execute the predefined CURL session (given by the *ch*).

curl_getinfo (PHP 4 >= 4.0.4)

Get information regarding a specific transfer

string **curl_getinfo** (resource ch, int opt) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

curl_init (PHP 4 >= 4.0.2)

Initialize a CURL session

int **curl_init** ([string url]) \linebreak

The **curl_init()** will initialize a new session and return a CURL handle for use with the curl_setopt(), curl_exec(), and curl_close() functions. If the optional *url* parameter is supplied then the CURLOPT_URL option will be set to the value of the parameter. You can manually set this using the curl_setopt() function.

Ejemplo 1. Initializing a new CURL session and fetching a webpage

```
<?php
$ch = curl_init();

curl_setopt ($ch, CURLOPT_URL, "http://www.zend.com/");
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);

curl_close ($ch);
?>
```

See also: curl_close(), curl_setopt()

curl_setopt (PHP 4 >= 4.0.2)

Set an option for a CURL transfer

bool **curl_setopt** (int ch, string option, mixed value) \linebreak

The **curl_setopt()** function will set options for a CURL session identified by the *ch* parameter. The *option* parameter is the option you want to set, and the *value* is the value of the option given by the *option*.

The *value* should be a long for the following options (specified in the *option* parameter):

- *CURLOPT_INFILESIZE*: When you are uploading a file to a remote site, this option should be used to tell PHP what the expected size of the infile will be.
- *CURLOPT_VERBOSE*: Set this option to a non-zero value if you want CURL to report everything that is happening.
- *CURLOPT_HEADER*: Set this option to a non-zero value if you want the header to be included in the output.
- *CURLOPT_NOPROGRESS*: Set this option to a non-zero value if you don't want PHP to display a progress meter for CURL transfers

  Nota: PHP automatically sets this option to a non-zero parameter, this should only be changed for debugging purposes.

- *CURLOPT_NOBODY*: Set this option to a non-zero value if you don't want the body included with the output.
- *CURLOPT_FAILONERROR*: Set this option to a non-zero value if you want PHP to fail silently if the HTTP code returned is greater than 300. The default behaviour is to return the page normally, ignoring the code.
- *CURLOPT_UPLOAD*: Set this option to a non-zero value if you want PHP to prepare for an upload.
- *CURLOPT_POST*: Set this option to a non-zero value if you want PHP to do a regular HTTP POST. This POST is a normal application/x-www-from-urlencoded kind, most commonly used by HTML forms.
- *CURLOPT_FTPLISTONLY*: Set this option to a non-zero value and PHP will just list the names of an FTP directory.
- *CURLOPT_FTPAPPEND*: Set this option to a non-zero value and PHP will append to the remote file instead of overwriting it.
- *CURLOPT_NETRC*: Set this option to a non-zero value and PHP will scan your ~./netrc file to find your username and password for the remote site that you're establishing a connection with.
- *CURLOPT_FOLLOWLOCATION*: Set this option to a non-zero value to follow any "Location: " header that the server sends as a part of the HTTP header (note this is recursive, PHP will follow as many "Location: " headers that it is sent.)
- *CURLOPT_PUT*: Set this option a non-zero value to HTTP PUT a file. The file to PUT must be set with the CURLOPT_INFILE and CURLOPT_INFILESIZE.
- *CURLOPT_MUTE*: Set this option to a non-zero value and PHP will be completely silent with regards to the CURL functions.

- *CURLOPT_TIMEOUT*: Pass a long as a parameter that contains the maximum time, in seconds, that you'll allow the curl functions to take.
- *CURLOPT_LOW_SPEED_LIMIT*: Pass a long as a parameter that contains the transfer speed in bytes per second that the transfer should be below during CURLOPT_LOW_SPEED_TIME seconds for PHP to consider it too slow and abort.
- *CURLOPT_LOW_SPEED_TIME*: Pass a long as a parameter that contains the time in seconds that the transfer should be below the CURLOPT_LOW_SPEED_LIMIT for PHP to consider it too slow and abort.
- *CURLOPT_RESUME_FROM*: Pass a long as a parameter that contains the offset, in bytes, that you want the transfer to start from.
- *CURLOPT_SSLVERSION*: Pass a long as a parameter that contains the SSL version (2 or 3) to use. By default PHP will try and determine this by itself, although, in some cases you must set this manually.
- *CURLOPT_TIMECONDITION*: Pass a long as a parameter that defines how the CURLOPT_TIMEVALUE is treated. You can set this parameter to TIMECOND_IFMODSINCE or TIMECOND_ISUNMODSINCE. This is a HTTP-only feature.
- *CURLOPT_TIMEVALUE*: Pass a long as a parameter that is the time in seconds since January 1st, 1970. The time will be used as specified by the CURLOPT_TIMEVALUE option, or by default the TIMECOND_IFMODSINCE will be used.

The *value* parameter should be a string for the following values of the *option* parameter:

- *CURLOPT_URL*: This is the URL that you want PHP to fetch. You can also set this option when initializing a session with the curl_init() function.
- *CURLOPT_USERPWD*: Pass a string formatted in the [username]:[password] manner, for PHP to use for the connection. connection.
- *CURLOPT_PROXYUSERPWD*: Pass a string formatted in the [username]:[password] format for connection to the HTTP proxy.
- *CURLOPT_RANGE*: Pass the specified range you want. It should be in the "X-Y" format, where X or Y may be left out. The HTTP transfers also support several intervals, seperated with commas as in X-Y,N-M.
- *CURLOPT_POSTFIELDS*: Pass a string containing the full data to post in an HTTP "POST" operation.
- *CURLOPT_REFERER*: Pass a string containing the "referer" header to be used in an HTTP request.
- *CURLOPT_USERAGENT*: Pass a string containing the "user-agent" header to be used in an HTTP request.
- *CURLOPT_FTPPORT*: Pass a string containing the which will be used to get the IP address to use for the ftp "PORT" instruction. The POST instruction tells the remote server to connect to our specified IP address. The string may be a plain IP address, a hostname, a network interface name (under UNIX), or just a plain '-' to use the systems default IP address.
- *CURLOPT_COOKIE*: Pass a string containing the content of the cookie to be set in the HTTP header.

- *CURLOPT_SSLCERT*: Pass a string containing the filename of PEM formatted certificate.
- *CURLOPT_SSLCERTPASSWD*: Pass a string containing the password required to use the CURLOPT_SSLCERT certificate.
- *CURLOPT_COOKIEFILE*: Pass a string containing the name of the file containing the cookiee data. The cookie file can be in Netscape format, or just plain HTTP-style headers dumped into a file.
- *CURLOPT_CUSTOMREQUEST*: Pass a string to be used instead of GET or HEAD when doing an HTTP request. This is useful for doing DELETE or another, more obscure, HTTP request.

  Nota: Don't do this without making sure your server supports the command first.

The following options expect a file descriptor that is obtained by using the fopen() function:

- *CURLOPT_FILE*: The file where the output of your transfer should be placed, the default is STDOUT.
- *CURLOPT_INFILE*: The file where the input of your transfer comes from.
- *CURLOPT_WRITEHEADER*: The file to write the header part of the output into.
- *CURLOPT_STDERR*: The file to write errors to instead of stderr.

curl_version (PHP 4 >= 4.0.2)

Return the current CURL version

string **curl_version** (void) \linebreak

The **curl_version()** function returns a string containing the current CURL version.

XIII. Funciones de pago electrónico

Estas funciones solo están disponibles si el intérprete ha sido compilado con `--with-cybercash=[DIR]`. Estas funciones han sido añadidas en PHP4.

cybercash_base64_decode (PHP 4)

string **cybercash_base64_decode** (string inbuff) \linebreak

cybercash_base64_encode (PHP 4)

???

string **cybercash_base64_encode** (string inbuff) \linebreak

cybercash_decr (PHP 4)

???

array **cybercash_decr** (string wmk, string sk, string inbuff) \linebreak

La función devuelve un array asociativo con los elementos "errcode" y, si "errcode" es `FALSE`, "outbuff" (string), "outLth" (long) y "macbuff" (string).

cybercash_encr (PHP 4)

???

array **cybercash_encr** (string wmk, string sk, string inbuff) \linebreak

La función devuelve un array asociativo con los elementos "errcode" y, si "errcode" es `FALSE`, "outbuff" (string), "outLth" (long) y "macbuff" (string).

XIV. Crédit Mutuel CyberMUT functions

Introducción

This extension allows you to process credit cards transactions using Crédit Mutuel CyberMUT system (http://www.creditmutuel.fr/centre_commercial/vendez_sur_internet.html).

CyberMUT is a popular Web Payment Service in France, provided by the Crédit Mutuel bank. If you are foreign in France, these functions will not be useful for you.

The use of these functions is almost identical to the original SDK functions, except for the parameters of return for cybermut_creerformulairecm() and cybermut_creerreponsecm(), which are returned directly by functions PHP, whereas they had passed in reference in the original functions.

These functions have been added in PHP 4.0.6.

> **Nota:** These functions only provide a link to CyberMUT SDK. Be sure to read the CyberMUT Developers Guide for full details of the required parameters.

Instalación

These functions are only available if PHP has been compiled with the `--with-cybermut[=DIR]` option, where DIR is the location of `libcm-mac.a` and `cm-mac.h`. You will require the appropriate SDK for your platform, which may be sent to you after your CyberMUT's subscription (contact them via Web, or go to the nearest Crédit Mutuel).

cybermut_creerformulairecm (PHP 4 >= 4.0.5)

Generate HTML form of request for payment

string **cybermut_creerformulairecm** (string url_CM, string version, string TPE, string montant, string ref_commande, string texte_libre, string url_retour, string url_retour_ok, string url_retour_err, string langue, string code_societe, string texte_bouton) \linebreak

cybermut_creerformulairecm() is used to generate the HTML form of request for payment.

Ejemplo 1. First step of payment (equiv cgi1.c)

```
<?php
// Directory where the keys are located
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

  $retour =  cybermut_creerformulairecm(
  "https://www.creditmutuel.fr/test/telepaiement/paiement.cgi",
  $VERSION,
  "1234567890",
  "300FRF",
  $REFERENCE,
  $TEXTE_LIBRE,
  $URL_RETOUR,
  $URL_RETOUR_OK,
  $URL_RETOUR_ERR,
  "francais",
  "company",
  "Paiement par carte bancaire");

  echo $retour;
?>
```

See also cybermut_testmac() and cybermut_creerreponsecm().

cybermut_creerreponsecm (PHP 4 >= 4.0.5)

Generate the acknowledgement of delivery of the confirmation of payment

string **cybermut_creerreponsecm** (string phrase) \linebreak

cybermut_creerreponsecm() returns a string containing delivery acknowledgement message.

The parameter is "OK" if the message of confirmation of the payment was correctly identified by cybermut_testmac(). Any other chain is regarded as an error message.

See also cybermut_creerformulairecm() and cybermut_testmac().

cybermut_testmac (PHP 4 >= 4.0.5)

Make sure that there no was data diddling contained in the received message of confirmation

bool **cybermut_testmac** (string code_MAC, string version, string TPE, string cdate, string montant, string ref_commande, string texte_libre, string code-retour) \linebreak

cybermut_testmac() is used to make sure that there was not data diddling contained in the received message of confirmation. Pay attention to parameters *code-retour* and *texte-libre*, which cannot be evaluated as is, because of the dash. You must retrieve them by using:

```
<?php
  $code_retour = $_GET["code-retour"];
  $texte_libre = $_GET["texte-libre"];
?>
```

Ejemplo 1. Last step of payment (equiv cgi2.c)

```
<?php
// Make sure that Enable Track Vars is ON.
// Directory where are located the keys
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

$texte_libre = $_GET["texte-libre"];
$code_retour = $_GET["code-retour"];

$mac_ok = cybermut_testmac($MAC,$VERSION,$TPE,$date,$montant,$reference,$texte_libre,$code_r

if ($mac_ok) {

  //
  // insert data processing here
  //
  //

  $result=cybermut_creerreponsecm("OK");
```

```
} else {
  $result=cybermut_creerreponsecm("Document Falsifie");
}

?>
```

See also cybermut_creerformulairecm() and cybermut_creerreponsecm().

XV. Cyrus IMAP administration functions

Introducción

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinamicamente en tiempo de ejecución.

`CYRUS_CONN_NONSYNCLITERAL` (integer)

`CYRUS_CONN_INITIALRESPONSE` (integer)

`CYRUS_CALLBACK_NUMBERED` (integer)

`CYRUS_CALLBACK_NOLITERAL` (integer)

cyrus_authenticate (PHP 4 >= 4.1.0)

Authenticate against a Cyrus IMAP server

bool **cyrus_authenticate** (resource connection [, string mechlist [, string service [, string user [, int minssf [, int maxssf]]]]]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cyrus_bind (PHP 4 >= 4.1.0)

Bind callbacks to a Cyrus IMAP connection

bool **cyrus_bind** (resource connection, array callbacks) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cyrus_close (PHP 4 >= 4.1.0)

Close connection to a Cyrus IMAP server

bool **cyrus_close** (resource connection) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cyrus_connect (PHP 4 >= 4.1.0)

Connect to a Cyrus IMAP server

resource **cyrus_connect** ([string host [, string port [, int flags]]]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cyrus_query (PHP 4 >= 4.1.0)

Send a query to a Cyrus IMAP server

bool **cyrus_query** (resource connection, string query) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

cyrus_unbind (PHP 4 >= 4.1.0)

Unbind ...

bool **cyrus_unbind** (resource connection, string trigger_name) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

XVI. Character type functions

Introducción

The functions provided by this extension check whether a character or string falls into a certain character class according to the current locale (see also setlocale()).

When called with an integer argument these functions behave exactly like their C counterparts from "ctype.h".

When called with a string argument they will check every character in the string and will only return `TRUE` if every character in the string matches the requested criteria.

Passing anything else but a string or integer will return `FALSE` immediately.

Requerimientos

None besides functions from the standard C library which are always available.

Instalación

Beginning with PHP 4.2.0 these functions are enabled by default. For older versions you have to configure and compile PHP with `--enable-ctype`.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

Esta extensión no define ningún tipo de recurso.

Constantes predefinidas

Esta extensión no define ninguna constante.

ctype_alnum (PHP 4 >= 4.0.4)

Check for alphanumeric character(s)

bool **ctype_alnum** (string text) \linebreak

Returns `TRUE` if every character in *`text`* is either a letter or a digit, `FALSE` otherwise. In the standard `C` locale letters are just `[A-Za-z]`. The function is equivalent to `(ctype_alpha($text) || ctype_digit($text))`.

See also ctype_alpha(), ctype_digit(), and setlocale().

ctype_alpha (PHP 4 >= 4.0.4)

Check for alphabetic character(s)

bool **ctype_alpha** (string text) \linebreak

Returns `TRUE` if every character in *`text`* is a letter from the current locale, `FALSE` otherwise. In the standard `C` locale letters are just `[A-Za-z]` and **ctype_alpha()** is equivalent to `(ctype_upper($text) || ctype_lower($text))`, but other languages have letters that are considered neither upper nor lower case.

See also ctype_upper(), ctype_lower(), and setlocale().

ctype_cntrl (PHP 4 >= 4.0.4)

Check for control character(s)

bool **ctype_cntrl** (string text) \linebreak

Returns `TRUE` if every character in *`text`* has a special control function, `FALSE` otherwise. Control characters are e.g. line feed, tab, esc.

ctype_digit (PHP 4 >= 4.0.4)

Check for numeric character(s)

bool **ctype_digit** (string text) \linebreak

Returns `TRUE` if every character in *`text`* is a decimal digit, `FALSE` otherwise.

See also ctype_alnum() and ctype_xdigit().

ctype_graph (PHP 4 >= 4.0.4)

Check for any printable character(s) except space

bool **ctype_graph** (string text) \linebreak

Returns `TRUE` if every character in *`text`* is printable and actually creates visible output (no white space), `FALSE` otherwise.

See also ctype_alnum(), ctype_print(), and ctype_punct().

ctype_lower (PHP 4 >= 4.0.4)

Check for lowercase character(s)

bool **ctype_lower** (string text) \linebreak

Returns `TRUE` if every character in *`text`* is a lowercase letter in the current locale.

See also ctype_alpha() and ctype_upper().

ctype_print (PHP 4 >= 4.0.4)

Check for printable character(s)

bool **ctype_print** (string text) \linebreak

Returns `TRUE` if every character in *`text`* will actually create output (including blanks). Returns `FALSE` if *`text`* contains control characters or characters that do not have any output or control function at all.

See also ctype_cntrl(), ctype_graph(), and ctype_punct().

ctype_punct (PHP 4 >= 4.0.4)

Check for any printable character which is not whitespace or an alphanumeric character

bool **ctype_punct** (string text) \linebreak

Returns `TRUE` if every character in *`text`* is printable, but neither letter, digit or blank, `FALSE` otherwise.

See also ctype_cntrl(), ctype_graph(), and **ctype_punct()**.

ctype_space (PHP 4 >= 4.0.4)

Check for whitespace character(s)

bool **ctype_space** (string text) \linebreak

Returns `TRUE` if every character in *`text`* creates some sort of white space, `FALSE` otherwise. Besides the blank character this also includes tab, vertical tab, line feed, carriage return and form feed characters.

ctype_upper (PHP 4 >= 4.0.4)

Check for uppercase character(s)

bool **ctype_upper** (string text) \linebreak

Returns `TRUE` if every character in *`text`* is a uppercase letter in the current locale.

See also ctype_alpha() and ctype_lower().

ctype_xdigit (PHP 4 >= 4.0.4)

Check for character(s) representing a hexadecimal digit

bool **ctype_xdigit** (string text) \linebreak

Returns `TRUE` if every character in *`text`* is a hexadecimal 'digit', that is a decimal digit or a character from `[A-Fa-f]` , `FALSE` otherwise.

See also ctype_digit().

XVII. Funciones de la capa de abstraccion de bases de datos (dbm-style)

Estas funciones son la base para el acceso a bases de datos del estilo Berkeley DB.

Este es un nivel de abstraccion general para varias bases de datos. Como tal su funcionalidad esta limitada a un grupo de modernas bases de datos como Sleepycat Software's DB2 (). (Esta no debe confundirse con IBM DB2 software, la cual es soportada mediante las funciones ODBC.)

El comportamiento de varios aspectos depende de la implementacion de la base de datos. Funciones como dba_optimize() y dba_sync() cumpliran su funcionalidad con unas bases de datos pero no con otras.

Los siguientes manejadores (handlers) estan soportados:

- dbm es el mas antiguo (original) tipo de base de datos de la familia de Berkeley DB. Se debe evitar su uso, si es posible. Nosotros no soportamos las funciones de compatibilidad de DB2 y gdbm, porque ellas solo son compatibles a nivel de codigo fuente, pero no pueden manejar el formato original dbm.
- ndbm es un tipo mas nuevo y mas flexible que dbm. Todavia tiene la mayoria de las limitaciones de dbm (Por lo tanto es descartado).
- gdbm es el gestor de bases de datos de GNU (database manager) ().
- db2 es Sleepycat Software's DB2 (). Es descrito como "un conjunto de herramientas de programacion que proveen acceso de alto nivel a bases de datos en aplicaciones standalone o en el modelo cliente/servidor. "
- cdb es "una rapida, de confianza, sencilla herramienta para la creacion y lectura de bases de datos constantes." Fue creada por el autor de qmail y puede encontrarse en here (). Como la base es constante solo se soportan las operaciones de lectura.

Ejemplo 1. Ejemplo de DBA

```
<?php

$id = dba_open("/tmp/test.db", "n", "db2");

if(!$id) {
    echo "dba_open failed\n";
    exit;
}

dba_replace("key", "This is an example!", $id);

if(dba_exists("key", $id)) {
    echo dba_fetch("key", $id);
    dba_delete("key", $id);
}

dba_close($id);
```

```
?>
```

DBA es "binary safe" y no tiene ningun limite arbitrario. Hereda todas sus limitaciones de la implementacion de base de datos que tenga.

Todos las bases de datos basadas en ficheros deben proveer un mecanismo para establecer el modo a la hora de crear nuevas bases de datos, si ello es posible. Habitualmente este modo es pasado como el cuarto argumento en dba_open() o en dba_popen().

Se puede acceder a todas las entradas de una base de datos de modo secuencial (lineal) usando las funciones dba_firstkey() y dba_nextkey(). No se puede cambiar la base de datos mientras se recorre (traversing) por ella.

Ejemplo 2. Recorriendo una base de datos

```
<?php

# ...open database...

$key = dba_firstkey($id);

while($key != false) {
    if(...) { # remember the key to perform some action later
        $handle_later[] = $key;
    }
    $key = dba_nextkey($id);
}

for($i = 0; $i < count($handle_later); $i++)
    dba_delete($handle_later[$i], $id);

?>
```

dba_close (PHP 3>= 3.0.8, PHP 4)

Cerrar uba base de datos

void **dba_close** (int handle) \linebreak

dba_close() cierra la conexion con una base de datos previamente abierta y libera todos los recursos especificados por *handle*.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_close() no devuelve ningun valor.

Ver tambien: dba_open() dba_popen()

dba_delete (PHP 3>= 3.0.8, PHP 4)

Borra una entrada especificada por la clave key

bool **dba_delete** (string key, int handle) \linebreak

dba_delete() borra la entrada especificada por *key* de la base de datos especificada por *handle*.

key es la clave de la entrada que es borrada.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_delete() devuelve TRUE o FALSE, si la entrada es borrada o no, respectivamente.

Ver tambien: dba_exists() dba_fetch() dba_insert() dba_replace()

dba_exists (PHP 3>= 3.0.8, PHP 4)

Comprueba si la clave key existe

bool **dba_exists** (string key, int handle) \linebreak

dba_exists() comprueba si la clave *key* existe en la base de datos especificada por *handle*.

key es la clave para la que se realiza la comprobacion.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_exists() devuelve TRUE o FALSE, si la clave es hallada o no, respectivamente.

Ver tambien: dba_fetch() dba_delete() dba_insert() dba_replace()

dba_fetch (PHP 3>= 3.0.8, PHP 4)

Extrae los datos especificados por la clave key

string **dba_fetch** (string key, int handle) \linebreak

dba_fetch() extrae los datos especificados por la clave *key* de la base de datos determinada por *handle*.

key es la clave de la entrada de los datos que queremos extraer.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_fetch() devuelve la cadena asociada o FALSE, si el par key/data es hallado o no, respectivamente.

Ver tambien: dba_exists() dba_delete() dba_insert() dba_replace()

dba_firstkey (PHP 3>= 3.0.8, PHP 4)

Conseguir la primera clave

string **dba_firstkey** (int handle) \linebreak

dba_firstkey() devuelve la primera clave de la base de datos especificada por *handle* y resetea el puntero interno de claves. Esto permite una busqueda lineal por toda la base de datos.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_firstkey() devuelve la clave o FALSE en funcion de si tiene exito o falla, respectivamente.

Ver tambien: dba_nextkey()

dba_insert (PHP 3>= 3.0.8, PHP 4)

Insertar una entrada

bool **dba_insert** (string key, string value, int handle) \linebreak

dba_insert() inserta la entrada descrita con *key* y *value* dentro de la base de datos especificada por *handle*. Fallara si ya existe una entrada con el mismo parametro *key*.

key es la clave de la entrada a ser insertada.

value es el valor a ser insertado.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_insert() devuelve TRUE o FALSE, en funcion de si tiene exito o falla, respectivamente.

Ver tambien: dba_exists() dba_delete() dba_fetch() dba_replace()

dba_nextkey (PHP 3>= 3.0.8, PHP 4)

Extraer la siguiente clave

string **dba_nextkey** (int handle) \linebreak

dba_nextkey() devuelve la siguiente clave de la base de datos especificada por *handle* e incrementa el puntero de claves interno.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_nextkey() devuelve la clave o FALSE dependiendo de si tiene exito o falla, respectivamente.

Ver tambien: dba_firstkey()

dba_open (PHP 3>= 3.0.8, PHP 4)

Abrir una base de datos

int **dba_open** (string path, string mode, string handler [, ...]) \linebreak

dba_open() establece una instancia para *path* con *mode* usando *handler*.

path normalmente es el "path" en el sistema de ficheros.

mode es "r" para acceso de lectura, "w" para lectura/escritura de una base de datos ya existente, "c" para lectura/escritura y creacion de una base datos si esta no existe, y "n" para crear, truncar y lectura/escritura.

handler es el nombre de el manejador (handler) que sera usado para el acceso al *path*. Es pasado como un parametro opcional a **dba_open()** y puede usarse en lugar de ella.

dba_open() devuelve un valor positivo de handler o FALSE, en el caso de que la apertura de la base de datos se realice o si falla, respectivamente.

Ver tambien: dba_popen() dba_close()

dba_optimize (PHP 3>= 3.0.8, PHP 4)

Optimiza la base de datos

bool **dba_optimize** (int handle) \linebreak

dba_optimize() optimiza la base de datos especificada por *handle*.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_optimize() devuelve TRUE o FALSE, si la optimizacion tiene exito o falla, respectivamente.

Ver tambien: dba_sync()

dba_popen (PHP 3>= 3.0.8, PHP 4)

Apertura persistente de una base de datos

int **dba_popen** (string path, string mode, string handler [, ...]) \linebreak

dba_popen() establece una instancia persistente para *path* con *mode* usando *handler*.

path normalmente es el "path" en el sistema de ficheros.

mode es "r" para acceso de lectura, "w" para lectura/escritura de una base de datos ya existente, "c" para lectura/escritura y creacion de una base datos si esta no existe, y "n" para crear, truncar y lectura/escritura.

handler es el nombre del manejador (handler) que sera usado para el acceso al *path*. Es pasado como un parametro opcional a **dba_popen()** y puede usarse en lugar de ella.

dba_popen() devuelve un valor positivo de handler o FALSE, en el caso de que la apertura de la base de datos se realice o si falla, respectivamente.

Ver tambien: dba_open() dba_close()

dba_replace (PHP 3>= 3.0.8, PHP 4)

Reemplaza o inserta una entrada

bool **dba_replace** (string key, string value, int handle) \linebreak

dba_replace() reemplaza o inserta la entrada descrita con *key* y *value* dentro de la base de datos especificada por *handle*.

key es la clave de la entrada a insertar.

value es el valor a ser insertado.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_replace() devuelve TRUE o FALSE, dependiendo de si tiene exito o falla respectivamente.

Ver tambien: dba_exists() dba_delete() dba_fetch() dba_insert()

dba_sync (PHP 3>= 3.0.8, PHP 4)

Sincroniza la base de datos

bool **dba_sync** (int handle) \linebreak

dba_sync() sincroniza la base de datos especificada por *handle*. Esto probablemente realice una escritura fisica en el disco, si es soportado.

handle es un manejador (handle) de la base de datos devuelto por dba_open().

dba_sync() devuelve `TRUE` o `FALSE`, si la sincronizacion tiene exito o falla, respectivamente.

Ver tambien: dba_optimize()

XVIII. Funciones de fecha y hora

checkdate (PHP 3, PHP 4)

valida una fecha u hora

int **checkdate** (int month, int day, int year) \linebreak

Devuelve un valor verdadero si la fecha dada es válida; en caso contrario, devuelve un valor falso. Comprueba la validez de la fecha formada por los argumentos. Se considera válida una fecha si:

- el año está entre 0 y 32767, ambos incluidos
- el mes está entre 1 y 12, ambos incluidos
- el día está en el rango permitido para el mes dado. Se tienen en consideración los años bisiestos.

date (PHP 3, PHP 4)

da formato a la fecha/hora local

string **date** (string format [, int timestamp]) \linebreak

Devuelve una cadena formateada de acuerdo con la cadena de formato dada, utilizando el valor de *timestamp* dado o la hora local actual si no hay parámetro.

Se reconocen los siguientes caracteres en la cadena de formato:

- a - "am" o "pm"
- A - "AM" o "PM"
- d - día del mes, dos dígitos con cero a la izquierda; es decir, de "01" a "31"
- D - día de la semana, en texto, con tres letras; por ejemplo, "Fri"
- F - mes, en texto, completo; por ejemplo, "January"
- h - hora, de "01" a "12"
- H - hora, de "00" a "23"
- g - hour, sin ceros, de "1" a "12"
- G - hour, sin ceros; de "0" a "23"
- i - minutos; de "00" a "59"
- j - día del mes sin cero inicial; de "1" a "31"
- l (’L’ minúscula) - día de la semana, en texto, completo; por ejemplo, "Friday"
- L - "1" or "0", según si el año es bisiesto o no
- m - mes; de "01" a "12"
- n - mes sin cero inicial; de "1" a "12"
- M - mes, en texto, 3 letras; por ejemplo, "Jan"

- s - segundos; de "00" a "59"
- S - sufijo ordinal en inglés, en texto, 2 caracteres; por ejemplo, "th", "nd"
- t - número de días del mes dado; de "28" a "31"
- U - segundos desde el valor de ’epoch’
- w - día de la semana, en número, de "0" (domingo) a "6" (sábado)
- Y - año, cuatro cifras; por ejemplo, "1999"
- y - año, dos cifras; por ejemplo, "99"
- z - día del año; de "0" a "365"
- Z - diferencia horaria en segundos (de "-43200" a "43200")

Los caracteres no reconocidos se imprimen tal cual. El formato "Z" siempre devuelve "0" en la función **gmdate()()**

Ejemplo 1. Ejemplo de date()

```
print (date("l dS of F Y h:i:s A"));
print ("July 1, 2000 is on a " . date("l", mktime(0,0,0,7,1,2000)));
```

Es posible usar **date()** y mktime() juntas para obtener fechas futuras o pasadas.

Ejemplo 2. Ejemplo de date() y mktime()

```
$tomorrow  = mktime(0,0,0,date("m")  ,date("d")+1,date("Y"));
$lastmonth = mktime(0,0,0,date("m")-1,date("d"),  date("Y"));
$nextyear  = mktime(0,0,0,date("m"),  date("d"),  date("Y")+1);
```

Para dar formato a fechas en otros idiomas, se deben usar las funciones setlocale() y strftime().

Ver también gmdate() y mktime().

getdate (PHP 3, PHP 4)

obtiene información de fecha y hora

array **getdate** (int timestamp) \linebreak

Devuelve un array asociativo que contiene la información de fecha del valor timestamp como los siguientes elementos:

- "seconds" - segundos
- "minutes" - minutos

- "hours" - horas
- "mday" - día del mes
- "wday" - día de la semana, en número
- "mon" - mes, en número
- "year" - año, en número
- "yday" - día del año, en número; por ejemplo, "299"
- "weekday" - día de la semana, en texto, completo; por ejemplo, "Friday"
- "month" - mes, en texto, completo; por ejemplo, "January"

gettimeofday (PHP 3>= 3.0.7, PHP 4)

obtiene la hora actual

array **gettimeofday** (void) \linebreak

Es un interfaz para gettimeofday(2). Devuelve un array asociativo que contiene los datos devueltos por esta llamada al sistema.

- "sec" - segundos
- "usec" - microsegundos
- "minuteswest" - minutos al oeste de Greenwich
- "dsttime" - tipo de corrección dst

gmdate (PHP 3, PHP 4)

da formato a una fecha/hora GMT/CUT

string **gmdate** (string format, int timestamp) \linebreak

Idéntica a la función **data()** excepto en que la hora devuelta es la de Greenwich (GMT). Por ejemplo, si se utiliza en Finlandia (GMT +0200), la primera línea del ejemplo devuelve "Jan 01 1998 00:00:00", mientras la segunda imprime "Dec 31 1997 22:00:00".

Ejemplo 1. Ejemplo de gmdate()

```
echo date( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
echo gmdate( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
```

Ver también date(), mktime() y gmmktime().

gmmktime (PHP 3, PHP 4)

obtiene el valor timestamp UNIX de una fecha GMT

int **gmmktime** (int hour, int minute, int second, int month, int day, int year [, int is_dst]) \linebreak

Idéntica a mktime(), excepto en que los parámetros representan una fecha GMT.

gmstrftime (PHP 3>= 3.0.12, PHP 4)

da formato a una fecha/hora GMT/CUT según las convenciones locales

string **gmstrftime** (string format, int timestamp) \linebreak

Se comporta como strftime(), excepto en que la hora devuelta es la de Greenwich (GMT). Por ejemplo, si se utiliza en la zona horaria EST (GMT -0500), la primera línea del ejemplo imprime "Dec 31 1998 20:00:00", mientras la segunda imprime "Jan 01 1999 01:00:00".

Ejemplo 1. Ejemplo de gmstrftime()

```
setlocale ('LC_TIME','en_US');
echo strftime ("%b %d %Y %H:%M:%S",mktime(20,0,0,12,31,98))."\n";
echo gmstrftime ("%b %d %Y %H:%M:%S",mktime(20,0,0,12,31,98))."\n";
```

Ver también strftime().

localtime (PHP 4)

Obtener la hora local

array **localtime** ([int muestra_de_tiempo [, bool es_asociativo]]) \linebreak

La función **localtime()** devuelve un vector idético al de la estructura devuelta en C por la llamada a la misma función. El primer parámetro que se le pasa a **localtime()** es el timestamp, una representació de una fecha/hora concretas. Si no se proporciona, se utilizará la hora actual. El segundo argumento de **localtime()** es *es_asociativo*. Si está a 0 o no es proporcionado, el vector se devuelve como un vector normal, indizado numéricamente. Si el argumento está a 1, el vector devuelto es un vector asociativo conteniendo los diferentes elementos de la estructura devuelta por C al llamar a la función localtime. Los nombres de las diferentes claves del vector asociativo se encuentran a continuación:

- "tm_sec" - segundos

- "tm_min" - minutos
- "tm_hour" - horas
- "tm_mday" - día del mes
- "tm_mon" - mes del año, empezando en 0 que es Enero
- "tm_year" - Años que hacen desde 1900
- "tm_wday" - Día de la semana
- "tm_yday" - Día del año
- "tm_isdst" - Si el cambio de hora para el ahorro energético tiene efecto o no

microtime (PHP 3, PHP 4)

devuelve el valor timestamp UNIX actual con microsegundos

string **microtime** (void) \linebreak

Devuelve la cadena "msec sec", donde sec es la hora actual en número de segundos desde el valor Unix Epoch (0:00:00 del 1 de enero de 1970, hora GMT), y msec es la parte de microsegundos. Esta función sólo está disponible en sistemas operativos con admiten la llamada al sistema gettimeofday().

Ver también time().

mktime (PHP 3, PHP 4)

obtiene el timestamp UNIX de una fecha

int **mktime** (int hour, int minute, int second, int month, int day, int year [, int is_dst]) \linebreak

Advertencia: Véase el extraño orden de los argumentos, que se diferencia del orden de argumentos en una llamada mktime() de UNIX y que no permite eliminar parámetros de derecha a izquierda (ver abajo). Es un error común mezclar estos valores en un script.

Devuelve el valor timestamp Unix correspondiente a los argumentos dados. El timestamp es un entero de tipo long que contiene el número de segundos entre el valor Unix Epoch (1 de enero de 1970) y la hora especificada.

Se pueden eliminar argumentos en orden de derecha a izquierda; en los argumentos omitidos se toma el valor de la fecha y hora locales.

`is_dst` puede ponerse a 1 si la hora corresponde a horario de verano, 0 si no, o -1 (valor por omisión) si no se sabe.

Nota: `is_dst` se añadió en la versión 3.0.10.

mktime() es útil para realizar cálculos y validaciones con fechas, ya que calcula automáticamente el valor correcto para una entrada fuera de rango. Por ejemplo, cada una de las líneas siguientes produce la cadena "Jan-01-1998".

Ejemplo 1. Ejemplo de mktime()

```
echo date( "M-d-Y", mktime(0,0,0,12,32,1997) );
echo date( "M-d-Y", mktime(0,0,0,13,1,1997) );
echo date( "M-d-Y", mktime(0,0,0,1,1,1998) );
```

El último día de cada mes se puede expresar como el día "0" del mes siguiente, no el día -1. Los dos ejemplos siguientes producen la cadena "The last day in Feb 2000 is: 29".

Ejemplo 2. El último día del próximo mes

```
$lastday=mktime(0,0,0,3,0,2000);
echo strftime("Last day in Feb 2000 is: %d",$lastday);

$lastday=mktime(0,0,0,4,-31,2000);
echo strftime("Last day in Feb 2000 is: %d",$lastday);
```

Ver también date() y time().

strftime (PHP 3, PHP 4)

da formato a la hora o fecha local de acuerdo con las convenciones locales

string **strftime** (string format, int timestamp) \linebreak

Devuelve una cadena formateada según la cadena de formato dada utilizando el valor *timestamp* o la hora local actual. Los nombres del mes y el día de la semana y otras cadenas dependientes del idioma respetan lo establecido con setlocale().

Se reconocen los siguientes especificadores de conversión en la cadena de formato:

- %a - nombre del día de la semana abreviado
- %A - nombre del día de la semana completo
- %b - nombre del mes abreviado
- %B - nombre del mes completo
- %c - representación de fecha y hora preferidas en el idioma actual
- %d - día del mes en número (de 00 a 31)

- %H - hora como un número de 00 a 23
- %I - hora como un número de 01 a 12
- %j - día del año como un número de 001 a 366
- %m - mes como un número de 01 a 12
- %M - minuto en número
- %p - ‘am’ o ‘pm’, según la hora dada, o las cadenas correspondientes en el idioma actual
- %S - segundos en número
- %U - número de la semana en el año, empezando con el primer domingo como el primer día de la primera semana
- %W - número de la semana en el año, empezando con el primer lunes como el primer día de la primera semana
- %w - día de la semana en número (el domingo es el 0)
- %x - representación preferida de la fecha sin la hora
- %X - representación preferida de la hora sin la fecha
- %y - año en número de 00 a 99
- %Y - año en número de cuatro cifras
- %Z - nombre o abreviatura de la zona horaria
- %% - carácter ‘%’

Ejemplo 1. Ejemplo de strftime()

```
setlocale ("LC_TIME", "C");
print(strftime("%A in Finnish is "));
setlocale ("LC_TIME", "fi_FI");
print(strftime("%A, in French "));
setlocale ("LC_TIME", "fr_CA");
print(strftime("%A and in German "));
setlocale ("LC_TIME", "de_DE");
print(strftime("%A.\n"));
```

Este ejemplo funciona si se tienen los respectivos ‘locales’ instalados en el sistema.

Ver también setlocale() y mktime().

strtotime (PHP 3>= 3.0.12, PHP 4)

Procesar cualquier descripción textual de fecha/hora en Inglés convirtiéndola en una timestamp de UNIX.

int **strtotime** (string hora [, int ahora]) \linebreak

La función espera que se le pase una cadena conteniendo una fecha en formato Inglés e intentará procesarla y convertirla a una timestamp (muestra de tiempo) de UNIX relativa a la timestamp proporcionada en *ahora*, o la hora actual si no se indica ninguna. Si falla, devolverá `-1`.

Dado que **strtotime()** obra de acuerdo con la sintaxis de fechas de GNU, puede echar un vistazo a la página del manual GNU titulada Date Input Formats (http://www.gnu.org/manual/tar-1.12/html_chapter/tar_7.html) (Formatos de entrada de fechas). La sintaxis descrita ahí es válida para el parátro *hora*.

Ejemplo 1. Ejemplos con strtotime()

```
echo strtotime ("now"), "\n";
echo strtotime ("10 September 2000"), "\n";
echo strtotime ("+1 day"), "\n";
echo strtotime ("+1 week"), "\n";
echo strtotime ("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime ("next Thursday"), "\n";
echo strtotime ("last Monday"), "\n";
```

Ejemplo 2. Comprobando si falla

```
$str = 'No v&aacute;lida';
if (($timestamp = strtotime($str)) === -1) {
    echo "La cadena ($str) no es v&aacute;lida.";
} else {
    echo "$str == ". date('l dS of F Y h:i:s A',$timestamp);
}
```

> **Nota:** El rango válido de una timestamp suele ser desde Fri, 13 Dec 1901 20:45:54 GMT (Viernes, 13 de diciembre) a Tue, 19 Jan 2038 03:14:07 GMT (Martes, 19 de enero). (Estas son las fechas que corresponden a los valores mínimo y máximo de un entero con signo de 32 bits.)

time (PHP 3, PHP 4)

devuelve el timestamp UNIX actual

int **time** (void) \linebreak

Devuelve la hora actual como número de segundos transcurridos desde las 00:00:00 del 1 de enero de 1970 GMT (Unix Epoch).

Ver también date().

XIX. Funciones para dBase

Estas funciones permiten el acceso a datos almacenados en formato dBase (dbf).

No hay soporte para índices o campos Memo. Tampoco hay soporte para bloqueo: si dos procesos concurrentes en el servidor modifican el mismo fichero dBase, probablemente se destruirán los datos.

A diferencia de las bases de datos SQL, las "bases de datos" dBase no pueden cambiar su definición. Una vez creado el fichero, la definición de la base de datos es fija. No hay índices que aceleren la búsqueda u organicen los datos de distinto modo. Los ficheros dBase son simples ficheros secuenciales con registros de longitud fija. Los nuevos registros se añaden al final del fichero y los registros borrados se conservan hasta que se llama a la función **dbase_pack()()**.

Se recomienda no utilizar ficheros dBase como bases de datos, sino elegir cualquier servidor SQL; MySQL o Postgres son opciones habituales con PHP. El soporte para dBase se proporciona para permitir importar y exportar datos a y desde la base de datos web, ya que este formato de ficheros es aceptado habitualmente por las hojas de datos y los organizadores de Windows. La importación y exportación de datos es lo único para lo que sirve el soporte dBase.

dbase_add_record (PHP 3, PHP 4)

añade un registro a un fichero dBase

bool **dbase_add_record** (int dbase_identifier, array record) \linebreak

Añade los datos de *record* a la base de datos. Si el número de elementos del registro proporcionado no es igual al número de campos de la base de datos, la operación fallará y la función devolverá FALSE.

dbase_close (PHP 3, PHP 4)

cierra un fichero dBase

bool **dbase_close** (int dbase_identifier) \linebreak

Cierra el fichero asociado con *dbase_identifier*.

dbase_create (PHP 3, PHP 4)

crea una base de datos dBase

int **dbase_create** (string filename, array fields) \linebreak

El parámetro *fields* es un array de arrays, cada uno de los cuales describe el formato de un campo de la base de datos. Cada campo consiste de un nombre, un carácter que indica el tipo de campo, una longitud, y una precisión.

Los tipos de campos disponibles son:

L

Lógico. No tienen longitud ni precisión.

M

Memo. (Sin soporte en PHP.) No tienen longitud ni precisión.

D

Fecha (almacenada como AAAAMMDD). No tienen longitud ni precisión.

N

Número. Tienen longitud y precisión (número de cifras tras el punto decimal).

C

Cadena.

Si la base de datos se crea con éxito, se devuelve un dbase_identifier; en caso contrario, devuelve `FALSE`.

Ejemplo 1. Crear un fichero dBase

```
// "database" name
$dbname = "/tmp/test.dbf";

// database "definition"
$def =
    array(
        array("date",     "D"),
        array("name",     "C",  50),
        array("age",      "N",   3, 0),
        array("email",    "C", 128),
        array("ismember", "L")
    );

// creation
if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
```

dbase_delete_record (PHP 3, PHP 4)

borra un registro del fichero dBase

bool **dbase_delete_record** (int dbase_identifier, int record) \linebreak

Marca el registro *record* para ser borrado del fichero de datos. Para eliminar realmente el registro del fichero, debe llamarse a la función dbase_pack().

dbase_get_record_with_names (PHP 3>= 3.0.4, PHP 4)

lee un registro de un fichero dBase como array asociativo

array **dbase_get_record_with_names** (int dbase_identifier, int record) \linebreak

Devuelve los datos del registro *record* en un array asociativo. El array incluye también un elemento con índice 'deleted' que vale 1 si el registro ha sido marcado para borrar (ver dbase_delete_record().

Cada campo se convierte al tipo PHP apropiado. (Las fechas se transforman en cadenas.)

dbase_get_record (PHP 3, PHP 4)

lee un registro de un fichero dBase

array **dbase_get_record** (int dbase_identifier, int record) \linebreak

Devuelve los datos del registro *record* en un array. El array se indexa a partir de 0, e incluye un elemento con el índice asociativo 'deleted', que vale 1 si el registro ha sido marcado para borrar (ver dbase_delete_record().

Cada campo se convierte al tipo PHP apropiado. (Las fechas se guardan como cadenas.)

dbase_numfields (PHP 3, PHP 4)

cuenta el número de campos en un fichero dBase

int **dbase_numfields** (int dbase_identifier) \linebreak

Devuelve el número de campos (columnas) en el fichero especificado. Los números de campo va de 0 a dbase_numfields($db)-1, mientras los números de registros van de 1 a dbase_numrecords($db).

Ejemplo 1. Uso de dbase_numfields()

```
$rec = dbase_get_record($db, $recno);
$nf  = dbase_numfields($db);
for ($i=0; $i < $nf; $i++) {
    print $rec[$i]."<br>\n";
}
```

dbase_numrecords (PHP 3, PHP 4)

cuenta el número de registros en un fichero dBase

int **dbase_numrecords** (int dbase_identifier) \linebreak

Devuelve el número de registros (filas) en el fichero especificado. Los números de registro van de 1 a dbase_numrecords($db), mientras los números de campo van de 0 a dbase_numfields($db)-1.

dbase_open (PHP 3, PHP 4)

abre un fichero dBase

int **dbase_open** (string filename, int flags) \linebreak

Los "flags" son los que utiliza la llamada al sistema open(). Normalmente, 0 significa sólo lectura, 1 sólo escritura y 2 lectura y escritura.

Devuelve un dbase_identifier del fichero abierto, o `FALSE` si no pudo abrirse el fichero.

dbase_pack (PHP 3, PHP 4)

"empaqueta" un fichero dBase

bool **dbase_pack** (int dbase_identifier) \linebreak

Empaqueta el fichero especificado, borrando definitivamente todos los registros marcados con la función dbase_delete_record().

dbase_replace_record (PHP 3>= 3.0.11, PHP 4)

reemplaza un registro en un fichero dBase

bool **dbase_replace_record** (int dbase_identifier, array record, int dbase_record_number) \linebreak

Reemplaza los datos asociados con el registro *`record_number`* con los datos de *`record`* en el fichero de datos. Si el número de elementos del registro proporcionado no es igual al número de campos de la base de datos, la operación fallará y la función devolverá `FALSE`.

`dbase_record_number` es un entero en el rango de 1 al número de registros en el fichero de datos (devuelto por la función dbase_numrecords()).

XX. Funciones dbm

Estas funcione le permiten almacenar registros en una base de datos estilo dbm. Este tipo de base de datos (soportadas por las librerías db y gdbm de Berkeley, así como por algunas librerías del sistema y por una librería incluída para acceso a archivos de texto) guarda pares clave/valor (en oposición a los registros completos soportados por las bases de datos relacionales).

Ejemplo 1. ejemplo de dbm

```
$dbm = dbmopen("vistoya", "w");
if (dbmexists($dbm, $idusuario)) {
  $visto_ya = dbmfetch($dbm, $idusuario);
} else {
  dbminsert($dbm, $idusuario, time());
}
do_stuff();
dbmreplace($dbm, $idusuario, time());
dbmclose($dbm);
```

dblist (PHP 3, PHP 4)

describe la librería compatible dbm que se está usando

string **dblist** (void) \linebreak

dbmclose (PHP 3, PHP 4)

cierra una base de datos dbm

bool **dbmclose** (int identif_dbm) \linebreak

Desbloquea y cierra la base de datos especificada.

dbmdelete (PHP 3, PHP 4)

borra el valor de una clave de una base de datos dbm

bool **dbmdelete** (int identif_dbm, string clave) \linebreak

Borra el valor para la *clave* en la base de datos.

Devuelve FALSE si la clave no existía en la base de datos.

dbmexists (PHP 3, PHP 4)

dice si existe un valor para una clave dada en la base de datos dbm

bool **dbmexists** (int identif_dbm, string clave) \linebreak

Devuelve TRUE si hay un valor asociado con la *clave*.

dbmfetch (PHP 3, PHP 4)

obtiene un valor para una clave desde la base de datos dbm

string **dbmfetch** (int identif_dbm, string clave) \linebreak

Devuelve el valor asociado con la *clave*.

dbmfirstkey (PHP 3, PHP 4)

obtiene la primera clave de una base de datos dbm

string **dbmfirstkey** (int identif_dbm) \linebreak

Devuelve la primera clave de la base de datos. Nótese que no se garantiza ningún orden en particular, pues la base de datos se crea utilizando una tabla hash, que no garantiza ordenación alguna.

dbminsert (PHP 3, PHP 4)

inserta un valor para una clave en la base de datos dbm

int **dbminsert** (int identif_dbm, string clave, string valor) \linebreak

Añade el valor a la base de datos con la clave especificada.

Devuelve -1 si la base de datos se abrío en modo sólo lectura, 0 si la inserción tuvo éxito y 1 si la clave ya existía (para sustituir el valor, utilice dbmreplace().)

dbmnextkey (PHP 3, PHP 4)

obtiene la siguiente clave de una base de datos dbm

string **dbmnextkey** (int identif_dbm, string clave) \linebreak

Devuelve la clave que sigue a *clave*. Llamando a dbmfirstkey() seguida de llamadas sucesivas a **dbmnextkey()** se pueden visitar todos los pares clave/valor de la base de datos dbm. Por ejemplo:

Ejemplo 1. Visitanco cada par clave/valor en una base de datos dbm.

```
$clave = dbmfirstkey($id_dbm);
while ($clave) {
    echo "$clave = " . dbmfetch($id_dbm, $clave) . "\n";
    $clave = dbmnextkey($id_dbm, $clave);
}
```

dbmopen (PHP 3, PHP 4)

abre una base de datos dbm

int **dbmopen** (string fichero, string indicadores) \linebreak

El primer argumento es el nombre con sendero completo del archivo dbm que se va a abrir y el segundo es el modo de apertura, que puede ser "r", "n", "c" o "w", que significan sólo lectura, nuevo (implica lectura/escritura y suele truncar una base de datos si ya existía con ese nombre), crear (implica lectura/escritura, pero sin truncar la base de datos) y abrir para lectura/escritura, respectivamente.

Devuelve un identificador que se pasa al resto de funciones dbm si tiene éxito, o `FALSE` si falla.

Si se utiliza el soporte de ndbm, este creará los archivos fichero.dir y fichero.pag. gdbm sólo utiliza un archivo y lo mismo hace el soporte interno de archivos de texto, mientras que el db de Berkeley crea un archivo fichero.db. Nótese que el PHP hace su propio bloqueo de archivo sobre el que pudiera realizar la propia librería dbm. El PHP no borra los archivos .lck que crea. Los utiliza simplemente como i-nodos fijos en los que hacer el bloqueo. Para más información sobre archivos dbm, vea las páginas man de su Unix o obtenga el gdbm de GNU desde `ftp://prep.ai.mit.edu/pub/gnu`.

dbmreplace (PHP 3, PHP 4)

sustituye el valor de una clave en la base de datos dbm

bool **dbmreplace** (int identif_dbm, string clave, string valor) \linebreak

Sustituye el valor para la clave especificada de la base de datos.

También añadirá la clave a la base de datos si no existía antes.

XXI. dbx functions

Introducción

The dbx module is a database abstraction layer (db 'X', where 'X' is a supported database). The dbx functions allow you to access all supported databases using a single calling convention. The dbx-functions themselves do not interface directly to the databases, but interface to the modules that are used to support these databases.

Requerimientos

To be able to use a database with the dbx-module, the module must be either linked or loaded into PHP, and the database module must be supported by the dbx-module. Currently, following databases are supported, but others will follow (soon, I hope :-):

- FrontBase (available from PHP 4.1.0).
- Microsoft SQL Server
- MySQL
- ODBC
- PostgreSQL
- Sybase-CT (available from PHP 4.2.0).

Documentation for adding additional database support to dbx can be found at http://www.guidance.nl/php/dbx/doc/.

Instalación

In order to have these functions available, you must compile PHP with dbx support by using the `--enable-dbx` option and all options for the databases that will be used, e.g. for MySQL you must also specify `--with-mysql=[DIR]`. To get other supported databases to work with the dbx-module refer to

their specific documentation.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

There are two resource types used in the dbx module. The first one is the link-object for a database connection, the second a result-object which helds the result of a query.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión

ha sido o bien compilada dentro de PHP o grabada dinamicamente en tiempo de ejecución.

`DBX_MYSQL` (integer)

`DBX_ODBC` (integer)

`DBX_PGSQL` (integer)

`DBX_MSSQL` (integer)

`DBX_FBSQL` (integer)

`DBX_OCI8` (integer)

`DBX_SYBASECT` (integer)

`DBX_PERSISTENT` (integer)

`DBX_RESULT_INFO` (integer)

`DBX_RESULT_INDEX` (integer)

`DBX_RESULT_ASSOC` (integer)

`DBX_CMP_NATIVE` (integer)

`DBX_CMP_TEXT` (integer)

`DBX_CMP_NUMBER` (integer)

`DBX_CMP_ASC` (integer)

`DBX_CMP_DESC` (integer)

dbx_close (PHP 4 >= 4.0.6)

Close an open connection/database

bool **dbx_close** (object link_identifier) \linebreak

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

Ejemplo 1. dbx_close() example

```
<?php
$link = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Could not connect");

print("Connected successfully");
dbx_close($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

See also: dbx_connect().

dbx_compare (PHP 4 >= 4.1.0)

Compare two rows for sorting purposes

int **dbx_compare** (array row_a, array row_b, string column_key [, int flags]) \linebreak

dbx_compare() returns `0` if the `row_a[$column_key]` is equal to `row_b[$column_key]`, and `1` or `-1` if the former is greater or is smaller than the latter one, respectively, or vice versa if the *`flag`* is set to `DBX_CMP_DESC`. **dbx_compare()** is a helper function for dbx_sort() to ease the make and use of the custom sorting function.

The *`flags`* can be set to specify comparison direction:

- `DBX_CMP_ASC` - ascending order
- `DBX_CMP_DESC` - descending order

and the preferred comparison type:

- `DBX_CMP_NATIVE` - no type conversion
- `DBX_CMP_TEXT` - compare items as strings
- `DBX_CMP_NUMBER` - compare items numerically

One of the direction and one of the type constant can be combined with bitwise OR operator (|). The default value for the *flags* parameter is DBX_CMP_ASC | DBX_CMP_NATIVE.

Ejemplo 1. dbx_compare() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
    if ( !$rv ) {
        $rv = dbx_compare ($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect (DBX_ODBC, "", "db", "username", "password")
    or die ("Could not connect");

$result = dbx_query ($link, "SELECT id, parentid, description FROM table ORDER BY id");
    // data in $result is now ordered by id

dbx_sort ($result, "user_re_order");
    // date in $result is now ordered by parentid (descending), then by id

dbx_close ($link);
?>
```

See also dbx_sort().

dbx_connect (PHP 4 >= 4.0.6)

Open a connection/database

object **dbx_connect** (mixed module, string host, string database, string username, string password [, int persistent]) \linebreak

dbx_connect() returns an object on success, FALSE on error. If a connection has been made but the database could not be selected, the connection is closed and FALSE is returned. The *persistent* parameter can be set to DBX_PERSISTENT, if so, a persistent connection will be created.

The *module* parameter can be either a string or a constant, though the latter form is preferred. The possible values are given below, but keep in mind that they only work if the module is actually loaded.

- DBX_MYSQL or "mysql"
- DBX_ODBC or "odbc"
- DBX_PGSQL or "pgsql"

- `DBX_MSSQL` or "mssql"
- `DBX_FBSQL` or "fbsql" (available from PHP 4.1.0)
- `DBX_SYBASECT` or "sybase_ct" (available from PHP 4.2.0)

The *`host`*, *`database`*, *`username`* and *`password`* parameters are expected, but not always used depending on the connect functions for the abstracted module.

The returned `object` has three properties:

database

It is the name of the currently selected database.

handle

It is a valid handle for the connected database, and as such it can be used in module-specific functions (if required).

```
$link = dbx_connect (DBX_MYSQL, "localhost", "db", "username", "password");
mysql_close ($link->handle); // dbx_close($link) would be better here
```

module

It is used internally by dbx only, and is actually the module number mentioned above.

Ejemplo 1. dbx_connect() example

```
<?php
$link = dbx_connect (DBX_ODBC, "", "db", "username", "password", DBX_PERSISTENT)
    or die ("Could not connect");

print ("Connected successfully");
dbx_close ($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

See also: dbx_close().

dbx_error (PHP 4 >= 4.0.6)

Report the error message of the latest function call in the module (not just in the connection)

string **dbx_error** (object link_identifier) \linebreak

dbx_error() returns a string containing the error message from the last function call of the abstracted module (e.g. mysql module). If there are multiple connections in the same module, just the last error is given. If there are connections on different modules, the latest error is returned for the module specified by the *link_identifier* parameter.

Ejemplo 1. dbx_error() example

```
<?php
$link   = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Could not connect");

$result = dbx_query($link, "select id from non_existing_table");
if ( $result == 0 ) {
    echo dbx_error ($link);
}
dbx_close ($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

The error message for Microsoft SQL Server is actually the result of the mssql_get_last_message() function.

dbx_query (PHP 4 >= 4.0.6)

Send a query and fetch all results (if any)

object **dbx_query** (object link_identifier, string sql_statement [, long flags]) \linebreak

dbx_query() returns an object or `1` on success, and `0` on failure. The result object is returned only if the query given in *sql_statement* produces a result set.

Ejemplo 1. How to handle the returned value

```
<?php
$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("Could not connect");
```

```
$result = dbx_query($link, 'SELECT id, parentid, description FROM table');

if ( is_object($result) ) {
    // ... do some stuff here, see detailed examples below ...
    // first, print out field names and types
    // then, draw a table filled with the returned field values
}
else if ( $result == 1 ) {
    echo("Query executed successfully, but no result set returned");
}
else {
    exit("Query failed");
}

dbx_close($link);
?>
```

The *flags* parameter is used to control the amount of information that is returned. It may be any combination of the following constants with the bitwise OR operator (|):

`DBX_RESULT_INDEX`

It is *always* set, that is, the returned object has a data property which is a 2 dimensional array indexed numerically. For example, in the expression `data[2][3] 2` stands for the row (or record) number and `3` stands for the column (or field) number. The first row and column are indexed at 0.

If `DBX_RESULT_ASSOC` is also specified, the returning object contains the information related to `DBX_RESULT_INFO` too, even if it was not specified.

`DBX_RESULT_INFO`

It provides info about columns, such as field names and field types.

`DBX_RESULT_ASSOC`

It effects that the field values can be accessed with the respective column names used as keys to the returned object's data property.

Associated results are actually references to the numerically indexed data, so modifying `data[0][0]` causes that `data[0]['field_name_for_first_column']` is modified as well.

Note that `DBX_RESULT_INDEX` is always used, regardless of the actual value of *flags* parameter. This means that the following combinations is effective only:

- `DBX_RESULT_INDEX`
- `DBX_RESULT_INDEX | DBX_RESULT_INFO`

- `DBX_RESULT_INDEX` | `DBX_RESULT_INFO` | `DBX_RESULT_ASSOC` - this is the default, if *`flags`* is not specified.

The returing `object` has four or five properties depending on *`flags`*:

handle

It is a valid handle for the connected database, and as such it can be used in module specific functions (if required).

```
$result = dbx_query ($link, "SELECT id FROM table");
mysql_field_len ($result->handle, 0);
```

cols and rows

These contain the number of columns (or fields) and rows (or records) respectively.

```
$result = dbx_query ($link, 'SELECT id FROM table');
echo $result->rows; // number of records
echo $result->cols; // number of fields
```

info (optional)

It is returned only if either `DBX_RESULT_INFO` or `DBX_RESULT_ASSOC` is specified in the *`flags`* parameter. It is a 2 dimensional array, that has two named rows (`name` and `type`) to retrieve column information.

Ejemplo 2. lists each field's name and type

```
$result = dbx_query ($link, 'SELECT id FROM table',
                     DBX_RESULT_INDEX | DBX_RESULT_INFO);

for ($i = 0; $i < $result->cols; $i++ ) {
    echo $result->info['name'][$i] . "\n";
    echo $result->info['type'][$i] . "\n";
}
```

data

This property contains the actual resulting data, possibly associated with column names as well depending on *flags*. If DBX_RESULT_ASSOC is set, it is possible to use $result->data[2]["field_name"].

Ejemplo 3. outputs the content of data property into HTML table

```
$result = dbx_query ($link, 'SELECT id, parentid, description FROM table');

echo "<table>\n";
foreach ( $result->data as $row ) {
    echo "<tr>\n";
    foreach ( $row as $field ) {
        echo "<td>$field</td>";
    }
    echo "</tr>\n";
}
echo "</table>\n";
```

Nota: Always refer to the module-specific documentation as well.

See also: dbx_connect().

dbx_sort (PHP 4 >= 4.0.6)

Sort a result from a dbx_query by a custom sort function

bool **dbx_sort** (object result, string user_compare_function) \linebreak

Devuelve TRUE si todo fue bien, FALSE en caso de fallo.

Nota: It is always better to use ORDER BY SQL clause instead of **dbx_sort()**, if possible.

Ejemplo 1. dbx_sort() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
```

```
    if ( !$rv ) {
        $rv = dbx_compare ($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect (DBX_ODBC, "", "db", "username", "password")
    or die ("Could not connect");

$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
    // data in $result is now ordered by id

dbx_sort ($result, "user_re_order");
    // data in $result is now ordered by parentid (descending), then by id

dbx_close ($link);
?>
```

See also dbx_compare().

XXII. DB++ Functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Introducción

db++, made by the German company Concept asa (http://www.concept-asa.de/), is a relational database system with high performance and low memory and disk usage in mind. While providing SQL as an additional language interface, it is not really a SQL database in the first place but provides its own AQL query language which is much more influenced by the relational algebra then SQL is.

Concept asa always had an interest in supporting open source languages, db++ has had Perl and Tcl call interfaces for years now and uses Tcl as its internal stored procedure language.

Requerimientos

This extension relies on external client libraries so you have to have a db++ client installed on the system you want to use this extension on.

Concept asa (http://www.concept-asa.de/) provides db++ Demo versions (http://www.concept-asa.de/down-eng.html) and documentation (http://www.concept-asa.de/downloads/doc-eng.tar.gz) for Linux, some other UNIX versions. There is also a Windows version of db++, but this extension doesn't support it (yet).

Instalación

In order to build this extension yourself you need the db++ client libraries and header files to be installed on your system (these are included in the db++ installation archives by default). You have to run **configure** with option `--with-dbplus` to build this extension.

configure looks for the client libraries and header files under the default paths `/usr/dbplus`, `/usr/local/dbplus` and `/opt/dblus`. If you have installed db++ in a different place you have add

the installation path to the **configure** option like this: `--with-dbplus=/your/installation/path`.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

dbplus_relation

Most db++ functions operate on or return *`dbplus_relation`* resources. A *`dbplus_relation`* is a handle to a stored relation or a relation generated as the result of a query.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinamicamente en tiempo de ejecución.

db++ error codes

Tabla 1. DB++ Error Codes

PHP Constant	db++ constant	meaning
`DBPLUS_ERR_NOERR` (integer)	ERR_NOERR	Null error condition
`DBPLUS_ERR_DUPLICATE` (integer)	ERR_DUPLICATE	Tried to insert a duplicate tuple
`DBPLUS_ERR_EOSCAN` (integer)	ERR_EOSCAN	End of scan from rget()
`DBPLUS_ERR_EMPTY` (integer)	ERR_EMPTY	Relation is empty (server)
`DBPLUS_ERR_CLOSE` (integer)	ERR_CLOSE	The server can’t close
`DBPLUS_ERR_WLOCKED` (integer)	ERR_WLOCKED	The record is write locked
`DBPLUS_ERR_LOCKED` (integer)	ERR_LOCKED	Relation was already locked
`DBPLUS_ERR_NOLOCK` (integer)	ERR_NOLOCK	Relation cannot be locked
`DBPLUS_ERR_READ` (integer)	ERR_READ	Read error on relation
`DBPLUS_ERR_WRITE` (integer)	ERR_WRITE	Write error on relation
`DBPLUS_ERR_CREATE` (integer)	ERR_CREATE	Create() system call failed
`DBPLUS_ERR_LSEEK` (integer)	ERR_LSEEK	Lseek() system call failed

PHP Constant	db++ constant	meaning
DBPLUS_ERR_LENGTH (integer)	ERR_LENGTH	Tuple exceeds maximum length
DBPLUS_ERR_OPEN (integer)	ERR_OPEN	Open() system call failed
DBPLUS_ERR_WOPEN (integer)	ERR_WOPEN	Relation already opened for writing
DBPLUS_ERR_MAGIC (integer)	ERR_MAGIC	File is not a relation
DBPLUS_ERR_VERSION (integer)	ERR_VERSION	File is a very old relation
DBPLUS_ERR_PGSIZE (integer)	ERR_PGSIZE	Relation uses a different page size
DBPLUS_ERR_CRC (integer)	ERR_CRC	Invalid crc in the superpage
DBPLUS_ERR_PIPE (integer)	ERR_PIPE	Piped relation requires lseek()
DBPLUS_ERR_NIDX (integer)	ERR_NIDX	Too many secondary indices
DBPLUS_ERR_MALLOC (integer)	ERR_MALLOC	Malloc() call failed
DBPLUS_ERR_NUSERS (integer)	ERR_NUSERS	Error use of max users
DBPLUS_ERR_PREEXIT (integer)	ERR_PREEXIT	Caused by invalid usage
DBPLUS_ERR_ONTRAP (integer)	ERR_ONTRAP	Caused by a signal
DBPLUS_ERR_PREPROC (integer)	ERR_PREPROC	Error in the preprocessor
DBPLUS_ERR_DBPARSE (integer)	ERR_DBPARSE	Error in the parser
DBPLUS_ERR_DBRUNERR (integer)	ERR_DBRUNERR	Run error in db
DBPLUS_ERR_DBPREEXIT (integer)	ERR_DBPREEXIT	Exit condition caused by prexit() * procedure
DBPLUS_ERR_WAIT (integer)	ERR_WAIT	Wait a little (Simple only)
DBPLUS_ERR_CORRUPT_TUPLE (integer)	ERR_CORRUPT_TUPLE	A client sent a corrupt tuple
DBPLUS_ERR_WARNING0 (integer)	ERR_WARNING0	The Simple routines encountered a non fatal error which was corrected
DBPLUS_ERR_PANIC (integer)	ERR_PANIC	The server should not really die but after a disaster send ERR_PANIC to all its clients
DBPLUS_ERR_FIFO (integer)	ERR_FIFO	Can’t create a fifo
DBPLUS_ERR_PERM (integer)	ERR_PERM	Permission denied
DBPLUS_ERR_TCL (integer)	ERR_TCL	TCL_error
DBPLUS_ERR_RESTRICTED (integer)	ERR_RESTRICTED	Only two users
DBPLUS_ERR_USER (integer)	ERR_USER	An error in the use of the library by an application programmer

PHP Constant	db++ constant	meaning
`DBPLUS_ERR_UNKNOWN` (integer)	ERR_UNKNOWN	

dbplus_add (4.1.0 - 4.2.1 only)

Add a tuple to a relation

int **dbplus_add** (resource relation, array tuple) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

This function will add a tuple to a relation. The `tuple` data is an array of attribute/value pairs to be inserted into the given `relation`. After successful execution the `tuple` array will contain the complete data of the newly created tuple, including all implicitly set domain fields like sequences.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_aql (4.1.0 - 4.2.1 only)

Perform AQL query

resource **dbplus_aql** (string query [, string server [, string dbpath]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_aql() will execute an AQL `query` on the given `server` and `dbpath`.

On success it will return a relation handle. The result data may be fetched from this relation by calling dbplus_next() and **dbplus_current()**. Other relation access functions will not work on a result relation.

Further information on the AQL A... Query Language is provided in the original db++ manual.

dbplus_chdir (4.1.0 - 4.2.1 only)

Get/Set database virtual current directory

string **dbplus_chdir** ([string newdir]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_chdir() will change the virtual current directory where relation files will be looked for by dbplus_open(). **dbplus_chdir()** will return the absolute path of the current directory. Calling **dbplus_chdir()** without giving any `newdir` may be used to query the current working directory.

dbplus_close (4.1.0 - 4.2.1 only)

Close a relation

int **dbplus_close** (resource relation) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Calling **dbplus_close()** will close a relation previously opened by dbplus_open().

dbplus_curr (4.1.0 - 4.2.1 only)

Get current tuple from relation

int **dbplus_curr** (resource relation, array tuple) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_curr() will read the data for the current tuple for the given `relation` and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_errcode (4.1.0 - 4.2.1 only)

Get error string for given errorcode or last error

string **dbplus_errcode** (int errno) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_errcode() returns a cleartext error string for the error code passed as `errno` of for the result code of the last db++ operation if no parameter is given.

dbplus_errno (4.1.0 - 4.2.1 only)

Get error code for last operation

int **dbplus_errno** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_errno() will return the error code returned by the last db++ operation.

See also dbplus_errcode().

dbplus_find (4.1.0 - 4.2.1 only)

Set a constraint on a relation

int **dbplus_find** (resource relation, array constraints, mixed tuple) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_find() will place a constraint on the given relation. Further calls to functions like dbplus_curr() or dbplus_next() will only return tuples matching the given constraints.

Constraints are triplets of strings containing of a domain name, a comparison operator and a comparison value. The *constraints* parameter array may consist of a collection of string arrays, each of which contains a domain, an operator and a value, or of a single string array containing a multiple of three elements.

The comparison operator may be one of the following strings: '==', '>', '>=', '<', '<=', '!=', '~' for a regular expression match and 'BAND' or 'BOR' for bitwise operations.

See also dbplus_unselect().

dbplus_first (4.1.0 - 4.2.1 only)

Get first tuple from relation

int **dbplus_first** (resource relation, array tuple) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_curr() will read the data for the first tuple for the given *relation*, make it the current tuple and pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_curr(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_flush (4.1.0 - 4.2.1 only)

Flush all changes made on a relation

int **dbplus_flush** (resource relation) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_flush() will write all changes applied to *relation* since the last flush to disk.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_freealllocks (4.1.0 - 4.2.1 only)

Free all locks held by this client

int **dbplus_freealllocks** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_freeaalllocks() will free all tuple locks held by this client.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freerlocks().

dbplus_freelock (4.1.0 - 4.2.1 only)

Release write lock on tuple

int **dbplus_freelock** (resource relation, string tname) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_freelock() will release a write lock on the given *tuple* previously obtained by dbplus_getlock().

See also dbplus_getlock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_freerlocks (4.1.0 - 4.2.1 only)

Free all tuple locks on given relation

int **dbplus_freerlocks** (resource relation) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_freerlocks() will free all tuple locks held on the given `relation`.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freealllocks().

dbplus_getlock (4.1.0 - 4.2.1 only)

Get a write lock on a tuple

int **dbplus_getlock** (resource relation, string tname) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_getlock() will request a write lock on the specified `tuple`. It will return zero on success or a non-zero error code, especially DBPLUS_ERR_WLOCKED, on failure.

See also dbplus_freelock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_getunique (4.1.0 - 4.2.1 only)

Get a id number unique to a relation

int **dbplus_getunique** (resource relation, int uniqueid) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_getunique() will obtain a number guaranteed to be unique for the given `relation` and will pass it back in the variable given as `uniqueid`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_info (4.1.0 - 4.2.1 only)

???

int **dbplus_info** (resource relation, string key, array) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Not implemented yet.

dbplus_last (4.1.0 - 4.2.1 only)

Get last tuple from relation

int **dbplus_last** (resource relation, array tuple) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_curr() will read the data for the last tuple for the given `relation`, make it the current tuple and pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_next().

dbplus_lockrel (unknown)

Request write lock on relation

int **dbplus_lockrel** (resource relation) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_lockrel() will request a write lock on the given relation. Other clients may still query the relation, but can't alter it while it is locked.

dbplus_next (4.1.0 - 4.2.1 only)

Get next tuple from relation

int **dbplus_next** (resource relation, array) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_curr() will read the data for the next tuple for the given `relation`, will make it the current tuple and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_last().

dbplus_open (4.1.0 - 4.2.1 only)

Open relation file

resource **dbplus_open** (string name) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The relation file `name` will be opened. `name` can be either a file name or a relative or absolute path name. This will be mapped in any case to an absolute relation file path on a specific host machine and server.

On success a relation file resource (cursor) is returned which must be used in any subsequent commands referencing the relation. Failure leads to a zero return value, the actual error code may be asked for by calling dbplus_errno().

dbplus_prev (4.1.0 - 4.2.1 only)

Get previous tuple from relation

int **dbplus_prev** (resource relation, array tuple) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_curr() will read the data for the next tuple for the given `relation`, will make it the current tuple and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_next(), and dbplus_last().

dbplus_rchperm (4.1.0 - 4.2.1 only)

Change relation permissions

int **dbplus_rchperm** (resource relation, int mask, string user, string group) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_rchperm() will change access permissions as specified by `mask`, `user` and `group`. The values for these are operating system specific.

dbplus_rcreate (4.1.0 - 4.2.1 only)

Creates a new DB++ relation

resource **dbplus_rcreate** (string name, mixed domlist [, boolean overwrite]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_rcreate() will create a new relation named `name`. An existing relation by the same name will only be overwritten if the relation is currently not in use and `overwrite` is set to TRUE.

`domlist` should contain the domain specification for the new relation within an array of domain description strings. (**dbplus_rcreate()** will also accept a string with space delimited domain description strings, but it is recommended to use an array). A domain description string consists of a domain name unique to this relation, a slash and a type specification character. See the db++ documentation, especially the dbcreate(1) manpage, for a description of available type specifiers and their meanings.

dbplus_rcrtexact (4.1.0 - 4.2.1 only)

Creates an exact but empty copy of a relation including indices

resource **dbplus_rcrtexact** (string name, resource relation, boolean overwrite) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_rcrtexact() will create an exact but empty copy of the given `relation` under a new `name`. An existing relation by the same `name` will only be overwritten if `overwrite` is TRUE and no other process is currently using the relation.

dbplus_rcrtlike (4.1.0 - 4.2.1 only)

Creates an empty copy of a relation with default indices

resource **dbplus_rcrtlike** (string name, resource relation, int flag) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_rcrtexact() will create an empty copy of the given *relation* under a new *name*, but with default indices. An existing relation by the same *name* will only be overwritten if *overwrite* is TRUE and no other process is currently using the relation.

dbplus_resolve (4.1.0 - 4.2.1 only)

Resolve host information for relation

int **dbplus_resolve** (string relation_name) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_resolve() will try to resolve the given *relation_name* and find out internal server id, real hostname and the database path on this host. The function will return an array containing these values under the keys 'sid', 'host' and 'host_path' or FALSE on error.

See also dbplus_tcl().

dbplus_restorepos (4.1.0 - 4.2.1 only)

???

int **dbplus_restorepos** (resource relation, array tuple) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Not implemented yet.

dbplus_rkeys (4.1.0 - 4.2.1 only)

Specify new primary key for a relation

resource **dbplus_rkeys** (resource relation, mixed domlist) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_rkeys() will replace the current primary key for `relation` with the combination of domains specified by `domlist`.

`domlist` may be passed as a single domain name string or as an array of domain names.

dbplus_ropen (4.1.0 - 4.2.1 only)

Open relation file local

resource **dbplus_ropen** (string name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_ropen() will open the relation `file` locally for quick access without any client/server overhead. Access is read only and only **dbplus_current()** and dbplus_next() may be applied to the returned relation.

dbplus_rquery (4.1.0 - 4.2.1 only)

Perform local (raw) AQL query

int **dbplus_rquery** (string query, string dbpath) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_rquery() performs a local (raw) AQL query using an AQL interpreter embedded into the db++ client library. **dbplus_rquery()** is faster than dbplus_aql() but will work on local data only.

dbplus_rrename (4.1.0 - 4.2.1 only)

Rename a relation

int **dbplus_rrename** (resource relation, string name) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_rrename() will change the name of `relation` to `name`.

dbplus_rsecindex (4.1.0 - 4.2.1 only)

Create a new secondary index for a relation

resource **dbplus_rsecindex** (resource relation, mixed domlist, int type) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_rsecindex() will create a new secondary index for `relation` with consists of the domains specified by `domlist` and is of type `type`

`domlist` may be passed as a single domain name string or as an array of domain names.

dbplus_runlink (4.1.0 - 4.2.1 only)

Remove relation from filesystem

int **dbplus_runlink** (resource relation) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_unlink() will close and remove the `relation`.

dbplus_rzap (4.1.0 - 4.2.1 only)

Remove all tuples from relation

int **dbplus_rzap** (resource relation) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_rzap() will remove all tuples from `relation`.

dbplus_savepos (4.1.0 - 4.2.1 only)

???

int **dbplus_savepos** (resource relation) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Not implemented yet.

dbplus_setindex (4.1.0 - 4.2.1 only)

???

int **dbplus_setindex** (resource relation, string idx_name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Not implemented yet.

dbplus_setindexbynumber (4.1.0 - 4.2.1 only)

???

int **dbplus_setindexbynumber** (resource relation, int idx_number) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Not implemented yet.

dbplus_sql (4.1.0 - 4.2.1 only)

Perform SQL query

resource **dbplus_sql** (string query, string server, string dbpath) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Not implemented yet.

dbplus_tcl (4.1.0 - 4.2.1 only)

Execute TCL code on server side

int **dbplus_tcl** (int sid, string script) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

A db++ server will prepare a TCL interpreter for each client connection. This interpreter will enable the server to execute TCL code provided by the client as a sort of stored procedures to improve the performance of database operations by avoiding client/server data transfers and context switches.

dbplus_tcl() needs to pass the client connection id the TCL *`script`* code should be executed by. dbplus_resolve() will provide this connection id. The function will return whatever the TCL code returns or a TCL error message if the TCL code fails.

See also dbplus_resolve().

dbplus_tremove (4.1.0 - 4.2.1 only)

Remove tuple and return new current tuple

int **dbplus_tremove** (resource relation, array tuple [, array current]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_tremove() removes *`tuple`* from *`relation`* if it perfectly matches a tuple within the relation. *`current`*, if given, will contain the data of the new current tuple after calling **dbplus_tremove()**.

dbplus_undo (4.1.0 - 4.2.1 only)

???

int **dbplus_undo** (resource relation) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Not implemented yet.

dbplus_undoprepare (4.1.0 - 4.2.1 only)

???

int **dbplus_undoprepare** (resource relation) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Not implemented yet.

dbplus_unlockrel (4.1.0 - 4.2.1 only)

Give up write lock on relation

int **dbplus_unlockrel** (resource relation) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_unlockrel() will release a write lock previously obtained by dbplus_lockrel().

dbplus_unselect (4.1.0 - 4.2.1 only)

Remove a constraint from relation

int **dbplus_unselect** (resource relation) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Calling **dbplus_unselect()** will remove a constraint previously set by dbplus_find() on `relation`.

dbplus_update (4.1.0 - 4.2.1 only)

Update specified tuple in relation

int **dbplus_update** (resource relation, array old, array new) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_update() replaces the tuple given by `old` with the data from `new` if and only if `old` completely matches a tuple within `relation`.

dbplus_xlockrel (4.1.0 - 4.2.1 only)

Request exclusive lock on relation

int **dbplus_xlockrel** (resource relation) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_xlockrel() will request an exclusive lock on `relation` preventing even read access from other clients.

See also dbplus_xunlockrel().

dbplus_xunlockrel (4.1.0 - 4.2.1 only)

Free exclusive lock on relation

int **dbplus_xunlockrel** (resource relation) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dbplus_xunlockrel() will release an exclusive lock on `relation` previously obtained by dbplus_xlockrel().

XXIII. Direct IO functions

Introducción

PHP supports the direct io functions as described in the Posix Standard (Section 6) for performing I/O functions at a lower level than the C-Language stream I/O functions (fopen, fread,..).

Requerimientos

Estas funciones están disponibles como parte del módulo estandar, el cual está siempre disponible.

Installation

To get these functions to work, you have to configure PHP with `--enable-dio`.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

One resource type is defined by this extension: a file descriptor returnded by dio_open().

Constantes predefinidas

Esta extensión no define ninguna constante.

dio_close (PHP 4 >= 4.2.0)

Closes the file descriptor given by fd

void **dio_close** (resource fd) \linebreak

The function **dio_close()** closes the file descriptor *resource*.

dio_fcntl (PHP 4 >= 4.2.0)

Performs a c library fcntl on fd

mixed **dio_fcntl** (resource fd, int cmd [, mixed arg]) \linebreak

The **dio_fcntl()** function performs the operation specified by *cmd* on the file descriptor *fd*. Some commands require additional arguments *args* to be supplied.

arg is an associative array, when *cmd* is F_SETLK or F_SETLLW, with the following keys:

- "start" - offset where lock begins
- "length" - size of locked area. zero means to end of file
- "wenth" - Where l_start is relative to: can be SEEK_SET, SEEK_END and SEEK_CUR
- "type" - type of lock: can be F_RDLCK (read lock), F_WRLCK (write lock) or F_UNLCK (unlock)

cmd can be one of the following operations:

- F_SETLK - Lock is set or cleared. If the lock is held by someone else **dio_fcntl()** returns -1.
- F_SETLKW - like F_SETLK, but in case the lock is held by someone else, **dio_fcntl()** waits until the lock is released.
- F_GETLK - **dio_fcntl()** returns an associative array (as described above) if someone else prevents lock. If there is no obstruction key "type" will set to F_UNLCK.
- F_DUPFD - finds the lowest numbered available file descriptor greater or equal than *arg* and returns them.

dio_open (PHP 4 >= 4.2.0)

Opens a new filename with specified permissions of flags and creation permissions of mode

resource **dio_open** (string filename, int flags [, int mode]) \linebreak

dio_open() opens a file and returns a new file descriptor for it, or -1 if any error occurred. If *flags* is O_CREAT, optional third parameter *mode* will set the mode of the file (creation permissions). The *flags* parameter can be one of the following options:

- O_RDONLY - opens the file for read access
- O_WRONLY - opens the file for write access
- O_RDWR - opens the file for both reading and writing

The *flags* parameter can also include any combination of the following flags:

- O_CREAT - creates the file, if it doesn't already exist
- O_EXCL - if both, O_CREAT and O_EXCL are set, **dio_open()** fails, if file already exists
- O_TRUNC - if file exists, and its opened for write access, file will be truncated to zero length.
- O_APPEND - write operations write data at the end of file
- O_NONBLOCK - sets non blocking mode

dio_read (PHP 4 >= 4.2.0)

Reads n bytes from fd and returns them, if n is not specified, reads 1k block

string **dio_read** (resource fd [, int n]) \linebreak

The function **dio_read()** reads and returns *n* bytes from file with descriptor *resource*. If *n* is not specified, **dio_read()** reads 1K sized block and returns them.

dio_seek (PHP 4 >= 4.2.0)

Seeks to pos on fd from whence

int **dio_seek** (resource fd, int pos, int whence) \linebreak

The function **dio_seek()** is used to change the file position of the file with descriptor *resource*. The parameter *whence* specifies how the position *pos* should be interpreted:

- SEEK_SET - specifies that *pos* is specified from the beginning of the file
- SEEK_CUR - Specifies that *pos* is a count of characters from the current file position. This count may be positive or negative
- SEEK_END - Specifies that *pos* is a count of characters from the end of the file. A negative count specifies a position within the current extent of the file; a positive count specifies a position past the current end. If you set the position past the current end, and actually write data, you will extend the file with zeros up to that position

dio_stat (PHP 4 >= 4.2.0)

Gets stat information about the file descriptor fd

array **dio_stat** (resource fd) \linebreak

Function **dio_stat()** returns information about the file with file descriptor *`fd`*. **dio_stat()** returns an associative array with the following keys:

- "device" - device
- "inode" - inode
- "mode" - mode
- "nlink" - number of hard links
- "uid" - user id
- "gid" - group id
- "device_type" - device type (if inode device)
- "size" - total size in bytes
- "blocksize" - blocksize
- "blocks" - number of blocks allocated
- "atime" - time of last access
- "mtime" - time of last modification
- "ctime" - time of last change

On error **dio_stat()** returns NULL.

dio_truncate (PHP 4 >= 4.2.0)

Truncates file descriptor fd to offset bytes

bool **dio_truncate** (resource fd, int offset) \linebreak

Function **dio_truncate()** causes the file referenced by *`fd`* to be truncated to at most *`offset`* bytes in size. If the file previously was larger than this size, the extra data is lost. If the file previously was shorter, it is unspecified whether the file is left unchanged or is extended. In the latter case the extended part reads as zero bytes. Returns 0 on success, otherwise -1.

dio_write (PHP 4 >= 4.2.0)

Writes data to fd with optional truncation at length

int **dio_write** (resource fd, string data [, int len]) \linebreak

The function **dio_write()** writes up to *len* bytes from *data* to file *fd*. If *len* is not specified, **dio_write()** writes all *data* to the specified file. **dio_write()** returns the number of bytes written to *fd*.

XXIV. Funciones con directorios

chdir (PHP 3, PHP 4)

cambia de directorio

int **chdir** (string directory) \linebreak

Cambia el directorio PHP actual a *directory*. Devuelve FALSE si no puede cambiar al directorio, TRUE si todo va bien.

chroot (PHP 4 >= 4.0.5)

change the root directory

bool **chroot** (string directory) \linebreak

Changes the root directory of the current process to *directory*. Returns FALSE if unable to change the root directory, TRUE otherwise.

Nota: It's not wise to use this function when running in a webserver environment, because it's not possible to reset the root directory to / again at the end of the request. This function will only function correct when running as CGI this way.

dir (PHP 3, PHP 4)

clase directorio

new **dir** (string directory) \linebreak

Un mecanismo semi-orientado a objetos para leer directorios. El parametro *directory* abre el directorio. Dos propiedades estan disponibles cuando el directorio ha sido abierto. La propiedad de manejo puede ser usada con otras funciones de directorios tal como readdir(), rewinddir() y closedir(). La propiedad de trayectoria (path) es fijada para encaminar el directorio que ha sido abierto. Tres metodos estan disponibles: leer, rebobinar y cerrar.

Ejemplo 1. dir() Ejemplo

```
$d = dir("/etc");
echo "Handle: ".$d->handle."<br>\n";
echo "Path: ".$d->path."<br>\n";
while($entry=$d->read()) {
echo $entry."<br>\n";
}
$d->close();
```

closedir (PHP 3, PHP 4)

cierra el manejador de directorios

void **closedir** (int dir_handle) \linebreak

Cierra la secuencia de directorio determinada por *dir_handle*. La secuencia debe de haber sido abierta previamente con opendir().

getcwd (PHP 4)

gets the current working directory

string **getcwd** (void) \linebreak

Returns the current working directory.

See also chdir().

opendir (PHP 3, PHP 4)

abre el manejador de directorios

int **opendir** (string path) \linebreak

Devuelve un manejador de directorio para ser usado con las llamadas closedir(), readdir() y rewinddir().

readdir (PHP 3, PHP 4)

lee las entradas del manejador de directorios

string **readdir** (int dir_handle) \linebreak

Devuelve el nombre del siguiente fichero en el directorio. Los nombres de ficheros no son devueltos en ningun orden especial .

Ejemplo 1. Listar todos los ficheros en un directorio

```
<?php
$handle=opendir('.');
echo "Directory handle: $handle\n";
```

```
echo "Files:\n";
while ($file = readdir($handle)) {
 echo "$file\n";
 }
 closedir($handle);
 ?>
```

Tener en cuenta que **readdir()** devolvera tambien . y .. Si no quereis estas entradas podeis borrarlas:

Ejemplo 2. Listar todos los ficheros en un directorio excepto . y ..

```
<?php
$handle=opendir('.');
while ($file = readdir($handle)) {
 if ($file != "." && $file != "..") {
   echo "$file\n";
}
}
closedir($handle);
?>
```

rewinddir (PHP 3, PHP 4)

rebobinar el manejador de directorios

void **rewinddir** (int dir_handle) \linebreak

Inizializa la secuencia de directorio determinada por *dir_handle* al principio del directorio.

XXV. Funciones de DOM XML

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Estas funciones son disponibles solamente si PHP fué configurado con `--with-dom=[DIR]`, usando al libreria de XML de GNOME. Usted va a necesitar como mínimo libxml-2.0.0 (la versión beta no trabajará). Estas functiones fueron añadidas en PHP4.

Este module define las siguientes constantes:

Tabla 1. Constantes de XML

Constante	Valor	Descripción
XML_ELEMENT_NODE	1	
XML_ATTRIBUTE_NODE	2	
XML_TEXT_NODE	3	
XML_CDATA_SECTION_NODE	4	
XML_ENTITY_REF_NODE	5	
XML_ENTITY_NODE	6	
XML_PI_NODE	7	
XML_COMMENT_NODE	8	
XML_DOCUMENT_NODE	9	
XML_DOCUMENT_TYPE_NODE	10	
XML_DOCUMENT_FRAG_NODE	11	
XML_NOTATION_NODE	12	
XML_GLOBAL_NAMESPACE	1	
XML_LOCAL_NAMESPACE	2	

Este modulo define un número de clases. Las funciones de DOM XML devuelven un árbol conteniendo la structura del documento XML, en el cual cada nodo es un objecto perteneciente a una de estas clases.

DomAttribute->name (unknown)

Returns name of attribute

bool **DomAttribute->name** (void) \linebreak

This function returns the name of the attribute.

See also DomAttribute_value().

DomAttribute->specified (unknown)

Checks if attribute is specified

bool **DomAttribute->specified** (void) \linebreak

Check DOM standard for a detailed explanation.

DomAttribute->value (unknown)

Returns value of attribute

bool **DomAttribute->value** (void) \linebreak

This function returns the value of the attribute.

See also DomAttribute_name().

DomDocument->add_root [deprecated] (unknown)

Adds a root node

resource **DomDocument->add_root** (string name) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Adds a root element node to a dom document and returns the new node. The element name is given in the passed parameter.

Ejemplo 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->add_root("HTML");
$head = $root->new_child("HEAD", "");
$head->new_child("TITLE", "Hier der Titel");
echo htmlentities($doc->dump_mem());
?>
```

DomDocument->create_attribute (unknown)

Create new attribute

object **DomDocument->create_attribute** (string name, string value) \linebreak

This function returns a new instance of class `DomAttribute`. The name of the attribute is the value of the first parameter. The value of the attribute is the value of the second parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_cdata_section(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_cdata_section (unknown)

Create new cdata node

string **DomDocument->create_cdata_section** (string content) \linebreak

This function returns a new instance of class `DomCData`. The content of the cdata is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_comment (unknown)

Create new comment node

object **DomDocument->create_comment** (string content) \linebreak

This function returns a new instance of class `DomComment`. The content of the comment is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_element (unknown)

Create new element node

object **DomDocument->create_element** (string name) \linebreak

This function returns a new instance of class `DomElement`. The tag name of the element is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), **DomDocument_create_text()**, DomDocument_create_comment(), DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_entity_reference (unknown)

object **DomDocument->create_entity_reference** (string content) \linebreak

This function returns a new instance of class `DomEntityReference`. The content of the entity reference is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_cdata_section(), DomDocument_create_processing_instruction(), DomDocument_create_attribute(), DomNode_insert_before().

DomDocument->create_processing_instruction (unknown)

Creates new PI node

string **DomDocument->create_processing_instruction** (string content) \linebreak

This function returns a new instance of class `DomCData`. The content of the pi is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_cdata_section(), DomDocument_create_attribute(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_text_node (unknown)

Create new text node

object **DomDocument->create_text_node** (string content) \linebreak

This function returns a new instance of class `DomText`. The content of the text is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), DomDocument_create_comment(), **DomDocument_create_text()**, DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->doctype (unknown)

Returns the document type

object **DomDocument->doctype** (void) \linebreak

This function returns an object of class `DomDocumentType`. In versions of PHP before 4.3 this has been the class `Dtd`, but the DOM Standard does not know such a class.

See also the methods of class `DomDocumentType`.

DomDocument->document_element (unknown)

Returns root element node

object **DomDocument->document_element** (void) \linebreak

This function returns the root element node of a document.

The following example returns just the element with name CHAPTER and prints it. The other node -- the comment -- is not returned.

Ejemplo 1. Retrieving root element

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
print_r($root);
?>
```

DomDocument->dump_file (unknown)

Dumps the internal XML tree back into a file

string **DomDocument->dump_file** (string filename [, bool compressionmode [, bool format]]) \linebreak

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The *`format`* specifies whether the output should be neatly formatted, or not. The first parameter specifies the name of the filename and the second parameter, whether it should be compressed or not.

Ejemplo 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
```

```
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
$doc->dump_file("/tmp/test.xml", false, true);
?>
```

See also DomDocument_dump_mem() DomDocument_html_dump_mem().

DomDocument->dump_mem (unknown)

Dumps the internal XML tree back into a string

string **DomDocument->dump_mem** ([bool format]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The *`format`* specifies whether the output should be neatly formatted, or not.

Ejemplo 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->dump_mem(true));
echo "</PRE>";
?>
```

Nota: The first parameter was added in PHP 4.3.0.

See also DomDocument_dump_file(), DomDocument_html_dump_mem().

DomDocument->get_element_by_id (unknown)

Searches for an element with a certain id

object **DomDocument->get_element_by_id** (string id) \linebreak

This function is similar to DomDocument_get_elements_by_tagname() but searches for an element with a given id. According to the DOM standard this requires a DTD which defines the attribute ID to be of type ID, though the current implementation simply does an xpath search for "//*[@ID = '%s']". This does not comply to the DOM standard which requires to return null if it is not known which attribute is of type id. This behaviour is likely to be fixed, so do not rely on the current behaviour.

See also DomDocument_get_elements_by_tagname()

DomDocument->get_elements_by_tagname (unknown)

array **DomDocument->get_elements_by_tagname** (string name) \linebreak

See also DomDocument_add_root()

DomDocument->html_dump_mem (unknown)

Dumps the internal XML tree back into a string as HTML

string **DomDocument->html_dump_mem** (void) \linebreak

Creates an HTML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below.

Ejemplo 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
```

```
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->html_dump_mem());
echo "</PRE>";
?>
```

See also DomDocument_dump_file(), DomDocument_html_dump_mem().

DomDocumentType->entities (unknown)

Returns list of entities

array **DomDocumentType->entities** (void) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomDocumentType->internal_subset (unknown)

Returns internal subset

bool **DomDocumentType->internal_subset** (void) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomDocumentType->name (unknown)

Returns name of document type

string **DomDocumentType->name** (void) \linebreak

This function returns the name of the document type.

DomDocumentType->notations (unknown)

Returns list of notations

array **DomDocumentType->notations** (void) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomDocumentType->public_id (unknown)

Returns public id of document type

string **DomDocumentType->public_id** (void) \linebreak

This function returns the public id of the document type.

The following example echos nothing.

Ejemplo 1. Retrieving the public id

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$doctype = $dom->doctype();
echo $doctype->public_id();
?>
```

DomDocumentType->system_id (unknown)

Returns system id of document type

string **DomDocumentType->system_id** (void) \linebreak

Returns the system id of the document type.

The following example echos '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'.

Ejemplo 1. Retrieving the system id

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$doctype = $dom->doctype();
echo $doctype->system_id();
?>
```

DomElement->get_attribute_node (unknown)

Returns value of attribute

object **DomElement->get_attribute_node (** object attr) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomElement->get_attribute (unknown)

Returns value of attribute

object **DomElement->get_attribute** (string name) \linebreak

Returns the attribute with name *name* of the current node.

See also DomElement_set_attribute()

DomElement->get_elements_by_tagname (unknown)

Adds new attribute

bool **DomElement->get_elements_by_tagname** (string name) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomElement->has_attribute (unknown)

Adds new attribute

bool **DomElement->has_attribute** (string name) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomElement->remove_attribute (unknown)

Adds new attribute

bool **DomElement->remove_attribute** (string name) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomElement->set_attribute (unknown)

Adds new attribute

bool **DomElement->set_attribute** (string name, string value) \linebreak

Sets an attribute with name *name* ot the given value. If the attribute does not exist, it will be created.

Ejemplo 1. Setting an attribute

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

See also DomElement_get_attribute()

DomElement->tagname (unknown)

Returns name of element

string **DomElement->tagname** (void) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomNode->append_child (unknown)

Adds new child at the end of the children

object **DomNode->append_child** (object newnode) \linebreak

This functions appends a child to an existing list of children or creates a new list of children. The child can be created with e.g. DomDocument_create_element(), **DomDocument_create_text()** etc. or simply by using any other node.

Before a new child is appended it is first duplicated. Therefore the new child is a completely new copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of a xml document. The return value is the appended child. If you plan to do further modifications on the appended child you must use the returned node.

The following example will add a new element node to a fresh document and sets the attribute "align" to "left".

Ejemplo 1. Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

The above example could also be written as the following:

Ejemplo 2. Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node->set_attribute("align", "left");
$newnode = $doc->append_child($node);
?>
```

A more comples example is the one below. It first searches for a certain element, duplicates it including its children and adds it as a sibling. Finally a new attribute is added to one of the children of the new sibling and the whole document is dumped.

Ejemplo 3. Adding a child

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$parent = $element->parent_node();
$newnode = $parent->append_child($element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

echo "<PRE>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</PRE>";
?>
```

The above example could also be done with DomNode_insert_before() instead of DomNode_append_child().

See also DomNode_insert_before().

DomNode->append_sibling (unknown)

Adds new sibling to a node

object **DomNode->append_sibling** (object newnode) \linebreak

This functions appends a sibling to an existing node. The child can be created with e.g. DomDocument_create_element(), **DomDocument_create_text()** etc. or simply by using any other node.

Before a new sibling is added it is first duplicated. Therefore the new child is a completely new copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of a xml document. The return value is the added sibling. If you plan to do further modifications on the added sibling you must use the returned node.

This function has been added to provide the behaviour of DomNode_append_child() as it works till PHP 4.2.

See also **DomNode_append_before()**.

DomNode->attributes (unknown)

Returns list of attributes

array **DomNode->attributes** (void) \linebreak

This function only returns an array of attributes if the node is of type XML_ELEMENT_NODE.

DomNode->child_nodes (unknown)

Returns children of node

array **DomNode->child_nodes** (void) \linebreak

Returns all children of the node.

See also DomNode_next_sibling(), DomNode_previous_sibling().

DomNode->clone_node (unknown)

Clones a node

object **DomNode->clone_node** (void) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomNode->dump_node (unknown)

Dumps a single node

string **DomNode->dump_node** (void) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

See also DomDocument_dump_mem().

DomNode->first_child (unknown)

Returns first child of node

bool **DomNode->first_child** (void) \linebreak

Returns the first child of the node.

See also DomNode_last_child(), DomNode_next_sibling(), DomNode_previous_sibling().

DomNode->get_content (unknown)

Gets content of node

string **DomNode->get_content** (void) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomNode->has_attributess (unknown)

Checks if node has attributes

bool **DomNode->has_attributes** (void) \linebreak

This function checks if the node has attributes.

See also DomNode_has_child_nodes().

DomNode->has_child_nodes (unknown)

Checks if node has children

bool **DomNode->has_child_nodes** (void) \linebreak

This function checks if the node has children.

See also DomNode_child_nodes().

DomNode->insert_before (unknown)

Inserts new node as child

object **DomNode->insert_before** (object newnode, object refnode) \linebreak

This function inserts the new node *`newnode`* right before the node *`refnode`*. The return value is the inserted node. If you plan to do further modifications on the appended child you must use the returned node.

DomNode_insert_before() is very similar to DomNode_append_child() as the following example shows which does the same as the example at DomNode_append_child().

Ejemplo 1. Adding a child

```
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$newnode = $element->insert_before($element, $element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

echo "<PRE>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</PRE>";
```

See also DomNode_append_child().

DomNode->is_blank_node (unknown)

Checks if node is blank

bool **DomNode->is_blank_node** (void) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomNode->last_child (unknown)

Returns last child of node

object **DomNode->last_child** (void) \linebreak

Returns the last child of the node.

See also DomNode_first_child(), DomNode_next_sibling(), DomNode_previous_sibling().

DomNode->next_sibling (unknown)

Returns the next sibling of node

object **DomNode->next_sibling** (void) \linebreak

This function returns the next sibling of the current node. If there is no next sibling it returns false. You can use this function to iterate over all children of a node as shown in the example.

Ejemplo 1. Iterate over children

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$child = $element->first_child();
```

```
while($child) {
   print_r($child);
   $child = $child->next_sibling();
}
?>
```

See also DomNode_previous_sibling().

DomNode->node_name (unknown)

Returns name of node

string **DomNode->node_name** (void) \linebreak

Returns name of the node. The name has different meanings for the different types of nodes as illustrated in the following table.

Tabla 1. Meaning of value

Type	Meaning
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	#cdata-section
DomComment	#comment
DomDocument	#document
DomDocumentType	document type name
DomElement	tag name
DomEntity	name of entity
DomEntityReference	name of entity reference
DomNotation	notation name
DomProcessingInstruction	target
DomText	#text

DomNode->node_type (unknown)

Returns type of node

int **DomNode->node_type** (void) \linebreak

Returns the type of the node. All possible types are listed in the table in the introduction.

DomNode->node_value (unknown)

Returns value of a node

string **DomNode->node_value** (void) \linebreak

Returns value of the node. The value has different meanings for the different types of nodes as illustrated in the following table.

Tabla 1. Meaning of value

Type	Meaning
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	content
DomComment	content of comment
DomDocument	null
DomDocumentType	null
DomElement	null
DomEntity	null
DomEntityReference	null
DomNotation	null
DomProcessingInstruction	entire content without target
DomText	content of text

DomNode->owner_document (unknown)

Returns the document this node belongs to

object **DomNode->owner_document** (void) \linebreak

This function returns the document the current node belongs to.

The following example will create two identical lists of children.

Ejemplo 1. Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->owner_document();
$children = $doc2->children();
print_r($children);
?>
```

See also DomNode_insert_before().

DomNode->parent_node (unknown)

Returns the parent of the node

object **DomNode->parent_node** (void) \linebreak

This function returns the parent node.

The following example will show two identical lists of children.

Ejemplo 1. Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->parent_node();
$children = $doc2->children();
print_r($children);
?>
```

DomNode->prefix (unknown)

Returns name space prefix of node

string **DomNode->prefix** (void) \linebreak

Returns the name space prefix of the node.

DomNode->previous_sibling (unknown)

Returns the previous sibling of node

object **DomNode->previous_sibling** (void) \linebreak

This function returns the previous sibling of the current node.

See also DomNode_next_sibling().

DomNode->remove_child (unknown)

Removes child from list of children

object **DomNode->remove_child** (object oldchild) \linebreak

This functions removes a child from a list of children. If child cannot be removed or is not a child the function will return false. If the child could be removed the functions returns the old child.

Ejemplo 1. Removing a child

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$children = $element->child_nodes();
$child = $element->remove_child($children[0]);

echo "<PRE>";
$xmlfile = $dom->dump_mem(true);
echo htmlentities($xmlfile);
echo "</PRE>";
```

```
?>
```

See also DomNode_append_child().

DomNode->replace_child (unknown)

Replaces a child

object **DomNode->replace_child** (object oldnode, object newnode) \linebreak

This function replaces the child *oldnode* with the passed new node. If the new node is already a child it will not be added a second time. If the old node cannot be found the function returns false. If the replacement succeds the old node is returned.

See also DomNode_append_child()

DomNode->replace_node (unknown)

Replaces node

object **DomNode->replace_node** (object newnode) \linebreak

This function replaces an existing node with the passed new node. Before the replacement *newnode* is copied if it has a parent to make sure a node which is already in the document will not be inserted a second time. This behaviour enforces doing all modifications on the node before the replacement or to refetch the inserted node afterwards with functions like DomNode_first_child(), DomNode_child_nodes() etc..

See also DomNode_append_child()

DomNode->set_content (unknown)

Sets content of node

bool **DomNode->set_content** (void) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomNode->set_name (unknown)

Sets name of node

bool **DomNode->set_name** (void) \linebreak

Sets name of node.

See also DomNode_node_name().

DomNode->unlink_node (unknown)

Deletes node

object **DomNode->unlink_node** (void) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomProcessingInstruction->data (unknown)

Returns data of pi node

string **DomProcessingInstruction->data** (void) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

DomProcessingInstruction->target (unknown)

Returns target of pi node

string **DomProcessingInstruction->target** (void) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

domxml_new_doc (PHP 4 >= 4.2.1)

Creates new empty XML document

object **domxml_new_doc** (string version) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Creates a new dom document from scratch and returns it.

See also DomDocument_add_root()

domxml_open_file (PHP 4 >= 4.2.1)

Creates a DOM object from XML file

object **domxml_open_file** (string filename) \linebreak

The function parses the XML document in the file named *`filename`* and returns an object of class "Dom document", having the properties as listed above. The file is accessed read-only.

Ejemplo 1. Opening a xml document from a file

```
<?php

if(!$dom = domxml_open_file("example.xml")) {
```

```
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
?>
```

See also domxml_open_mem(), domxml_new_doc().

domxml_open_mem (PHP 4 >= 4.2.1)

Creates a DOM object of an XML document

object **domxml_open_mem** (string str) \linebreak

The function parses the XML document in *str* and returns an object of class "Dom document", having the properties as listed above. This function, domxml_open_file() or domxml_new_doc() must be called before any other function calls.

Ejemplo 1. Opening a xml document in a string

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
?>
```

See also domxml_open_file(), domxml_new_doc().

domxml_version (PHP 4 >= 4.1.0)

Get XML library version

string **domxml_version** (void) \linebreak

This function returns the version of the XML library version currently used.

domxml_xmltree (PHP 4 >= 4.2.1)

Creates a tree of PHP objects from an XML document

object **domxml_xmltree** (string str) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The function parses the XML document in `str` and returns a tree PHP objects as the parsed document. This function is isolated from the other functions, which means you cannot access the tree with any of the other functions. Modifying it, for example by adding nodes, makes no sense since there is currently no way to dump it as an XML file. However this function may be valuable if you want to read a file and investigate the content.

xpath_eval_expression (PHP 4 >= 4.0.4)

Evaluates the XPath Location Path in the given string

array **xpath_eval_expression** (object xpath_context) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

See also xpath_eval()

xpath_eval (PHP 4 >= 4.0.4)

Evaluates the XPath Location Path in the given string

array **xpath_eval** (object xpath context) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

See also xpath_new_context()

xpath_new_context (PHP 4 >= 4.0.4)

Creates new xpath context

object **xpath_new_context** (object dom document) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

See also xpath_eval()

xptr_eval (PHP 4 >= 4.0.4)

Evaluate the XPtr Location Path in the given string

int **xptr_eval** ([object xpath_context, string eval_str]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xptr_new_context (PHP 4 >= 4.0.4)

Create new XPath Context

string **xptr_new_context** ([object doc_handle]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

XXVI. .NET functions

Introducción

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

dotnet_load (unknown)

Loads a DOTNET module

int **dotnet_load** (string assembly_name [, string datatype_name [, int codepage]]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

XXVII. Error Handling and Logging Functions

These are functions dealing with error handling and logging. They allow you to define your own error handling rules, as well as modify the way the errors can be logged. This allows you to change and enhance error reporting to suit your needs.

With the logging functions, you can send messages directly to other machines, to an email (or email to pager gateway!), to system logs, etc., so you can selectively log and monitor the most important parts of your applications and websites.

The error reporting functions allow you to customize what level and kind of error feedback is given, ranging from simple notices to customized functions returned during errors.

error_log (PHP 3, PHP 4)

envía un mensaje de error a algún lugar

int **error_log** (string message, int message_type [, string destination [, string extra_headers]]) \linebreak

Envía un mensaje de error al log de errores del servidor web, a un puerto TCP o a un fichero. El primer parámetro, *message* (mensaje), es el mensaje de error que debe ser registrado. El segundo parámetro, *message_type* (tipo de mensaje) indica el lugar al que debe dirigirse:

Tabla 1. error_log() tipos de log

0	*message* es enviado al registro de sistema de PHP, utilizando el mecanismo de registro de sistema del Sistema Operativo, o a un fichero, dependiendo del valor de la directiva de configuración error_log
1	*message* es enviado por correo electrónico a la dirección del parámetro *destination* (destino). Este es el único tipo de mensaje donde se utiliza el cuarto parámetro, *extra_headers*. Este tipo de mensaje utiliza la misma funcionalidad interna que mail() realiza.
2	*message* es enviado a través de la conexión de depuración de PHP. Esta opción está disponible sólo si la depuración remota ha sido activada. En este caso el parámetro *destination* especifica el nombre de host o dirección IP y, opcionalmente, el número de puerto del socket que recibe la información de depuración.
3	*message* es añadido al fichero *destination*.

Ejemplo 1. error_log() ejemplos

```
// Send notification through the server log if we can not
// connect to the database.
if (!Ora_Logon($username, $password)) {
    error_log("Oracle database not available!", 0);
}

// Notify administrator by email if we run out of FOO
if (!($foo = allocate_new_foo()) {
    error_log("Big trouble, we're all out of FOOs!", 1,
              "operator@mydomain.com");
}

// other ways of calling error_log():
error_log("You messed up!", 2, "127.0.0.1:7000");
```

```
error_log("You messed up!", 2, "loghost");
error_log("You messed up!", 3, "/var/tmp/my-errors.log");
```

error_reporting (PHP 3, PHP 4)

establece que errores PHP son registrados

int **error_reporting** ([int level]) \linebreak

Establece el nivel de registro de los errores PHP y devuelve el nivel anterior. El nivel de registro es una máscara de bits de los valores siguientes (siga los enlaces a los valores internos para obtener sus significados):

Tabla 1. error_reporting() valores de bit

valor	nombre interno
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING

restore_error_handler (PHP 4 >= 4.0.1)

Restores the previous error handler function

void **restore_error_handler** (void) \linebreak

Used after changing the error handler function using set_error_handler(), to revert to the previous error handler (which could be the built-in or a user defined function)

See also error_reporting(), set_error_handler(), trigger_error(), user_error()

set_error_handler (PHP 4 >= 4.0.1)

Sets a user-defined error handler function.

string **set_error_handler** (string error_handler) \linebreak

Sets a user function (*error_handler*) to handle errors in a script. Returns the previously defined error handler (if any), or FALSE on error. This function can be used for defining your own way of handling errors during runtime, for example in applications in which you need to do cleanup of data/files when a critical error happens, or when you need to trigger an error under certain conditions (using trigger_error())

The user function needs to accept 2 parameters: the error code, and a string describing the error. The example below shows the handling of internal execptions by triggering errors and handling them with a user defined function:

Ejemplo 1. Error handling with set_error_handler() and trigger_error()

```
<?php

// redefine the user error constants - PHP4 only
define (FATAL,E_USER_ERROR);
define (ERROR,E_USER_WARNING);
define (WARNING,E_USER_NOTICE);

// set the error reporting level for this script
error_reporting  (FATAL + ERROR + WARNING);

// error handler function
function myErrorHandler ($errno, $errstr) {
    switch ($errno) {
    case FATAL:
    echo "<b>FATAL</b> [$errno] $errstr<br>\n";
    echo "  Fatal error in line ".__LINE__." of file ".__FILE__;
    echo ", PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
    echo "Aborting...<br>\n";
    exit -1;
    break;
    case ERROR:
    echo "<b>ERROR</b> [$errno] $errstr<br>\n";
    break;
    case WARNING:
    echo "<b>WARNING</b> [$errno] $errstr<br>\n";
    break;
    default:
    echo "Unkown error type: [$errno] $errstr<br>\n";
    break;
    }
}

// function to test the error handling
function scale_by_log ($vect, $scale) {
    if ( !is_numeric($scale) || $scale <= 0 )
    trigger_error("log(x) for x <= 0 is undefined, you used: scale = $scale",
      FATAL);
    if (!is_array($vect)) {
    trigger_error("Incorrect input vector, array of values expected", ERROR);
    return null;
    }
```

```
    for ($i=0; $i<count($vect); $i++) {
    if (!is_numeric($vect[$i]))
    trigger_error("Value at position $i is not a number, using 0 (zero)",
      WARNING);
    $temp[$i] = log($scale) * $vect[$i];
    }
    return $temp;
}

// set to the user defined error handler
$old_error_handler = set_error_handler("myErrorHandler");

// trigger some errors, first define a mixed array with a non-numeric item
echo "vector a\n";
$a = array(2,3,"foo",5.5,43.3,21.11);
print_r($a);

// now generate second array, generating a warning
echo "----\nvector b - a warning (b = log(PI) * a)\n";
$b = scale_by_log($a, M_PI);
print_r($b);

// this is trouble, we pass a string instead of an array
echo "----\nvector c - an error\n";
$c = scale_by_log("not array",2.3);
var_dump($c);

// this is a critical error, log of zero or negative number is undefined
echo "----\nvector d - fatal error\n";
$d = scale_by_log($a, -2.5);

?>
```

And when you run this sample script, the output will be

```
vector a
Array
(
    [0] => 2
    [1] => 3
    [2] => foo
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
----
vector b - a warning (b = log(PI) * a)
<b>WARNING</b> [1024] Value at position 2 is not a number, using 0 (zero)<br>
Array
(
    [0] => 2.2894597716988
```

```
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
)
----
vector c - an error
<b>ERROR</b> [512] Incorrect input vector, array of values expected<br>
NULL
----
vector d - fatal error
<b>FATAL</b> [256] log(x) for x <= 0 is undefined, you used: scale = -2.5<br>
  Fatal error in line 16 of file trigger_error.php, PHP 4.0.1pl2 (Linux)<br>
Aborting...<br>
```

See also error_reporting(), restore_error_handler(), trigger_error(), user_error()

trigger_error (PHP 4 >= 4.0.1)

Generates a user-level error/warning/notice message

void **trigger_error** (string error_msg [, int error_type]) \linebreak

Used to trigger a user error condition, it can be used by in conjunction with the built-in error handler, or with a user defined function that has been set as the new error handler (set_error_handler()). This function is useful when you need to generate a particular response to an exception at runtime. For example:

```
if (assert ($divisor == 0))
   trigger_error ("Cannot divide by zero", E_USER_ERROR);
```

Nota: See set_error_handler() for a more extensive example.

See also error_reporting(), set_error_handler(), restore_error_handler(), user_error()

user_error (PHP 4)

Generates a user-level error/warning/notice message

void **user_error** (string error_msg [, int error_type]) \linebreak

This is an alias for the function trigger_error().

See also error_reporting(), set_error_handler(), restore_error_handler(), and trigger_error()

XXVIII. FrontBase Functions

Introducción

These functions allow you to access FrontBase database servers. More information about FrontBase can be found at http://www.frontbase.com/.

Documentation for FrontBase can be found at http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation.

Frontbase support has been added to PHP 4.0.6.

Requerimientos

You must install the FrontBase database server or at least the fbsql client libraries to use this functions. You can get FrontBase from http://www.frontbase.com/.

Instalación

In order to have these functions available, you must compile PHP with fbsql support by using the `--with-fbsql` option. If you use this option without specifying the path to fbsql, PHP will search for the fbsql client libraries in the default installation location for the platform. Users who installed FrontBase in a non standard directory should always specify the path to fbsql: `--with-fbsql=/path/to/fbsql`. This will force PHP to use the client libraries installed by FrontBase, avoiding any conflicts.

Configuración en tiempo de ejecución

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión

ha sido o bien compilada dentro de PHP o grabada dinamicamente en tiempo de ejecución.

`FBSQL_ASSOC` (integer)

`FBSQL_NUM` (integer)

`FBSQL_BOTH` (integer)

`FBSQL_LOCK_DEFERRED` (integer)

`FBSQL_LOCK_OPTIMISTIC` (integer)

`FBSQL_LOCK_PESSIMISTIC` (integer)

`FBSQL_ISO_READ_UNCOMMITTED` (integer)

`FBSQL_ISO_READ_COMMITTED` (integer)

`FBSQL_ISO_REPEATABLE_READ` (integer)

`FBSQL_ISO_SERIALIZABLE` (integer)

`FBSQL_ISO_VERSIONED` (integer)

`FBSQL_UNKNOWN` (integer)

`FBSQL_STOPPED` (integer)

`FBSQL_STARTING` (integer)

`FBSQL_RUNNING` (integer)

`FBSQL_STOPPING` (integer)

`FBSQL_NOEXEC` (integer)

fbsql_affected_rows (PHP 4 >= 4.0.6)

Get number of affected rows in previous FrontBase operation

int **fbsql_affected_rows** ([resource link_identifier]) \linebreak

fbsql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with `link_identifier`. If the link identifier isn't specified, the last link opened by fbsql_connect() is assumed.

> **Nota:** If you are using transactions, you need to call **fbsql_affected_rows()** after your INSERT, UPDATE, or DELETE query, not after the commit.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

> **Nota:** When using UPDATE, FrontBase will not update columns where the new value is the same as the old value. This creates the possibility that **fbsql_affected_rows()** may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

If the last query failed, this function will return -1.

See also: fbsql_num_rows().

fbsql_autocommit (PHP 4 >= 4.0.6)

Enable or disable autocommit

bool **fbsql_autocommit** (resource link_identifier [, bool OnOff]) \linebreak

fbsql_autocommit() returns the current autocommit status. if the optional OnOff parameter is given the auto commit status will be changed. With OnOff set to `TRUE` each statement will be committed automatically, if no errors was found. With OnOff set to `FALSE` the user must commit or rollback the transaction using either fbsql_commit() or fbsql_rollback().

See also: fbsql_commit() and fbsql_rollback()

fbsql_change_user (unknown)

Change logged in user of the active connection

resource **fbsql_change_user** (string user, string password [, string database [, resource link_identifier]]) \linebreak

fbsql_change_user() changes the logged in user of the current active connection, or the connection given by the optional parameter link_identifier. If a database is specified, this will default or current database after the user has been changed. If the new user and password authorization fails, the current connected user stays active.

fbsql_close (PHP 4 >= 4.0.6)

Close FrontBase connection

boolean **fbsql_close** ([resource link_identifier]) \linebreak

Returns: `TRUE` on success, `FALSE` on error.

fbsql_close() closes the connection to the FrontBase server that's associated with the specified link identifier. If *`link_identifier`* isn't specified, the last opened link is used.

Using **fbsql_close()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

Ejemplo 1. fbsql_close() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);
?>
```

See also: fbsql_connect() and fbsql_pconnect().

fbsql_commit (PHP 4 >= 4.0.6)

Commits a transaction to the database

bool **fbsql_commit** ([resource link_identifier]) \linebreak

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

fbsql_commit() ends the current transaction by writing all inserts, updates and deletes to the disk and unlocking all row and table locks held by the transaction. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_rollback()

fbsql_connect (PHP 4 >= 4.0.6)

Open a connection to a FrontBase Server

resource **fbsql_connect** ([string hostname [, string username [, string password]]]) \linebreak

Returns a positive FrontBase link identifier on success, or an error message on failure.

fbsql_connect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *hostname* = 'NULL', *username* = '_SYSTEM' and *password* = empty password.

If a second call is made to **fbsql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling fbsql_close().

Ejemplo 1. fbsql_connect() example

```
<?php

    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);

?>
```

See also fbsql_pconnect() and fbsql_close().

fbsql_create_blob (PHP 4 >= 4.2.0)

Create a BLOB

string **fbsql_create_blob** (string blob_data [, resource link_identifier]) \linebreak

Returns: A resource handle to the newly created blob.

fbsql_create_blob() creates a blob from blob_data. The returned resource handle can be used with insert and update commands to store the blob in the database.

Ejemplo 1. fbsql_create_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
```

```
    $filename = "blobfile.bin";
    $fp = fopen($filename, "rb");
    $blobdata = fread($fp, filesize($filename));
    fclose($fp);

    $blobHandle = fbsql_create_blob($blobdata, $link);

    $sql = "INSINSERT INTO BLOB_TABLE (BLOB_COLUMN) VALUES ($blobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_clob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_clob (PHP 4 >= 4.2.0)

Create a CLOB

string **fbsql_create_clob** (string clob_data [, resource link_identifier]) \linebreak

Returns: A resource handle to the newly created CLOB.

fbsql_create_clob() creates a clob from clob_data. The returned resource handle can be used with insert and update commands to store the clob in the database.

Ejemplo 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $filename = "clob_file.txt";
    $fp = fopen($filename, "rb");
    $clobdata = fread($fp, filesize($filename));
    fclose($fp);

    $clobHandle = fbsql_create_clob($clobdata, $link);

    $sql = "INSERT INTO CLOB_TABLE (CLOB_COLUMN) VALUES ($clobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_db (PHP 4 >= 4.0.6)

Create a FrontBase database

bool **fbsql_create_db** (string database name [, resource link_identifier]) \linebreak

fbsql_create_db() attempts to create a new database on the server associated with the specified link identifier.

Ejemplo 1. fbsql_create_db() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    if (fbsql_create_db ("my_db")) {
        print("Database created successfully\n");
    } else {
        printf("Error creating database: %s\n", fbsql_error ());
    }
?>
```

See also: fbsql_drop_db().

fbsql_data_seek (PHP 4 >= 4.0.6)

Move internal result pointer

bool **fbsql_data_seek** (resource result_identifier, int row_number) \linebreak

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

fbsql_data_seek() moves the internal row pointer of the FrontBase result associated with the specified result identifier to point to the specified row number. The next call to fbsql_fetch_row() would return that row.

`Row_number` starts at 0.

Ejemplo 1. fbsql_data_seek() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");

    fbsql_select_db ("samp_db")
        or die ("Could not select database");
```

```
    $query = "SELECT last_name, first_name FROM friends;";
    $result = fbsql_query ($query)
        or die ("Query failed");

    # fetch rows in reverse order

    for ($i = fbsql_num_rows ($result) - 1; $i >=0; $i--) {
        if (!fbsql_data_seek ($result, $i)) {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }

        if(!($row = fbsql_fetch_object ($result)))
            continue;

        printf("%s %s<BR>\n", $row->last_name, $row->first_name);
    }

    fbsql_free_result ($result);
?>
```

fbsql_database_password (PHP 4 >= 4.0.6)

Sets or retrieves the password for a FrontBase database

string **fbsql_database_password** (resource link_identifier [, string database_password]) \linebreak

Returns: The database password associated with the link identifier.

fbsql_database_password() sets and retrieves the database password used by the connection. if a database is protected by a database password, the user must call this function before calling fbsql_select_db(). if the second optional parameter is given the function sets the database password for the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if fbsql_connect() was called, and use it.

This function does not change the database password in the database nor can it be used to retrive the database password for a database.

Ejemplo 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    fbsql_database_password($link, "secret db password");
    fbsql_select_db($database, $link);
?>
```

See also: fbsql_connect(), fbsql_pconnect() and fbsql_select_db().

fbsql_database (PHP 4 >= 4.0.6)

Get or set the database name used with a connection

string **fbsql_database** (resource link_identifier [, string database]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

fbsql_db_query (PHP 4 >= 4.0.6)

Send a FrontBase query

resource **fbsql_db_query** (string database, string query [, resource link_identifier]) \linebreak

Returns: A positive FrontBase result identifier to the query result, or `FALSE` on error.

fbsql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the FrontBase server and if no such link is found it'll try to create one as if fbsql_connect() was called with no arguments

See also fbsql_connect().

fbsql_db_status (PHP 4 >= 4.1.0)

Get the status for a given database

int **fbsql_db_status** (string database_name [, resource link_identifier]) \linebreak

Returns: An integer value with the current status.

fbsql_db_status() requests the current status of the database specified by *`database_name`*. If the *`link_identifier`* is omitted the default link_identifier will be used.

The return value can be one of the following constants:

- `FALSE` - The exec handler for the host was invalid. This error will occur when the link_identifier connects directly to a database by using a port number. FBExec can be available on the server but no connection has been made for it.
- FBSQL_UNKNOWN - The Status is unknown.
- FBSQL_STOPPED - The database is not running. Use fbsql_start_db() to start the database.
- FBSQL_STARTING - The database is starting.
- FBSQL_RUNNING - The database is running and can be used to perform SQL operations.
- FBSQL_STOPPING - The database is stopping.
- FBSQL_NOEXEC - FBExec is not running on the server and it is not possible to get the status of the database.

See also: fbsql_start_db() and fbsql_stop_db().

fbsql_drop_db (PHP 4 >= 4.0.6)

Drop (delete) a FrontBase database

bool **fbsql_drop_db** (string database_name [, resource link_identifier]) \linebreak

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

fbsql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

fbsql_errno (PHP 4 >= 4.0.6)

Returns the numerical value of the error message from previous FrontBase operation

int **fbsql_errno** ([resource link_identifier]) \linebreak

Returns the error number from the last fbsql function, or `0` (zero) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_errno()** to retrieve the error code. Note that this function only returns the error code from the most recently executed fbsql function (not including fbsql_error() and **fbsql_errno()**), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
```

```
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: fbsql_error() and fbsql_warnings().

fbsql_error (PHP 4 >= 4.0.6)

Returns the text of the error message from previous FrontBase operation

string **fbsql_error** ([resource link_identifier]) \linebreak

Returns the error text from the last fbsql function, or ″ (the empty string) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_error()** to retrieve the error text. Note that this function only returns the error text from the most recently executed fbsql function (not including **fbsql_error()** and fbsql_errno()), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: fbsql_errno() and fbsql_warnings().

fbsql_fetch_array (PHP 4 >= 4.0.6)

Fetch a result row as an associative array, a numeric array, or both

array **fbsql_fetch_array** (resource result [, int result_type]) \linebreak

Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

fbsql_fetch_array() is an extended version of fbsql_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must the numeric index of the column or make an alias for the column.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

An important thing to note is that using **fbsql_fetch_array()** is NOT significantly slower than using fbsql_fetch_row(), while it provides a significant added value.

The optional second argument *result_type* in **fbsql_fetch_array()** is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

For further details, see also fbsql_fetch_row() and fbsql_fetch_assoc().

Ejemplo 1. fbsql_fetch_array() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select user_id, fullname from table");
while ($row = fbsql_fetch_array ($result)) {
    echo "user_id: ".$row["user_id"]."<br>\n";
    echo "user_id: ".$row[0]."<br>\n";
    echo "fullname: ".$row["fullname"]."<br>\n";
    echo "fullname: ".$row[1]."<br>\n";
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_assoc (PHP 4 >= 4.0.6)

Fetch a result row as an associative array

array **fbsql_fetch_assoc** (resource result) \linebreak

Returns an associative array that corresponds to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_assoc() is equivalent to calling fbsql_fetch_array() with FBSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way fbsql_fetch_array() originally worked. If you need the numeric indices as well as the associative, use fbsql_fetch_array().

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use fbsql_fetch_array() and have it return the numeric indices as well.

An important thing to note is that using **fbsql_fetch_assoc()** is NOT significantly slower than using fbsql_fetch_row(), while it provides a significant added value.

For further details, see also fbsql_fetch_row() and fbsql_fetch_array().

Ejemplo 1. fbsql_fetch_assoc() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select * from table");
while ($row = fbsql_fetch_assoc ($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_field (PHP 4 >= 4.0.6)

Get column information from a result and return as an object

object **fbsql_fetch_field** (resource result [, int field_offset]) \linebreak

Returns an object containing field information.

fbsql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **fbsql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be `NULL`
- type - the type of the column

Ejemplo 1. fbsql_fetch_field() example

```
<?php
fbsql_connect ($host, $user, $password)
    or die ("Could not connect");
$result = fbsql_db_query ("database", "select * from table")
    or die ("Query failed");
# get column metadata
$i = 0;
while ($i < fbsql_num_fields ($result)) {
    echo "Information for column $i:<BR>\n";
    $meta = fbsql_fetch_field ($result);
    if (!$meta) {
        echo "No information available<BR>\n";
    }
    echo "<PRE>
max_length:   $meta->max_length
name:         $meta->name
not_null:     $meta->not_null
table:        $meta->table
type:         $meta->type
</PRE>";
    $i++;
}
fbsql_free_result ($result);
?>
```

See also fbsql_field_seek().

fbsql_fetch_lengths (PHP 4 >= 4.0.6)

Get the length of each output in a result

array **fbsql_fetch_lengths** ([resource result]) \linebreak

Returns: An array that corresponds to the lengths of each field in the last row fetched by fbsql_fetch_row(), or `FALSE` on error.

fbsql_fetch_lengths() stores the lengths of each result column in the last row returned by fbsql_fetch_row(), fbsql_fetch_array() and fbsql_fetch_object() in an array, starting at offset 0.

See also: fbsql_fetch_row().

fbsql_fetch_object (PHP 4 >= 4.0.6)

Fetch a result row as an object

object **fbsql_fetch_object** (resource result [, int result_type]) \linebreak

Returns an object with properties that correspond to the fetched row, or `FALSE` if there are no more rows.

fbsql_fetch_object() is similar to fbsql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *`result_type`* is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

Speed-wise, the function is identical to fbsql_fetch_array(), and almost as quick as fbsql_fetch_row() (the difference is insignificant).

Ejemplo 1. fbsql_fetch_object() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database", "select * from table");
while ($row = fbsql_fetch_object ($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
fbsql_free_result ($result);
?>
```

See also: fbsql_fetch_array() and fbsql_fetch_row().

fbsql_fetch_row (PHP 4 >= 4.0.6)

Get a result row as an enumerated array

array **fbsql_fetch_row** (resource result) \linebreak

Returns: An array that corresponds to the fetched row, or `FALSE` if there are no more rows.

fbsql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **fbsql_fetch_row()** would return the next row in the result set, or `FALSE` if there are no more rows.

See also: fbsql_fetch_array(), fbsql_fetch_object(), fbsql_data_seek(), fbsql_fetch_lengths(), and fbsql_result().

fbsql_field_flags (PHP 4 >= 4.0.6)

Get the flags associated with the specified field in a result

string **fbsql_field_flags** (resource result, int field_offset) \linebreak

fbsql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using explode().

fbsql_field_len (PHP 4 >= 4.0.6)

Returns the length of the specified field

int **fbsql_field_len** (resource result, int field_offset) \linebreak

fbsql_field_len() returns the length of the specified field.

fbsql_field_name (PHP 4 >= 4.0.6)

Get the name of the specified field in a result

string **fbsql_field_name** (resource result, int field_index) \linebreak

fbsql_field_name() returns the name of the specified field index. *`result`* must be a valid result identifier and *`field_index`* is the numerical offset of the field.

> **Nota:** *`field_index`* starts at 0.
>
> e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Ejemplo 1. fbsql_field_name() example

```
// The users table consists of three fields:
//   user_id
//   username
//   password.

$res = fbsql_db_query("users", "select * from users", $link);
```

```
echo fbsql_field_name($res, 0) . "\n";
echo fbsql_field_name($res, 2);
```

The above example would produce the following output:

```
user_id
password
```

fbsql_field_seek (PHP 4 >= 4.0.6)

Set result pointer to a specified field offset

bool **fbsql_field_seek** (resource result, int field_offset) \linebreak

Seeks to the specified field offset. If the next call to fbsql_fetch_field() doesn't include a field offset, the field offset specified in **fbsql_field_seek()** will be returned.

See also: fbsql_fetch_field().

fbsql_field_table (PHP 4 >= 4.0.6)

Get name of the table the specified field is in

string **fbsql_field_table** (resource result, int field_offset) \linebreak

Returns the name of the table that the specified field is in.

fbsql_field_type (PHP 4 >= 4.0.6)

Get the type of the specified field in a result

string **fbsql_field_type** (resource result, int field_offset) \linebreak

fbsql_field_type() is similar to the fbsql_field_name() function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as

detailed in the FrontBase documentation (http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation).

Ejemplo 1. fbsql_field_type() example

```
<?php

fbsql_connect ("localhost", "_SYSTEM", "");
fbsql_select_db ("wisconsin");
$result = fbsql_query ("SELECT * FROM onek;");
$fields = fbsql_num_fields ($result);
$rows   = fbsql_num_rows ($result);
$i = 0;
$table = fbsql_field_table ($result, $i);
echo "Your '".$table."' table has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = fbsql_field_type  ($result, $i);
    $name  = fbsql_field_name  ($result, $i);
    $len   = fbsql_field_len   ($result, $i);
    $flags = fbsql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
fbsql_close();

?>
```

fbsql_free_result (PHP 4 >= 4.0.6)

Free result memory

bool **fbsql_free_result** (resource result) \linebreak

fbsql_free_result() will free all memory associated with the result identifier *`result`*.

fbsql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

fbsql_get_autostart_info (PHP 4 >= 4.1.0)

No description given yet

array **fbsql_get_autostart_info** ([resource link_identifier]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

fbsql_hostname (PHP 4 >= 4.0.6)

Get or set the host name used with a connection

string **fbsql_hostname** (resource link_identifier [, string host_name]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

fbsql_insert_id (PHP 4 >= 4.0.6)

Get the id generated from the previous INSERT operation

int **fbsql_insert_id** ([resource link_identifier]) \linebreak

fbsql_insert_id() returns the ID generated for an column defined as DEFAULT UNIQUE by the previous INSERT query using the given *`link_identifier`*. If *`link_identifier`* isn't specified, the last opened link is assumed.

fbsql_insert_id() returns 0 if the previous query does not generate an DEFAULT UNIQUE value. If you need to save the value for later, be sure to call fbsql_insert_id() immediately after the query that generates the value.

> **Nota:** The value of the FrontBase SQL function `LAST_INSERT_ID()` always contains the most recently generated DEFAULT UNIQUE value, and is not reset between queries.

fbsql_list_dbs (PHP 4 >= 4.0.6)

List databases available on a FrontBase server

resource **fbsql_list_dbs** ([resource link_identifier]) \linebreak

fbsql_list_dbs() will return a result pointer containing the databases available from the current fbsql daemon. Use the fbsql_tablename() function to traverse this result pointer.

Ejemplo 1. fbsql_list_dbs() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');
$db_list = fbsql_list_dbs($link);

while ($row = fbsql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
```

The above example would produce the following output:

```
database1
database2
database3
...
```

Nota: The above code would just as easily work with fbsql_fetch_row() or other similar functions.

fbsql_list_fields (PHP 4 >= 4.0.6)

List FrontBase result fields

resource **fbsql_list_fields** (string database_name, string table_name [, resource link_identifier]) \linebreak

fbsql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with fbsql_field_flags(), fbsql_field_len(), fbsql_field_name(), and fbsql_field_type().

A result identifier is a positive integer. The function returns -1 if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@fbsql()` then this error string will also be printed out.

Ejemplo 1. fbsql_list_fields() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');

$fields = fbsql_list_fields("database1", "table1", $link);
$columns = fbsql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo fbsql_field_name($fields, $i) . "\n";;
}
```

The above example would produce the following output:

```
field1
field2
field3
...
```

fbsql_list_tables (PHP 4 >= 4.0.6)

List tables in a FrontBase database

resource **fbsql_list_tables** (string database [, resource link_identifier]) \linebreak

fbsql_list_tables() takes a database name and returns a result pointer much like the fbsql_db_query() function. The fbsql_tablename() function should be used to extract the actual table names from the result pointer.

fbsql_next_result (PHP 4 >= 4.0.6)

Move the internal result pointer to the next result

bool **fbsql_next_result** (resource result_id) \linebreak

When sending more than one SQL statement to the server or executing a stored procedure with multiple results will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the words from the new result set. The function will return `TRUE` if an additional result set was available or `FALSE` otherwise.

Ejemplo 1. fbsql_next_result() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret");
    fbsql_select_db("MyDB", $link);
    $SQL = "Select * from table1; select * from table2;";
    $rs = fbsql_query($SQL, $link);
    do {
        while ($row = fbsql_fetch_row($rs)) {
        }
    } while (fbsql_next_result($rs));
    fbsql_free_result($rs);
    fbsql_close ($link);
?>
```

fbsql_num_fields (PHP 4 >= 4.0.6)

Get number of fields in result

int **fbsql_num_fields** (resource result) \linebreak

fbsql_num_fields() returns the number of fields in a result set.

See also: fbsql_db_query(), fbsql_query(), fbsql_fetch_field(), and fbsql_num_rows().

fbsql_num_rows (PHP 4 >= 4.0.6)

Get number of rows in result

int **fbsql_num_rows** (resource result) \linebreak

fbsql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows returned from a INSERT, UPDATE or DELETE query, use fbsql_affected_rows().

Ejemplo 1. fbsql_num_rows() example

```
<?php

$link = fbsql_connect("localhost", "username", "password");
fbsql_select_db("database", $link);

$result = fbsql_query("SELECT * FROM table1;", $link);
$num_rows = fbsql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

See also: fbsql_affected_rows(), fbsql_connect(), fbsql_select_db(), and fbsql_query().

fbsql_password (PHP 4 >= 4.0.6)

Get or set the user password used with a connection

string **fbsql_password** (resource link_identifier [, string password]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

fbsql_pconnect (PHP 4 >= 4.0.6)

Open a persistent connection to a FrontBase Server

resource **fbsql_pconnect** ([string hostname [, string username [, string password]]]) \linebreak

Returns: A positive FrontBase persistent link identifier on success, or `FALSE` on error.

fbsql_pconnect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *`host`* = 'localhost', *`username`* = "_SYSTEM" and *`password`* = empty password.

fbsql_pconnect() acts very much like fbsql_connect() with two major differences.

To set Frontbase server port number, use fbsql_select_db().

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use.

This type of links is therefore called 'persistent'.

fbsql_query (PHP 4 >= 4.0.6)

Send a FrontBase query

resource **fbsql_query** (string query [, resource link_identifier]) \linebreak

fbsql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *`link_identifier`* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if fbsql_connect() was called with no arguments, and use it.

> **Nota:** The query string shall always end with a semicolon.

fbsql_query() returns `TRUE` (non-zero) or `FALSE` to indicate whether or not the query succeeded. A return value of `TRUE` means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **fbsql_query()** fails and returns `FALSE`:

Ejemplo 1. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

The following query is semantically invalid if `my_col` is not a column in the table `my_tbl`, so **fbsql_query()** fails and returns `FALSE`:

Ejemplo 2. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

fbsql_query() will also fail and return `FALSE` if you don't have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call fbsql_num_rows() to find out how many rows were returned for a SELECT statement or fbsql_affected_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

For SELECT statements, **fbsql_query()** returns a new result identifier that you can pass to fbsql_result(). When you are done with the result set, you can free the resources associated with it by calling fbsql_free_result(). Although, the memory will automatically be freed at the end of the script's execution.

See also: fbsql_affected_rows(), fbsql_db_query(), fbsql_free_result(), fbsql_result(), fbsql_select_db(), and fbsql_connect().

fbsql_read_blob (PHP 4 >= 4.2.0)

Read a BLOB from the database

string **fbsql_read_blob** (string blob_handle [, resource link_identifier]) \linebreak

Returns: A string containing the BLOB specified by blob_handle.

fbsql_read_blob() reads BLOB data from the database. If a select statement contains BLOB and/or BLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_blob()** to get the actual BLOB data from the database.

Ejemplo 1. fbsql_read_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT BLOB_COLUMN FROM BLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the blob data for teh first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
```

```
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the BLOB data in the first row
    $blob_data = fbsql_read_blob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), **fbsql_read_blob()**, fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_read_clob (PHP 4 >= 4.2.0)

Read a CLOB from the database

string **fbsql_read_clob** (string clob_handle [, resource link_identifier]) \linebreak

Returns: A string containing the CLOB specified by clob_handle.

fbsql_read_clob() reads CLOB data from the database. If a select statement contains BLOB and/or CLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_clob()** to get the actual CLOB data from the database.

Ejemplo 1. fbsql_read_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT CLOB_COLUMN FROM CLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the clob data for teh first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the CLOB data in the first row
    $clob_data = fbsql_read_clob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), **fbsql_read_clob()**, and fbsql_set_lob_mode().

fbsql_result (PHP 4 >= 4.0.6)

Get result data

mixed **fbsql_result** (resource result, int row [, mixed field]) \linebreak

fbsql_result() returns the contents of one cell from a FrontBase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tabledname.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **fbsql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **fbsql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: fbsql_fetch_row(), fbsql_fetch_array(), and fbsql_fetch_object().

fbsql_rollback (PHP 4 >= 4.0.6)

Rollback a transaction to the database

bool **fbsql_rollback** ([resource link_identifier]) \linebreak

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

fbsql_rollback() ends the current transaction by rolling back all statements issued since last commit. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_commit()

fbsql_select_db (PHP 4 >= 4.0.6)

Select a FrontBase database

bool **fbsql_select_db** (string database_name [, resource link_identifier]) \linebreak

Returns: `TRUE` on success, `FALSE` on error.

fbsql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if fbsql_connect() was called, and use it.

The client contacts FBExec to obtain the port number to use for the connection to the database. If the database name is a number the system will use that as a port number and it will not ask FBExec for the port number. The FrontBase server can be stared as FRontBase -FBExec=No -port=<port number> <database name>.

Every subsequent call to fbsql_query() will be made on the active database.

if the database is protected with a database password, the user must call fbsql_database_password() before selecting the database.

See also: fbsql_connect(), fbsql_pconnect(), fbsql_database_password() and fbsql_query().

fbsql_set_lob_mode (PHP 4 >= 4.2.0)

Set the LOB retrieve mode for a FrontBase result set

bool **fbsql_set_lob_mode** (resource result, string database_name) \linebreak

Returns: `TRUE` on success, `FALSE` on error.

fbsql_set_lob_mode() sets the mode for retrieving LOB data from the database. When BLOB and CLOB data is stored in FrontBase it can be stored direct or indirect. Direct stored LOB data will always be fetched no matter the setting of the lob mode. If the LOB data is less than 512 bytes it will always be stored directly.

- FBSQL_LOB_DIRECT - LOB data is retrieved directly. When data is fetched from the database with fbsql_fetch_row(), and other fetch functions, all CLOB and BLOB columns will be returned as ordinary columns. This is the default value on a new FrontBase result.
- FBSQL_LOB_HANDLE - LOB data is retrieved as handles to the data. When data is fetched from the database with **fbsql_fetch_row ()**, and other fetch functions, LOB data will be returned as a handle to the data if the data is stored indirect or the data if it is stored direct. If a handle is returned it will be a 27 byte string formatted as "@’000000000000000000000000’".

See also: fbsql_create_blob(), fbsql_create_clob(), fbsql_read_blob(), and fbsql_read_clob().

fbsql_set_transaction (PHP 4 >= 4.2.0)

Set the transaction locking and isolation

void **fbsql_set_transaction** (resource link_identifier, int Locking, int Isolation) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

fbsql_start_db (PHP 4 >= 4.0.6)

Start a database on local or remote server

bool **fbsql_start_db** (string database_name [, resource link_identifier]) \linebreak

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

fbsql_start_db()

See also: fbsql_db_status() and fbsql_stop_db().

fbsql_stop_db (PHP 4 >= 4.0.6)

Stop a database on local or remote server

bool **fbsql_stop_db** (string database_name [, resource link_identifier]) \linebreak

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

fbsql_stop_db()

See also: fbsql_db_status() and fbsql_start_db().

fbsql_tablename (PHP 4 >= 4.2.0)

Get table name of field

string **fbsql_tablename** (resource result, int i) \linebreak

fbsql_tablename() takes a result pointer returned by the fbsql_list_tables() function as well as an integer index and returns the name of a table. The fbsql_num_rows() function may be used to determine the number of tables in the result pointer.

Ejemplo 1. fbsql_tablename() example

```
<?php
fbsql_connect ("localhost", "_SYSTEM", "");
$result = fbsql_list_tables ("wisconsin");
$i = 0;
while ($i < fbsql_num_rows ($result)) {
    $tb_names[$i] = fbsql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

fbsql_username (PHP 4 >= 4.0.6)

Get or set the host user used with a connection

string **fbsql_username** (resource link_identifier [, string username]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

fbsql_warnings (PHP 4 >= 4.0.6)

Enable or disable FrontBase warnings

bool **fbsql_warnings** ([bool OnOff]) \linebreak

Returns `TRUE` if warnings is turned on otherwise `FALSE`.

fbsql_warnings() enables or disables FrontBase warnings.

XXIX. Funciones filePro

Estas funciones permiten acceso en modo de solo-lectura a datos guardados en bases de datos filePro.

filePro es una marca registrada de fP Technologies, Inc. Mas informacion sobre filePro puede encontrarse en http://www.fptech.com/.

filepro_fieldcount (PHP 3, PHP 4)

encuentra cuantos campos existen en una base de datos filePro

int **filepro_fieldcount** (void) \linebreak

Devuelve el numero de campos (columnas) existentes en la base de datos filePro abierta.

Ver tambien filepro().

filepro_fieldname (PHP 3, PHP 4)

obtiene el nombre de un campo

string **filepro_fieldname** (int field_number) \linebreak

Devuelve el nombre del campo correspondiente a *`field_number`*.

filepro_fieldtype (PHP 3, PHP 4)

obtiene el tipo de campo

string **filepro_fieldtype** (int field_number) \linebreak

Devuelve el tipo de campo del campo correspondiente a *`field_number`*.

filepro_fieldwidth (PHP 3, PHP 4)

obtiene la anchura de un campo

int **filepro_fieldwidth** (int field_number) \linebreak

Devuelve la anchura de el campo correspondiente a *`field_number`*.

filepro_retrieve (PHP 3, PHP 4)

extrae informacion de una base de datos filePro

string **filepro_retrieve** (int row_number, int field_number) \linebreak

Devuelve la informacion de la base de datos contenida en la localizacion especificada.

filepro_rowcount (PHP 3, PHP 4)

encuentra cuantas filas existen en una base de datos filePro

int **filepro_rowcount** (void) \linebreak

Devuelve el numero de filas (entradas) existentes en la base de datos filePro abierta.

Ver tambien filepro().

filepro (PHP 3, PHP 4)

lee y verifica el fichero de mapeo

bool **filepro** (string directory) \linebreak

Lee y verifica el fichero de mapeo, guardando la relacion de campos e informacion.

Ningun bloqueo es realizado, por ello, no se deberia modificar la base de datos filePro cuando puede ser abierta con PHP.

XXX. Funciones del sistema de ficheros

basename (PHP 3, PHP 4)

Devuelve la parte del path correspondiente al nombre del fichero

string **basename** (string path) \linebreak

Dada una cadena (string) que contiene el path de un fichero, esta función devuelve el nombre base del fichero.

En Windows, tanto la barra (/) como la barra inversa (\) pueden usarse como caracter separador en el path. En otros entornos, se usa la barra directa (/).

Ejemplo 1. Ejemplo de basename()

```
$path = "/home/httpd/html/index.php3";
$file = basename($path); // $file toma el valor "index.php3"
```

Ver también: dirname()

chgrp (PHP 3, PHP 4)

Cambia el grupo de un fichero

int **chgrp** (string filename, mixed group) \linebreak

Trata de cambiar el grupo al que pertenece el fichero filename al grupo group. Sólo el superusuario puede cambiar el grupo de un fichero arbitrariamente; otros usuarios pueden cambiar el grupo del fichero a cualquier grupo del cual el usuario sea miembro.

Devuelve `TRUE` en caso de éxito; en otro caso devuelve `FALSE`.

En Windows, no hace nada y devuelve `TRUE`.

Ver también chown() y chmod().

chmod (PHP 3, PHP 4)

Cambia permisos de un fichero

int **chmod** (string filename, int mode) \linebreak

Trata de cambiar los permisos del fichero especificado por *`filename`* a los permisos dados por *`mode`*.

Cuidado que *`mode`* no es asumido de forma automática como un valor octal. Para asegurar que se hace la operación esperada necesitas anteponer un cero (0) como prefijo del parámetro *`mode`*:

```
chmod( "/somedir/somefile", 755 );   // decimal; probablemente incorrecto
chmod( "/somedir/somefile", 0755 );  // octal; valor correcto de mode
```

Devuelve `TRUE` en caso de éxito y `FALSE` en otro caso.

Ver también chown() y chgrp().

chown (PHP 3, PHP 4)

Cambia el propietario de un fichero

int **chown** (string filename, mixed user) \linebreak

Trata de cambiar el propietario del fichero filename al usuario user. Sólo el superusuario puede cambiar el propietario de un fichero.

Devuelve `TRUE` en caso de éxito; en otro caso devuelve `FALSE`.

Nota: En Windows, no hace nada y devuelve `TRUE`.

Ver también **chown()** y chmod().

clearstatcache (PHP 3, PHP 4)

Limpia la cache de estado de un fichero

void **clearstatcache** (void) \linebreak

Invocar la llamada del sistema stat o lstat es bastante costoso en la mayoría de los sistemas. Por lo tanto, el resultado de la última llamada a cualquiera de las funciones de estado (listadas abajo) es guardado para usarlo en la próxima llamada de este tipo empleando el mismo nombre de fichero. Si deseas forzar un nuevo chequeo del estado del fichero, por ejemplo si el fichero está siendo chequeado muchas veces y puede cambiar o desaparecer, usa esta función para borrar los resultados almacenados en memoria de la última llamada.

Este valor sólo es cacheado durante el tiempo de vida de una petición simple.

Entre las funciones afectadas se incluyen stat(), lstat(), file_exists(), is_writeable(), is_readable(), is_executable(), is_file(), is_dir(), is_link(), filectime(), fileatime(), filemtime(), fileinode(), filegroup(), fileowner(), filesize(), filetype(), y fileperms().

copy (PHP 3, PHP 4)

Copia un fichero

int **copy** (string source, string dest) \linebreak

Hace una copia de un fichero. Devuelve `TRUE` si la copia tiene éxito, y `FALSE` en otro caso.

Ejemplo 1. Ejemplo de copy()

```
if (!copy($file, $file.'.bak')) {
    print("failed to copy $file...<br>\n");
}
```

Ver también: rename().

delete (unknown)

Una entrada manual inútil

void **delete** (string file) \linebreak

Esto es una entrada manual inútil para satisfacer a esas personas que están buscando unlink() o unset() en el lugar equivocado.

Ver también: unlink() para borrar ficheros, unset() para borrar variables.

dirname (PHP 3, PHP 4)

Devuelve la parte del path correspondiente al directorio

string **dirname** (string path) \linebreak

Dada una cadena (string) conteniendo el path a un fichero, esta función devolverá el nombre del directorio.

En Windows, tanto la barra (`/`) como la barra inversa (`\`) son usadas como separadores de caracteres. En otros entornos, debe usarse la barra directa (`/`).

Ejemplo 1. Ejemplo de dirname()

```
$path = "/etc/passwd";
$file = dirname($path); // $file toma el valor "/etc"
```

Ver también: basename()

disk_free_space (PHP 4 >= 4.1.0)

Returns available space in directory

float **disk_free_space** (string directory) \linebreak

Given a string containing a directory, this function will return the number of bytes available on the corresponding filesystem or disk partition.

Ejemplo 1. disk_free_space() example

```
$df = disk_free_space("/"); // $df contains the number of bytes
                            // available on "/"
```

disk_total_space (PHP 4 >= 4.1.0)

Returns the total size of a directory

float **disk_total_space** (string directory) \linebreak

Given a string containing a directory, this function will return the total number of bytes on the corresponding filesystem or disk partition.

Ejemplo 1. disk_total_space() example

```
$df = disk_total_space("/"); // $df contains the total number of
                             // bytes available on "/"
```

diskfreespace (PHP 3>= 3.0.7, PHP 4)

Devuelve el espacio disponible en un directorio

float **diskfreespace** (string directory) \linebreak

Dada una cadena (string) conteniendo el nombre de un directorio, esta función devolverá el número de bytes disponibles en el disco correspondiente.

Ejemplo 1. Ejemplo de diskfreespace()

```
$df = diskfreespace("/"); // $df contiene el numero de bytes
                          // disponibles en "/"
```

fclose (PHP 3, PHP 4)

Cierra el apuntador a un fichero abierto

int **fclose** (int fp) \linebreak

Se cierra el fichero apuntado por fp.

Devuelve `TRUE` en caso de éxito y `FALSE` en caso de fallo.

El apuntador al fichero debe ser válido y debe apuntarse a un fichero abierto con éxito con fopen() o con fsockopen().

feof (PHP 3, PHP 4)

Verifica si el apuntador a un fichero está al final del fichero (end-of-file)

int **feof** (int fp) \linebreak

Devuelve `TRUE` si el apuntador del fichero está en EOF o si ocurre un error; en otro caso devuelve `FALSE`.

The file pointer must be valid, and must point to a file El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito por fopen(), popen(), o fsockopen().

fflush (PHP 4 >= 4.0.1)

Flushes the output to a file

int **fflush** (int fp) \linebreak

This function forces a write of all buffered output to the to the resource pointed to by the file handle *`fp`*. Returns `TRUE` if successful, `FALSE` otherwise.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen().

fgetc (PHP 3, PHP 4)

Obtiene un caracter del fichero apuntado

string **fgetc** (int fp) \linebreak

Devuelve una cadena (string) conteniendo un simple caracter leido del fichero apuntado por fp. Devuelve `FALSE` para EOF (como hace feof()).

El apuntador al fichero debe ser valido, y debe apuntar a un fichero abierto con éxito por fopen(), popen(), o fsockopen().

Ver también fread(), fopen(), popen(), fsockopen(), y fgets().

fgetcsv (PHP 3>= 3.0.8, PHP 4)

Obtiene una línea del fichero apuntado y extrae los campos CSV

array **fgetcsv** (int fp, int length [, string delimiter]) \linebreak

Parecida a fgets() excepto que fgetcsv() parsea la línea que lee buscando campos en formato CSV y devuelve un array conteniendo los campos leidos. El delimitador de campo es una coma, a menos que se especifique otro delimitador con el tercer parámetro opcional.

fp debe ser un apuntador válido a un fichero abierto con éxito por fopen(), popen(), o fsockopen()

la longitud debe ser mayor que la línea más larga que pueda encontrarse en le fichero CSV (permitiendo arrastrar caracteres de fin de línea)

fgetcsv() devuelve `FALSE` en caso de error, incluyendo el fin de fichero.

NOTA: Una línea en blanco en un fichero CSV se devuelve como un array que contiene un único campo nulo, y esto no será tratado como un error.

Ejemplo 1. Ejemplo de fgetcsv() - Leer e imprimir el contenido completo de un fichero CSV

```
$row = 1;
$fp = fopen ("test.csv","r");
while ($data = fgetcsv ($fp, 1000, ",")) {
    $num = count ($data);
    print "<p> $num fields in line $row: <br>";
    $row++;
```

```
    for ($c=0; $c<$num; $c++) {
        print $data[$c] . "<br>";
    }
}
fclose ($fp);
```

fgets (PHP 3, PHP 4)

Obtiene una línea del fichero apuntado

string **fgets** (int fp, int length) \linebreak

Devuelve una cadena de como mucho length - 1 bytes leidos del fichero apuntado por fp. La lectura acaba cuando son leidos length - 1 bytes, cuando se llega a una nueva línea (el caracter de nueva línea se incluye en el valor devuelto), o cuando se llega a un EOF (lo que ocurra primero).

Si ocurre un error, devuelve `FALSE`.

Fallos Comunes:

Los que hayan usado la semantica de 'C' de la función fgets deben darse cuenta de la diferencia que hay en como el EOF es devuelto por esta función.

El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito con fopen(), popen(), o fsockopen().

A continuación un ejemplo sencillo:

Ejemplo 1. Leyendo un fichero línea por línea

```
$fd = fopen ("/tmp/inputfile.txt", "r");
while (!feof($fd)) {
    $buffer = fgets($fd, 4096);
    echo $buffer;
}
fclose ($fd);
```

Ver también fread(), fopen(), popen(), fgetc(), y fsockopen().

fgetss (PHP 3, PHP 4)

Obtiene una línea del fichero apuntado y quita las etiquetas HTML

string **fgetss** (int fp, int length [, string allowable_tags]) \linebreak

Idéntica a fgets(), excepto que fgetss trata de quitar cualquier etiqueta HTML y PHP del texto que lee.

Se puede utilizar el tercer parámetro opcional para especificar etiquetas que no deben de quitarse.

Nota: *allowable_tags* fue añadido en PHP 3.0.13, PHP4B3.

Ver también fgets(), fopen(), fsockopen(), popen(), y strip_tags().

file_exists (PHP 3, PHP 4)

Verifica si un fichero existe

int **file_exists** (string filename) \linebreak

Devuelve TRUE si el fichero especificado por *filename* existe; y FALSE en otro caso.

El resultado de esta función es cacheado. Ver clearstatcache() para más detalles.

file_get_contents (PHP 4 CVS only)

Reads entire file into a string

string **file_get_contents** (string filename [, int use_include_path]) \linebreak

Identical to readfile(), except that **file_get_contents()** returns the file in a string.

Nota: This function was introduced in PHP 4.3.0.

Nota: Esta función es segura bináriamente.

Sugerencia: Puede usar una URL como nombre de archivo con esta función si los "fopen wrappers" han sido activados. Consulte fopen() para más detalles.

See also: fgets(), file(), fread(), include(), and readfile().

file_get_wrapper_data (PHP 4 CVS only)

Retrieves header/meta data from "wrapped" file pointers

mixed **file_get_wrapper_data** (int fp) \linebreak

This function returns header or meta data from files opened with fopen(). This is useful to return the response headers for HTTP connections, or some other statistics for other resources.

The format of the returned data is deliberately undocumented at this time, and depends on which wrapper(s) were used to open the file.

Nota: This function was introduced in PHP 4.3.0.

file_register_wrapper (PHP 4 CVS only)

Register a URL wrapper implemented as a PHP class

boolean **file_register_wrapper** (string protocol, string classname) \linebreak

This function is currently only documented by the example below:

Ejemplo 1. Implementing a base64 encoding protocol

```
class Base64EncodingStream {
    var $fp = null;

    function stream_open($path, $mode, $options, &$opened_path)
    {
        $this->fp = fopen($path, $mode);
        return is_resource($this->fp);
    }
    function stream_close()
    {
        fclose($this->fp);
    }
    function stream_read($count)
    {
        return false; // We only allow writing
    }
    function stream_write($data)
    {
        return fwrite($this->fp, base64_encode($data));
    }
    function stream_flush()
    {
        fflush($this->fp);
        return true;
    }
    function stream_seek($offset, $whence)
    {
        return false;
```

```
    }
    function stream_gets()
    {
        return false;
    }
    function stream_tell()
    {
        return false;
    }
    function stream_eof()
    {
        return false;
    }
}
file_register_wrapper("base64", "Base64EncodingStream")
    or die("Failed to register protocol");

copy("/tmp/inputfile.txt", "base64:///tmp/outputfile.txt");
readfile("/tmp/outputfile");
```

file_register_wrapper() will return false if the *protocol* already has a handler, or if "fopen wrappers" are disabled.

> **Nota:** This function was introduced in PHP 4.3.0.

file (PHP 3, PHP 4)

lee un fichero completo hacia un array

array **file** (string filename [, int use_include_path]) \linebreak

Idéntica a readfile(), excepto que **file()** devuelve el fichero en un array. Cada elemento del array corresponde a una línea del fichero, con el caracter de nueva línea incluido.

Se puede utilizar el segundo parámetro opcional y ponerle el valor "1", si también se quiere buscar el fichero en el include_path.

Ver también readfile(), fopen(), y popen().

fileatime (PHP 3, PHP 4)

Obtiene la última fecha de acceso a un fichero

int **fileatime** (string filename) \linebreak

Devuelve la fecha a la que el fichero fue accedido por última vez, o `FALSE` en caso de error.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

filectime (PHP 3, PHP 4)

Obtiene la fecha de cambio del inode del fichero

int **filectime** (string filename) \linebreak

Devuelve el momento en el que el fichero fue cambiado por última vez, o `FALSE` en caso de error.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

filegroup (PHP 3, PHP 4)

Obtiene el grupo de un fichero

int **filegroup** (string filename) \linebreak

Devuelve el identificador (ID) de grupo del propietario del fichero, o `FALSE` en caso de un error. El ID del grupo es devuelto en formato numérico, usar posix_getgrgid() para obtener el nombre del grupo.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

fileinode (PHP 3, PHP 4)

Obtiene el inode del fichero

int **fileinode** (string filename) \linebreak

Devuelve el número de inode del fichero, o `FALSE` en caso de un error.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

filemtime (PHP 3, PHP 4)

Obtiene la fecha de modificación del fichero

int **filemtime** (string filename) \linebreak

Devuelve el momento en el que el fichero fue modificado por última vez, o `FALSE` en caso de un error.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

fileowner (PHP 3, PHP 4)

Obtiene el propietario del fichero

int **fileowner** (string filename) \linebreak

Devuelve el identificador (ID) de usuario del propietario del fichero, o `FALSE` en caso de error. El ID de usuario se devuelve en formato numérico, usar posix_getpwuid() para obtener el nombre del usuario.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

fileperms (PHP 3, PHP 4)

Obtiene los permisos del fichero

int **fileperms** (string filename) \linebreak

Devuelve los permisos del fichero, o `FALSE` en caso de error.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

filesize (PHP 3, PHP 4)

Obtiene el tamaño del fichero

int **filesize** (string filename) \linebreak

Devuelve el tamaño del fichero, o `FALSE` en caso de error.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

filetype (PHP 3, PHP 4)

Obtiene el tipo de fichero

string **filetype** (string filename) \linebreak

Devuelve el tipo de fichero. Valores posibles son fifo, char, dir, block, link, file, y unknown.

Devuelve `FALSE` si ocurre un error.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

flock (PHP 3>= 3.0.7, PHP 4)

Bloqueo de ficheros portable y asesorado

bool **flock** (int fp, int operation) \linebreak

PHP soporta un método portable de bloquear ficheros completos de manera asesorada (lo que significa que todos los programas que acceden tienen que usar el mismo modo de bloqueo o éste no funcionará).

flock() opera sobre *fp* el cual debe ser un apuntador a un fichero abierto. *operation* toma uno de los siguientes valores:

- Para que adquiera un bloqueo compartido (lectura), fija *operation* a 1.
- Para adquirir un bloqueo exclusivo (escritura), fija *operation* a 2.
- Para liberar un bloqueo (compartido o exclusivo), fija *operation* a 3.
- Si no quieres que **flock()** bloquee mientras está activado, suma 4 al valor de *operation*.

flock() permite establece un modelo simple de lectura/escritura el cual puede usarse en prácticamente cualquier plataforma (incluyendo la mayoria de sistemas Unix e incluso Windows).

flock() devuelve `TRUE` en caso de éxito y `FALSE` en caso de error (ej. cuando no se puede establecer un bloqueo).

fopen (PHP 3, PHP 4)

Abre un fichero o una URL

int **fopen** (string filename, string mode [, int use_include_path]) \linebreak

Si *filename* comienza con "http://" (no es sensible a mayúsculas), se abre una conexión HTTP 1.0 hacia el servidor especificado y se devuelve un apuntador de fichero al comienzo del texto de respuesta.

No maneja redirecciones HTTP, por eso se debe incluir una barra final cuando se trata de directorios.

Si *filename* comienza con "ftp://" (no es sensible a mayúsculas), se abre una conexión ftp hacia el servidor especificado y se devuelve un apuntador al fichero requerido. Si el servidor no soporta ftp en modo pasivo, esto fallará. Se pueden abrir fichero via ftp para leer o para escribir (pero no ambas cosas simultáneamente).

Si *filename* no comienza con nada de lo anterior, el fichero se abre del sistema de ficheros, y se devuelve un apuntador al fichero abierto.

Si el abrir el fichero falla, la función devuelve `FALSE`.

mode puede ser cualquiera de lo siguiente:

- 'r' - Abre para sólo lectura; sitúa el apuntador del fichero al comienzo del mismo.
- 'r+' - Abre para lectura y escritura; situa el apuntador del fichero al comienzo del fichero.

- ’w’ - Abre para sólo escritura; sitúa el apuntador del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
- ’w+’ - Abre el fichero para lectura y escritura; sitúa el apuntador del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
- ’a’ - Abre sólo para escribir (añadir); sitúa el apuntador del fichero al final del mismo. Si el fichero no existe, trata de crearlo.
- ’a+’ - Abre para lectura y escritura (añadiendo); sitúa el apuntador del fichero al final del mismo. Si el fichero no existe, trata de crearlo.

Además, *mode* puede contener la letra ’b’. Esto es útil para sistemas que diferencian entre ficheros binarios y de texto (ej. es inútil en Unix). Si no se necesita, será ignorado.

Puede usarse el tercer parámetro opcional y fijarlo a "1", si también se quiere buscar el fichero en el include_path.

Ejemplo 1. Ejemplo de fopen()()

```
$fp = fopen("/home/rasmus/file.txt", "r");
$fp = fopen("http://www.php.net/", "r");
$fp = fopen("ftp://user:password@example.com/", "w");
```

Si experimentas problemas a la hora de leer y escribir a ficheros y estas usando la version de PHP como módulo para el servidor, recuerda que debes asegurar que los ficheros y directorios que estas usando son accesibles al proceso servidor.

En la plataforma Windows, ten cuidado de escribir correctamente las barras invertidas en el path del fichero (poniéndolas dobles), o usa barras directas.

```
$fp = fopen("c:\\data\\info.txt", "r");
```

Ver también fclose(), fsockopen(), y popen().

fpassthru (PHP 3, PHP 4)

Saca todos los datos restantes del fichero apuntado

int **fpassthru** (int fp) \linebreak

Lee hasta el EOF del fichero apuntado y escribe los resultados a la salida estándar (stdout).

Si ocurre un error, **fpassthru()** devuelve `FALSE`.

El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito por fopen(), popen(), o fsockopen(). El fichero se cierra cuando **fpassthru()** termina de leerlo (dejando *`fp`* sin ninguna utilidad).

Si sólo quieres volcar el contenido de un fichero a stdout puedes If you just want to dump the contents of a file to stdout you may usar la función readfile(), la cual te libra de la llamada a fopen().

Ver también readfile(), fopen(), popen(), y fsockopen()

fputs (PHP 3, PHP 4)

Escribe en el fichero apuntado

int **fputs** (int fp, string str [, int length]) \linebreak

fputs() es un alias de fwrite(), y es idéntico a él en todo. Notar que el parámetro *`length`* es opcional y si no se pone la cadena entera será escrita.

fread (PHP 3, PHP 4)

Lee ficheros en plan binario

string **fread** (int fp, int length) \linebreak

fread() lee hasta *`length`* bytes del apuntador de fichero referenciado por *`fp`*. La lectura acaba cuando *`length`* bytes se han leido o se alcanza EOF, lo que ocurra primero.

```
// Mete el contenido de un fichero en una cadena
$filename = "/usr/local/something.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

Ver también fwrite(), fopen(), fsockopen(), popen(), fgets(), fgetss(), file(), y fpassthru().

fscanf (PHP 4 >= 4.0.1)

Parses input from a file according to a format

mixed **fscanf** (int handle, string format [, string var1]) \linebreak

The function **fscanf()** is similar to sscanf(), but it takes its input from a file associated with *handle* and interprets the input according to the specified *format*. If only two parameters were passed to this function, the values parsed will be returned as an array. Otherwise, if optional parameters are passed, the function will return the number of assigned values. The optional parameters must be passed by reference.

Any whitespace in the format string matches any whitespace in the input stream. This means that even a tab \n in the format string can match a single space character in the input stream.

Ejemplo 1. fscanf() Example

```
$fp = fopen ("users.txt","r");
while ($userinfo = fscanf ($fp, "%s\t%s\t%s\n")) {
    list ($name, $profession, $countrycode) = $userinfo;
    //... do something with the values
}
fclose($fp);
```

Ejemplo 2. users.txt

```
javier  argonaut        pe
hiroshi sculptor        jp
robert  slacker us
luigi   florist it
```

See also fread(), fgets(), fgetss(), sscanf(), printf(), and sprintf().

fseek (PHP 3, PHP 4)

Sitúa el apuntador a un fichero

int **fseek** (int fp, int offset) \linebreak

Fija el indicador de posición del fichero referenciado por fp a tantos bytes como indica offset. Es equivalente a la llamada (en C) `fseek(fp, offset, SEEK_SET)`.

Si va bien, devuelve 0; en otro caso, devuelve -1. Tener en cuenta que situarse más alla de EOF no se considera un error.

No puede usarse sobre apuntadores de ficheros devueltos por fopen() si usan los formatos "http://" or "ftp://".

Ver también ftell() y rewind().

fstat (PHP 4)

Gets information about a file using an open file pointer

array **fstat** (int fp) \linebreak

Gathers the statistics of the file opened by the file pointer fp. This function is similar to the stat() function except that it operates on an open file pointer instead of a filename.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. number of links
4. user id of owner
5. group id owner
6. device type if inode device *
7. size in bytes
8. time of last access
9. time of last modification
10. time of last change
11. blocksize for filesystem I/O *
12. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1

The results of this function are cached. See clearstatcache() for more details.

ftell (PHP 3, PHP 4)

Pregunta por la posición del apuntador de lectura/escritura de un fichero

int **ftell** (int fp) \linebreak

Devuelve la posición del apuntador de fichero referenciado por fp; es decir, la distancia en la secuencia del fichero.

Si ocurre un error, devuelve `FALSE`.

El apuntador al fichero debe ser válido, y debe referirse a The file pointer must be valid, and must point to a file un fichero abierto con éxito por fopen() o popen().

Ver también fopen(), popen(), fseek() y rewind().

ftruncate (PHP 4)

Truncates a file to a given length

int **ftruncate** (int fp, int size) \linebreak

Takes the filepointer, *fp*, and truncates the file to length, *size*. This function returns TRUE on success and FALSE on failure.

fwrite (PHP 3, PHP 4)

Escribe ficheros en plan binario

int **fwrite** (int fp, string string [, int length]) \linebreak

fwrite() escribe el contenido de *string* al fichero apuntado por *fp*. Si se da el argumento *length*, la escritura acaba antes de que *length* bytes sean escritos o se alcance el final de *string*, lo que ocurra primero.

Tener en cuenta que si se da el argumento *length*, entonces la opción de configuración magic_quotes_runtime será ignorada y los caracteres de barra no se quitarán de la cadena *string*.

Ver también fread(), fopen(), fsockopen(), popen(), y fputs().

glob (PHP 4 CVS only)

Find pathnames matching a pattern

array **glob** (string pattern [, int flags]) \linebreak

The **glob()** function searches for all the pathnames matching *pattern* according to the rules used by the shell. No tilde expansion or parameter substitution is done.

Returns an array containing the matched files/directories or FALSE on error.

Nota: This function is disabled in safe mode and therefore will always return FALSE in safe mode.

Ejemplo 1. Convenient way how glob() can replace opendir() and friends.

```
foreach (glob("*.txt") as $filename) {
    echo "$filename size " . filesize($filename) . "\n";
}
```

This could result in the following output:

```
funclist.txt size 44686
funcsummary.txt size 267625
quickref.txt size 137820
```

See also opendir(), readdir() and closedir().

is_dir (PHP 3, PHP 4)

Dice si el fichero nombrado es un directorio

bool **is_dir** (string filename) \linebreak

Devuelve `TRUE` si el nombre del fichero existe y es un directorio.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

Ver también is_file() y is_link().

is_executable (PHP 3, PHP 4)

Dice si el fichero nombrado es ejecutable

bool **is_executable** (string filename) \linebreak

Devuelve `TRUE` si el fichero indicado existe y es ejecutable.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

Ver también is_file() y is_link().

is_file (PHP 3, PHP 4)

Dice si el fichero nombrado es un fichero regular

bool **is_file** (string filename) \linebreak

Devuelve `TRUE` si el fichero nombrado existe y es un fichero regular.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

Ver también is_dir() y is_link().

is_link (PHP 3, PHP 4)

Dice si el fichero indicado es un enlace simbólico

bool **is_link** (string filename) \linebreak

Devuelve `TRUE` si el fichero indicado existe y es un enlace simbólico.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

Ver también is_dir() y is_file().

is_readable (PHP 3, PHP 4)

Dice si el fichero indicado se puede leer

bool **is_readable** (string filename) \linebreak

Devuelve `TRUE` si el fichero indicado existe y se puede leer.

Recuerda que PHP puede acceder al fichero con el identificador de usuario con el que el servidor web se ejecuta (a menudo 'nobody'). No se tienen en cuenta las limitaciones de modos seguros.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

Ver también is_writeable().

is_uploaded_file (PHP 3>= 3.0.17, PHP 4 >= 4.0.3)

Tells whether the file was uploaded via HTTP POST

bool **is_uploaded_file** (string filename) \linebreak

Returns `TRUE` if the file named by `filename` was uploaded via HTTP POST. This is useful to help ensure that a malicious user hasn't tried to trick the script into working on files upon which it should not be working--for instance, `/etc/passwd`.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

is_uploaded_file() is available only in versions of PHP 3 after PHP 3.0.16, and in versions of PHP 4 after 4.0.2. If you are stuck using an earlier version, you can use the following function to help protect yourself:

> **Nota:** The following example will *not* work in versions of PHP 4 after 4.0.2. It depends on internal functionality of PHP which changed after that version.

```
<?php
```

```
/* Userland test for uploaded file. */
function is_uploaded_file($filename) {
    if (!$tmp_file = get_cfg_var('upload_tmp_dir')) {
        $tmp_file = dirname(tempnam(", "));
    }
    $tmp_file .= '/' . basename($filename);
    /* User might have trailing slash in php.ini... */
    return (ereg_replace('/+', '/', $tmp_file) == $filename);
}

/* This is how to use it, since you also don't have
 * move_uploaded_file() in these older versions: */
if (is_uploaded_file($HTTP_POST_FILES['userfile'])) {
    copy($HTTP_POST_FILES['userfile'], "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack: filename '$HTTP_POST_FILES[userfile]'.";
}
?>
```

See also move_uploaded_file(), and the section Handling file uploads for a simple usage example.

is_writable (PHP 4)

Tells whether the filename is writable

bool **is_writable** (string filename) \linebreak

Returns TRUE if the filename exists and is writable. The filename argument may be a directory name allowing you to check if a directory is writeable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also is_readable().

is_writeable (PHP 3, PHP 4)

Dice si se puede escribir en el fichero indicado

bool **is_writeable** (string filename) \linebreak

Devuelve `TRUE` si el fichero indicado existe y se puede escribir en él. El argumento filename puede ser el nombre de un directorio, lo que permite verificar si un directorio tiene permiso de escritura.

Recuerda que PHP puede acceder al fichero con el identificador de usuario con el que el servidor web se ejecuta (a menudo 'nobody'). No se tienen en cuenta las limitaciones de modos seguros.

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

Ver también is_readable().

link (PHP 3, PHP 4)

Crea un enlace fuerte

int **link** (string target, string link) \linebreak

link() crea un enlace fuerte.

Ver también symlink() para crear enlaces débiles, y readlink() junto con linkinfo().

linkinfo (PHP 3, PHP 4)

Consigue información sobre un enlace

int **linkinfo** (string path) \linebreak

linkinfo() da el campo st_dev de la extructura stat de UNIX C devuelto por la llamada al sistema lstat. Esta función se usa para verificar si un enlace (apuntado por *`path`*) existe realmente (usando el mismo método que la macro S_ISLNK definida en stat.h). Devuelve 0 o `FALSE` en caso de error.

Ver también symlink(), link(), y readlink().

lstat (PHP 3>= 3.0.4, PHP 4)

Da información sobre un fichero o enlace simbólico

array **lstat** (string filename) \linebreak

Reúne los datos del fichero o enlace simbólico indicado por filename. Esta función es idéntica a la función stat() excepto que si el nombre en el parámetro *`filename`* es un enlace simbólico, son devueltos los datos (status) del enlace simbólico, y no los del fichero al que apunta el enlace simbólico.

Devuelve un array conteniendo los datos del fichero con los siguientes elementos:

1. dispositivo (device)
2. inode

3. número de enlaces
4. id de usuario del propietario
5. id de grupo del propietario
6. tipo de disposivo si es un inode device *
7. tamaño en bytes
8. fecha del último acceso
9. fecha de la última modificación
10. fecha del último cambio
11. tamaño de bloque para el sistema I/O *
12. número de bloques ocupados

* - sólo válido en sistemas que soportan el tipo st_blksize --otros sistemas (como Windows) devuelven -1

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

mkdir (PHP 3, PHP 4)

Crea un directorio

int **mkdir** (string pathname, int mode) \linebreak

Trata de crear el directorio especificado por pathname.

Ten en cuenta que debes especifiar el modo como un número octal, lo que significa que debes anteponerle un 0 al número.

```
mkdir ("/path/to/my/dir", 0700);
```

Devuelve `TRUE` en caso de éxito y `FALSE` en caso de fallo.

Ver también rmdir().

move_uploaded_file (PHP 4 >= 4.0.3)

Moves an uploaded file to a new location

bool **move_uploaded_file** (string filename, string destination) \linebreak

This function checks to ensure that the file designated by *`filename`* is a valid upload file (meaning that it was uploaded via PHP's HTTP POST upload mechanism). If the file is valid, it will be moved to the filename given by *`destination`*.

If *filename* is not a valid upload file, then no action will occur, and **move_uploaded_file()** will return FALSE.

If *filename* is a valid upload file, but cannot be moved for some reason, no action will occur, and **move_uploaded_file()** will return FALSE. Additionally, a warning will be issued.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

Nota: Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.

Nota: move_uploaded_file() is not affected by the normal safe-mode UID-restrictions. This is not unsafe because **move_uploaded_file()** only operates on files uploaded via PHP.

Aviso

If the destination file already exists, it will be overwritten.

See also is_uploaded_file(), and the section Handling file uploads for a simple usage example.

parse_ini_file (PHP 4)

Parse a configuration file

array **parse_ini_file** (string filename [, bool process_sections]) \linebreak

parse_ini_file() loads in the ini file specified in *filename*, and returns the settings in it in an associative array. By setting the last *process_sections* parameter to TRUE, you get a multidimensional array, with the section names and settings included. The default for *process_sections* is FALSE

Nota: This function has nothing to do with the php.ini file. It is already processed, the time you run your script. This function can be used to read in your own application's configuration files.

Nota: If a value in the ini file contains any non-alphanumeric characters it needs to be enclosed in double-quotes (").

Nota: Since PHP 4.2.1 this function is also affected by safe_mode and open_basedir.

The structure of the ini file is similar to that of the `php.ini`'s.

> **Aviso**
>
> If the ini file you are trying to parse is malformed, PHP will exit.

Ejemplo 1. Contents of sample.ini

```
; This is a sample configuration file
; Comments start with ';', as in php.ini

[first_section]
one = 1
five = 5

[second_section]
path = /usr/local/bin
URL = "http://www.example.com/~username"
```

Ejemplo 2. parse_ini_file() example

```
<?php

// Parse without sections
$ini_array = parse_ini_file("sample.ini");
print_r($ini_array);

// Parse with sections
$ini_array = parse_ini_file("sample.ini", TRUE);
print_r($ini_array);

?>
```

Would produce:

```
Array
(
    [one] => 1
    [five] => 5
```

```
    [path] => /usr/local/bin
    [URL] => http://www.example.com/~username
)
Array
(
    [first_section] => Array
        (
            [one] => 1
            [five] => 5
        )

    [second_section] => Array
        (
            [path] => /usr/local/bin
            [URL] => http://www.example.com/~username
        )

)
```

pathinfo (PHP 4 >= 4.0.3)

Returns information about a file path

array **pathinfo** (string path) \linebreak

pathinfo() returns an associative array containing information about *path*. The following array elements are returned: *dirname*, *basename* and *extension*.

Ejemplo 1. pathinfo() Example

```
<?php

$path_parts = pathinfo("/www/htdocs/index.html");

echo $path_parts["dirname"] . "\n";
echo $path_parts["basename"] . "\n";
echo $path_parts["extension"] . "\n";

?>
```

Would produce:

```
/www/htdocs
index.html
html
```

See also dirname(), basename(), parse_url() and realpath().

pclose (PHP 3, PHP 4)

Cierra el fichero de proceso apuntado

int **pclose** (int fp) \linebreak

Cierra un fichero que representa un tuberia (pipe) abierta con popen().

El apuntador al fichero debe ser válido, y debe haber sido devuelto por una llamada con éxito a popen().

Devuelve el estado de terminación del proceso que estaba ejecutándose.

Ver también popen().

popen (PHP 3, PHP 4)

Abre el fichero de proceso apuntado

int **popen** (string command, string mode) \linebreak

Abre una tubería (pipe) a un proceso ejecutado haciendo fork al comando dado por command

Devuelve un apuntador de fichero idéntico al devuelto por fopen(), excepto que este es unidireccional (sólo puede usarse o para leer o para escribir) y debe cerrarse con pclose(). Este apuntador puede usarse con fgets(), fgetss(), y fputs().

Si ocurre un error, devuelve `FALSE`.

```
$fp = popen ("/bin/ls", "r");
```

Ver también pclose().

readfile (PHP 3, PHP 4)

Muestra el contenido de un fichero

int **readfile** (string filename [, int use_include_path]) \linebreak

Lee un fichero y lo escribe a la salida estándar.

Devuelve el número de bytes leidos del fichero. Si ocurre un error, se devuelve `FALSE` y a menos que la función fuera llamada como @readfile, se imprime un mensaje de error

Si *`filename`* comienzo por "http://" (no es sensible a mayúsculas), se abre una conexión HTTP 1.0 al servidor especificado y el texto de la respuesta se escribe a la salida estándar.

No maneja redirecciones HTTP, por eso se debe incluir una barra final cuando se trata de directorios.

Si *`filename`* comienza con "ftp://" (no es sensible a mayúsculas), se abre una conexión ftp al servidor especificado y el fichero que se pide se escribe en la salida estándar. Si el servidor no soporta ftp en modo pasivo, la función fallará.

Si *`filename`* no comienza con ninguna de las cadenas anteriores, el fichero será abierto del sistema de ficheros y su contenido escrito en la salida estándar.

Se puede usar el segundo parámetro opcional y fijarlo a "1", si si quieres que también se busque el fichero en el include_path.

Ver también fpassthru(), file(), fopen(), include(), require(), y virtual().

readlink (PHP 3, PHP 4)

Devuelve el objetivo de un enlace simbólico

string **readlink** (string path) \linebreak

readlink() hace lo mismo que la función C readlink C y devuelve el contenido del path del enlace simbólico o 0 en caso de error.

Ver también symlink(), **readlink()** y linkinfo().

realpath (PHP 4)

Returns canonicalized absolute pathname

string **realpath** (string path) \linebreak

realpath() expands all symbolic links and resolves references to '/./', '/../' and extra '/' characters in the input *`path`* and return the canonicalized absolute pathname. The resulting path will have no symbolic link, '/./' or '/../' components.

realpath() returns `FALSE` on failure, e.g. if the file does not exists.

Ejemplo 1. realpath() example

```
$real_path = realpath ("../../index.php");
```

See also: basename(), dirname(), and pathinfo().

rename (PHP 3, PHP 4)

Renombra un fichero

int **rename** (string oldname, string newname) \linebreak

Trata de renombrar *oldname* como *newname*.

Devuelve TRUE en caso de éxito y FALSE en caso de fallo.

rewind (PHP 3, PHP 4)

Rebobina la posición del apuntador al fichero

int **rewind** (int fp) \linebreak

Fija el indicador de posición del fichero dado por fp al comienzo de del fichero.

Si ocurre un error, devuelve 0.

El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito por fopen().

Ver también fseek() y ftell().

rmdir (PHP 3, PHP 4)

Elimina un directorio

int **rmdir** (string dirname) \linebreak

Trata de eliminar el directorio indicado por pathname. El directorio debe estar vacio, y los permisos relevantes deben permitir esto.

Si ocurre un error, devuelve 0.

Ver también mkdir().

set_file_buffer (PHP 3>= 3.0.8, PHP 4 >= 4.0.1)

Fija el buffer de fichero del fichero apuntado

int **fwrite** (int fp, int buffer) \linebreak

set_file_buffer() fija el buffer para operaciones de escritura en el apuntador de fichero *`fp`* con *`buffer`* bytes. Si *`buffer`* es 0 entonces las operaciones de escritura no usan un buffer intermedio.

La función devuelve 0 en caso de éxito, o EOF si la petición no se puede realizar.

Tener en cuenta que por defecto cualquier fopen hace una llamada a set_file_buffer de 8K.

Ver también fopen().

stat (PHP 3, PHP 4)

Da información sobre un fichero

array **stat** (string filename) \linebreak

Recoje los datos sobre el fichero indicado por filename.

Devuelve un array conteniendo los datos del fichero con los siguientes elementos:

1. dispositivo (device)
2. inode
3. modo de protección del inode
4. número de enlaces
5. id de usuario del propietario
6. id de grupo del propietario
7. tipo de dispositivo si es un inode device *
8. tamaño en bytes
9. fecha del último acceso access
10. fecha de la última modificación
11. fecha del último cambio
12. tamaño del bloque para el sistema I/O *
13. número de bloques ocupados

* - sólo válido en sistemas que soportan el tipo st_blksize --otros sistemas (como Windows) devuelven -1

Los resultados de esta función son cacheados. Ver clearstatcache() para más detalles.

symlink (PHP 3, PHP 4)

Crea un enlace simbólico

int **symlink** (string target, string link) \linebreak

symlink() crea un enlace simbólico del objetivo *target* con el nombre especificado por *link*.

Ver también link() para crear enlaces fuertes, y readlink() junto con linkinfo().

tempnam (PHP 3, PHP 4)

Crea un fichero de nombre único

string **tempnam** (string dir, string prefix) \linebreak

Crea un fichero temporal de nombre único en el directorio especificado. Si el directorio no existe **tempnam()** puede generar un fichero en el directorio temporal del sistema.

El comportamiento de la función **tempnam()** depende del sistema. En Windows la variable de entorno TMP se impone sobre el parámetro *dir*, en Linux la variable de entorno TMPDIR tiene preferencia, mientras que en SVR4 siempre se usará el parámetro *dir* si si el directorio al que apunta existe. Consulta la documentación del sistema sobre la función tempnam(3) en caso de duda.

Devuelve el nombre del nuevo fichero temporal, o una cadena nula en caso de fallo.

Ejemplo 1. Ejemplo de tempnam()

```
$tmpfname = tempnam ("/tmp", "FOO");
```

tmpfile (PHP 3>= 3.0.13, PHP 4)

Creates a temporary file

int **tmpfile** (void) \linebreak

Creates a temporary file with an unique name in write mode, returning a file handle similar to the one returned by fopen(). The file is automatically removed when closed (using fclose()), or when the script ends.

For details, consult your system documentation on the `tmpfile(3)` function, as well as the `stdio.h` header file.

Ejemplo 1. tmpfile() example

```
$temp = tmpfile();
fwrite($temp, "writing to tempfile");
fclose($temp); // this removes the file
```

See also tempnam().

touch (PHP 3, PHP 4)

Fija la fecha de modificación de un fichero

int **touch** (string filename, int time) \linebreak

Trata de fijar la fecha de modificación del fichero indicado por filename al valor dado por time. Si no se pone la opción time, se utiliza la fecha actual.

Si el fichero no existe, será creado.

Devuelve `TRUE` en caso de éxito y `FALSE` en otro caso.

umask (PHP 3, PHP 4)

Cambia la umask actual

int **umask** (int mask) \linebreak

umask() fija las umask PHP con la mascara & 0777 y y devuelve la antigua umask. Cuando PHP se está usando como un módulo del servidor, la umask se restaura cuando cada petición es finalizada.

umask() sin argumentos sólamente devuelve la umask actual.

unlink (PHP 3, PHP 4)

Borra un fichero

int **unlink** (string filename) \linebreak

Borra el fichero *`filename`*. Es similar a la función unlink() del Unix C.

Devuelve 0 o `FALSE` en caso de error.

Ver también rmdir() para borrar directorios.

XXXI. Funciones Forms Data Format (Formato de Datos de Formularios)

El Formato de Datos de Formulario (FDF) está diseñado para el manejo de formularios en archivos PDF. Se aconseja leer la información disponible en http://partners.adobe.com/asn/developer/acrosdk/forms.html para más información sobre lo que es FDF y cómo se usa en general.

> **Nota:** Actualmente Adobe sólo proporciona una versión compatible con libc5 para Linux. Las pruebas con glibc2 provocaron un fallo de segmentado. Si alguien es capaz de hacerla funcionar, por favor coméntelo en esta página.

> **Nota:** Si tiene problemas configurando php con soporte de fdftk, compruebe si el archivo de cabecera FdfTk.h y la librería libFdfTk.so están en su lugar correcto. Deberían encontrarse respectivamente en fdftk-dir/include y en fdftk-dir/lib. Este problema no se dará si se limita a desempaqueta la distribucón del FtdTk.

La idea general del FDF es similar a los formularios HTML. La diferencia básicamente está en el formato en que se transmiten los datos al servidor cuando se pulsa el botón de envío (este es realmente el Formato de Datos de Formulario) y el formato del formulario en sí mismo (que es el Formato de Documento Portable, PDF). Procesar lo datos del FDF es una de las características que proporcionan las funciones fdf. Pero aún hay más. Uno también puede tomar un formulario PDF y rellenar los campos de entrada con datos sin modificar el formulario en sí mismo. En dicho caso, lo que se hace es crear un documento FDF (fdf_create()), fijar los valores de cada campo de entrada (fdf_set_value()) y asociarlo con un formulario PDF (fdf_set_file()). Finalmente, debe ser enviado al navegador con el MimeType application/vnd.fdf. El plug-in de Acrobar reader de su navegador reconoce el MimeType, lee el formulario PDF asociado y rellena los datos a partir del documento FDF.

Los siguientes ejemplos muestran cómo se evalúan los datos de los formularios.

Ejemplo 1. Evaluando un documento FDF

```
<?php
// Guarda los datos FDF en un archivo temporal
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Abre archivo temporal y evalúa los datos
// El formulario pdf contenía varios campos de texto con los nombres
// volumen, fecha, comentario, editorial, preparador, y dos casillas de verificación
// muestra_editorial y muestra_preparador.
$fdf = fdf_open("test.fdf");
$volume = fdf_get_value($fdf, "volumen");
echo "El campo volumen tiene el valor '<B>$volume</B>'<BR>";
```

```
$date = fdf_get_value($fdf, "fecha");
echo "El campo fecha tiene el valor '<B>$date</B>'<BR>";

$comment = fdf_get_value($fdf, "comentario");
echo "El campo comentario tiene el valor '<B>$comment</B>'<BR>";

if(fdf_get_value($fdf, "muestra_editorial") == "On") {
  $publisher = fdf_get_value($fdf, "editorial");
  echo "El campo editorial tiene el valor '<B>$publisher</B>'<BR>";
} else
  echo "No se debe mostrar la editorial.<BR>";

if(fdf_get_value($fdf, "muestra_preparador") == "On") {
  $preparer = fdf_get_value($fdf, "preparador");
  echo "El campo preparador tiene el valor '<B>$preparer</B>'<BR>";
} else
  echo "No se debe mostrar el preparador.<BR>";
fdf_close($fdf);
?>
```

fdf_add_template (PHP 3>= 3.0.13, PHP 4)

Adds a template into the FDF document

bool **fdf_add_template** (int fdfdoc, int newpage, string filename, string template, int rename) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

fdf_close (PHP 3>= 3.0.6, PHP 4)

Cerrar un documento FDF

void **fdf_close** (int fdf_document) \linebreak

La función **fdf_close()** cierra un documento FDF.

Vea también fdf_open().

fdf_create (PHP 3>= 3.0.6, PHP 4)

Crear un documento FDF

int **fdf_create** (void) \linebreak

La función **fdf_create()** crea un documento FDF nuevo. Esta función se necesita si se desea rellenar los campos de entrada en un documento PDF.

Ejemplo 1. Rellenando un documento PDF

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, "volumen", $volume, 0);

fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type: application/vnd.fdf");
$fp = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

Vea también fdf_close(), fdf_save(), fdf_open().

fdf_get_file (PHP 3>= 3.0.6, PHP 4)

Obtener el valor de la clave /F

string **fdf_get_file** (int fdf_document) \linebreak

La función fdf_set_file() devuelve el valor de la clave /F.

Vea también fdf_set_file().

fdf_get_status (PHP 3>= 3.0.6, PHP 4)

Obtener el valor de la clave /STATUS

string **fdf_get_status** (int fdf_document) \linebreak

La función **fdf_get_status()** devuelve el valor de la clave /STATUS.

Vea también fdf_set_status().

fdf_get_value (PHP 3>= 3.0.6, PHP 4)

Obtener el valor de un campo

string **fdf_get_value** (int fdf_document, string fieldname) \linebreak

La función **fdf_get_value()** devuelve el valor de un campo.

Vea también fdf_set_value().

fdf_next_field_name (PHP 3>= 3.0.6, PHP 4)

Obtener el nombre del siguiente campo

string **fdf_next_field_name** (int fdf_document, string fieldname) \linebreak

La función **fdf_next_field_name()** devuelve el nombre del campo tras el campo *`fieldname`* o el nombre del primer campo si el segundo parámetro es `NULL`.

Vea también **fdf_set_field()**, **fdf_get_field()**.

fdf_open (PHP 3>= 3.0.6, PHP 4)

Abrir un documento FDF

int **fdf_open** (string filename) \linebreak

La función **fdf_open()** abre un archivo con datos de formulario. Este archivo debe contener los datos tal y como se devuelven en un formulario PDF. Actualmente dicho archivo debe crearse "manualmente" usando la función fopen() y escribiendo en éste el contenido de HTTP_FDF_DATA usando fwrite(). No existe un mecanismo similar al de los formularios HTML donde se crea una variable para cada campo de entrada.

Ejemplo 1. Accediendo a los datos del formulario

```
<?php
// Guarda los datos FDF en un archivo temporal
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Abre archivo temporal y evalúa los datos
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

Vea también fdf_close().

fdf_save (PHP 3>= 3.0.6, PHP 4)

Guardar un documeto FDF

int **fdf_save** (string filename) \linebreak

La función **fdf_save()** guarda un documento FDF. El kit de FDF proporciona una forma de volcar el documento a stdout si el parámetro *filename* es '.'. Esto no funciona si el PHP se utiliza como un módulo del apache. En tal caso se deberá escribir a un fichero y utilizar p. ej. fpassthru() para visualizarlo.

Vea también fdf_close() y el ejemplo para fdf_create().

fdf_set_ap (PHP 3>= 3.0.6, PHP 4)

Ajusta la apariencia de un campo

void **fdf_set_ap** (int fdf_document, string field_name, int face, string filename, int page_number) \linebreak

La función **fdf_set_ap()** ajusta la apariencia de un campo (p. ej. el valor de la clave /AP). Los valores posibles de *face* son 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_encoding (PHP 4 >= 4.1.0)

Sets FDF character encoding

bool **fdf_set_encoding** (int fdf_document, string encoding) \linebreak

fdf_set_encoding() sets the character encoding in FDF document *fdf_document*. *encoding* should be the valid encoding name. The valid encoding name in Acrobat 5.0 are "Shift-JIS", "UHC", "GBK","BigFive".

The **fdf_set_encoding()** is available in PHP 4.1.0 or later.

fdf_set_file (PHP 3>= 3.0.6, PHP 4)

Fijar el valor de la clave /F

void **fdf_set_file** (int fdf_document, string filename) \linebreak

La función **fdf_set_file()** fija el valor de la clave /F. La clave /F es simplemente una referencia a un formulario PDF que se va a rellenar con datos. En un entorno web es un URL (p.ej. http:/testfdf/resultlabel.pdf).

Vea también fdf_get_file() y el ejemplo para fdf_create().

fdf_set_flags (PHP 4 >= 4.0.2)

Sets a flag of a field

bool **fdf_set_flags** (int fdf_document, string fieldname, int whichFlags, int newFlags) \linebreak

The **fdf_set_flags()** sets certain flags of the given field *fieldname*.

See also fdf_set_opt().

fdf_set_javascript_action (PHP 4 >= 4.0.2)

Sets an javascript action of a field

bool **fdf_set_javascript_action** (int fdf_document, string fieldname, int trigger, string script) \linebreak

fdf_set_javascript_action() sets a javascript action for the given field *`fieldname`*.

See also fdf_set_submit_form_action().

fdf_set_opt (PHP 4 >= 4.0.2)

Sets an option of a field

bool **fdf_set_opt** (int fdf_document, string fieldname, int element, string str1, string str2) \linebreak

The **fdf_set_opt()** sets options of the given field *`fieldname`*.

See also fdf_set_flags().

fdf_set_status (PHP 3>= 3.0.6, PHP 4)

Fija el valor de la clave /STATUS

void **fdf_set_status** (int fdf_document, string status) \linebreak

La función **fdf_set_status()** fija el valor de la clave /STATUS.

Vea también fdf_get_status().

fdf_set_submit_form_action (PHP 4 >= 4.0.2)

Sets a submit form action of a field

bool **fdf_set_submit_form_action** (int fdf_document, string fieldname, int trigger, string script, int flags) \linebreak

The **fdf_set_submit_form_action()** sets a submit form action for the given field *`fieldname`*.

See also fdf_set_javascript_action().

fdf_set_value (PHP 3>= 3.0.6, PHP 4)

Fijar el valor de un campo

void **fdf_set_value** (int fdf_document, string fieldname, string value, int isName) \linebreak

La función **fdf_set_value()** fija el valor de un campo. El parámetro final determina si el valor del campo se deberá convertir a un Nombre PDF (`isName` = 1) o convertir en una Cadena PDF (`isName` = 0).

Vea también fdf_get_value().

XXXII. FriBiDi functions

Introducción

Requerimientos

Instalación

Configuración en tiempo de ejecución

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión

ha sido o bien compilada dentro de PHP o grabada dinamicamente en tiempo de ejecución.

`FRIBIDI_CHARSET_UTF8` (integer)

`FRIBIDI_CHARSET_8859_6` (integer)

`FRIBIDI_CHARSET_8859_8` (integer)

`FRIBIDI_CHARSET_CP1255` (integer)

`FRIBIDI_CHARSET_CP1256` (integer)

`FRIBIDI_CHARSET_ISIRI_3342` (integer)

fribidi_log2vis (PHP 4 >= 4.0.4)

Convert a logical string to a visual one

string **fribidi_log2vis** (string str, string direction, int charset) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

XXXIII. Funciones FTP

FTP es el acrónimo de File Transfer Protocol (Protocolo de Transferencia de Ficheros.

Cuando se utiliza el módulo FTP, se definen las siguientes constantes: FTP_ASCII, y FTP_BINARY.

ftp_cdup (PHP 3>= 3.0.13, PHP 4)

Cambia al directorio padre

int **ftp_cdup** (int ftp_stream) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`.

Cambia al directorio padre.

ftp_chdir (PHP 3>= 3.0.13, PHP 4)

Cambia de directorio en un servidor FTP

int **ftp_chdir** (int ftp_stream, string directory) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`.

Cambia al directorio especificado por el parámetro *`directory`*.

ftp_close (PHP 4 >= 4.2.0)

Closes an FTP connection

void **ftp_close** (resource ftp_stream) \linebreak

Nota: This function is only available in CVS.

ftp_close() closes *`ftp_stream`* and releases the resource. You can't reuse this resource but have to create a new one with ftp_connect().

ftp_connect (PHP 3>= 3.0.13, PHP 4)

Establece una conexión FTP

int **ftp_connect** (string host [, int port]) \linebreak

Si tiene éxito, devuelve un flujo FTP. En caso de error, devuelve `FALSE`.

ftp_connect() establece una conexión FTP al *`host`* especificado. El parámetro *`port`* especifica un puerto alternativo al que conectar. Si se omite o es cero, se usa el puerto FTP por defecto, 21.

ftp_delete (PHP 3>= 3.0.13, PHP 4)

Borra un fichero del servidor FTP.

int **ftp_delete** (int ftp_stream, string path) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`.

ftp_delete() borra el fichero especificado por el parámetro *`path`* del servidor FTP.

ftp_exec (PHP 4 >= 4.0.3)

Request execution of a program on the FTP server

bool **ftp_exec** (resource ftp_stream, string command) \linebreak

Sends a SITE EXEC *`command`* request to the FTP server. Returns `FALSE` if the request fails, returns the output of the command otherwise.

ftp_fget (PHP 3>= 3.0.13, PHP 4)

Descarga un fichero del servidor FTP y lo guarda en un fichero abierto.

int **ftp_fget** (int ftp_stream, int fp, string remote_file, int mode) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`.

ftp_fget() baja el fichero *`remote_file`* del servidor FTP, y lo escribe en el puntero a fichero *`fp`*. El modo de transferencia especificado por el parámetro *`mode`* debe ser FTP_ASCII o bien FTP_BINARY.

ftp_fput (PHP 3>= 3.0.13, PHP 4)

Sube un fichero abierto al servidor FTP.

int **ftp_fput** (int ftp_stream, string remote_file, int fp, int mode) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`

ftp_fput() sube los datos apuntados por el puntero a fichero *`fp`* hasta alcanzar el final del fichero. Los resultados se guardan en el fichero *`remote_file`* del FTP remoto. El modo de transferencia especificado por el parámetro *`mode`* debe ser FTP_ASCII o bien FTP_BINARY.

ftp_get_option (PHP 4 >= 4.2.0)

Retrieves various runtime behaviours of the current FTP stream

mixed **ftp_get_option** (resource ftp_stream, int option) \linebreak

Nota: This function is only available in CVS.

Returns the value on success, or FALSE if the given *option* is not supported. In the latter case, a warning message is also thrown.

This function returns the value for the requested *option* from the specified *ftp_stream* . Currently, the following options are supported:

Tabla 1. Supported runtime FTP options

FTP_TIMEOUT_SEC	Returns the current timeout used for network related operations.

Ejemplo 1. ftp_get_option() example

```
// Get the timeout of the given FTP stream
$timeout = ftp_get_option($conn_id, FTP_TIMEOUT_SEC);
```

ftp_get (PHP 3>= 3.0.13, PHP 4)

Descarga un fichero del servidor FTP.

int **ftp_get** (int ftp_stream, string local_file, string remote_file, int mode) \linebreak

Si tiene éxito, devuelve TRUE. En caso de error, devuelve FALSE.

ftp_get() baja el fichero *remote_file* del servidor FTP, y lo guarda localmente como *local_file*. El modo de transferencia especificado por el parámetro *mode* debe ser FTP_ASCII o bien FTP_BINARY.

ftp_login (PHP 3>= 3.0.13, PHP 4)

Comienza la sesion en una conexión FTP

int **ftp_login** (int ftp_stream, string username, string password) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`.

Inicia una sesion (envía identificador de usuario y contraseña) en el flujo FTP especificado.

ftp_mdtm (PHP 3>= 3.0.13, PHP 4)

Devuelve la fecha de última modificación del fichero especificado.

int **ftp_mdtm** (int ftp_stream, string remote_file) \linebreak

Si tiene éxito, devuelve una marca de tiempo UNIX (UNIX timestamp). En caso de error, devuelve -1.

ftp_mdtm() comprueba la fecha de última modificación de un fichero, y la devuelve como una marca de tiempo UNIX. Si se produce algún error, o el fichero no existe, devuelve -1. Tenga en cuenta que no todos los servidores soportan esta característica.

ftp_mkdir (PHP 3>= 3.0.13, PHP 4)

Crea un directorio

string **ftp_mkdir** (int ftp_stream, string directory) \linebreak

Si tiene éxito, devuelve el nombre del directorio recién creado. En caso de error, devuelve `FALSE`.

Crea el directorio especificado por el parámetro *`directory`*.

ftp_nlist (PHP 3>= 3.0.13, PHP 4)

Devuelve una lista de ficheros del directorio dado.

int **ftp_nlist** (int ftp_stream, string directory) \linebreak

Si tiene éxito, devuelve un array de nombres de fichero. En caso de error, devuelve `FALSE`.

ftp_pasv (PHP 3>= 3.0.13, PHP 4)

Activa o desactiva el modo pasivo.

int **ftp_pasv** (int ftp_stream, int pasv) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`.

ftp_pasv() activa el modo pasivo si el parámetro *`pasv`* es `TRUE` (desactiva el modo pasivo si *`pasv`* es `FALSE`.) En modo pasivo, las conexiones de datos son iniciadas por el cliente, en lugar de ser iniciadas por el servidor.

ftp_put (PHP 3>= 3.0.13, PHP 4)

Sube un fichero al servidor FTP.

int **ftp_put** (int ftp_stream, string remote_file, string local_file, int mode) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`.

ftp_put() sube el fichero local *`local_file`* al servidor FTP y lo guarda como *`remote_file`*. El modo de transferencia especificado por el parámetro *`mode`* debe ser FTP_ASCII o bien FTP_BINARY.

ftp_pwd (PHP 3>= 3.0.13, PHP 4)

Devuelve el nombre del directorio actual

int **ftp_pwd** (int ftp_stream) \linebreak

Devuelve el directorio actual, o `FALSE` en caso de error.

ftp_quit (PHP 3>= 3.0.13, PHP 4)

Cierra una conexión FTP

int **ftp_quit** (int ftp_stream) \linebreak

ftp_connect() cierra el flujo FTP *`ftp_stream`*.

ftp_rawlist (PHP 3>= 3.0.13, PHP 4)

Devuelve una lista detallada de ficheros del directorio dado.

int **ftp_rawlist** (int ftp_stream, string directory) \linebreak

ftp_rawlist() ejecuta el comando FTP LIST, y devuelve el resultado como un array. Cada elemento del array corresponde a una línea de texto. La salida no se procesa de ninguna forma. Se puede utilizar el

identificador de tipo de sistema devuelto por ftp_systype() para determinar cómo se deben interpretar los resultados.

ftp_rename (PHP 3>= 3.0.13, PHP 4)

Renombra un fichero del servidor FTP.

int **ftp_rename** (int ftp_stream, string from, string to) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`.

ftp_rename() renombra el fichero especificado por el parámetro *`from`* con el nuevo nombre *`to`*

ftp_rmdir (PHP 3>= 3.0.13, PHP 4)

Borra un directorio

int **ftp_rmdir** (int ftp_stream, string directory) \linebreak

Si tiene éxito, devuelve `TRUE`. En caso de error, devuelve `FALSE`.

Borra el directorio especificado por el parámetro *`directory`*.

ftp_set_option (PHP 4 >= 4.2.0)

Set miscellaneous runtime FTP options

bool **ftp_set_option** (resource ftp_stream, int option, mixed value) \linebreak

Nota: This function is only available in CVS.

Returns `TRUE` if the option could be set; `FALSE` if not. A warning message will be thrown if the *`option`* is not supported or the passed *`value`* doesn't match the expected value for the given *`option`*.

This function controls various runtime options for the specified FTP stream. The *`value`* parameter depends on which *`option`* parameter is chosen to be altered. Currently, the following options are supported:

Tabla 1. Supported runtime FTP options

FTP_TIMEOUT_SEC	Changes the timeout in seconds used for all network related functions. Parameter *value* has be to of type int and must be greater than 0. The default timeout is 90 seconds.

Ejemplo 1. ftp_set_option() example

```
// Set the network timeout down to 10 seconds
ftp_set_option($conn_id, FTP_TIMEOUT_SEC, 10);
```

ftp_site (PHP 3>= 3.0.15, PHP 4)

Sends a SITE command to the server

bool **ftp_site** (resource ftp_stream, string cmd) \linebreak

ftp_site() sends the command specified by *cmd* to the FTP server. SITE commands are not standardized, and vary from server to server. They are useful for handling such things as file permissions and group membership.

Returns `TRUE` on success, `FALSE` on error.

ftp_size (PHP 3>= 3.0.13, PHP 4)

Devuelve el tamaño del fichero especificado.

int **ftp_size** (int ftp_stream, string remote_file) \linebreak

Si tiene éxito devuelve el tamaño del fichero, o -1 en caso de error.

ftp_size() devuelve el tamaño de un fichero. Si ocurre algún error, o si el fichero no existe, devuelve -1. No todos los servidores soportan esta característica.

ftp_systype (PHP 3>= 3.0.13, PHP 4)

Devuelve el identificador de tipo de sistema del servidor FTP remoto.

int **ftp_systype** (int ftp_stream) \linebreak

Devuelve el tipo de sistema remoto, o `FALSE` en caso de error.

XXXIV. Function Handling functions

These functions all handle various operations involved in working with functions.

call_user_func_array (PHP 4 >= 4.0.4)

Call a user function given with an array of parameters

mixed **call_user_func_array** (string function_name [, array paramarr]) \linebreak

Call a user defined function given by *function_name*, with the parameters in *paramarr*. For example:

```
function debug($var, $val)
    echo "***DEBUGGING\nVARIABLE: $var\nVALUE:";
    if (is_array($val) || is_object($val) || is_resource($val))
        print_r($val);
    else
        echo "\n$val\n";
    echo "***\n";
}

$c = mysql_connect();
$host = $_SERVER["SERVER_NAME"];

call_user_func_array ('debug', array("host", $host));
call_user_func_array ('debug', array("c", $c));
call_user_func_array ('debug', array("_POST", $_POST));
```

See also: call_user_func(), call_user_method(), call_user_method_array().

Nota: This function was added to the CVS code after release of PHP 4.0.4pl1

call_user_func (PHP 3>= 3.0.3, PHP 4)

Call a user function given by the first parameter

mixed **call_user_func** (string function_name [, mixed parameter [, mixed ...]]) \linebreak

Call a user defined function given by the *function_name* parameter. Take the following:

```
function barber ($type) {
    print "You wanted a $type haircut, no problem";
}
call_user_func ('barber', "mushroom");
call_user_func ('barber', "shave");
```

create_function (PHP 4 >= 4.0.1)

Create an anonymous (lambda-style) function

string **create_function** (string args, string code) \linebreak

Creates an anonymous function from the parameters passed, and returns a unique name for it. Usually the *args* will be passed as a single quote delimited string, and this is also recommended for the *code*. The reason for using single quoted strings, is to protect the variable names from parsing, otherwise, if you use double quotes there will be a need to escape the variable names, e.g. `\$avar`.

You can use this function, to (for example) create a function from information gathered at run time:

Ejemplo 1. Creating an anonymous function with create_function()

```
$newfunc = create_function('$a,$b','return "ln($a) + ln($b) = ".log($a * $b);');
echo "New anonymous function: $newfunc\n";
echo $newfunc(2,M_E)."\n";
// outputs
// New anonymous function: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
```

Or, perhaps to have general handler function that can apply a set of operations to a list of parameters:

Ejemplo 2. Making a general processing function with create_function()

```
function process($var1, $var2, $farr) {
    for ($f=0; $f < count($farr); $f++)
    echo $farr[$f]($var1,$var2)."\n";
}

// create a bunch of math functions
$f1 = 'if ($a >=0) {return "b*a^2 = ".$b*sqrt($a);} else {return false;}';
$f2 = "return \"min(b^2+a, a^2,b) = \".min(\$a*\$a+\$b,\$b*\$b+\$a);";
$f3 = 'if ($a > 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b;} else {return false;}';
$farr = array(
    create_function('$x,$y', 'return "some trig: ".(sin($x) + $x*cos($y));'),
    create_function('$x,$y', 'return "a hypotenuse: ".sqrt($x*$x + $y*$y);'),
    create_function('$a,$b', $f1),
    create_function('$a,$b', $f2),
    create_function('$a,$b', $f3)
    );

echo "\nUsing the first array of anonymous functions\n";
echo "parameters: 2.3445, M_PI\n";
process(2.3445, M_PI, $farr);
```

```
// now make a bunch of string processing functions
$garr = array(
    create_function('$b,$a','if (strncmp($a,$b,3) == 0) return "** \"$a\" '.
    'and \"$b\"\n** Look the same to me! (looking at the first 3 chars)";'),
    create_function('$a,$b',' return "CRCs: ".crc32($a)." , ".crc32(b);'),
    create_function('$a,$b',' return "similar(a,b) = ".similar_text($a,$b,&$p)."($p%)";')
    );
echo "\nUsing the second array of anonymous functions\n";
process("Twas brilling and the slithy toves", "Twas the night", $garr);
```

and when you run the code above, the output will be:

```
Using the first array of anonymous functions
parameters: 2.3445, M_PI
some trig: -1.6291725057799
a hypotenuse: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594

Using the second array of anonymous functions
** "Twas the night" and "Twas brilling and the slithy toves"
** Look the same to me! (looking at the first 3 chars)
CRCs: -725381282 , 1908338681
similar(a,b) = 11(45.833333333333%)
```

But perhaps the most common use for of lambda-style (anonymous) functions is to create callback functions, for example when using array_walk() or usort()

Ejemplo 3. Using anonymous functions as callback functions

```
$av = array("the ","a ","that ","this ");
array_walk($av, create_function('&$v,$k','$v = $v."mango";'));
print_r($av);  // for PHP3 use var_dump()
// outputs:
// Array
// (
//   [0] => the mango
//   [1] => a mango
//   [2] => that mango
//   [3] => this mango
// )

// an array of strings ordered from shorter to longer
$sv = array("small","larger","a big string","it is a string thing");
print_r($sv);
// outputs:
// Array
// (
//   [0] => small
```

```
//   [1] => larger
//   [2] => a big string
//   [3] => it is a string thing
// )

// sort it from longer to shorter
usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);
// outputs:
// Array
// (
//   [0] => it is a string thing
//   [1] => a big string
//   [2] => larger
//   [3] => small
// )
```

func_get_arg (PHP 4)

Devuelve un elemento de la lista de argumentos.

int **func_get_arg** (int arg_num) \linebreak

Devuelve el argumento que está en la posición *arg_num* en la lista de argumentos de una función definida por el usuario. Los argumentos de la función se cuentan comenzando por la posición cero. **func_get_arg()** generará un aviso si se llama desde fuera de la definición de la función.

Si *arg_num* es mayor que el número de argumentos pasados realmente, se generará un aviso y **func_get_arg()** devolverá FALSE.

```
<?php
function foo() {
   $numargs = func_num_args();
   echo "Number of arguments: $numargs<br>\n";
   if ( $numargs >= 2 ) {
      echo "Second argument is: " . func_get_arg( 1 ) . "<br>\n";
   }
}

foo( 1, 2, 3 );
?>
```

func_get_arg() puede utilizarse conjuntamente con func_num_args() y func_get_args() para permitir a las funciones definidas por el usuario que acepten listas de argumentos de longitud variable.

Nota: Esta función fue añadida en PHP 4.

func_get_args (PHP 4)

Devuelve un array que contiene la lista de argumentos de una función.

int **func_get_args** (void) \linebreak

Devuelve un array en el que cada elemento es el miembro correspondiente de la lista de argumentos de la función definida por el usuario actual. **func_get_args()** generará un aviso si es llamada desde fuera de la definición de la función.

```
<?php
function foo() {
   $numargs = func_num_args();
   echo "Number of arguments: $numargs<br>\n";
   if ( $numargs >= 2 ) {
      echo "Second argument is: " . func_get_arg( 1 ) . "<br>\n";
   }
   $arg_list = func_get_args();
   for ( $i = 0; $i < $numargs; $i++ ) {
      echo "Argument $i is: " . $arg_list[$i] . "<br>\n";
   }
}

foo( 1, 2, 3 );
?>
```

func_get_args() puede utilizarse conjuntamente con func_num_args() y func_get_arg() para permitir a las funciones definidas por el usuario que acepten listas de argumentos de longitud variable.

Nota: Esta función fue añadida en PHP 4.

func_num_args (PHP 4)

Devuelve el número de argumentos pasados a la función.

int **func_num_args** (void) \linebreak

Devuelve el número de argumentos pasados a la función actual definida por el usuario. **func_num_args()** generará un aviso si es llamada desde fuera de la definición de la función.

```
<?php
function foo() {
   $numargs = func_num_args();
   echo "Number of arguments: $numargs\n";
}

foo( 1, 2, 3 );  // Prints 'Number of arguments: 3'
?>
```

func_num_args() puede utilizarse conjuntamente con func_get_arg() y func_get_args() para permitir a las funciones definidas por el usuario que acepten listas de argumentos de longitud variable.

Nota: Esta función fue añadida en PHP 4.

function_exists (PHP 3>= 3.0.7, PHP 4)

Devuelve `TRUE` si la función dada ha sido definida

int **function_exists** (string function_name) \linebreak

Consulta la lista de funciones definidas buscando *`function_name`* (nombre de función). Devuelve `TRUE` si encuentra el nombre de función dado, `FALSE` en otro caso.

get_defined_functions (PHP 4 >= 4.0.4)

Returns an array of all defined functions

array **get_defined_functions** (void) \linebreak

This function returns an multidimensional array containing a list of all defined functions, both built-in (internal) and user-defined. The internal functions will be accessible via `$arr["internal"]`, and the user defined ones using `$arr["user"]` (see example below).

```
function myrow($id, $data) {
    return "<tr><th>$id</th><td>$data</td></tr>\n";
}

$arr = get_defined_functions();
```

```
print_r($arr);
```

Will output something along the lines of:

```
Array
(
    [internal] => Array
        (
            [0] => zend_version
            [1] => func_num_args
            [2] => func_get_arg
            [3] => func_get_args
            [4] => strlen
            [5] => strcmp
            [6] => strncmp
            ...
            [750] => bcscale
            [751] => bccomp
        )

    [user] => Array
        (
            [0] => myrow
        )

)
```

See also get_defined_vars() and get_defined_constants().

register_shutdown_function (PHP 3>= 3.0.4, PHP 4)

Registra una función para su ejecución en el cierre.

int **register_shutdown_function** (string func) \linebreak

Registra la función nombrada en *func* para que se ejecute cuando el script procese su finalización.

Aviso:

Debido a que no se permite ningún tipo de salida en el navegador en esta función, no será capaz de depurarla utilizando sentencias como print o echo.

register_tick_function (PHP 4 >= 4.0.3)

Register a function for execution on each tick

void **register_tick_function** (string func [, mixed arg]) \linebreak

Registers the function named by *`func`* to be executed when a tick is called.

unregister_tick_function (PHP 4 >= 4.0.3)

De-register a function for execution on each tick

void **unregister_tick_function** (string func [, mixed arg]) \linebreak

De-registers the function named by *`func`* so it is no longer executed when a tick is called.

XXXV. GNU Gettext

bind_textdomain_codeset (PHP 4 >= 4.2.0)

Specify the character encoding in which the messages from the DOMAIN message catalog will be returned

string **bind_textdomain_codeset** (string domain, string codeset) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

bindtextdomain (PHP 3>= 3.0.7, PHP 4)

Establece la ruta para un dominio

string **bindtextdomain** (string domain, string directory) \linebreak

La función **bindtextdomain()** establece la ruta para el dominio.

dcgettext (PHP 3>= 3.0.7, PHP 4)

Omite el dominio para una única busqueda

string **dcgettext** (string domain, string message, int category) \linebreak

Esta función permite omitir el dominio actual para una busqueda de un mensaje. Además permite especificar una categoría.

dcngettext (PHP 4 >= 4.2.0)

Plural version of dcgettext

string **dcngettext** (string domain, string msgid1, string msgid2, int n, int category) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

dgettext (PHP 3>= 3.0.7, PHP 4)

Omite el dominio actual

string **dgettext** (string domain, string message) \linebreak

La función **dgettext()** permite omitir el dominio actual para una única busqueda.

dngettext (PHP 4 >= 4.2.0)

Plural version of dgettext

string **dngettext** (string domain, string msgid1, string msgid2, int n) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

gettext (PHP 3>= 3.0.7, PHP 4)

Realiza una busqueda del mensaje en el dominio actual

string **gettext** (string message) \linebreak

Esta función devuelve la traducción de la cadena si encuentra una en la tabla de traducciones, o el mensaje enviado si no se encuentra ninguna. Puede usar un carácter de subrayado como un alias para esta función.

Ejemplo 1. gettext()-check

```
<?php
// Establece el idioma en alemán
putenv ("LANG=de");

// Especifica la localización de las tablas de traducción
bindtextdomain ("myPHPApp", "./locale");

// Elige un dominio
```

```
textdomain ("myPHPApp");

// Imprime un mensaje de prueba
print (gettext ("Welcome to My PHP Application"));
?>
```

ngettext (PHP 4 >= 4.2.0)

Plural version of gettext

string **ngettext** (string msgid1, string msgid2, int n) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

textdomain (PHP 3>= 3.0.7, PHP 4)

Establece el dominio actual

int **textdomain** ([string library]) \linebreak

Esta función establece el dominio en el que se realizarán las busquedas provocadas por las llamadas a gettext(), normalmente el nombre dado a la aplicación. Se devuelve el dominio anterior. Puede llamar a la función sin parámetros para obtener el dominio actual sin necesidad de cambiarlo.

XXXVI. GMP functions

These functions allow you to work with arbitrary-length integers using GNU MP library. In order to have these functions available, you must compile PHP with GMP support by using the `--with-gmp` option.

You can download the GMP library from http://www.swox.com/gmp/. This site also has the GMP manual available.

You will need GMP version 2 or better to use these functions. Some functions may require more recent version of the GMP library.

These functions have been added in PHP 4.0.4.

> **Nota:** Most GMP functions accept GMP number arguments, defined as `resource` below. However, most of these functions will accept also numeric and string arguments, given it's possible to convert the latter to number. Also, if there's faster function that can operate on integer arguments, it would be used instead of slower function when supplied arguments are integers. This is done transparently, so the bootom line is that you can use integers in every function that expects GMP number. See also gmp_init() function.

Ejemplo 1. Factorial function using GMP

```
<?php
function fact ($x) {
    if ($x <= 1)
        return 1;
    else
        return gmp_mul ($x, fact ($x-1));
}

print gmp_strval (fact (1000)) . "\n";
?>
```

This will calculate factiorial of 1000 (pretty big number) very fast.

gmp_abs (PHP 4 >= 4.0.4)

Absolute value

resource **gmp_abs** (resource a) \linebreak

Returns absolute value of *a*.

gmp_add (PHP 4 >= 4.0.4)

Add numbers

resource **gmp_add** (resource a, resource b) \linebreak

Add two GMP numbers. The result will be GMP number representing the sum of the arguments.

gmp_and (PHP 4 >= 4.0.4)

Logical AND

resource **gmp_and** (resource a, resource b) \linebreak

Calculates logical AND of two GMP numbers.

gmp_clrbit (PHP 4 >= 4.0.4)

Clear bit

resource **gmp_clrbit** (resource &a, int index) \linebreak

Clears (sets to 0) bit *index* in *a*.

gmp_cmp (PHP 4 >= 4.0.4)

Compare numbers

int **gmp_cmp** (resource a, resource b) \linebreak

Returns positive value if `a > b`, zero if `a = b` and negative value if `a < b`.

gmp_com (PHP 4 >= 4.0.4)

Calculates one's complement of a

resource **gmp_com** (resource a) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

gmp_div_q (PHP 4 >= 4.0.4)

Divide numbers

resource **gmp_div_q** (resource a, resource b [, int round]) \linebreak

Divides *a* by *b* and returns the integer result. The result rounding is defined by the *round*, which can have the following values:

- *GMP_ROUND_ZERO*: The result is truncated towards 0.
- *GMP_ROUND_PLUSINF*: The result is rounded towards `+infinity`.
- *GMP_ROUND_MINUSINF*: The result is rounded towards `-infinity`.

This function can also be called as gmp_div().

See also gmp_div_r(), gmp_div_qr()

gmp_div_qr (PHP 4 >= 4.0.4)

Divide numbers and get quotient and remainder

array **gmp_div_qr** (resource n, resource d [, int round]) \linebreak

The function divides *n* by *d* and returns array, with the first element being `[n/d]` (the integer result of the division) and the second being `(n - [n/d] * d)` (the remainder of the division).

See the gmp_div_q() function for description of the *round* argument.

Ejemplo 1. Division of GMP numbers

```
<?php
    $a = gmp_init ("0x41682179fbf5");
    $res = gmp_div_qr ($a, "0xDEFE75");
    printf("Result is: q - %s, r - %s",
            gmp_strval ($res[0]), gmp_strval ($res[1]));
?>
```

See also gmp_div_q(), gmp_div_r().

gmp_div_r (PHP 4 >= 4.0.4)

Remainder of the division of numbers

resource **gmp_div_r** (resource n, resource d [, int round]) \linebreak

Calculates remainder of the integer division of *n* by *d*. The remainder has the sign of the *n* argument, if not zero.

See the gmp_div_q() function for description of the *round* argument.

See also gmp_div_q(), gmp_div_qr()

gmp_div (PHP 4 >= 4.0.4)

Divide numbers

resource **gmp_divexact** (resource a, resource b) \linebreak

This function is an alias to gmp_div_q().

gmp_divexact (PHP 4 >= 4.0.4)

Exact division of numbers

resource **gmp_divexact** (resource n, resource d) \linebreak

Divides *n* by *d*, using fast "exact division" algorithm. This function produces correct results only when it is known in advance that *d* divides *n*.

gmp_fact (PHP 4 >= 4.0.4)

Factorial

resource **gmp_fact** (int a) \linebreak

Calculates factorial (`a!`) of *a*.

gmp_gcd (PHP 4 >= 4.0.4)

Calculate GCD

resource **gmp_gcd** (resource a, resource b) \linebreak

Calculate greatest common divisor of *a* and *b*. The result is always positive even if either of or both input operands are negative.

gmp_gcdext (PHP 4 >= 4.0.4)

Calculate GCD and multipliers

array **gmp_gcdext** (resource a, resource b) \linebreak

Calculates g, s, and t, such that `a*s + b*t = g = gcd(a,b)`, where gcd is gretest common divisor. Returns array with respective elements g, s and t.

gmp_hamdist (PHP 4 >= 4.0.4)

Hamming distance

int **gmp_hamdist** (resource a, resource b) \linebreak

Returns the hamming distance between *a* and *a*. Both operands should be non-negative.

gmp_init (PHP 4 >= 4.0.4)

Create GMP number

resource **gmp_init** (mixed number) \linebreak

Creates a GMP number from integer or string. String representation can be decimal or hexadecimal. In the latter case, the string should start with `0x`.

Ejemplo 1. Creating GMP number

```
<?php
    $a = gmp_init (123456);
    $b = gmp_init ("0xFFFFDEBACDFEDF7200");
?>
```

Nota: It is not necessary to call this function if you want to use integer or string in place of GMP number in GMP functions, like gmp_add(). Function arguments are automatically converted to GMP numbers, if such conversion is possible and needed, using the same rules as **gmp_init()**.

gmp_intval (PHP 4 >= 4.0.4)

Convert GMP number to integer

int **gmp_intval** (resource gmpnumber) \linebreak

This function allows to convert GMP number to integer.

Aviso

This function returns useful result only if the number actually fits the PHP integer (i.e., signed long type). If you want just to print the GMP number, use gmp_strval().

gmp_invert (PHP 4 >= 4.0.4)

Inverse by modulo

resource **gmp_invert** (resource a, resource b) \linebreak

Computes the inverse of *a* modulo *b*. Returns FALSE if an inverse does not exist.

gmp_jacobi (PHP 4 >= 4.0.4)

Jacobi symbol

int **gmp_jacobi** (resource a, resource p) \linebreak

Computes Jacobi symbol (http://www.utm.edu/research/primes/glossary/JacobiSymbol.html) of *a* and *p*. *p* should be odd and must be positive.

gmp_legendre (PHP 4 >= 4.0.4)

Legendre symbol

int **gmp_legendre** (resource a, resource p) \linebreak

Compute the Legendre symbol (http://www.utm.edu/research/primes/glossary/LegendreSymbol.html) of *a* and *p*. *p* should be odd and must be positive.

gmp_mod (PHP 4 >= 4.0.4)

Modulo operation

resource **gmp_mod** (resource n, resource d) \linebreak

Calculates *n* modulo *d*. The result is always non-negative, the sign of *d* is ignored.

gmp_mul (PHP 4 >= 4.0.4)

Multiply numbers

resource **gmp_mul** (resource a, resource b) \linebreak

Multiplies *a* by *b* and returns the result.

gmp_neg (PHP 4 >= 4.0.4)

Negate number

resource **gmp_neg** (resource a) \linebreak

Returns -*a*.

gmp_or (PHP 4 >= 4.0.4)

Logical OR

resource **gmp_or** (resource a, resource b) \linebreak

Calculates logical inclusive OR of two GMP numbers.

gmp_perfect_square (PHP 4 >= 4.0.4)

Perfect square check

bool **gmp_perfect_square** (resource a) \linebreak

Returns `TRUE` if *`a`* is a prefect square, `FALSE` otherwise.

See also: gmp_sqrt(), gmp_sqrtrm().

gmp_popcount (PHP 4 >= 4.0.4)

Population count

int **gmp_popcount** (resource a) \linebreak

Return the population count of *`a`*.

gmp_pow (PHP 4 >= 4.0.4)

Raise number into power

resource **gmp_pow** (resource base, int exp) \linebreak

Raise *`base`* into power *`exp`*. The case of 0^0 yields 1. *`exp`* cannot be negative.

gmp_powm (PHP 4 >= 4.0.4)

Raise number into power with modulo

resource **gmp_powm** (resource base, resource exp, resource mod) \linebreak

Calculate (*`base`* raised into power *`exp`*) modulo *`mod`*. If *`exp`* is negative, result is undefined.

gmp_prob_prime (PHP 4 >= 4.0.4)

Check if number is "probably prime"

int **gmp_prob_prime** (resource a [, int reps]) \linebreak

If this function returns 0, *a* is definitely not prime. If it returns 1, then *a* is "probably" prime. If it returns 2, then *a* is surely prime. Reasonable values of *reps* vary from 5 to 10 (default being 10); a higher value lowers the probability for a non-prime to pass as a "probable" prime.

The function uses Miller-Rabin's probabilistic test.

gmp_random (PHP 4 >= 4.0.4)

Random number

resource **gmp_random** (int limiter) \linebreak

Generate a random number. The number will be up to *limiter* words long. If *limiter* is negative, negative numbers are generated.

gmp_scan0 (PHP 4 >= 4.0.4)

Scan for 0

int **gmp_scan0** (resource a, int start) \linebreak

Scans *a*, starting with bit *start*, towards more significant bits, until the first clear bit is found. Returns the index of the found bit.

gmp_scan1 (PHP 4 >= 4.0.4)

Scan for 1

int **gmp_scan1** (resource a, int start) \linebreak

Scans *a*, starting with bit *start*, towards more significant bits, until the first set bit is found. Returns the index of the found bit.

gmp_setbit (PHP 4 >= 4.0.4)

Set bit

resource **gmp_setbit** (resource &a, int index [, bool set_clear]) \linebreak

Sets bit *index* in *a*. *set_clear* defines if the bit is set to 0 or 1. By default the bit is set to 1.

gmp_sign (PHP 4 >= 4.0.4)

Sign of number

int **gmp_sign** (resource a) \linebreak

Return sign of *a* - 1 if *a* is positive and -1 if it's negative.

gmp_sqrt (PHP 4 >= 4.0.4)

Square root

resource **gmp_sqrt** (resource a) \linebreak

Calculates square root of *a*.

gmp_sqrtrm (unknown)

Square root with remainder

array **gmp_sqrtrm** (resource a) \linebreak

Returns array where first element is the integer square root of *a* (see also gmp_sqrt()), and the second is the remainder (i.e., the difference between *a* and the first element squared).

gmp_strval (PHP 4 >= 4.0.4)

Convert GMP number to string

string **gmp_strval** (resource gmpnumber [, int base]) \linebreak

Convert GMP number to string representation in base *base*. The default base is 10. Allowed values for the base are from 2 to 36.

Ejemplo 1. Converting GMP number to string

```
<?php
    $a = gmp_init("0x41682179fbf5");
    printf ("Decimal: %s, 36-based: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_sub (PHP 4 >= 4.0.4)

Subtract numbers

resource **gmp_sub** (resource a, resource b) \linebreak

Subtract *b* from *a* and returns the result.

gmp_xor (PHP 4 >= 4.0.4)

Logical XOR

resource **gmp_xor** (resource a, resource b) \linebreak

Calculates logical exclusive OR (XOR) of two GMP numbers.

XXXVII. Funciones HTTP

Estas funciones permiten manipular la salida que se envía al navegador remoto a nivel de protocolo HTTP.

header (PHP 3, PHP 4)

Manda una cabecera HTTP

int **header** (string string) \linebreak

La función **header()** se utiliza al comienzo de un fichero HTML para enviar las cadenas de texto de la cabecera HTTP. Consulte la Especificación HTTP 1.1 (http://www.w3.org/Protocols/rfc2616/rfc2616) para obtener más información sobre las cabeceras http. *Nota:* Recuerde que la función **header()** debe llamarse antes de que se genere salida alguna, bien con etiquetas HTML normales o con PHP. Un error muy frecuente consiste en leer código con include() o con auto_prepend, y que dicho código inserte espacios o líneas en blanco antes de llamar a **header()**.

Hay dos casos especiales de llamadas a header. La primera es la cabecera "Location". No sólo envía esta cabecera al navegador, sino que también devuelve un código de estado REDIRECT a Apache. Desde el punto de vista del programador de scripts esto no debería ser importante, pero para la gente que comprende las interioridades de Apache sí puede serlo.

```
header("Location: http://www.php.net");  /* Redirect browser to PHP web site */
exit;  /* Make sure that code below does not get executed when we redirect. */
```

El segundo caso especial se produce con cualquier cabecera que comience con la cadena, "HTTP/" (las mayúsculas no son significativas). Por ejemplo, si tiene la directiva ErrorDocument 404 de Apache apuntando a un script PHP, es una buena idea asegurarse de que su script de PHP genera realmente un 404. La primera cosa que debe hacer en su script sería:

```
header("http/1.0 404 Not Found");
```

Los scripts de PHP a menudo generan HTML dinámico que no debe almacenarse en la caché del navegador cliente o en la caché de cualquier proxy situado entre el servidor y el navegador cliente. Se puede obligar a muchos proxies y clientes a que deshabiliten el almacenamiento en caché con

```
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");             // Date in the past
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT"); // always modified
header("Cache-Control: no-cache, must-revalidate");           // HTTP/1.1
header("Pragma: no-cache");                                   // HTTP/1.0
```

headers_sent (PHP 3>= 3.0.8, PHP 4)

Returns `TRUE` if headers have been sent

bool **headers_sent** (void) \linebreak

This function returns `TRUE` if the HTTP headers have already been sent, `FALSE` otherwise.

See also header()

setcookie (PHP 3, PHP 4)

Envía una cookie

int **setcookie** (string name, string value, int expire, string path, string domain, int secure) \linebreak

setcookie() define una cookie para ser enviada con el resto de la información de la cabecera. Las cookies deben enviarse *antes* de mandar cualquier otra cabecera (esta es una restricción de las cookies, no de PHP). Esto requiere que sitúe las llamadas a esta función antes de cualquier etiqueta `<html>` o `<head>`.

Todos los parámetros excepto *`name`* son opcionales. Si sólo se especifica el parámetro name, la cookie con ese nombre se borrará del cliente remoto. También puede sustituir cualquier parámetro por una cadena de texto vacía (*""*) y saltar así ese parámetro. Los parámetros *`expire`* y *`secure`* son números enteros y no se pueden saltar con una cadena de texto vacía. En su lugar utilice un cero (*0*). El parámetro *`expire`* es un entero de tiempo típico de UNIX tal como lo devuelven las funciones time() o mktime(). El parámetro *`secure`* indica que la cookie se debe transmitir única y exclusivamente sobre una conexión segura HTTPS.

Fallos habituales:

Las cookies no se hacen visibles hasta la siguiente carga de una página para la que la cookie deba estar visible.

Las llamadas múltiples a **setcookie()** en el mismo script se ejecutarán en orden inverso. Si está intentando borrar una cookie antes de insertar otra, debe situar la llamada de inserción antes de la de borrado.

A continuación se muestran algunos ejemplos::

Ejemplo 1. setcookie(), ejemplos

```
setcookie("TestCookie","Test Value");
setcookie("TestCookie",$value,time()+3600);  /* expire in 1 hour */
setcookie("TestCookie",$value,time()+3600,"/~rasmus/",".utoronto.ca",1);
```

Tenga en cuenta que el campo value de la cookie se codifica como URL (urlencode) automáticamente cuando envía la cookie. Cuando ésta se recibe, se descodifica autómaticamente y se asigna a una variable con el mismo nombre que el nombre de la cookie. Para ver el contenido de nuestra cookie de prueba en un script, simplemente utilice uno de los siguientes ejemplos:

```
echo $TestCookie;
echo $HTTP_COOKIE_VARS["TestCookie"];
```

También puede utilizar arrays de cookies empleando la notación de array en el nombre de la cookie. Esto tiene como efecto establecer tantas cookies como elementos de array, pero cuando el script recibe la cookie, se guardan los valores en un array con el nombre de la cookie:

```
setcookie( "cookie[three]", "cookiethree" );
setcookie( "cookie[two]", "cookietwo" );
setcookie( "cookie[one]", "cookieone" );
if ( isset( $cookie ) ) {
   while( list( $name, $value ) = each( $cookie ) ) {
  echo "$name == $value<br>\n";
   }
}
```

Para obtener más información sobre las cookies, consulte la especificación de cookies de Netscape, que se encuentra en http://www.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 con Service Pack 1 no funciona correctamente con las cookies que tienen asociado el parámetro path.

Netscape Communicator 4.05 y Microsoft Internet Explorer 3.x funcionan aparentemente de manera incorrecta cuando no se especifican los parámetros path y time.

XXXVIII. Funciones para Hyperwave

Introducción

Hyperwave ha sido desarrollado en el IICM (http://www.iicm.edu) en Graz. Comenzó con el nombre Hyper-G y cambió a Hyperwave cuando fue comercializado (Si lo recuerdo bien, fue en 1996).

Hyperwave no es software gratuiro. La versión actual, 4.1, está disponible en www.hyperwave.com (http://www.hyperwave.com/). Se puede solicitar gratuitamente una versión limitada (30 días).

Hyperwave es un sistema de información similar a una base de datos (HIS, Hyperwave Information Server - Servidor Hyperwave de Información). Su objetivo es el almacenamiento y manipulación de documentos. Un documento puede ser cualquier bloque posible de datos que también puede ser almacenado en un archivo. Cada documento se acompaña por su registro de objeto. El registro de objeto contiene metadatos para el documento. Los metadatos son una lista de atributos que pueden ser extendidos por el usuario. Ciertos atributos siempre son fijados por el servidor Hyperwave, otros pueden ser modificados por el usuario. Un atributo es un par nombre/valor de la forma nombre=valor. El registro completo del objeto tiene tantos de estos pares como guste el usuario. El nombre de un atributo no tiene porqué ser único, p. ej. un título puede aparecer varias veces en el registro de un objeto. Esto tiene sentido si se desea especificar un título en diferentes idiomas. En dicho caso existe la convención de que cada valor de título esté precedido por la abreviatura de dos letras del idioma, seguida por dos puntos, como p. ej. ’en:Title in English’ o ’es:Título en Español’. Otros atributos tales como descripciones o palabras clave son canditatos potenciales a esta diferenciación. También se pueden reemplazar las abreviaturas de idioma por cualquier otra cadena siempre y cuando estén separadas por los dos puntos del resto del valor del atributo.

Cada registro de objeto tiene una representación nativa como cadena con cada par nombre/valor separado por una línea nueva. La extensión Hyperwave también conoce una segunda representación que consiste en un array asociativo donde el nombre del atributo es la clave. Los valores de atributo multilingües en sí mismos forman otro array asociativo donde la clave es la abreviatura del idioma. Realmente cualquier atributo múltiple forma una tabla asociativa donde la cadena a la izquierda de los dos puntos en el valor de atributo es la clave. (Esto no se ha implementado por completo. Sólo los atributos Title, Description y Keyword son tratados adecuadamente.)

Aparte de los documentos, todos los hiper-enlaces contenidos en un documento son almacenados también como registros de objeto. Cuando el documento sea insertado en la base de datos, los hiper-enlaces que haya en un documento serán borrados del mismo y almacenados como objetos individuales. El registro de objeto del enlace contiene información acerca de dónde comienza y dónde termina. Para recuperar el documento original se deberá recuperar el documento sin los enlaces y la lista de los mismos para reinsertarla (Las funciones hw_pipedocument() y hw_gettext() hacen esto para usted). La ventaja de separar los enlaces del documento es obvia. Una vez un documento al que apunta un enlace cambia de nombre, el enlace puede modificarse fácilmente. El documento que contiene el enlace no se ve afectado. Incluso se puede añadir un enlace a un documento sin alterarlo.

Decir que hw_pipedocument() y hw_gettext() hacen automáticamente la inserción de enlaces no es tan simple como suena. Insertar los enlaces implica una cierta jerarquía en los documentos. En un servidor web esto viene dado por el sistema de archivos, pero el Hyperwave tiene su propia jerarquía y los nombres no representan la posición de un objeto en dicha jerarquía. Por tanto, la creación de los enlaces precisa primeramente de realizar un mapeado entre el espacio de nombres y la jerarquía del Hyperwave y

el espacio de nombres respectivo de una jerarquía de web. La diferencia fundamental entre Hyperwave y la web es la distinción clara entre nombres y jerarquía que se da en el primero. El nombre no contiene ninguna información sobre la posición del objeto en la jerarquía. En la web, el nombre también contiene la información sobre la posición en la jerarquía del objeto. Esto nos lleva a dos posibles formas de mapeo. O bien se reflejan la jerarquía del Hyperwave y el nombre del objeto Hyperwave en el URL o sólo el nombre. Para facilitar las cosas, se utiliza el segundo método. El objeto Hyperwave de nombre ’mi_objeto’ es mapeado a ’http://host/mi_objeto’ sin importar dónde reside dentro de la jerarquía de Hyperwave. Un objeto con el nombre ’padre/mi_objeto’ podría ser el hijo de ’mi_objeto’ en la jerarquía Hyperwave, aunque en el espacio de nombres web aparezca justamente lo opuesto y el usuario pueda ser llevado a confusión. Esto sólo se puede evitar seleccionando nombres de objetos razonables.

Hecha esta decisión surge un segundo problema. ¿Cómo implicar al PHP? el URL http://host/mi_objeto no llamará a ningún script PHP a no ser que se le diga al servidor que lo transforme en p. ej. ’http://host/script_php3/mi_objeto’ y que el ’script_php3’ luego evalúe la variable $PATH_INFO y recupere el objeto con nombre ’mi_objeto’ del servidor Hyperwave. Hay sólo un pequeño inconveniente que se puede resolver fácilmente. Cuando se reescribe cualquier URL no se permite el acceso a ningún otro documento en el servidor web. Un script de PHP para buscar en el servidor Hyperwave sería imposible. Por lo tanto se necesitará al menos una segunda regla de reescritura para que excluya ciertos URL, como los que empiecen p. ej. por http://host/Hyperwave. Básicamente esto sería compartir un espacio de nombres entre el servidor web y el servidor Hyperwave.

Los enlaces se insertan en los documentos basándose en el mecanismo citado más arriba.

Se vuelve más complicado si el PHP no se está ejecutando como módulo del servidor o como script CGI, sino que se ejecuta como aplicación, p. ej. para volcar el contenido del servidor de Hyperwave a un CD-ROM. En dicho caso tiene sentido mantener la jerarquía Hyperwave y mapearla en el sistema de archivos. Esto entra conflicto con los nombres de los objetos si estos reflejan su propia jerarquía (p. ej. eligiendo nombres que comienzan por ’/’). Por tanto, la ’/’ tendrá que ser reemplazada por otro carácter, p. ej. ’_’ para continuar.

El protocolo de red para comunicarse con el servidor Hyperwave se denomina HG-CSP (http://www.hyperwave.de/7.17-hg-prot) (Hyper-G Client/Server Protocol, Protocolo Hyper-G Cliente/Servidor). Está basado en mensajes que inician ciertas acciones, p. ej. obtener el registro de un objeto. En versiones anteriores del Servidor Hyperwave se distribuyeron dos clientes nativos (Harmony, Amadeus) para la comunicación con el servidor. Ambos desaparecieron cuando se comercializó el Hyperwave. Para sustituírlo se proporcionó el llamado wavemaster. El wavemaster es como un conversor de protocolo de HTTP a HG-CSP. La idea es realizar toda la administración de la base de datos y la visualización de documentos con una interfaz web. El wavemaster implementa una serie de posicionadores para acciones que permiten personalizar la interfaz. Dicho conjunto de posicionadores se denomina el Lenguaje PLACE. El PLACE no posee muchas de las características de un lenguaje de programación real y las extensiones al mismo únicamente alargan la lista de posicionadores. Esto ha obligado al uso de JavaScript que, en mi opinión, no hace la vida más fácil.

Añadir soporte de Hyperwave al PHP rellenaría el espacio que deja la falta de un lenguaje de programación que permita personalizar la interfaz. El PHP implementa todos los mensajes definidos en el HG-CSP pero además proporciona comandos más poderosos, p. ej. recuperar documentos completos.

El Hyperwave tiene su propia terminología para dar nombre a ciertos tipos de información. Esta ha sido ampliamente extendida y anulada. Casi todas las funciones operan en uno de los siguientes tipos de

datos.

- ID de objeto: Un valor entero único paara cada objeto del servidor Hyperwave. También es uno de los atributos del registro de objeto (ObjectID). Los ID de objeto se usan generalmente como parámetros de entrada que especifican a un objeto.
- registro de objeto: Una cadena con pares atributo-valor con la forma atributo=valos. Los pares están separados unos de otros por un retorno de carro. Un registro de objeto puede convertirse fácilmente en una tabla (array) de objetos usando **hw_object2array()**. Varias funciones devuelven registros de objeto. Los nombres de dichas funciones terminan en obj.
- tabla de objetos: Una tabla asociativa con todos los atributos de un objeto. La clave es el nombre del atributo. Si un atributo aparece más de una vez en un registro de objeto será convertido en otra tabla asociativa o indizada. Los atributos que dependen del idioma (como el título, claves o descripción) se convertirán en una tabla asociativa con la abreviatura del idioma como clave. El resto de los atributos múltiples crearán una tabla indizada. Las funciones de PHP nunca devuelven tablas de objetos.
- hw_document: Este es un nuevo tipo de datos que almacena el documento actual, p. ej. HTML, PDF, etc. Está algo optimizado para documentos HTML pero puede usarse para cualquier formato.

Varias funciones que devuelven una tabla de registros de objeto también devuelven una tabla asociativa con información estadística sobre los mismos. La tabla es el último elemento del registro de objeto. La tabla estadística contiene los siguientes elementos:

Hidden

Número de registros de objeto con el atributo PresentationHints puesto a Hidden.

CollectionHead

Número de registros de objeto con el atributo PresentationHints puesto a CollectionHead.

FullCollectionHead

Número de registros de objeto con el atributo PresentationHints puesto a FullCollectionHead.

CollectionHeadNr

Índice a una tabla de regitros de objeto con el atributo PresentationHints puesto a CollectionHead.

FullCollectionHeadNr

Índice a una tabla de regitros de objeto con el atributo PresentationHints puesto a

FullCollectionHead.

Total

Total: Número de registros de objeto.

Integración con Apache

La extensión Hyperwave se utiliza mejor cuando el PHP se compila como un módulo de Apache. En tal caso el servidor Hyperwave subyacente puede ser ocultado casi por completo de los usuarios si el Apache utiliza su motor de re-escritura. Las siguientes instrucciones explicarán esto.

Como el PHP con soporte Hyperwave incluído en el Apache se ha diseñado para sustituir la solución nativa de Hyperwave basada en Wavemaster, asumiré que el servidor Apache sólo sirve como interfaz web para el Hyperwave. Esto no es necesario, pero simplifica la configuración. El concepto es bastante sencillo. Primeramente necesita un script PHP que evalúe la variable PATH_INFO y que trate su valor como el nombre de un objeto Hyperwave. Llamemos a este script 'Hyperwave'. El URL http://nombre.servidor/Hyperwave/nombre_de_objeto devolvería entonces el objeto Hyperwave llamado 'nombre_de_objeto'. Dependiendo del tipo del objeto, así reaccionará el script. Si es una colección, probablemente devolverá un lista de hijos. Si es un documento devolverá el tipo MIME y el contenido. Se puede mejorar ligeramente si se usa el motor de re-escritura del Apache. Desde el punto de vista del usuario será más sencillo si el URL http://nombre.servidor/nombre de objeto devuelve el objeto. La regla de reescritura es muy sencilla:

```
RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
```

Ahora todo URL apunta a un objeto en el servidor Hyperwave. Esto provoca un problema sencillo de resolver. No hay forma de ejecutar otro script, p. ej. para buscar, salvo el script 'Hyperwave'. Esto se puede solucionar con otra regla de reescritura como la siguiente:

```
RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
```

Esta reservará el directorio `/usr/local/apache/htdocs/hw` para script adicionales y otros archivos. Sólo hay que asegurarse que esta regla se evalúa antes de la otra. Sólo hay un pequeño inconveniente: todos los objetos Hyperwave cuyo nombre comienza por 'hw/' serán ocultados. así que asegúrese que no utiliza dichos nombres. Si necesita más directorios, p. ej. para imágenes, simplemente añada más reglas o sitúe los archivos en un solo directorio. Por último, no olvide conectar el motor de re-escritura con

```
RewriteEngine on
```

Mi experiencia me ha demostrado que necesitará los siguientes scripts:

- para devolver el script en sí
- para permitir las búsquedas
- para identificarse
- para ajustar su perfil
- uno para cada función adicional como mostrar los atributos del objeto, mostrar información sobre usuarios, mostrar el estado del servidor, etc.

Pendientes

Aún hay varias cosas pendientes:

- hw_InsertDocument debe dividirse en **hw_InsertObject()** y **hw_PutDocument()**.
- Los nombres de algunas funciones aún no están fijados.
- Muchas funciones precisan la conexión actual como primer parámetro. Esto obliga a escribir mucho, lo cual no es con frecuencia necesario si sólo hay una conexión abierta. Una conexión por defecto mejoraría esto.
- La conversión de registro de objeto a tabla de objeto necesita manipular cualquier atributo múltiple.

hw_Array2Objrec (PHP 3>= 3.0.4, PHP 4)

convierte atributos de tabla de objeto a registro de objeto

strin **hw_array2objrec** (array tabla_objetos) \linebreak

Convierte una *tabla_objetos* en un registro de objeto. Los atributos múltiples como 'Título' en distintos idiomas son tratados correctamente.

Vea también hw_objrec2array().

hw_changeobject (PHP 3>= 3.0.3, PHP 4)

Changes attributes of an object (obsolete)

void **hw_changeobject** (int link, int objid, array attributes) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

hw_Children (PHP 3>= 3.0.3, PHP 4)

id de objeto de los hijos

array **hw_children** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla de id de objeto. Cada uno de ellos pertenece a un hijo de la colección cuyo ID es *IDobjeto*. La tabla contiene tanto los documentos como las colecciones hijas.

hw_ChildrenObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de los hijos

array **hw_childrenobj** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla de registros de objeto. Cada uno de ellos pertenece a un hijo de la colección cuyo ID es *IDobjeto*. La tabla contiene tanto los documentos como las colecciones hijas.

hw_Close (PHP 3>= 3.0.3, PHP 4)

cierra la conexión Hyperwave

int **hw_close** (int conexión) \linebreak

Devuelve `FALSE` si la conexión no es un índice válido de conexión, y `TRUE` en caso contrario. Cierra la conexión dada con el servidor de Hyperwave.

hw_Connect (PHP 3>= 3.0.3, PHP 4)

abre una conexión

int **hw_connect** (string servidor, int puerto, string usuario, string clave) \linebreak

Abre una conexión con un servidor Hyperwave y devuelve un índice de conexión si hay éxito, o `FALSE` si la conexión no se pudo realizar. Cada argumento debe ser una cadena entrecomillada salvo el número de puerto. Los argumentos de *`usuario`* y *`clave`* son opcionales y pueden omitirse. En tal caso no se realizará identificación alguna con el servidor. Es similar a identificarse como el usuario 'anonymous'. Esta función devuelve un índice de conexión que se necesita para otras funciones Hyperwave. Puede tener varias conexiones abiertas a la vez. Recuerde que la clave no está encriptada.

Vea también **hw_pConnect()**.

hw_connection_info (PHP 3>= 3.0.3, PHP 4)

Prints information about the connection to Hyperwave server

void **hw_connection_info** (int link) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

hw_Cp (PHP 3>= 3.0.3, PHP 4)

copia objetos

int **hw_cp** (int conexión, array tabla_id_objeto, int id destino) \linebreak

Copia los objetos cuyos id se especifican en el segundo parámetro a la colección identificada como *`id destino`*.

El valor devuelto es el número de objetos copiados.

Vea también hw_mv().

hw_Deleteobject (PHP 3>= 3.0.3, PHP 4)

borra objetos

int **hw_deleteobject** (int conexión, int objeto_a_borrar) \linebreak

Borra el objeto con el id dado como segundo parámetro. Borrará todas las instancias del mismo.

Devuelve `TRUE` si no hay error o `FALSE` en caso contrario.

Vea también hw_mv().

hw_DocByAnchor (PHP 3>= 3.0.3, PHP 4)

id del objeto al que pertenece un enlace

int **hw_docbyanchor** (int conexión, int IDenlace) \linebreak

Devuelve el id de objeto del documento al que pertenece el *`IDenlace`*.

hw_DocByAnchorObj (PHP 3>= 3.0.3, PHP 4)

registro de objeto al que pertenece un enlace

string **hw_docbyanchorobj** (int conexión, int IDenlace) \linebreak

Devuelve el registro de objeto del documento al que pertenece el *`IDenlace`*.

hw_Document_Attributes (PHP 3>= 3.0.3, PHP 4)

object record of hw_document

string **hw_document_attributes** (int hw_document) \linebreak

Returns the object record of the document.

For backward compatibility, **hw_documentattributes()** is also accepted. This is deprecated, however.

See also hw_document_bodytag(), and hw_document_size().

hw_Document_BodyTag (PHP 3>= 3.0.3, PHP 4)

body tag of hw_document

string **hw_document_bodytag** (int hw_document) \linebreak

Returns the BODY tag of the document. If the document is an HTML document the BODY tag should be printed before the document.

See also hw_document_attributes(), and hw_document_size().

For backward compatibility, **hw_documentbodytag()** is also accepted. This is deprecated, however.

hw_Document_Content (PHP 3>= 3.0.3, PHP 4)

returns content of hw_document

string **hw_document_content** (int hw_document) \linebreak

Returns the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record.

See also hw_document_attributes(), hw_document_size(), and hw_document_setcontent().

hw_Document_SetContent (PHP 4)

sets/replaces content of hw_document

string **hw_document_setcontent** (int hw_document, string content) \linebreak

Sets or replaces the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record. If you provide this information in the content of the document too, the Hyperwave server will change the object record accordingly when the document is inserted. Probably not a very good idea. If this functions fails the document will retain its old content.

See also hw_document_attributes(), hw_document_size(), and hw_document_content().

hw_Document_Size (PHP 3>= 3.0.3, PHP 4)

size of hw_document

int **hw_document_size** (int hw_document) \linebreak

Returns the size in bytes of the document.

See also hw_document_bodytag(), and hw_document_attributes().

For backward compatibility, **hw_documentsize()** is also accepted. This is deprecated, however.

hw_dummy (PHP 3>= 3.0.3, PHP 4)

Hyperwave dummy function

string **hw_dummy** (int link, int id, int msgid) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

hw_EditText (PHP 3>= 3.0.3, PHP 4)

recupera documento de texto

int **hw_edittext** (int conexión, int documento_hw) \linebreak

Envía el documento de texto al servidor. El registro de objeto del documento no puede ser modificado mientras el documento es editado. Esta función sólo funcionará para objetos puros de texto. No abrirá ninguna conexión especial de datos y por tanto bloquea la conexión de control durante la transferencia.

Vea también **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_DocumentBodyTag()**, **hw_DocumentSize()**, **hw_OutputDocument()**, **hw_GetText()**.

hw_Error (PHP 3>= 3.0.3, PHP 4)

número de error

int **hw_error** (int conexión) \linebreak

Devuelve el último número de error. Si el valor devuelto es 0 no ha habido errores. El error está relacionado con el último comando.

hw_ErrorMsg (PHP 3>= 3.0.3, PHP 4)

devuelve un mensaje de error

string **hw_errormsg** (int conexión) \linebreak

Devuelve una cadena que contiene el último mensaje de error o 'No Error'. Si devuelve `FALSE` es que la función fracasó. El mensaje está relacionado con el último comando.

hw_Free_Document (PHP 3>= 3.0.3, PHP 4)

libera un documento_hw

int **hw_free_document** (int documento_hw) \linebreak

Libera la memoria ocupada por el documento Hyperwave.

hw_GetAnchors (PHP 3>= 3.0.3, PHP 4)

id de objeto de los enlaces de un documento

array **hw_getanchors** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla de id de objeto con los enlaces del documento cuyo ID es *`IDobjeto`*.

hw_GetAnchorsObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de los enlaces de un documento

array **hw_getanchorsobj** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla de registros de objeto con los enlaces del documento cuyo ID es *`IDobjeto`*.

hw_GetAndLock (PHP 3>= 3.0.3, PHP 4)

devuelve registro de objeto y lo bloquea

string **hw_getandlock** (int conexión, int IDobjeto) \linebreak

Devuelve el registro de objeto para el objeto con ID *`IDobjeto`*. También bloqueará el objeto, de modo que otros usuarios no podrán acceder al mismo hasta que sea desbloqueado.

Vea también **hw_Unlock()**, **hw_GetObject()**.

hw_GetChildColl (PHP 3>= 3.0.3, PHP 4)

id de objeto de colecciones hijas

array **hw_getchildcoll** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla de id de objeto. Cada ID de objeto pertenece a una colección hija de la colección con ID *IDobjeto*. La función no devolverá documentos hijos.

Vea también **hw_GetChildren()**, **hw_GetChildDocColl()**.

hw_GetChildCollObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de colecciones hijas

array **hw_getchildcollobj** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla de registros de objeto. Cada uno de ellos pertenece a una colección hija de la colección con ID *IDobjeto*. La función no devolverá documentos hijos.

Vea también **hw_ChildrenObj()**, **hw_GetChildDocCollObj()**.

hw_GetChildDocColl (PHP 3>= 3.0.3, PHP 4)

id de objeto de documentos hijos de una colección

array **hw_getchilddoccoll** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla de id de objeto para los documentos hijos de una colección.

Vea también **hw_GetChildren()**, **hw_GetChildColl()**.

hw_GetChildDocCollObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de documentos hijos de una colección

array **hw_getchilddoccollobj** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla de registros de objeto para los documentos hijos de una colección.

Vea también **hw_ChildrenObj()**, **hw_GetChildCollObj()**.

hw_GetObject (PHP 3>= 3.0.3, PHP 4)

registro de objeto

array **hw_getobject** (int conexión, [int|array] IDobjeto, string consulta) \linebreak

Devuelve el registro de objeto para el objeto cuyo ID es *`IDobjeto`* si el segundo parámetro es un entero. Si es una tabla la función devolverá una tabla de registros de objeto. En tal caso, el último parámetro, que es una cadena de consulta, también es evaluado.

La cadena de consulta tiene la sintáxis siguiente:

<expr> ::= "(" <expr> ")" |

"!" <expr> | /* NO */

<expr> "||" <expr> | /* O */

<expr> "&&" <expr> | /* Y */

<atributo> <operador> <valor>

<atributo> ::= /* cualquier atributo (Título, Autor, TipoDocumento ...) */

<operador> ::= "=" | /* igual */

"<" | /* menor que (comparación de cadenas) */

">" | /* mayor que (comparación de cadenas) */

"~" /* expresión regular */

La consulta permite seleccionar elementos de la lista de objetos dada. Al contrario de otras funciones de búsqueda, esta consulta no puede utilizar atributos indizados. El número de registros de objeto devueltos depende de la consulta y de si está permitido el acceso al objeto.

Vea también **hw_GetAndLock()**, **hw_GetObjectByQuery()**.

hw_GetObjectByQuery (PHP 3>= 3.0.3, PHP 4)

buscar objeto

array **hw_getobjectbyquery** (int conexión, string consulta, int max_resultados) \linebreak

Busca objetos en todo el servidor y devuelve una tabla de id de objeto. El número máximo de resultados será *`max_resultados`*. Si *`max_resultados`* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQueryObj()**.

hw_GetObjectByQueryColl (PHP 3>= 3.0.3, PHP 4)

buscar objeto en colección

array **hw_getobjectbyquerycoll** (int conexión, int IDobjeto, string consulta, int max_resultados) \linebreak

Busca objetos en la colección cuyo ID es *`IDobjeto`* y devuelve una tabla de ID de objeto. El número máximo de resultados será *`max_resultados`*. Si *`max_resultados`* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQueryCollObj()**.

hw_GetObjectByQueryCollObj (PHP 3>= 3.0.3, PHP 4)

buscar objeto en colección

array **hw_getobjectbyquerycollobj** (int conexión, int IDobjeto, string consulta, int max_resultados) \linebreak

Busca objetos en la colección cuyo ID es *`IDobjeto`* y devuelve una tabla de registros de objeto. El número máximo de resultados será *`max_resultados`*. Si *`max_resultados`* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQueryColl()**.

hw_GetObjectByQueryObj (PHP 3>= 3.0.3, PHP 4)

buscar objeto

array **hw_getobjectbyqueryobj** (int conexión, string consulta, int max_resultados) \linebreak

Busca objetos en todo el servidor y devuelve una tabla de registros de objeto. El número máximo de resultados será *`max_resultados`*. Si *`max_resultados`* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQuery()**.

hw_GetParents (PHP 3>= 3.0.3, PHP 4)

id de objeto de los padres

array **hw_getparentsobj** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla indizada de id de objeto. Cada una pertenece a un padre del objeto con ID *IDobjeto*.

hw_GetParentsObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de los padres

array **hw_getparentsobj** (int conexión, int IDobjeto) \linebreak

Devuelve una tabla indizada de registros de objeto junto a una tabla asociativa con información estadística sobre estos. La tabla asociativa es el último elemento de la tabla devuelta. Cada registro de objeto pertenece a un padre del objeto con ID *IDobjeto*.

hw_getrellink (PHP 3>= 3.0.3, PHP 4)

Get link from source to dest relative to rootid

string **hw_getrellink** (int link, int rootid, int sourceid, int destid) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

hw_GetRemote (PHP 3>= 3.0.3, PHP 4)

Obtiene un documento remoto

int **hw_getremote** (int conexión, int IDobjeto) \linebreak

Devuelve un documento remoto. Los documentos remotos en la notación de Hyperwave son los obtenidos de una fuente externa. Los documentos remotos típicos son páginas web externas o consultas a bases de datos. Para poder acceder a las fuentes externas a través de documentos remotos, el Hyperwave presenta el HGI (Hyperwave Gateway Interface - Interfaz de Pasarela de Hyperwave) que es similar al CGI. Actualmente, sólo se puede acceder a servidores ftp y http y a algunas bases de datos. Llamar a **hw_GetRemote()** devuelve el documento de la fuente externa. Si desea usar esta función debe familiarizarse con los HGI. Debería considerar el usar PHP en lugar del Hyperwave para acceder a fuentes externas. Añadir soporte de bases de datos a través de una pasarela Hyperwave sería más difícil que hacerlo en PHP.

Vea también **hw_GetRemoteChildren()**.

hw_GetRemoteChildren (PHP 3>= 3.0.3, PHP 4)

Obtiene el hijo del documento remoto

int **hw_getremotechildren** (int conexión, string registro de objeto) \linebreak

Devuelve el hijo de un documento remoto. Los hijos de documentos remotos son en sí mismos documentos remotos. Esto tiene sentido cuando se afina una consulta de bases de datos y se explica en la Guía de Programación de Hyperwave. Si el número de hijos es 1 la función devolverá el documento en sí mismo formateado con la Interfaz de Pasarela de Hyperwave (HGI). Si el número de hijos es mayor de 1 devolverá una tabla de registros de objeto, con cada uno posible candidato para otra llamada a **hw_GetRemoteChildren()**. Dichos registros de objeto son virtuales y no existen en el servidor Hyperwave, y por lo tanto no poseen un ID de objeto válido. La apariencia exacta de dicho registro de objeto depende del HGI. Si desea usar esta función deberá estar muy familiarizado con los HGI. También debería considerar el uso del PHP en lugar de Hyperwave para acceder a fuentes externas. Añadir soporte de bases de datos a través de una pasarela de Hyperwave resulta más difícil que hacerlo en PHP.

Vea también **hw_GetRemote()**.

hw_GetSrcByDestObj (PHP 3>= 3.0.3, PHP 4)

Devuelve los enlaces que apuntan al objeto

array **hw_getsrcbydestobj** (int conexión, int IDobjeto) \linebreak

Devuelve los registros de objeto de todos los enlaces que apuntan al objeto con ID *`IDobjeto`*. El objeto puede ser tanto un documento como un enlace de tipo destino.

Vea también **hw_GetAnchors()**.

hw_GetText (PHP 3>= 3.0.3, PHP 4)

obtiene un documento de texto

int **hw_gettext** (int conexión, int IDobjeto [, mixed IDraiz/prefijo]) \linebreak

Devuelve el documento con ID de objeto *`IDobjeto`*. Si el documento tiene enlaces que pueden ser insertados, serán insertados ahora. El parámetro opcional *`IDraiz/prefijo`* puede ser una cadena o un entero. Si es un entero determina la forma en que los enlaces se insertan en el documento. Por defecto es 0 y los enlaces se crean a partir del nombre del objeto de destino de los mismos. Esto es útil para aplicaciones web. Si un enlace apunta a un objeto con nombre ’pelicula_internet’ el enlace HTML será <A HREF="/pelicula_internet">. La posición actual del objeto de destino en la jerarquía de documentos es despreciada. Tendrá que ajustar su navegador web para que reescriba dicho URL a, por ejemplo, ’/mi_script.php3/pelicula_internet’. ’mi_script.php3’ deberá evaluar $PATH_INFO y recuperar el documento. Todos los enlaces tendrán el prefijo ’/mi_script.php3’. Si no desea este efecto puede fijar el

parámetro opcional *IDraiz/prefijo* al prefijo que desee en su lugar. En este caso deberá ser una cadena.

Si el *IDraiz/prefijo* es un entero distinto de 0, el enlace se construye con todos los nombres de objeto comenzando con el objeto de id *IDraiz/prefijo*, separado por una barra relativa al objeto actual. Si por ejemplo el documento anterior 'pelicula_internet' está situado en 'a-b-c-pelicula_internet' donde '-' es el separador entre niveles jerárquicos en el servidor Hyperwave y el documento fuente está situado en 'a-b-d-fuente', el enlace HTML resultante sería: <A HREF="../c/pelicula_internet">. Esto es útil cuando desea bajarse el contenido completo del servidor al disco y mapear la jerarquía de documentos en el sistema de archivos.

Esta función sólo trabajará en documentos de texto puros. No se abrirá una conexión de datos especial y por tanto bloqueará la conexión de control durante la transferencia.

Vea también **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_DocumentBodyTag()**, **hw_DocumentSize()**, **hw_OutputDocument()**.

hw_Username (unknown)

nombre del usuario actualmente conectado

string **hw_getusername** (int conexión) \linebreak

Devuelve el nombre de usuario de la conexión.

hw_Identify (PHP 3>= 3.0.3, PHP 4)

identificarse como usuario

int **hw_identify** (string nombre, string clave) \linebreak

Le identifica como el usuario *nombre* y *clave*. La identificación sólo es válida para la sesión actual. No pienso que esta función sea necesaria con mucha frecuancia. En muchos casos será más fácil identificarse al abrir la conexión.

Vea también **hw_Connect()**.

hw_InCollections (PHP 3>= 3.0.3, PHP 4)

comprueba si los id de objeto están en las colecciones

array **hw_incollections** (int conexión, array tabla_id_objeto, array tabla_id_colecciones, int colecciones_devueltas) \linebreak

Comprueba si el conjunto de objetos (documentos o colecciones) especificado por el parámetro *tabla_id_objeto* es parte de las colecciones enumeradas en *tabla_id_colecciones*. Cuando

el cuarto parámetro *colecciones_devueltas* es 0, el subconjunto de id de objeto que es parte de las colecciones (es decir, los documentos o colecciones hijos de una o más colecciones de id de colecciones o de sus subcolecciones, recursivamente) es devuelto en forma de tabla. Cuando el cuarto parámetro es 1, sin embargo, el conjunto de colecciones que tienen uno o más objetos de este subconjunto como hijos es devuelto como tabla. Esta opción permite a un cliente, p. ej., remarcar en una visualización gráfica la parte de la jerarquía de colecciones que contiene las coincidencias de una consulta previa.

hw_Info (PHP 3>= 3.0.3, PHP 4)

información sobre conexión

string **hw_info** (int conexión) \linebreak

Devuelve información sobre la conexión actual. La cadena devuelta tiene el siguiente formato: <Cadenaservidor>, <Anfitrión>, <Puerto>, <Usuario>, <Puerto del Usuario>, <Intercambio de bytes>

hw_InsColl (PHP 3>= 3.0.3, PHP 4)

insertar colección

int **hw_inscoll** (int conexión, int IDobjeto, array tabla_objetos) \linebreak

Inserta una nueva colección con los atributos de la *tabla_objetos* en la colección cuyo ID de objeto es *IDobjeto*.

hw_InsDoc (PHP 3>= 3.0.3, PHP 4)

insertar documento

int **hw_insdoc** (int conexión, int IDpadre, string registro_de_objeto, string texto) \linebreak

Inserta un nuevo documento con los atributos del *registro_de_objeto* en la colección cuyo ID de objeto es *IDpadre*. Esta función inserta tanto un registro de objeto sólo como un registro de objeto y el texto puro en ASCII dado en *texto* si este es especificado. si desea insertar un documento general de cualquier tipo, utilice en su lugar hw_insertdocument().

Vea también **hw_InsertDocument()**, **hw_InsColl()**.

hw_insertanchors (PHP 4 >= 4.0.4)

Inserts only anchors into text

string **hw_insertanchors** (int hwdoc, array anchorecs, array dest [, array urlprefixes]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

hw_InsertDocument (PHP 3>= 3.0.3, PHP 4)

subir cualquier objeto

int **hw_insertdocument** (int conexión, int id_padre, int documento_hw) \linebreak

Sube un documento a la colección dada por *`id_padre`*. El documento debe ser creado antes con la función **hw_NewDocument()**. Asegúrese que el registro de objeto del nuevo documento contenga al menos los atributos: Type, DocumentType, Title y Name (así, en inglés). Posiblemente desee fijar también el MimeType. La función devuelve la id de objeto del nuevo documento, o `FALSE`.

Vea también **hw_PipeDocument()**.

hw_InsertObject (PHP 3>= 3.0.3, PHP 4)

inserta un registro de objeto

int **hw_insertobject** (int conexión, string reg de objeto, string parametro) \linebreak

Inserta un objeto en el servidor. Este puede consistir en cualquier objeto hyperwave válido. Vea la documentación sobre el HG-CSP si desea información detallada sobre cuáles tienen que ser los parámetros.

Nota: Si se desea insertar un enlace, el atributo Position siempre se fijará a un valor comienzo/final o a 'invisible'. Las posiciones invisibles se necesitan si la anotación no tiene enlace correspondiente en el texto anotado.

Vea también **hw_PipeDocument()**, **hw_InsertDocument()**, **hw_InsDoc()**, **hw_InsColl()**.

hw_mapid (PHP 3>= 3.0.13, PHP 4)

Mapea in id global a un id virtual local

int **hw_mapid** (int conexión, int id servidor, int id objeto) \linebreak

Mapea un id de objeto global en un servidor hyperwave, incluso con aquellos a los que no se ha conectado con hw_connect(), sobre un id virtual de objeto. Este id virtual se puede utilizar como cualquier otro id de objeto, p. ej. para obtener el registro de objeto por medio de hw_getobject(). El id de servidor es la primera parte del id global de objeto (GOid) de aquel que es realmene el número IP expresado como entero.

Nota: Para usar esta función deberá activar el indicador F_DISTRIBUTED, que actualmenes sólo se puede fijar en tiempo de compilación desde hg_comm.c. Por defecto está inactivo. Lea el comentario al principio de hg_comm.c

hw_Modifyobject (PHP 3>= 3.0.7, PHP 4)

modifica el registro de objeto

int **hw_modifyobject** (int conexión, int objeto_a_cambiar, array eliminar, array añadir, int modo) \linebreak

Este comando permite eliminar, añadir, o modificar atributos individuales de un registro de objeto. El objeto está especificado por el ID de objeto *`objeto_a_cambiar`*. La primera tabla, *`eliminar`*, es la lista de atributos a eliminar. La segunda tabla, *`añadir`*, es la lista de atributos a añadir. Para modificar un atributo, hay que borrar el antiguo y añadir uno nuevo. **hw_modifyobject()** siempre eliminará los atributos antes de añadir los nuevos excepto si el valor del atributo a eliminar no es una cadena o una tabla.

El último parámetro determina si la modificación se realiza de manera recursiva. 1 quiere decir que sea recursiva. Si alguno de los objetos no se puede modificar, será ignorado sin avisar. Incluso hw_error() podría no indicar un error aunque alguno de los objetos no pueda ser modificado.

Las claves de ambas tablas son los nombres de los atributos. El valor de cada elemento de la tabla puede ser una tabla, una cadena o cualquier otra cosa. Si es una tabla, cada valor de atributo se construye como la clave de cada elemento más dos puntos y el valor de cada elemento. Si es una cadena se toma como valor del atributo. Una cadena vacía producirá la supresión completa del atributo. Si el valor no es ni cadena ni tabla, sino otra cosa, p. ej. un entero, no se realizará operación alguna en el atributo. Esto es necesario se desea añadir un atributo completamente nuevo, no solamente un nuevo valor para un atributo existente. Si la tabla eliminar contuviera una cadena vacía para dicho atributo, este se intentaría suprimir, lo que fallaría porque este no existe. La siguiente adición de un nuevo valor para dicho atributo también fallará. Fijando el valor para dicho atributo p. ej. a 0 hará que ni siquiera se intente eliminar, pero funcionará la adición del mismo.

Si desea cambiar el atributo ’Nombre’ con el valor actual ’libros’ por el de ’artículos’ deberá crear dos tablas y llamar a **hw_modifyobject()**.

Ejemplo 1. modificando un atributo

```
// $conexion es una conexión con el servidor Hyperwave
// $idobj es la ID del objeto a modificar
$tablasupr = array("Nombre" => "libros");
$tablaanad = array("Nombre" => "artículos");
$hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

Para borrar/añadir un par nombre=valor de/a el registro de objeto, simplemente pase la tabla eliminar/añadir y fije el último/tercer parámetro a tabla vacía. Si el atributo es el primero con dicho nombre que se añade, fije el valor del atributo en la tabla eliminar a un valor entero.

Ejemplo 2. añadiendo un atributo completamente nuevo

```
// $conexion es una conexión con el servidor Hyperwave
// $idobj es la ID del objeto a modificar
$tablasupr = array("Nombre" => 0);
$tablaanad = array("Nombre" => "artículos");
$hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

Nota: Los atributos plurilingües, p. ej. 'Título', se pueden modificar de dos maneras. O bien proporcionando los valores de los atributos en su forma nativa 'lenguaje':'título', bien proporcionando una tabla con los elementos para cada lenguaje según se describe más arriba. El ejemplo anterior podría quedar entonces:

Ejemplo 3. modificando el atributo Título

```
$tablasupr = array("Título" => "es:Libros");
$tablaanad = array("Título" => "es:Artículos");
$hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

o

Ejemplo 4. modificando el atributo Título

```
$tablasupr = array("Título" => array("es" => "Libros"));
$tablaanad = array("Título" => array("es" => "Artículos", "ge"=>"Artikel"));
$hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

Esto elimina el título español 'Libros' y añade el título español 'Artículos' y el título alemán 'Artikel'.

Ejemplo 5. eliminando atributos

```
$tablasupr = array("Título" => "");
$tablaanad = array("Tñitulo" => "es:Artículos");
$hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

Nota: Esto eliminará todos los atributos con el nombre 'Título' y añadirá un nuevo atributo 'Título'. Esto es útil cuando se desea eliminar atributos de forma recursiva.

Nota: Si necesita eliminar todos los atributos con un cierto nombre tendrá que pasar una cadena vacía como el valor del atributo.

Nota: Sólo los atributos 'Title', 'Description' y 'Keyword' (así, en inglés) manejarán de forma apropiada el prefijo de idioma. Si estos atributos no llevaran prefijo de idioma, se les asignariá el prefijo 'xx'.

Nota: El atributo 'Name' es bastante especial. En algunos casos no puede ser completamente eliminado. Obtendrá un mensaje de error 'Change of base attribute' ('Cambio de atributo base', no está muy claro cuando ocurre). Por tanto, tendrá siempre que añadir un nuevo atributo Name primero y luego eliminar el anterior.

Nota: No debe rodear esta función de llamadas a hw_getandlock() ni a hw_unlock(). **hw_modifyobject()** ya lo hace internamente.

Devuelve `TRUE` si no hay error o `FALSE` en caso contrario.

hw_Mv (PHP 3>= 3.0.3, PHP 4)

mueve objetos

int **hw_mv** (int conexión, array tabla de id de objeto, int id origen, int id destino) \linebreak

Mueve los objetos cuyas id se especifican en el segundo parámetro desde la colección con id *`id origen`* hasta la colección con el id *`id destino`*. Si el id de destino es 0, los objetos serán disociados de la colección origen. Si esta fuese la última instancia del objeto, este sería borrado. Si desea borrar todas las instancias de una vez, utilice hw_deleteobject().

El valor devuelto es el número de objetos movidos.

Vea también hw_cp(), hw_deleteobject().

hw_New_Document (PHP 3>= 3.0.3, PHP 4)

crear nuevo documento

int **hw_new_document** (string registro_de_objeto, string datos_documento, int tama_documento) \linebreak

Devuelve un nuevo documento Hyperwave en el que los datos del documento están fijados a *datos_documento* y el registro de objeto a *registro_de_objeto*. La longitud de los *datos_documento* debe pasarse en *tama_documento*. Esta función no inserta el documento en el servidor Hyperwave.

Vea también **hw_FreeDocument()**, **hw_DocumentSize()**, **hw_DocumentBodyTag()**, **hw_OutputDocument()**, **hw_InsertDocument()**.

hw_Objrec2Array (PHP 3>= 3.0.3, PHP 4)

convierte atributos de registro de objeto a tabla de objetos

array **hw_objrec2array** (string registro_de_objeto) \linebreak

Convierte un *registro_de_objeto* en una tabla de objetos. Las claves de la tabla resultante son los nombres de los atributos. Los atributos múltiples como 'Título' en distintos idiomas forman su propia tabla. Las claves de esta tabla son las partes a la izquierda de los dos puntos del valor del atributo. Actualmente, sólo los atributos 'Title', 'Description' y 'Keyword' (así, en inglés) son correctamente tratados. Otros atributos múltiples generan una tabla indizada. Actualmente, sólo el atributo 'Group' es tratado correctamente.

Vea también hw_array2objrec().

hw_Output_Document (PHP 3>= 3.0.3, PHP 4)

prints hw_document

int **hw_output_document** (int hw_document) \linebreak

Prints the document without the BODY tag.

For backward compatibility, **hw_outputdocument()** is also accepted. This is deprecated, however.

hw_pConnect (PHP 3>= 3.0.3, PHP 4)

hacer una conexión de base de datos permanente

int **hw_pconnect** (string servidor, int puerto, string usuario, string clave) \linebreak

Devuelve un índice de conexión si hay éxito, o `FALSE` si la conexión no puede hacerse. Abre una conexión permanente a un servidor Hyperwave. Cada uno de los argumentos debe ser una cadena entrecomillada excepto el número de puerto. El argumento *`usuario`* y la *`clave`* son opcionales y pueden omitirse. En tal caso no se realizará ninguna identificación con el servidor. Es similar a la identificación anónima del usuario. Esta función devuelve un índice de conexión que se necesita para otras funciones de Hyperwave. Se pueden tener varias conexiones permanentes abiertas a la vez.

Vea también **hw_Connect()**.

hw_PipeDocument (PHP 3>= 3.0.3, PHP 4)

recupera cualquier documento

int **hw_pipedocument** (int conexión, int IDobjeto) \linebreak

Devuelve el documento Hyperwave cuyo ID de objeto es *`IDobjeto`*. Si el documento tiene enlaces que pueden ser insertados, deberán haberse insertado ya. El documento será transferido a través de una conexión de datos especial que no bloquea la conexión de control.

Vea también **hw_GetText()** para saber más sobre inserción de enlaces, **hw_FreeDocument()**, **hw_DocumentSize()**, **hw_DocumentBodyTag()**, **hw_OutputDocument()**.

hw_Root (PHP 3>= 3.0.3, PHP 4)

ID del objeto raíz

int **hw_root** ( ) \linebreak

Devuelve la ID de objeto de la colección hiper-raíz. Actualmente siempre vale 0. La colección hija de la hiper-raíz es la colección raíz del servidor al que se ha conectado.

hw_setlinkroot (PHP 3>= 3.0.3, PHP 4)

Set the id to which links are calculated

void **hw_setlinkroot** (int link, int rootid) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

hw_stat (PHP 3>= 3.0.3, PHP 4)

Returns status string

string **hw_stat** (int link) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

hw_Unlock (PHP 3>= 3.0.3, PHP 4)

desbloquea objeto

int **hw_unlock** (int conexión, int IDobjeto) \linebreak

Desbloquea un documento para que otros usuarios puedan acceder al mismo de nuevo.

Vea también **hw_GetAndLock()**.

hw_Who (PHP 3>= 3.0.3, PHP 4)

Lista de los usuarios actualmente conectados

int **hw_who** (int conexión) \linebreak

Devuelve una tabla de los usuarios actualmente conectados al servidor Hyperwave. Cada elemento de esta tabla contiene en sí mismo los elementos ID, nombre, sistema, onSinceDate (conectadoDesdeFecha), onSinceTime (conectadoDesdeHora), TotalTime (TiempoTotal) y self (propio). 'self' es 1 si esta línea corresponde al usuario que realizó la petición.

XXXIX. Hyperwave API functions

Introduction

Hyperwave has been developed at IICM (http://www.iicm.edu/) in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (If I remember properly it was in 1996).

Hyperwave is not free software. The current version, 5.5, is available at www.hyperwave.com (http://www.hyperwave.com/). A time limited version can be ordered for free (30 days).

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user.

Requirements

Since 2001 there is a Hyperwave SDK available. It supports Java, JavaScript and C++. This PHP Extension is based on the C++ interface. In order to activate the hwapi support in PHP you will have to install the Hyperwave SDK first and configure PHP with `--with-hwapi=<dir$gt;`.

Classes

The API provided by the HW_API extension is fully object oriented. It is very similar to the C++ interface of the Hyperwave SDK. It consist of the following classes.

- `HW_API`
- `HW_API_Object`
- `HW_API_Attribute`
- `HW_API_Error`
- `HW_API_Content`
- `HW_API_Reason`

Some basic classes like `HW_API_String`, `HW_API_String_Array`, etc., which exist in the Hyperwave SDK have not been implemented since PHP has powerful replacements for them.

Each class has certain method, whose names are identical to its counterparts in the Hyperwave SDK. Passing arguments to this function differs from all the other PHP Extension but is close to the C++ API of the HW SDK. Instead of passing serval parameters they are all put into an associated array and passed

as one paramter. The names of the keys are identical to those documented in the HW SDK. The common parameters are listed below. If other parameters are required they will be documented if needed.

- `objectIdentifier` The name or id of an object, e.g. "rootcollection", "0x873A8768 0x00000002".
- `parentIdentifier` The name or id of an object which is considered to be a parent.
- `object` An instance of class HW_API_Object.
- `parameters` An instance of class HW_API_Object.
- `version` The version of an object.
- `mode` An integer value determine the way an operation is executed.
- `attributeSelector` Any array of strings, each containing a name of an attribute. This is used if you retrieve the object record and want to include certain attributes.
- `objectQuery` A query to select certain object out of a list of objects. This is used to reduce the number of objects which was delivered by a function like **hw_api->children()** or **hw_api->find()**.

Integration with Apache

The integration with Apache and possible other servers is already described in the Hyperwave Modul which has been the first extension to connect a Hyperwave Server.

hw_api_attribute->key (unknown)

Returns key of the attribute

string **key** (void) \linebreak

Returns the name of the attribute.

See also hwapi_attribute_value().

hw_api_attribute->langdepvalue (unknown)

Returns value for a given language

string **langdepvalue** (string language) \linebreak

Returns the value in the given language of the attribute.

See also hwapi_attribute_value().

hw_api_attribute->value (unknown)

Returns value of the attribute

string **value** (void) \linebreak

Returns the value of the attribute.

See also hwapi_attribute_key(), hwapi_attribute_values().

hw_api_attribute->values (unknown)

Returns all values of the attribute

array **values** (void) \linebreak

Returns all values of the attribute as an array of strings.

See also hwapi_attribute_value().

hw_api_attribute (unknown)

Creates instance of class hw_api_attribute

object **attribute** ([string name [, string value]]) \linebreak

Creates a new instance of hw_api_attribute with the given name and value.

hw_api->checkin (unknown)

Checks in an object

object **checkin** (array parameter) \linebreak

This function checks in an object or a whole hiearchie of objects. The parameters array contains the required element 'objectIdentifier' and the optional element 'version', 'comment', 'mode' and 'objectQuery'. 'version' sets the version of the object. It consists of the major and minor version separated by a period. If the version is not set, the minor version is incremented. 'mode' can be one of the following values:

HW_API_CHECKIN_NORMAL

Checks in and commits the object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

If the object to check in is a collection, all children will be checked in recursively if they are documents. Trying to check in a collection would result in an error.

HW_API_CHECKIN_FORCE_VERSION_CONTROL

Checks in an object even if it is not under version control.

HW_API_CHECKIN_REVERT_IF_NOT_CHANGED

Check if the new version is different from the last version. Unless this is the case the object will be checked in.

HW_API_CHECKIN_KEEP_TIME_MODIFIED

Keeps the time modified from the most recent object.

HW_API_CHECKIN_NO_AUTO_COMMIT

The object is not automatically commited on checkin.

See also hwapi_checkout().

hw_api->checkout (unknown)

Checks out an object

object **checkout** (array parameter) \linebreak

This function checks out an object or a whole hiearchie of objects. The parameters array contains the required element ’objectIdentifier’ and the optional element ’version’, ’mode’ and ’objectQuery’. ’mode’ can be one of the following values:

HW_API_CHECKIN_NORMAL

Checks out an object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

If the object to check out is a collection, all children will be checked out recursively if they are documents. Trying to check out a collection would result in an error.

See also hwapi_checkin().

hw_api->children (unknown)

Returns children of an object

array **children** (array parameter) \linebreak

Retrieves the children of a collection or the attributes of a document. The children can be further filtered by specifying an object query. The parameter array contains the required elements ’objectIdentifier’ and the optional elements ’attributeSelector’ and ’objectQuery’.

The return value is an array of objects of type `HW_API_Object` or `HW_API_Error`.

See also hwapi_parents().

hw_api_content->mimetype (unknown)

Returns mimetype

string **mimetype** (void) \linebreak

Returns the mimetype of the content.

hw_api_content->read (unknown)

Read content

string **read** (string buffer, integer len) \linebreak

Reads *`len`* bytes from the content into the given buffer.

hw_api->content (unknown)

Returns content of an object

object **content** (array parameter) \linebreak

This function returns the content of a document as an object of type `hw_api_content`. The parameter array contains the required elements 'objectidentifier' and the optional element 'mode'. The mode can be one of the constants HW_API_CONTENT_ALLLINKS, HW_API_CONTENT_REACHABLELINKS or HW_API_CONTENT_PLAIN. HW_API_CONTENT_ALLLINKS means to insert all anchors even if the destination is not reachable. HW_API_CONTENT_REACHABLELINKS tells **hw_api_content()** to insert only reachable links and HW_API_CONTENT_PLAIN will lead to document without any links.

hw_api->copy (unknown)

Copies physically

object **copy** (array parameter) \linebreak

This function will make a physical copy including the content if it exists and returns the new object or an error object. The parameter array contains the required elements 'objectIdentifier' and 'destinationParentIdentifier'. The optional parameter is 'attributeSelector''

See also hwapi_move(), hwapi_link().

hw_api->dbstat (unknown)

Returns statistics about database server

object **dbstat** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_hwstat(), hwapi_ftstat().

hw_api->dcstat (unknown)

Returns statistics about document cache server

object **dcstat** (array parameter) \linebreak

See also hwapi_hwstat(), hwapi_dbstat(), hwapi_ftstat().

hw_api->dstanchors (unknown)

Returns a list of all destination anchors

object **dstanchors** (array parameter) \linebreak

Retrieves all destination anchors of an object. The parameter array contains the required element ’objectIdentifier’ and the optional elements ’attributeSelector’ and ’objectQuery’.

See also hwapi_srcanchors().

hw_api->dstofsrcanchors (unknown)

Returns destination of a source anchor

object **dstofsrcanchors** (array parameter) \linebreak

Retrieves the destination object pointed by the specified source anchors. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element ’objectIdentifier’ and the optional element ’attributeSelector’.

See also hwapi_srcanchors(), hwapi_dstanchors(), hwapi_objectbyanchor().

hw_api_error->count (unknown)

Returns number of reasons

int **count** (void) \linebreak

Returns the number of error reasons.

See also hwapi_error_reason().

hw_api_error->reason (unknown)

Returns reason of error

object **reason** (void) \linebreak

Returns the first error reason.

See also hwapi_error_count().

hw_api->find (unknown)

Search for objects

array **find** (array parameter) \linebreak

This functions searches for objects either by executing a key or/and full text query. The found objects can further be filtered by an optional object query. They are sorted by their importance. The second search operation is relatively slow and its result can be limited to a certain number of hits. This allows to perform an incremental search, each returning just a subset of all found documents, starting at a given index. The parameter array contains the 'keyquery' or/and 'fulltextquery' depending on who you would like to search. Optional parameters are 'objectquery', 'scope', 'lanugages' and 'attributeselector'. In case of an incremental search the optional parameters 'startIndex', numberOfObjectsToGet' and 'exactMatchUnit' can be passed.

hw_api->ftstat (unknown)

Returns statistics about fulltext server

object **ftstat** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_dbstat(), hwapi_hwstat().

hwapi_hgcsp (unknown)

Returns object of class hw_api

object **hwapi_hgcsp** (string hostname [, int port]) \linebreak

Opens a connection to the Hyperwave server on host *`hostname`*. The protocol used is HGCSP. If you do not pass a port number, 418 is used.

See also **hwapi_hwtp()**.

hw_api->hwstat (unknown)

Returns statistics about Hyperwave server

object **hwstat** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_dbstat(), hwapi_ftstat().

hw_api->identify (unknown)

Log into Hyperwave Server

object **identify** (array parameter) \linebreak

Logs into the Hyperwave Server. The parameter array must contain the elements ’username’ und ’password’.

The return value will be an object of type `HW_API_Error` if identification failed or TRUE if it was successful.

hw_api->info (unknown)

Returns information about server configuration

object **info** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_dbstat(), hwapi_ftstat(), hwapi_hwstat().

hw_api->insert (unknown)

Inserts a new object

object **insert** (array parameter) \linebreak

Insert a new object. The object type can be user, group, document or anchor. Depending on the type other object attributes has to be set. The parameter array contains the required elements ’object’ and ’content’ (if the object is a document) and the optional parameters ’parameters’, ’mode’ and ’attributeSelector’. The ’object’ must contain all attributes of the object. ’parameters’ is an object as well holding futher attributes like the destination (attribute key is ’Parent’). ’content’ is the content of the document. ’mode’ can be a combination of the following flags:

HW_API_INSERT_NORMAL

The object in inserted into the server.

HW_API_INSERT_FORCE-VERSION-CONTROL

HW_API_INSERT_AUTOMATIC-CHECKOUT

HW_API_INSERT_PLAIN

HW_API_INSERT_KEEP_TIME_MODIFIED

HW_API_INSERT_DELAY_INDEXING

See also hwapi_replace().

hw_api->insertanchor (unknown)

Inserts a new object of type anchor

object **insertanchor** (array parameter) \linebreak

This function is a shortcut for hwapi_insert(). It inserts an object of type anchor and sets some of the attributes required for an anchor. The parameter array contains the required elements 'object' and 'documentIdentifier' and the optional elements 'destinationIdentifier', 'parameter', 'hint' and 'attributeSelector'. The 'documentIdentifier' specifies the document where the anchor shall be inserted. The target of the anchor is set in 'destinationIdentifier' if it already exists. If the target does not exists the element 'hint' has to be set to the name of object which is supposed to be inserted later. Once it is inserted the anchor target is resolved automatically.

See also hwapi_insertdocument(), hwapi_insertcollection(), hwapi_insert().

hw_api->insertcollection (unknown)

Inserts a new object of type collection

object **insertcollection** (array parameter) \linebreak

This function is a shortcut for hwapi_insert(). It inserts an object of type collection and sets some of the attributes required for a collection. The parameter array contains the required elements 'object' and 'parentIdentifier' and the optional elements 'parameter' and 'attributeSelector'. See hwapi_insert() for the meaning of each element.

See also hwapi_insertdocument(), hwapi_insertanchor(), hwapi_insert().

hw_api->insertdocument (unknown)

Inserts a new object of type document

object **insertdocument** (array parameter) \linebreak

This function is a shortcut for hwapi_insert(). It inserts an object with content and sets some of the attributes required for a document. The parameter array contains the required elements ’object’, ’parentIdentifier’ and ’content’ and the optional elements ’mode’, ’parameter’ and ’attributeSelector’. See hwapi_insert() for the meaning of each element.

See also hwapi_insert() hwapi_insertanchor(), hwapi_insertcollection().

hw_api->link (unknown)

Creates a link to an object

object **link** (array parameter) \linebreak

Creates a link to an object. Accessing this link is like accessing the object to links points to. The parameter array contains the required elements ’objectIdentifier’ and ’destinationParentIdentifier’. ’destinationParentIdentifier’ is the target collection.

The function returns true on success or an error object.

See also hwapi_copy().

hw_api->lock (unknown)

Locks an object

object **lock** (array parameter) \linebreak

Locks an object for exclusive editing by the user calling this function. The object can be only unlocked by this user or the sytem user. The parameter array contains the required element ’objectIdentifier’ and the optional parameters ’mode’ and ’objectquery’. ’mode’ determines how an object is locked. HW_API_LOCK_NORMAL means, an object is locked until it is unlocked. HW_API_LOCK_RECURSIVE is only valid for collection and locks all objects within th collection und possible subcollections. HW_API_LOCK_SESSION means, an object is locked only as long as the session is valid.

See also hwapi_unlock().

hw_api->move (unknown)

Moves object between collections

object **move** (array parameter) \linebreak

See also hw_objrec2array().

hw_api_content (unknown)

Create new instance of class hw_api_content

string **content** (string content, string mimetype) \linebreak

Creates a new content object from the string *`content`*. The mimetype is set to *`mimetype`*.

hw_api_object->assign (unknown)

Clones object

object **assign** (array parameter) \linebreak

Clones the attributes of an object.

hw_api_object->attreditable (unknown)

Checks whether an attribute is editable

bool **attreditable** (array parameter) \linebreak

hw_api_object->count (unknown)

Returns number of attributes

int **count** (array parameter) \linebreak

hw_api_object->insert (unknown)

Inserts new attribute

bool **insert** (object attribute) \linebreak

Adds an attribute to the object. Returns true on success and otherwise false.

See also hwapi_object_remove().

hw_api_object (unknown)

Creates a new instance of class hw_api_object

object **hw_api_object** (array parameter) \linebreak

See also hwapi_lock().

hw_api_object->remove (unknown)

Removes attribute

bool **remove** (string name) \linebreak

Removes the attribute with the given name. Returns true on success and otherwise false.

See also hwapi_object_insert().

hw_api_object->title (unknown)

Returns the title attribute

string **title** (array parameter) \linebreak

hw_api_object->value (unknown)

Returns value of attribute

string **value** (string name) \linebreak

Returns the value of the attribute with the given name or false if an error occured.

hw_api->object (unknown)

Retrieve attribute information

object **hw_api->object** (array parameter) \linebreak

This function retrieves the attribute information of an object of any version. It will not return the document content. The parameter array contains the required elements 'objectIdentifier' and the optional elements 'attributeSelector' and 'version'.

The returned object is an instance of class `HW_API_Object` on success or `HW_API_Error` if an error occured.

This simple example retrieves an object and checks for errors.

Ejemplo 1. Retrieve an object

```
<?php
function handle_error($error) {
  $reason = $error->reason(0);
  echo "Type: <B>";
  switch($reason->type()) {
    case 0:
      echo "Error";
      break;
    case 1:
      echo "Warning";
      break;
    case 2:
      echo "Message";
      break;
  }
  echo "</B><BR>\n";
  echo "Description: ".$reason->description("en")."<BR>\n";
}

function list_attr($obj) {
  echo "<TABLE>\n";
  $count = $obj->count();
  for($i=0; $i<$count; $i++) {
    $attr = $obj->attribute($i);
    printf("  <TR><TD ALIGN=right bgcolor=#c0c0c0><B>%s</B></TD><TD bgcolor=#F0F0F0>%s</TD>\
             $attr->key(), $attr->value());
  }
  echo "</TABLE>\n";
}

$hwapi = hwapi_hgcsp($g_config[HOSTNAME]);
$parms = array("objectIdentifier"=>"rootcollection, "attributeSelector"=>array("Title", "Nam
$root = $hwapi->object($parms);
if(get_class($root) == "HW_API_Error") {
  handle_error($root);
  exit;
}
list_attr($root);
?>
```

See also hwapi_content().

hw_api->objectbyanchor (unknown)

Returns the object an anchor belongs to

object **objectbyanchor** (array parameter) \linebreak

This function retrieves an object the specified anchor belongs to. The parameter array contains the required element 'objectIdentifier' and the optional element 'attributeSelector'.

See also **hwapi_dstofsrcanchor()**, hwapi_srcanchors(), hwapi_dstanchors().

hw_api->parents (unknown)

Returns parents of an object

array **parents** (array parameter) \linebreak

Retrieves the parents of an object. The parents can be further filtered by specifying an object query. The parameter array contains the required elements 'objectidentifier' and the optional elements 'attributeselector' and 'objectquery'.

The return value is an array of objects of type `HW_API_Object` or `HW_API_Error`.

See also hwapi_children().

hw_api_reason->description (unknown)

Returns description of reason

string **description** (void) \linebreak

Returns the description of a reason

hw_api_reason->type (unknown)

Returns type of reason

object **type** (void) \linebreak

Returns the type of a reason.

hw_api->remove (unknown)

Delete an object

object **remove** (array parameter) \linebreak

This function removes an object from the specified parent. Collections will be removed recursively. You can pass an optional object query to remove only those objects which match the query. An object will be deleted physically if it is the last instance. The parameter array contains the required elements 'objectidentifier' and 'parentidentifier'. If you want to remove a user or group 'parentidentifier' can be skiped. The optional parameter 'mode' determines how the deletion is performed. In normal mode the object will not be removed physically until all instances are removed. In physical mode all instances of the object will be deleted imediately. In removelinks mode all references to and from the objects will be deleted as well. In nonrecursive the deletion is not performed recursive. Removing a collection which is not empty will cause an error.

See also hwapi_move().

hw_api->replace (unknown)

Replaces an object

object **replace** (array parameter) \linebreak

Replaces the attributes and the content of an object The parameter array contains the required elements 'objectIdentifier' and 'object' and the optional parameters 'content', 'parameters', 'mode' and 'attributeSelector'. 'objectIdentifier' contains the object to be replaced. 'object' contains the new object. 'content' contains the new content. 'parameters' contain extra information for HTML documents. HTML_Language is the letter abbreviation of the language of the title. HTML_Base sets the base attribute of the HTML document. 'mode' can be a combination of the following flags:

HW_API_REPLACE_NORMAL

The object on the server is replace with the object passed.

HW_API_REPLACE_FORCE_VERSION_CONTROL

HW_API_REPLACE_AUTOMATIC_CHECKOUT

HW_API_REPLACE_AUTOMATIC_CHECKIN

HW_API_REPLACE_PLAIN

HW_API_REPLACE_REVERT_IF_NOT_CHANGED

HW_API_REPLACE_KEEP_TIME_MODIFIED

See also hwapi_insert().

hw_api->setcommitedversion (unknown)

Commits version other than last version

object **setcommitedversion** (array parameter) \linebreak

Commits a version of a docuemnt. The commited version is the one which is visible to users with read access. By default the last version is the commited version.

See also hwapi_checkin(), hwapi_checkout(), **hwapi_revert()**.

hw_api->srcanchors (unknown)

Returns a list of all source anchors

object **srcanchors** (array parameter) \linebreak

Retrieves all source anchors of an object. The parameter array contains the required element 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

See also hwapi_dstanchors().

hw_api->srcsofdst (unknown)

Returns source of a destination object

object **srcsofdst** (array parameter) \linebreak

Retrieves all the source anchors pointing to the specified destination. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element 'objectIdentifier' and the optional element 'attributeSelector' and 'objectQuery'. The function returns an array of objects or an error.

See also **hwapi_dstofsrcanchor()**.

hw_api->unlock (unknown)

Unlocks a locked object

object **unlock** (array parameter) \linebreak

Unlocks a locked object. Only the user who has locked the object and the system user may unlock an object. The parameter array contains the required element 'objectIdentifier' and the optional parameters 'mode' and 'objectquery'. The meaning of 'mode' is the same as in function hwapi_lock().

Returns true on success or an object of class HW_API_Error.

See also hwapi_lock().

hw_api->user (unknown)

Returns the own user object

object **user** (array parameter) \linebreak

See also hwapi_userlist().

hw_api->userlist (unknown)

Returns a list of all logged in users

object **userlist** (array parameter) \linebreak

See also hwapi_user().

XL. Funciones para ICAP - Internet Calendar Application Protocol

Para hacer funcionar estas funciones, deberá compilar el PHP con `--with-icap`. Eso indicará que se precisa la instalación de la librería ICAP. Obtenga la última versión en http://icap.chek.com/, compílela e instálela.

icap_close (unknown)

Cierra un stream ICAP

int **icap_close** (int stream_icap, int banderas) \linebreak

Cierra el stream ICAP dado.

icap_create_calendar (PHP 4)

Create a new calendar

string **icap_create_calendar** (int stream_id, string calendar) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

icap_delete_calendar (PHP 4)

Delete a calendar

string **icap_delete_calendar** (int stream_id, string calendar) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

icap_delete_event (PHP 4)

Borra un evento de un calendario ICAP

int **icap_delete_event** (int id_evento) \linebreak

icap_delete_event() borra el evento de calendario especificado por el *`id_evento`*.

Devuelve `TRUE`.

icap_fetch_event (PHP 4)

Obtiene un evento del stream de calendario/

object **icap_fetch_event** (stream stream_icap, id id_evento, options opciones) \linebreak

icap_fetch_event() obtiene el evento del stream de calendario especificado por el parámetro *`id_evento`*.

Devuelve un objeto de evento compuesto por:

- int id - ID de dicho evento.
- int public - `TRUE` si el evento es público, `FALSE` si es privado.
- string category - Cadena con la categoría del evento.
- string title - Cadena de título del evento.
- string description - Cadena de descripción del evento.
- int alarm - Número de minutos antes que el evento envíe una alarma/recordatorio.
- object start - Objeto que contiene una entrada datetime (fecha/hora).
- object end - Objeto que contiene una entrada datetime.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos
- int sec - segundos

icap_list_alarms (PHP 4)

Devuelve una lista de los eventos que una alarma ha disparado en el instante dado

array **icap_list_alarms** (stream stream_icap, datetime instante_alarma) \linebreak

Devuelve una tabla de identificadores de evento para los que una alarma debiera apagarse en el instante indicado.

La función **icap_list_alarms()** toma una estructura datetime para un stream de calendario. Se devuelve una tabla de los identificadores de evento de todas las alarmas que debieran apagarse en el instante indicado.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos
- int sec - segundos

icap_list_events (PHP 4)

Devuelve una lista de eventos entre dos instantes dados

array **icap_list_events** (stream stream_icap, datetime instante_inicio, datetime instante_final) \linebreak

Devuelve una tabla de ID de evento que están entre los dos instantes dados.

La función **icap_list_events()** toma un instante de inicio y uno de final para un stream de calendario. Se devuelve una tabla de ID de evento que están entre los instantes dados.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos
- int sec - segundos

icap_open (PHP 4)

Abre una conexión ICAP

stream **icap_open** (string calendario, string usuario, string clave, string opciones) \linebreak

Si hay éxito devuelve un stream (flujo) ICAP, o `FALSE` en caso de error.

icap_open() abre una conexión ICAP con el servidor de *calendario* especificado. Si se especifica el parámetro opcional *opciones*, también éste es pasado a dicho buzón.

icap_rename_calendar (PHP 4)

Rename a calendar

string **icap_rename_calendar** (int stream_id, string old_name, string new_name) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

icap_reopen (PHP 4)

Reopen ICAP stream to new calendar

int **icap_reopen** (int stream_id, string calendar [, int options]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

icap_snooze (PHP 4)

Apaga la alarma de un evento

int **icap_snooze** (int id_evento) \linebreak

icap_snooze() apaga la alarma del evento de calendario especificado por el *id_evento*.

Returns TRUE.

icap_store_event (PHP 4)

Almacena un evento en un calendario ICAP

int **icap_store_event** (int stream_icap, object evento) \linebreak

icap_store_event() Guarda un evento en un calendario ICAP. Un objeto de evento consiste en:

- int public - 1 si es público, 0 si es privado;
- string category - Cadena con la categoría del evento.
- string title - Cadena de título del evento.
- string description - Cadena de descripción del evento.
- int alarm - Número de minutos antes que el evento envíe una alarma/recordatorio.
- object start - Objeto que contiene una entrada datetime (fecha/hora).
- object end - Objeto que contiene una entrada datetime.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos
- int sec - segundos

Devuelve `TRUE` en caso de éxito y `FALSE` en caso de error.

XLI. iconv functions

Introducción

This module contains an interface to the iconv library functions. The Iconv library functions convert files between various character sets encodings. The supported character sets depend on iconv() implementation on your system. Note that iconv() function in some system is not work well as you expect. In this case, you should install libiconv library.

Requerimientos

You must have iconv() function in standard C library or libiconv installed on your system. libiconv library is available from http://www.gnu.org/software/libiconv/.

Instalación

To be able to use the functions defined in this module you must compile your PHP interpreter using the configure line `--with-iconv`.

Configuración en tiempo de ejecución

Esta extensión no define ninguna directiva de configuración.

Tipos de recursos

Esta extensión no define ningún tipo de recurso.

Constantes predefinidas

Esta extensión no define ninguna constante.

iconv_get_encoding (PHP 4 >= 4.0.5)

Get current setting for character encoding conversion

array **iconv_get_encoding** ([string type]) \linebreak

It returns the current settings of ob_iconv_handler() as array or `FALSE` in failure.

See also: iconv_set_encoding() and ob_iconv_handler().

iconv_set_encoding (PHP 4 >= 4.0.5)

Set current setting for character encoding conversion

array **iconv_set_encoding** (string type, string charset) \linebreak

It changes the value of *`type`* to *`charset`* and returns `TRUE` in success or `FALSE` in failure.

Ejemplo 1. iconv_set_encoding() example:

```
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
```

See also: iconv_get_encoding() and ob_iconv_handler().

iconv (PHP 4 >= 4.0.5)

Convert string to requested character encoding

string **iconv** (string in_charset, string out_charset, string str) \linebreak

It converts the string *`string`* encoded in *`in_charset`* to the string encoded in *`out_charset`*. It returns the converted string or `FALSE`, if it fails.

Ejemplo 1. iconv() example:

```
echo iconv("ISO-8859-1","UTF-8","This is test.");
```

ob_iconv_handler (PHP 4 >= 4.0.5)

Convert character encoding as output buffer handler

array **ob_iconv_handler** (string contents, int status) \linebreak

It converts the string encoded in *internal_encoding* to *output_encoding*.

internal_encoding and *output_encoding* should be defined by iconv_set_encoding() or in configuration file.

Ejemplo 1. ob_iconv_handler() example:

```
ob_start("ob_iconv_handler"); // start output buffering
```

See also: iconv_get_encoding() and iconv_set_encoding().

XLII. Funciones para imágenes

Puede usar las funciones para imágenes en PHP para obtener el tamaño de imágenes JPEG, GIF, PNG, SWF, TIFF y JPEG2000, y si tiene instalada la biblioteca GD (disponible en http://www.boutell.com/gd/) también podrá crear y manipular imágenes.

Si tiene PHP compilado con `--enable-exif` puede trabajar con la información guardada en las cabeceras de las imágenes JPEG y TIFF. Estas funciones no requieren la biblioteca GD.

Los formatos de imágenes que puede manipular dependen de la versión de GD que instale y de cualquier otra biblioteca que GD pueda necesitar para acceder a esos formatos de imagen. Las versiones de GD anteriores a la GD-1.6 soportan imágenes en formato gif y no soportan png, mientras que las versiones superiores a la GD-1.6 soportan el formato png y no el gif.

Antes de poder leer y escribir imágenes en formato jpeg, deberá obtener e instalar jpeg-6b (disponible en ftp://ftp.uu.net/graphics/jpeg/), y después recompilar GD para poder hacer uso de jpeg-6b. También tendrá que compilar PHP con la opción `--with-jpeg-dir=/ruta/a/jpeg-6b`.

Para añadir el soporte de fuentes Type 1, puede instalar t1lib (disponible en ftp://sunsite.unc.edu/pub/Linux/libs/graphics/), y entonces añadir la opción `--with-t1lib[=dir]` al recompilar.

exif_imagetype (PHP 4 CVS only)

Determine the type of an image

int|false **exif_imagetype** (string filename) \linebreak

exif_imagetype() reads the first bytes of an image and checks its signature. When a correct signature is found a constant will be returned otherwise the return value is `FALSE`. The return value is the same value that getimagesize() returns in index 2 but this function is much faster.

The following constants are defined: 1 = IMAGETYPE_GIF, 2 = IMAGETYPE_JPG, 3 = IMAGETYPE_PNG, 4 = IMAGETYPE_SWF, 5 = IMAGETYPE_PSD, 6 = IMAGETYPE_BMP, 7 = IMAGETYPE_TIFF_II (intel byte order), 8 = IMAGETYPE_TIFF_MM (motorola byte order), 9 = IMAGETYPE_JPC, 10 = IMAGETYPE_JP2, 11 = IMAGETYPE_JPX, 12 = IMAGETYPE_SWC.

This function can be used to avoid calls to other exif functions with unsupported file teypes or in conjunction with $_SERVER['HTTP_ACCEPT'] to check whether or not the viewer is able to see a specific image in his browser.

Nota: This function is only available in PHP 4 compiled using `--enable-exif`.

This function does not require the GD image library.

See also getimagesize().

exif_read_data (PHP 4 >= 4.2.0)

Read the EXIF headers from JPEG or TIFF

array **exif_read_data** (string filename [, string sections [, bool arrays [, bool thumbnail]]]) \linebreak

The **exif_read_data()** function reads the EXIF headers from a JPEG or TIFF image file. It returns an associative array where the indexes are the header names and the values are the values associated with those headers. If no data can be returned the result is `FALSE`.

`filename` is the name of the file to read. This cannot be a url.

`sections` a comma separated list of sections that need to be present in file to produce a result array.

FILE	FileName, FileSize, FileDateTime, SectionsFound
COMPUTED	html, Width, Height, IsColor and some more if available.
ANY_TAG	Any information that has a Tag e.g. IFD0, EXIF, ...
IFD0	All tagged data of IFD0. In normal imagefiles this contains image size and so forth.

THUMBNAIL	A file is supposed to contain a thumbnail if it has a second IFD. All tagged information about the embedded thumbnail is stored in this section.
COMMENT	Comment headers of JPEG images.
EXIF	The EXIF section is a sub section of IFD0. It contains more detailed information about an image. Most of these entries are digital camera related.

arrays specifies whether or not each section becomes an array. The sections *FILE*, *COMPUTED* and *THUMBNAIL* allways become arrays as they may contain values whose names are conflict with other sections.

thumbnail whether or not to read the thumbnail itself and not only its tagged data.

> **Nota:** Exif headers tend to be present in JPEG/TIFF images generated by digital cameras, but unfortunately each digital camera maker has a different idea of how to actually tag their images, so you can't always rely on a specific Exif header being present.

Ejemplo 1. exif_read_data() example

```
<?php
echo "test1.jpg:<br />\n";
$exif = exif_read_data ('tests/test1.jpg','IFD0');
echo $exif===false ? "No header data found.<br />\n" : "Image contains headers<br />";
$exif = exif_read_data ('tests/test2.jpg',0,true);
echo "test2.jpg:<br />\n";
foreach($exif as $key=>$section) {
    foreach($section as $name=>$val) {
        echo "$key.$name: $val<br />\n";
    }
}?>
```

The first call fails because the image has no header information.

```
test1.jpg:
No header data found.
test2.jpg:
FILE.FileName: test2.jpg
FILE.FileDateTime: 1017666176
FILE.FileSize: 1240
FILE.FileType: 2
FILE.SectionsFound: ANY_TAG, IFD0, THUMBNAIL, COMMENT
COMPUTED.html: width="1" height="1"
```

```
COMPUTED.Height: 1
COMPUTED.Width: 1
COMPUTED.IsColor: 1
COMPUTED.ByteOrderMotorola: 1
COMPUTED.UserComment: Exif test image.
COMPUTED.UserCommentEncoding: ASCII
COMPUTED.Copyright: Photo (c) M.Boerger, Edited by M.Boerger.
COMPUTED.Copyright.Photographer: Photo (c) M.Boerger
COMPUTED.Copyright.Editor: Edited by M.Boerger.
IFD0.Copyright: Photo (c) M.Boerger
IFD0.UserComment: ASCII
THUMBNAIL.JPEGInterchangeFormat: 134
THUMBNAIL.JPEGInterchangeFormatLength: 523
COMMENT.0: Comment #1.
COMMENT.1: Comment #2.
COMMENT.2: Comment #3end
```

Nota: If the image contains any IFD0 data then COMPUTED contains the entry ByteOrderMotorola which is 0 for little-endian (intel) and 1 for big-endian (motorola) byte order. This was added in PHP 4.3.

When an Exif header contains a Copyright note this itself can contain two values. As the solution is inconsitent in the Exif 2.10 standard the COMPUTED section will return both entries *Copyright.Photographer* and *Copyright.Editor* while the IFD0 sections contains the byte array with the NULL character that splits both entries. Or just the first entry if the datatype was wrong (normal behaviour of Exif). The COMPUTED will contain also an entry *Copyright* Which is either the original copyright string or it is a comma separated list of photo and editor copyright.

Nota: The tag UserComment has the same problem as the Copyright tag. It can store two values first the encoding used and second the value itself. If so the IFD section only contains the encoding or a byte array. The COMPUTED section will store both in the entries *UserCommentEncoding* and *UserComment*. The entry *UserComment* is available in both cases so it should be used in preference to the value in IFD0 section.

If the user comment uses Unicode or JIS encoding and the module mbstring is available this encoding will automatically changed according to the exif ini settings. This was added in PHP 4.3.

Nota: Height and Width are computed the same way getimagesize() does so their values must not be part of any header returned. Also html is a height/width text string to be used inside normal HTML.

Nota: Starting from PHP 4.3 the function can read all embedded IFD data including arrays (returned as such). Also the size of an embedded thumbnail is returned in *THUMBNAIL* subarray and the function **exif_read_data()** can return thumbnails in TIFF format. Last but not least there is no longer a maximum legth for returned values (not until memory limit is reached).

Nota: This function is only available in PHP 4 compiled using `--enable-exif`. Its functionality and behaviour has changed in PHP 4.2. Earlier versions are very unstable.

Since PHP 4.3 user comment can automatically change encoding if PHP 4 was compiled using `--enable-mbstring`.

This function does not require the GD image library.

See also exif_thumbnail() and getimagesize().

exif_thumbnail (PHP 4 >= 4.2.0)

Retrieve the embedded thumbnail of a TIFF or JPEG image

string **exif_thumbnail** (string filename [, int &width [, int &height]]) \linebreak

exif_thumbnail() reads the embedded thumbnail of a TIFF or JPEG image. If the image contains no thumbnail `FALSE` will be returned.

Both parameters *`width`* and *`height`* are available since PHP 4.3 and return the size of the thumbnail. It is possible that **exif_thumbnail()** cannot create an image but determine its size. In this case the return value is `FALSE` but *`width`* and *`height`* are set.

Starting from version PHP 4.3 the function **exif_thumbnail()** can return thumbnails in TIFF format.

Nota: This function is only available in PHP 4 compiled using `--enable-exif`. Its functionality and behaviour has changed in PHP 4.2

This function does not require the GD image library.

See also exif_read_data().

GetImageSize (PHP 3, PHP 4)

Obtiene el tamaño de una imagen GIF, JPG o PNG

array **getimagesize** (string filename [, array imageinfo]) \linebreak

La función **GetImageSize()** determinará el tamaño de cualquier fichero de imagen GIF, JPG o PNG y devolverá sus dimensiones junto al tipo de fichero en una cadena de texto que pueda ser usada en una marca HTML `IMG` normal.

Devuelve una matriz con 4 elementos. El índice 0 contiene la anchura de la imagen en pixels. El índice 1 contiene la altura. El índice 2 es una marca indicando el tipo de imagen. 1 = GIF, 2 = JPG, 3 = PNG. El índice 3 es una cadena de texto con el string correcto "height=xxx width=xxx" para ser usado directamente en una marca IMG.

Ejemplo 1. GetImageSize

```
<?php $size = GetImageSize("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>>
```

El parámetro opcional *`imageinfo`* permite extraer información adicional del fichero de imagen. Actualmente esto devolverá las diferentes marcas APP de los JPG en una matriz asociada. Algunos programas usan estas marcas APP para incluir información textual en las imagenes. Uno bastante común incluye información IPTC http://www.iptc.org/ en la marca APP13. Puede usar la función iptcparse() para convertir la marca binaria APP13 en algo leible.

Ejemplo 2. GetImageSize devolviendo IPTC

```
<?php
    $size = GetImageSize("testimg.jpg",&$info);
    if (isset($info["APP13"])) {
        $iptc = iptcparse($info["APP13"]);
        var_dump($iptc);
    }
?>
```

Nota: Esta función no requiere la libreria de imagenes GD.

image_type_to_mime_type (unknown)

Get Mime-Type for image-type returned by getimagesize, exif_read_data, exif_thumbnail, exif_imagetype

string **image_type_to_mime_type** (int imagetype) \linebreak

The **image_type_to_mime_type()** function will determine the Mime-Type for an IMAGETYPE constant.

Ejemplo 1. image_type_to_mime_type (file)

```
<?php
header ("Content-type: ".image_type_to_mime_type (IMAGETYPE_PNG));
?>
```

Nota: This function does not require the GD image library.

See also getimagesize(), exif_imagetype(), exif_read_data() and exif_thumbnail().

image2wbmp (PHP 4 >= 4.0.5)

Output image to browser or file

int **image2wbmp** (resource image [, string filename [, int threshold]]) \linebreak

image2wbmp() creates the WBMP file in filename from the image *image*. The *image* argument is the return from imagecreate().

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using header(), you can create a PHP script that outputs WBMP images directly.

Nota: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also imagewbmp().

imagealphablending (PHP 4 >= 4.0.6)

Set the blending mode for an image

int **imagealphablending** (resource image, bool blendmode) \linebreak

imagealphablending() allows for two different modes of drawing on truecolor images. In blending mode, the alpha channel component of the color supplied to all drawing function, such as imagesetpixel() determines how much of the underlying color should be allowed to shine through. As a result, gd automatically blends the existing color at that point with the drawing color, and stores the result in the

image. The resulting pixel is opaque. In non-blending mode, the drawing color is copied literally with its alpha channel information, replacing the destination pixel. Blending mode is not available when drawing on palette images. If *blendmode* is TRUE, then blending mode is enabled, otherwise disabled.

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1

ImageArc (PHP 3, PHP 4)

Dibuja una elipse parcial

int **imagearc** (int im, int cx, int cy, int w, int h, int s, int e, int col) \linebreak

ImageArc dibuja una elipse parcial centrada en cx, cy (la esquina superior izquierda es 0,0) en la imagen que representa im. w y h especifican la anchura y altura respectivamente mientras que los puntos de inicio y final vienen indicados por los parámetros s y e en grados.

ImageChar (PHP 3, PHP 4)

Dibuja un carácter horizontalmente

int **imagechar** (int im, int font, int x, int y, string c, int col) \linebreak

ImageChar dibuja el primer carácter de *c* en la imagen identificada como *id* con su esquina superior izquierda en *x*, *y* (arriba izquierda es 0,0) con el color *col*. Si la fuente es 1, 2, 3, 4 o 5 se usa una fuente predefinida (números mayores corresponden con tamaños mayores).

Vea también imageloadfont().

ImageCharUp (PHP 3, PHP 4)

Dibuja un carácter vertical

int **imagecharup** (int im, int font, int x, int y, string c, int col) \linebreak

ImageCharUp dibuja el caracter c verticalmente en la imagen identificado como im en las coordenadas x, y (arriba izquierda es 0,0) con el color col. Si la fuente es 1, 2, 3, 4 o 5 se usa una fuente predefinida.

Vea también imageloadfont().

ImageColorAllocate (PHP 3, PHP 4)

Reserva un color para una imagen

int **imagecolorallocate** (int im, int red, int green, int blue) \linebreak

ImageColorAllocate devuelve un identificador del color representado por la mezcla de los componentes RGB dados. El parámetro im es el resultado de la función imagecreate(). ImageColorAlocate tiene que ser invocada para crear cada color que va a ser usado por la imagen que representa im.

```
$white = ImageColorAllocate($im, 255,255,255);
$black = ImageColorAllocate($im, 0,0,0);
```

ImageColorAt (PHP 3, PHP 4)

Obtiende el índice del color de un pixel

int **imagecolorat** (int im, int x, int y) \linebreak

Devuelve el índice del color del pixel especificado en la posición de la imagen.

Vea también imagecolorset() y imagecolorsforindex().

ImageColorClosest (PHP 3, PHP 4)

Obtiene el índice del color más cercano al color especificado

int **imagecolorclosest** (int im, int red, int green, int blue) \linebreak

Devuelve el índice del color de la paleta de la imagen que sea más "cercano" al valor RGB especificado.

La "distancia" entre el color deseado y cada color de la paleta es calculada como si los valores RGB representasen puntos en un espacio tridimensional.

Vea también imagecolorexact().

imagecolorclosestalpha (PHP 4 >= 4.0.6)

Get the index of the closest color to the specified color + alpha

int **imagecolorclosestalpha** (resource image, int red, int green, int blue, int alpha) \linebreak

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value and *alpha* level.

See also imagecolorexactalpha().

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1

imagecolorclosesthwb (PHP 4 >= 4.0.1)

Get the index of the color which has the hue, white and blackness nearest to the given color

int **imagecolorclosesthwb** (resource image, int red, int green, int blue) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imagecolordeallocate (PHP 3>= 3.0.6, PHP 4)

De-allocate a color for an image

int **imagecolordeallocate** (resource image, int color) \linebreak

The **imagecolordeallocate()** function de-allocates a color previously allocated with the imagecolorallocate() function.

```
$white = imagecolorallocate ($im, 255, 255, 255);
imagecolordeallocate ($im, $white);
```

ImageColorExact (PHP 3, PHP 4)

Devuelve el índice del color especificado

int **imagecolorexact** (int im, int red, int green, int blue) \linebreak

Devuelve el índice del color especificado de la paleta de la imagen.

Si el color no existe en la paleta de la imagen, se devuelve el valor -1.

Vea también imagecolorclosest().

imagecolorexactalpha (PHP 4 >= 4.0.6)

Get the index of the specified color + alpha

int **imagecolorexactalpha** (resource image, int red, int green, int blue, int alpha) \linebreak

Returns the index of the specified color+alpha in the palette of the image.

If the color does not exist in the image's palette, -1 is returned.

See also imagecolorclosestalpha().

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageColorResolve (PHP 3>= 3.0.2, PHP 4)

Devuelve el índice del color especificado o su posible alternativa más cercana

int **imagecolorresolve** (int im, int red, int green, int blue) \linebreak

Esta función garantiza el resultado de un índice de color para un color solicitado, ya sea el color exacto o su alternativa más cercana.

Vea también imagecolorclosest().

imagecolorresolvealpha (PHP 4 >= 4.0.6)

Get the index of the specified color + alpha or its closest possible alternative

int **imagecolorresolvealpha** (resource image, int red, int green, int blue, int alpha) \linebreak

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

See also imagecolorclosestalpha().

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1

ImageColorSet (PHP 3, PHP 4)

Establece el color para el índice de la paleta especificado

bool **imagecolorset** (int im, int index, int red, int green, int blue) \linebreak

Establece el índice especificado de la paleta con el color introduciod. Esto es útil para crear efectos de rellenado en imagenes con paletas sin la sobrecarga de tener que realizar el rellenado.

Vea también imagecolorat().

ImageColorsForIndex (PHP 3, PHP 4)

Obtiene los colores de un índice

array **imagecolorsforindex** (int im, int index) \linebreak

Devuelve una matriz asociativa con las claves red, green y blue que contienen los valores apropiados para el color especificado en el índice.

Vea también imagecolorat() y imagecolorexact().

ImageColorsTotal (PHP 3, PHP 4)

Encuentra el número de colores de la paleta de una imagen

int **imagecolorstotal** (int im) \linebreak

Encuentra el número de colores de la paleta de una imagen.

Vea también imagecolorat() y imagecolorsforindex().

ImageColorTransparent (PHP 3, PHP 4)

Define un color como transparente

int **imagecolortransparent** (int im [, int col]) \linebreak

ImageColorTransparent establece como color transparente a col en la imagen im. im es el identificador de imagen devuelto por imagecreate() y col es el identificador de color devuelto por imagecolorallocate().

Se devuelve el identificador del color transparente (o el actual, si no se especifica ninguno).

imagecopy (PHP 3>= 3.0.6, PHP 4)

Copy part of an image

int **imagecopy** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h) \linebreak

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*.

imagecopymerge (PHP 4 >= 4.0.1)

Copy and merge part of an image

int **imagecopymerge** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct) \linebreak

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to imagecopy().

Nota: This function was added in PHP 4.0.6

imagecopymergegray (PHP 4 >= 4.0.6)

Copy and merge part of an image with gray scale

int **imagecopymergegray** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct) \linebreak

imagecopymergegray() copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to imagecopy().

This function is identical to imagecopymerge() except that when merging it preservese the hue of the source by converting the destination pixels to gray scale before the copy operation.

Nota: This function was added in PHP 4.0.6

imagecopyresampled (PHP 4 >= 4.0.6)

Copia y reescala parte de una imagen con remuestreo

int **imagecopyresampled** (resource img_dst, resource img_org, int Xdst, int Ydst, int Xorg, int Yorg, int ancho_dst, int alto_dst, int ancho_org, int alto_org) \linebreak

imagecopyresampled() copia una porción rectangular de una imagen sobre otra, suavizando los valores de los píxeles mediante interpolación, de forma que al reducir el tamaño de una imagen aún mantiene una buena claridad. *img_dst* es la imagen de destino, *img_org* es el identificador de la imagen de origen. Si las coordenadas de origen y destino y ancho y alto son diferentes, se encogerá o agrandará el fragmento de imagen según sea necesario. Las coordenadas son relativas a la esquina superior izquierda. Esta función se puede usar para copiar regiones dentro de la misma imagen (si *img_dst* es la misma que *img_org*) pero si las regiones se superponen, los resultados serán impredecibles.

Vea también imagecopyresized().

Nota: Esta función fue añadida en PHP 4.0.6 y requiere GD 2.0.1 o superior

ImageCopyResized (PHP 3, PHP 4)

Copia y redimensiona parte de una imagen

int **imagecopyresized** (int dst_im, int src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH) \linebreak

ImageCopyResize copia una porción rectangular de una imagen hacia otra imagen. *dst_im* es la imagen de destino, *src_im* es el identificador de la imagen origen. Si la altura y anchura de las coordenadas de origen y destino difieren se realizará un estrechamiento o un estiramiento apropiado del fragmento de la imagen. Las coordenadas van localizadas sobre la esquina superior izquierda. Esta función se puede usar para copiar regiones dentro de la misma imagen (si *dst_im* es igual que *src_im*) pero si las regiones se solapan los resultados seran impredecibles.

ImageCreate (PHP 3, PHP 4)

Crea una nueva imagen

int **imagecreate** (int x_size, int y_size) \linebreak

ImageCreate devuelve un identificador de imagen representando una imagen en blanco de tamaño x_size por y_size.

imagecreatefromgd2 (PHP 4 >= 4.1.0)

Create a new image from GD2 file or URL

resource **imagecreatefromgd2** (string filename) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imagecreatefromgd2part (PHP 4 >= 4.1.0)

Create a new image from a given part of GD2 file or URL

resource **imagecreatefromgd2part** (string filename, int srcX, int srcY, int width, int height) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imagecreatefromgd (PHP 4 >= 4.1.0)

Create a new image from GD file or URL

resource **imagecreatefromgd** (string filename) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ImageCreateFromGif (PHP 3, PHP 4)

Crea una nueva imagen desde un fichero o una URL

int **imagecreatefromgif** (string filename) \linebreak

imagecreatefromgif() devuelve un identificador de imagen representando la imagen obtenida del nombre del fichero dado.

imagecreatefromgif() devuelve una cadena vacia si hay algun fallo. Además muestra un mensaje de error, que desafortunadamente se representa como un link roto en un navegador. Para depurarlo fácilmente el siguiente ejemplo producirá un error de GIF:

Ejemplo 1. Ejemplo de control de un error durante la creación (cortesía vic@zymsys.com)

```
function LoadGif($imgname)
{
  $im = @imagecreatefromgif($imgname); /* Attempt to open */
  if ($im == "") { /* See if it failed */
    $im = ImageCreate(150,30); /* Create a blank image */
    $bgc = ImageColorAllocate($im,255,255,255);
    $tc  = ImageColorAllocate($im,0,0,0);
    ImageFilledRectangle($im,0,0,150,30,$bgc);
    ImageString($im,1,5,5,"Error loading $imgname",$tc); /* Output an errmsg */
  }
  return $im;
}
```

Nota: Desde que todo el soporte a GIFs ha sido eliminado en la libreria GD a partir de la versión 1.6, esta función no está disponible si está usando esa versión de la libreria GD.

imagecreatefromjpeg (PHP 3>= 3.0.16, PHP 4)

Crea una imagen nueva desde un archivo o URL

resource **imagecreatefromjpeg** (string nombre_archivo) \linebreak

imagecreatefromjpeg() devuelve un identificador de imagen que representa a la imagen obtenida a partir del nombre de archivo indicado.

imagecreatefromjpeg() devuelve una cadena vacía si ha fallado. También escribe un mensaje de error, que desafortunadamente se muestra en el navegador como un enlace roto. Para depurar con mayor comodidad, el ejemplo siguiente producirá un JPEG erróneo:

Ejemplo 1. Ejemplo de cómo manipular un error durante la creación (cortesía de vic@zymsys.com)

```
function CargarJpeg ($nombreimg) {
    $im = @imagecreatefromjpeg ($nombreimg); /* Intento de apertura */
    if (!$im) { /* Comprobar si ha fallado */
        $im  = imagecreate (150, 30); /* Crear una imagen en blanco */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Mostrar un mensaje de error */
        imagestring ($im, 1, 5, 5, "Error cargando $nombreimg", $tc);
    }
    return $im;
}
```

imagecreatefrompng (PHP 3>= 3.0.13, PHP 4)

Crea una imagen nueva desde un fichero o URL

resource **imagecreatefrompng** (string nombre_archivo) \linebreak

imagecreatefrompng() devuelve un identificador de imagen que representa a la imagen obtenida a partir del nombre de archivo indicado.

imagecreatefrompng() devuelve una cadena vacía si ha fallado. También escribe un mensaje de error, que desafortunadamente se muestra en el navegador como un enlace roto. Para depurar con mayor comodidad, el ejemplo siguiente producirá un JPEG erróneo:

Ejemplo 1. Ejemplo de cómo manipular un error durante la creación (cortesía de vic@zymsys.com)

```
function CargarJpeg ($nombreimg) {
    $im = @imagecreatefrompng ($nombreimg); /* Intento de apertura */
    if (!$im) { /* Comprobar si ha fallado */
        $im  = imagecreate (150, 30); /* Crear una imagen en blanco */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Mostrar un mensaje de error */
        imagestring ($im, 1, 5, 5, "Error cargando $nombreimg", $tc);
    }
    return $im;
}
```

imagecreatefromstring (PHP 4 >= 4.0.4)

Create a new image from the image stream in the string

resource **imagecreatefromstring** (string image) \linebreak

imagecreatefromstring() returns an image identifier representing the image obtained from the given string.

imagecreatefromwbmp (PHP 4 >= 4.0.1)

Create a new image from file or URL

resource **imagecreatefromwbmp** (string filename) \linebreak

imagecreatefromwbmp() returns an image identifier representing the image obtained from the given filename.

imagecreatefromwbmp() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error WBMP:

Ejemplo 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadWBMP ($imgname) {
    $im = @imagecreatefromwbmp ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = imagecreate (20, 20); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 10, 10, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

Nota: WBMP support is only available if PHP was compiled against GD-1.8 or later.

imagecreatefromxbm (PHP 4 >= 4.0.1)

Create a new image from file or URL

resource **imagecreatefromxbm** (string filename) \linebreak

imagecreatefromxbm() returns an image identifier representing the image obtained from the given filename.

imagecreatefromxpm (PHP 4 >= 4.0.1)

Create a new image from file or URL

resource **imagecreatefromxpm** (string filename) \linebreak

imagecreatefromxpm() returns an image identifier representing the image obtained from the given filename.

imagecreatetruecolor (PHP 4 >= 4.0.6)

Crea una imagen nueva en color real (true color)

resource **imagecreatetruecolor** (int anchura, int altura) \linebreak

imagecreatetruecolor() devuelve un identificador de imagen representando una imagen en blanco de tamaño *anchura* por *altura*.

Nota: Esta función fue añadida en PHP 4.0.6 y requiere GD 2.0.1 o superior

ImageDashedLine (PHP 3, PHP 4)

Dibuja una línea discontinua

int **imagedashedline** (int im, int x1, int y1, int x2, int y2, int col) \linebreak

ImageLine dibuja una línea discontinua desde x1,y1 hasta x2, y2 (arriba izquierda es 0.0) en la imagen im con el color col.

Vea también imageline().

ImageDestroy (PHP 3, PHP 4)

Destruye una imagen

int **imagedestroy** (int im) \linebreak

ImageDestroy libera la memoria asociada a la imagen im. im es la imagen devuelta por la función imagecreate().

imageellipse (PHP 4 >= 4.0.6)

Draw an ellipse

int **imageellipse** (resource image, int cx, int cy, int w, int h, int col) \linebreak

imageellipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively. The color of the ellipse is specified by *color*.

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.2 or later

See also imagearc().

ImageFill (PHP 3, PHP 4)

Relleno

int **imagefill** (int im, int x, int y, int col) \linebreak

ImageFill realiza un relleno empezando en la coordenada x,y (arriba izquierda es 0,0) con el color col en la imagen im.

imagefilledarc (PHP 4 >= 4.0.6)

Draw a partial ellipse and fill it

int **imagefilledarc** (resource image, int cx, int cy, int w, int h, int s, int e, int col, int style) \linebreak

imagefilledarc() draws a partial ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the *s* and *e* arguments. *style* is a bitwise OR of the following possibilities:

1. IMG_ARC_PIE
2. IMG_ARC_CHORD
3. IMG_ARC_NOFILL
4. IMG_ARC_EDGED

IMG_ARC_PIE and IMG_ARC_CHORD are mutually exclusive; IMG_ARC_CHORD just connects the starting and ending angles with a straight line, while IMG_ARC_PIE produces a rounded edge. IMG_ARC_NOFILL indicates that the arc or chord should be outlined, not filled. IMG_ARC_EDGED, used together with IMG_ARC_NOFILL, indicates that the beginning and ending angles should be connected to the center - this is a good way to outline (rather than fill) a 'pie slice'.

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1

imagefilledellipse (PHP 4 >= 4.0.6)

Draw a filled ellipse

int **imagefilledellipse** (resource image, int cx, int cy, int w, int h, int col) \linebreak

imagefilledellipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively. The ellipse is filled using *color*

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

See also imagefilledarc().

ImageFilledPolygon (PHP 3, PHP 4)

Dibuja un polígono relleno

int **imagefilledpolygon** (int im, array points, int num_points, int col) \linebreak

ImageFilledPolygon crea un polígono relleno en la imagen im, points es una matriz PHP conteniendo los vértices del poligono, por ejemplo. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. num_points es el número total de vértices.

ImageFilledRectangle (PHP 3, PHP 4)

dibuja un rectángulo relleno

int **imagefilledrectangle** (int im, int x1, int y1, int x2, int y2, int col) \linebreak

ImageFilledRectangle crea un rectángulo relleno con color col en la imagen im comenzando con las coordenadas superiores izquierdas x1, y1 y finalizando en las coordenadas inferiores derechas x2, y2. 0,0 es la esquina superior izquierda de la imagen.

ImageFillToBorder (PHP 3, PHP 4)

Relleno de un color especifico

int **imagefilltoborder** (int im, int x, int y, int border, int col) \linebreak

ImageFillToBorder realiza un relleno hasta el color del borde que está definido por border. El punto de inicio para el relleno es x,y (arriba izquierda es 0,0) y la región se rellena con el color col.

ImageFontHeight (PHP 3, PHP 4)

Devuelve la altura de una fuente

int **imagefontheight** (int font) \linebreak

Devuelve la altura en pixels de un carácter en un fuente específica.

Vea también imagefontwidth() y imageloadfont().

ImageFontWidth (PHP 3, PHP 4)

Devuelve la anchura de una fuente

int **imagefontwidth** (int font) \linebreak

Devuelve la anchura en pixels de un carácter en un fuente específica.

Vea también imagefontheight() y imageloadfont().

imageftbbox (PHP 4 >= 4.1.0)

Give the bounding box of a text using fonts via freetype2

array **imageftbbox** (int size, int angle, string font_file, string text [, array extrainfo]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imagefttext (PHP 4 >= 4.1.0)

Write text to the image using fonts using FreeType 2

array **imagefttext** (resource image, int size, int angle, int x, int y, int col, string font_file, string text [, array extrainfo]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imagegammacorrect (PHP 3>= 3.0.13, PHP 4)

Apply a gamma correction to a GD image

int **imagegammacorrect** (resource image, float inputgamma, float outputgamma) \linebreak

The **imagegammacorrect()** function applies gamma correction to a gd image stream (*image*) given an input gamma, the parameter *inputgamma* and an output gamma, the parameter *outputgamma*.

imagegd2 (PHP 4 >= 4.1.0)

Output GD2 image to browser or file

int **imagegd2** (resource image [, string filename]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imagegd (PHP 4 >= 4.1.0)

Output GD image to browser or file

int **imagegd** (resource image [, string filename]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ImageGif (PHP 3, PHP 4)

Envía una imagen al navegador o a un fichero

int **imagegif** (int im, string filename) \linebreak

imagegif() crea el fichero GIF en filename a partir de la imagen *im*. El parámetro *im* es el resultado de usar la función imagecreate().

El formato de la imagen será GIF87a a menos que la imagen se halla hecho transparente con imagecolortransparent(), en cuyo caso el formato de la imagen será GIF89a.

El parametro del nombre del fuichero es opcional, y si se deja en blanco, la imagen será mostrada directamente. Enviando un tipo de imagen/gif usando la función header(), puede crear un script PHP que muestre imagenes GIF directamente.

Nota: Desde que todo el soporte a GIFs ha sido eliminado en la libreria GD a partir de la versión 1.6, esta función no está disponible si está usando esa versión de la libreria GD.

ImageInterlace (PHP 3, PHP 4)

Activa o desactiva el entrelazado

int **imageinterlace** (int im [, int interlace]) \linebreak

ImageInterlace() activa o desactiva el bit de entrelazado. Si interlace es 1 la imagen im será entrelazada, y si interlace es 0 el bit de entrelazado se desactiva.

Esta función devuelve como ha cambiado el estado del bit de entralazado de la imagen.

imagejpeg (PHP 3>= 3.0.16, PHP 4)

Output image to browser or file

int **imagejpeg** (resource image [, string filename [, int quality]]) \linebreak

imagejpeg() creates the JPEG file in filename from the image *image*. The *image* argument is the return from the imagecreate() function.

The filename argument is optional, and if left off, the raw image stream will be output directly. To skip the filename argument in order to provide a quality argument just use an empty string (''). By sending an image/jpeg content-type using header(), you can create a PHP script that outputs JPEG images directly.

Nota: JPEG support is only available if PHP was compiled against GD-1.8 or later.

quality is optional, and ranges from 0 (worst quality, smaller file) to 100 (best quality, biggest file). The default is the default IJG quality value (about 75).

If you want to output Progressive JPEGs, you need to set interlacing on with imageinterlace().

See also imagepng(), imagegif(), imagewbmp(), imageinterlace() and imagetypes().

ImageLine (PHP 3, PHP 4)

Dibuja una línea

int **imageline** (int im, int x1, int y1, int x2, int y2, int col) \linebreak

ImageLine dibuja una línea desde x1,y1 hasta x2,y2 (arriba izquierda es 0,0) en la imagen im con el color col.

Vea también imagecreate() y imagecolorallocate().

ImageLoadFont (PHP 3, PHP 4)

Carga una fuente nueva

int **imageloadfont** (string file) \linebreak

ImageLoadFont carga una fuente de bitmaps definida por el usuario y devuelve un identificador para esa fuente (que siempre es mayor de 5, de forma que no pueda entrar en conflicto con las fuentes predefinidas)..

El formato del fichero de la fuente es actualmente binario y dependiente de la arquitectura. Esto significa que tiene que generar los ficheros de las fuentes en el mismo tipo de CPU que la que tiene la máquina que está ejecutando PHP.

Tabla 1. Formato del fichero de fuentes

Posición en bytes	tipo de datos C	Descripción
byte 0-3	int	número de carácteres en la fuente
byte 4-7	int	valor del primer carácter de la fuente (normalmente 32 para el espacio)
byte 8-11	int	Anchura en pixels de cada carácter
byte 12-15	int	Altura en pixels de cada carácter
byte 16-	char	matriz con los datos del carácter, un byte por pixel en cada carácter, haciendo un total de (número carácteres* altura*anchura) bytes.

Vea también **ImageFontWidth()** y **ImageFontHeight()**.

imagepalettecopy (PHP 4 >= 4.0.1)

Copy the palette from one image to another

int **imagepalettecopy** (resource destination, resource source) \linebreak

imagepalettecopy() copies the palette from the *`source`* image to the *`destination`* image.

imagepng (PHP 3>= 3.0.13, PHP 4)

Output a PNG image to either the browser or a file

int **imagepng** (resource image [, string filename]) \linebreak

The **imagepng()** outputs a GD image stream (*image*) in PNG format to standard output (usually the browser) or, if a filename is given by the *filename* it outputs the image to the file.

```
<?php
$im = imagecreatefrompng ("test.png");
imagepng ($im);
?>
```

See also imagegif(), imagewbmp(), imagejpeg(), imagetypes().

ImagePolygon (PHP 3, PHP 4)

Dibuja un polígono

int **imagepolygon** (int im, array points, int num_points, int col) \linebreak

ImagePolygon crea un polígono en la imagen id. points es un array PHP conteniendo los vértices del polígono. de la siguiente forma points[0] = x0, points1 = y0, points[2] = x1, points[3] = y1, etc. num_points es el número total de vértices.

Vea también imagecreate().

ImagePSBBox (PHP 3>= 3.0.9, PHP 4)

Devuelve el borde que rodea un rectángulo de texto usando fuentes PostScript Type1

array **imagepsbbox** (string text, int font, int size, int space, int width, float angle) \linebreak

size representa pixels.

space perimte cambiar el valor por defecto de un espacio en una fuentes. Este valor es añadido al valor normal y puede ser negativo.

tightness permite controlar la cantidad de espacio en blanco entre carácteres. Este valor se añade a la anchura normal del carácter y puede ser negativo.

angle viene dado en grados.

Los parámetros *space* y *tightness* vienen expresados en unidades de espacio de carácteres, donde una unidad es 1/1000 el borde de una M.

Los parámetros *space*, *tightness* y *angle* son opcionales.

El borde es calculado usando la información disponible de las métricas del carácter, y desafortunadamente tiende a diferir ligeramente de los resultados obtenidos de digitalizar el texto. Si el ángulo es de 0 grados, puede esperar que el texto necesite un pixel más en cada dirección.

Esta función devuelve un array conteniuendo los siguientes elementos:

0	coordenada x inferior izquierda
1	coordenada y inferior izquierda
2	coordenada x superior derecha
3	coordenada y superior derecha

Vea también imagepstext().

ImagePSCopyFont (PHP 3>= 3.0.9, PHP 4)

hace una copia de una fuente ya cargada para futuras modificaciones

int **imagepscopyfont** (int fontindex) \linebreak

Use esta función si necesita hacer modificaciones en la fuente, por ejemplo expander/condensar, inclinarla o cambiar su vector de codificación de caracteres, pero también necesita mantener la fuente original. Note que la fuente que quiera copiar debe haber sido obtenida usando imagepsloadfont(), no una fuente que sea una copia de otra. Aunque puede hacer modificaciones antes de copiarla.

Si usa esta función, *debe* liberar las fuentes obtenidas de esta manera. De otra forma su script se *colgará*.

En el caso de que todo vaya bien, devolverá un índice de fuente válido que puede ser usado para futuos propósitos. De otra forma la función devolverá FALSE e imprimirá un mensaje indicando que es lo que ha ido mal.

Vea también **imageploadpsfont()**.

ImagePSEncodeFont (PHP 3>= 3.0.9, PHP 4)

Cambia el vector de codificación de caracteres de una fuente

int **imagepsencodefont** (string encodingfile) \linebreak

Carga un vector de codificación de caracteres desde un archivo y cambia el vector de codificación de las fuentes a él. Loads a character encoding vector from from a file and changes the fonts encoding vector to it. En las fuentes PostScript normalmente faltan muchos caracteres por encima de 127, seguramente

quiera cambiar esto si emplea u idioma distinto del inglés.El formato exacto de este archivo está descrito en la documentación de T1libs. T1lib viene con dos archivos listos para usar, IsoLatin1.enc y IsoLatin2.enc.

Si se encuentra usando esta función todo el tiempo, una forma mucho mejor de definir la codificación es establecer ps.default_encoding en el archivo de configuración para que apunte al archivo de codificación correcto y todas las fuentes que cargue tendrán de forma automática la codificación correcta.

imagepsextendfont (PHP 3>= 3.0.9, PHP 4)

Extend or condense a font

bool **imagepsextendfont** (int font_index, float extend) \linebreak

Extend or condense a font (*`font_index`*), if the value of the *`extend`* parameter is less than one you will be condensing the font.

ImagePSFreeFont (PHP 3>= 3.0.9, PHP 4)

Libera la memoria usada por un fuente PostScript Type 1

void **imagepsfreefont** (int fontindex) \linebreak

Vea también imagepsloadfont().

ImagePSLoadFont (PHP 3>= 3.0.9, PHP 4)

Carga una fuente PostScript Type 1 desde un fichero

int **imagepsloadfont** (string filename) \linebreak

En el caso de que todo vaya bien, tiene que devolver un índice de fuente correcto que puede ser usado para futuras operaciones. En caso contrario la función devuelve `FALSE` e imprime un mensaje describiendo donde ha fallado

Vea también imagepsfreefont().

imagepsslantfont (PHP 3>= 3.0.9, PHP 4)

Slant a font

bool **imagepsslantfont** (int font_index, float slant) \linebreak

Slant a font given by the *font_index* parameter with a slant of the value of the *slant* parameter.

ImagePSText (PHP 3>= 3.0.9, PHP 4)

Para dibujar una cadena de texto sobre una imagen usando fuentes PostScript Type1

array **imagepstext** (int image, string text, int font, int size, int foreground, int background, int x, int y [, int space [, int tightness [, float angle [, int antialias_steps]]]]) \linebreak

size viene expresado en pixels.

foreground es el color en el cual el texto será pintado. *background* es el color en el que el texto tratará de resaltar con antialiaing. Los pixels con el color *background* no se pintan, de forma que la imagen de fondo no necesita ser de un color sólido.

Las coordenadas dadas por *x*, *y* definirán el origen (o punto de referencia) del primer carácter (la esquina superior izquierda del carácter). Esto es diferente de la función **ImageString()**, donde x, y definen la esquina superiro derecha del primer carácter. Consulte la documentación de PostScript sobre fuentes y su sistema de medidas si tiene algún problema entendiendo como funcion.

space permite cambiar el valor por defecto de un espacio en la fuente. Esta cantidad es sumada al valor normal y puede ser negativa.

tightness permite controlar la cantidad de espacio en blanco entre caracteres. Esta cantidad es sumada al valor normal y puede ser negativa.

angle viene en grados.

antialias_steps permite controlar el número de colores usados para el antialising del texto. Los valores permitidos son 4 y 16. El valor superior está recomendado para textos con tamaños inferiores a 20, donde el efecto en la calidad del texto es bastante visible. Con tamaños superiores use 4. Hace un menor uso de cálculo.

Parameters *space* y *tightness* están expresados en unidades de espacio de caracteres, donde 1 unidad es 1/1000 de una M mayúscula.

Los parámetros *space*, *tightness*, *angle* y *antialias* son opcionales.

Esta función devuelve una matriz conteniendo los siguientes elementos:

0	coordenada x inferior izquierda
1	coordenada y inferior izquierda
2	coordenada x superior derecha
3	coordenada y superior derecha

Vea también imagepsbbox().

ImageRectangle (PHP 3, PHP 4)

Dibuja un rectángulo

int **imagerectangle** (int im, int x1, int y1, int x2, int y2, int col) \linebreak

ImageRectangle crea un rectángulo de color col en la imagen im comenzando en la coordenada superior izquierda x1,y1 y finalizando en la coordenada inferior derecha x2,y2. 0,0 es la esquina superior izquierda de la imagen.

imagesetbrush (PHP 4 >= 4.0.6)

Set the brush image for line drawing

int **imagesetbrush** (resource image, resource brush) \linebreak

imagesetbrush() sets the brush image to be used by all line drawing functions (such as imageline() and imagepolygon()) when drawing with the special colors `IMG_COLOR_BRUSHED` or `IMG_COLOR_STYLEDBRUSHED`.

Nota: You need not take special action when you are finished with a brush, but if you destroy the brush image, you must not use the `IMG_COLOR_BRUSHED` or `IMG_COLOR_STYLEDBRUSHED` colors until you have set a new brush image!

Nota: This function was added in PHP 4.0.6

ImageSetPixel (PHP 3, PHP 4)

Dibuja un pixel

int **imagesetpixel** (int im, int x, int y, int col) \linebreak

ImageSetPixel dibuja un pixel x,y (arriba izquierda 0,0) en la imagen im con color col.

Vea también imagecreate() y imagecolorallocate().

imagesetstyle (PHP 4 >= 4.0.6)

Set the style for line drawing

int **imagesetstyle** (resource image, array style) \linebreak

imagesetstyle() sets the style to be used by all line drawing functions (such as imageline() and imagepolygon()) when drawing with the special color `IMG_COLOR_STYLED` or lines of images with color `IMG_COLOR_STYLEDBRUSHED`.

The *`style`* parameter is an array of pixels. Following example script draws a dashed line from upper left to lower right corner of the canvas:

Ejemplo 1. imagesetstyle

```
<?php
header ("Content-type: image/png");
$im  = imagecreate (100, 100);
$w   = imagecolorallocate ($im, 255, 255, 255);
$red = imagecolorallocate ($im, 255, 0, 0);

/* Draw a dashed line, 5 red pixels, 5 white pixels */
$style = array ($red,$red,$red,$red,$red,$w,$w,$w,$w,$w);
imagesetstyle ($im, $style);
imageline ($im, 0, 0, 100, 100, IMG_COLOR_STYLED);

/* Draw a line of happy faces using imagesetbrush() with imagesetstyle */
$style = array ($w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$red);
imagesetstyle ($im, $style);

$brush = imagecreatefrompng ("http://www.libpng.org/pub/png/images/smile.happy.png");
imagecolortransparent ($brush, $w);
imagesetbrush ($im, $brush);
imageline ($im, 100, 0, 0, 100, IMG_COLOR_STYLEDBRUSHED);

imagepng ($im);
imagedestroy ($im);
?>
```

See also imagesetbrush(), imageline().

Nota: This function was added in PHP 4.0.6

imagesetthickness (PHP 4 >= 4.0.6)

Set the thickness for line drawing

void **imagesetthickness** (resource image, int thickness) \linebreak

imagesetthickness() sets the thickness of the lines drawn when drawing rectangles, polygons, ellipses etc. etc. to *`thickness`* pixels.

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagesettile (PHP 4 >= 4.0.6)

Set the tile image for filling

int **imagesettile** (resource image, resource tile) \linebreak

imagesettile() sets the tile image to be used by all region filling functions (such as imagefill() and imagefilledpolygon()) when filling with the special color `IMG_COLOR_TILED`.

A tile is an image used to fill an area with a repeated pattern. *Any* GD image can be used as a tile, and by setting the transparent color index of the tile image with imagecolortransparent(), a tile allows certain parts of the underlying area to shine through can be created.

Nota: You need not take special action when you are finished with a tile, but if you destroy the tile image, you must not use the `IMG_COLOR_TILED` color until you have set a new tile image!

Nota: This function was added in PHP 4.0.6

ImageString (PHP 3, PHP 4)

Dibuja una cadena de texto horizintalmente

int **imagestring** (int im, int font, int x, int y, string s, int col) \linebreak

ImageString dibuja la cadena s en la imagen identificada por im en las coordenadas x,y (arriba izquierda es 0,0) en el color col. Si la fuente es 1, 2, 3, 4 o 5, se emplea una fuente interna.

Vea también imageloadfont().

ImageStringUp (PHP 3, PHP 4)

Dibuja una cadena de texto verticalmente

int **imagestringup** (int im, int font, int x, int y, string s, int col) \linebreak

ImageStringUp dibuja la cadena s verticalmente en la imagen identificada por im en las coordenadas x,y (arriba izquierda es 0,0) en el color col. Si la fuente es 1, 2, 3, 4 o 5, se usa una fuente interna.

Vea también imageloadfont().

ImageSX (PHP 3, PHP 4)

Obtiene la anchura de la imagen

int **imagesx** (int im) \linebreak

ImageSX devuelva la anchura de la imagen identificado por im.

Vea también imagecreate() y imagesy().

ImageSY (PHP 3, PHP 4)

Obtiene la altura de la imagen

int **imagesy** (int im) \linebreak

ImageSY devuelve la altura de la imagen identificada por im.

Vea también imagecreate() y imagesx().

imagetruecolortopalette (PHP 4 >= 4.0.6)

Convert a true color image to a palette image

void **imagetruecolortopalette** (resource image, bool dither, int ncolors) \linebreak

imagetruecolortopalette() converts a truecolor image to a palette image. The code for this function was originally drawn from the Independent JPEG Group library code, which is excellent. The code has been modified to preserve as much alpha channel information as possible in the resulting palette, in addition to preserving colors as well as possible. This does not work as well as might be hoped. It is usually best to simply produce a truecolor output image instead, which guarantees the highest output quality.

dither indicates if the image should be dithered - if it is TRUE then dithering will be used which will result in a more speckled image but with better color approximation.

ncolors sets the maximum number of colors that should be retained in the palette.

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageTTFBBox (PHP 3>= 3.0.1, PHP 4)

Devuelve un caja que rodea al texto usando fuentes TypeType

array **ImageTTFBBox** (int size, int angle, string fontfile, string text) \linebreak

Esta función calcula y devuelve un rectángulo en pixels que encierra un texto con TrueType.

text

La cadena que ha de ser medida.

size

El tamaño de la fuente.

fontfile

El nombre del archivo de fuente TrueType. (Puede ser también una URL.)

angle

Ángulo en grados en el *text* que va a ser medido.

ImageTTFBBox() devuelve una matriz con 8 elementos representando cuatro puntos que hacen una caja rodeando al texto:

0	esquina inferior izquierda, posición X
1	esquina inferior izquierda, posición Y
2	esquina inferior derecha, posición X
3	esquina inferior derecha, posición Y
4	esquina superior derecha, posición X
5	esquina superior derecha, posición Y
6	esquina superior izquierda, posición X
7	esquina superior izquierda, posición Y

Los puntos son relativos a *text* a pesar del ángulo, de forma que "superior izquierda" significa la esquina superior izquierda viendo el texto horizontalmente.

Esta función requiere tanto la libreria GD como la libreria Freetype.

Vea también **ImageTTFText()**.

ImageTTFText (PHP 3, PHP 4)

Escribe texto en la imagen usando fuentes TrueType

array **ImageTTFText** (int im, int size, int angle, int x, int y, int col, string fontfile, string text) \linebreak

ImageTTFText escribe la cadena*text* en la imagen identificada por *im*, comenzando en las coordenadas *x,y* (arriba izquierda es 0,0), con un ángulo de *angle* en el color *col*, usando el ficheor de fuente TrueType identificado por *fontfile*.

Las coordenadas dadas por *x*,*y* definirán el punto base del primer carácter. (la esquina inferior izquierda del carácter). Esto es diferente de la función **ImageString()**, donde x,y definen la esquina superior derecha del primer carácter.

El *angle* viene dado en grados, donde 0 grados representa el texto de izquierda a derecha (dirección las 3 en punto), y valores superiores representan una rotación en el sentido de las agujas de un reloj. (ej. un valor de 90 representaría un texto que fuese de abajo hacia arriba).

fontfile es la ruta hacia la fuente TrueType que desee usar.

text es la cadena de texto que puede incluir secuencias de caracteres UTF-8 (de la forma: &123;) para acceder a caracteres de la fuente más allá de los primeros 255.

col es el índice de color. El uso de un índice de color negativo tiene el efecto de desactivar el antialiasing.

ImageTTFText() devuelve una matriz con 8 elementos representando cuatro puntos que hace una caja que cubre el texto.El orden de los puntos en arriba izquierda, arriba derecha, abajo derecha, abajo izquierda. Los puntos son relativos al texto a pesar del ángulo, por lo que "arriba izquierda" significa en la esquina superior izquierda cuando ve el texto horizontalmente.

Este script de ejemplo producirá un GIF negro de 400x30 pixels, con las palabras "Testing..." en blanco con la fuente Arial.

Ejemplo 1. ImageTTFText

```
<?php
Header("Content-type: image/gif");
$im = imagecreate(400,30);
$black = ImageColorAllocate($im, 0,0,0);
$white = ImageColorAllocate($im, 255,255,255);
ImageTTFText($im, 20, 0, 10, 20, $white, "/path/arial.ttf", "Testing... Omega: &#937;");
ImageGif($im);
ImageDestroy($im);
?>
```

Esta función requiere la libreria GD y la libreria FreeType (http://www.freetype.org/).

Vea también **ImageTTFBBox()**.

imagetypes (PHP 3 CVS only, PHP 4 >= 4.0.2)

Return the image types supported by this PHP build

int **imagetypes** (void) \linebreak

This function returns a bit-field corresponding to the image formats supported by the version of GD linked into PHP. The following bits are returned, `IMG_GIF` | `IMG_JPG` | `IMG_PNG` | `IMG_WBMP`. To check for PNG support, for example, do this:

Ejemplo 1. imagetypes

```
<?php
if (imagetypes() & IMG_PNG) {
    echo "PNG Support is enabled";
}
?>
```

imagewbmp (PHP 3>= 3.0.15, PHP 4 >= 4.0.1)

Output image to browser or file

int **imagewbmp** (resource image [, string filename [, int foreground]]) \linebreak

imagewbmp() creates the WBMP file in filename from the image *`image`*. The *`image`* argument is the return from the imagecreate() function.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using header(), you can create a PHP script that outputs WBMP images directly.

Nota: WBMP support is only available if PHP was compiled against GD-1.8 or later.

Using the optional *`foreground`* parameter, you can set the foreground color. Use an identifier obtained from imagecolorallocate(). The default foreground color is black.

See also image2wbmp(), imagepng(), imagegif(), imagejpeg(), imagetypes().

iptcembed (PHP 3>= 3.0.7, PHP 4)

Embed binary IPTC data into a JPEG image

array **iptcembed** (string iptcdata, string jpeg_file_name [, int spool]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

iptcparse (PHP 3>= 3.0.6, PHP 4)

Parse a binary IPTC http://www.iptc.org/ block into single tags.

array **iptcparse** (string iptcblock) \linebreak

This function parses a binary IPTC block into its single tags. It returns an array using the tagmarker as an index and the value as the value. It returns `FALSE` on error or if no IPTC data was found. See getimagesize() for a sample.

jpeg2wbmp (PHP 4 >= 4.0.5)

Convert JPEG image file to WBMP image file

int **jpeg2wbmp** (string jpegname, string wbmpname, int d_height, int d_width, int threshold) \linebreak

Converts the *`jpegname`* JPEG file to WBMP format, and saves it as *`wbmpname`*. With the *`d_height`* and *`d_width`* you specify the height and width of the destination image.

Nota: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also png2wbmp().

png2wbmp (PHP 4 >= 4.0.5)

Convert PNG image file to WBMP image file

int **png2wbmp** (string pngname, string wbmpname, int d_height, int d_width, int threshold) \linebreak

Converts the *`pngname`* PNG file to WBMP format, and saves it as *`wbmpname`*. With the *`d_height`* and *`d_width`* you specify the height and width of the destination image.

Nota: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also jpeg2wbmp().

read_exif_data (PHP 4 >= 4.0.1)

Reads header information stored in TIFF and JPEG images

array **exif_read_data** (string filename, string sections, bool arrays, bool thumbnail) \linebreak

Nota: The **read_exif_data()** function is an alias for exif_read_data().

See also exif_thumbnail().

XLIII. Funciones IMAP

Para hacer funcionar estas funciones, debe compilar PHP con `--with-imap`. Esto requiere que la libreria c-client esté instalada. Obtenga la última versión de ftp://ftp.cac.washington.edu/imap/ y compílela. Luego copie `c-client/c-client.a` al directorio `/usr/local/lib` o a cualquier otro directorio de su LINK path y copie `c-client/rfc822.h`, `mail.h` y `linkage.h` al directorio `/usr/local/include` o a cualquier otro de su INCLUDE path.

Decir que estas funciones no están limitadas al protocolo IMAP, a pesar de sus nombres. La librería subyacente c-client también soporta NNTP, POP3 y métodos de acceso local a buzones de correo. Vea imap_open() para una mayor información.

imap_8bit (PHP 3, PHP 4)

Convierte una cadena de 8bit a una cadena quoted-printable

string **imap_8bit** (string string) \linebreak

Convierte una cadena de 8bit a una cadena quoted-printable.

Devuelve una cadena quoted-printable

imap_alerts (PHP 3>= 3.0.12, PHP 4)

Esta función devuelve todos los mensajes de alerta IMAP (si hubo) que han ocurrido durante la petición de la pagina o desde que la pila de alertas fue inicializada.

array **imap_alerts** (void) \linebreak

Esta función devuelve un array con todos los mensajes de alerta IMAP generados desde la última llamada a **imap_alerts()**, o el comienzo de la pagina. Cuando se llama a **imap_alerts()**, la pila de alerta es inicializada. La especificación IMAP requiere que estos mensajes sean pasados al usuario.

imap_append (PHP 3, PHP 4)

Agrega una cadena de mensaje al buzón especificado

int **imap_append** (int imap_stream, string mbox, string message, string flags) \linebreak

Devuelve `TRUE` si no hay error y `FALSE` en caso contrario.

imap_append() agrega una cadena de mensaje al buzón especificado *`mbox`*. Si se especifica el parámetro *`flags`*, escribe las opciones o condiciones establecidas en el parámetro *`flags`* al buzón.

Cuando conecte con el servidor Cyrus IMAP, debe usar "\r\n" como finalizador de linea en vez de "\n" o la operación fallará.

imap_base64 (PHP 3, PHP 4)

Decodifica texto codificado en BASE64

string **imap_base64** (string text) \linebreak

imap_base64() decodifica texto codificado en BASE-64. El mensaje decodificado es devuelto como una cadena.

imap_binary (PHP 3>= 3.0.2, PHP 4)

Convierte una cadena de 8bit a una cadena base64

string **imap_binary** (string string) \linebreak

Convierte una cadena de 8bit a una cadena base64.

Devuleve una cadena base64.

imap_body (PHP 3, PHP 4)

Lee el cuerpo del mensaje

string **imap_body** (int imap_stream, int msg_number, int flags) \linebreak

imap_body() devuelve el cuerpo del mensaje, numerado `msg_number` del buzón actual. Los `flags` opcionales son una máscara de bit con una o mas de las siguientes:

- FT_UID - El msgno es un UID
- FT_PEEK - No activar \Seen flag si no está ya activa
- FT_INTERNAL - La cadena devuelta está en formato interno, no canoniza a CRLF.

imap_bodystruct (PHP 3>= 3.0.4, PHP 4)

Read the structure of a specified body section of a specific message

object **imap_bodystruct** (int stream_id, int msg_no, int section) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imap_check (PHP 3, PHP 4)

Comprueba el estado del buzón actual

object **imap_check** (int imap_stream) \linebreak

Devuelve información acerca del buzón actual. Devuelve `FALSE` si falla.

La función **imap_check()** comprueba el estado del buzón actual en el servidor y devuelve la información en un objeto con las siguientes propiedades.

Date : fecha del mensaje
Driver : controlador
Mailbox : nombre del buzón
Nmsgs : número de mensajes
Recent : número de mensajes recientes

imap_clearflag_full (PHP 3>= 3.0.3, PHP 4)

Limpia los flags de los mensajes

string **imap_clearflag_full** (int stream, string sequence, string flag, string options) \linebreak

Esta funcion elimina el flag especificado del conjunto de flags activos para los mensajes en la secuencia especificada.

Las opciones son una máscara de bit con uno o más de los siguientes:

ST_UID El argumento sequence contiene UIDs en vez de números secuenciales

imap_close (PHP 3, PHP 4)

Cierra una sesión IMAP

int **imap_close** (int imap_stream, int flags) \linebreak

Cierra una sesión imap. Toma un parámetro *`flag`* opcional, CL_EXPUNGE, el cual purgará el buzón de forma trasparente antes de cerrarla.

imap_createmailbox (PHP 3, PHP 4)

Crea un buzón nuevo

int **imap_createmailbox** (int imap_stream, string mbox) \linebreak

imap_createmailbox() crea un buzón nuevo especificado por *mbox* (ver imap_open() para el formato del parámetro *mbox*).

Devuelve TRUE si no hay error y FALSE en caso contrario.

Ver También imap_renamemailbox() y imap_deletemailbox().

imap_delete (PHP 3, PHP 4)

Marca un mensaje para ser borrado en el buzón actual

int **imap_delete** (int imap_stream, int msg_number) \linebreak

Devuelve TRUE.

La función **imap_delete()** marca el mensaje referenciado por *msg_number* para su eliminación. El borrado físico de los mensajes es realizado por imap_expunge().

imap_deletemailbox (PHP 3, PHP 4)

Elimina un buzón

int **imap_deletemailbox** (int imap_stream, string mbox) \linebreak

imap_deletemailbox() elimina el buzón especificado (ver imap_open() para el formato del *mbox*).

Devuelve TRUE si no hay error y FALSE en caso contrario.

Ver También imap_createmailbox() y **imap_reanmemailbox()**.

imap_errors (PHP 3>= 3.0.12, PHP 4)

Esta función devuelve todos los errores IMAP (si hubo) que han occurrido durante la petición de la página o desde que la pila de errores se inicializó.

array **imap_errors** (void) \linebreak

Esta función devuelve un array de todos los mensajes de error IMAP generados desde la última llamada a **imap_errors()**, o el principio de la página. Cuando se llama a **imap_errors()**, la pila de errores se inicializa.

ATENCIàN: esta función no esta disponible aún en PHP4.

imap_expunge (PHP 3, PHP 4)

Elimina todos los mensajes marcados como borrados

int **imap_expunge** (int imap_stream) \linebreak

imap_expunge() elimina todos los mensajes marcados por la función imap_delete().

Devuleve `TRUE`.

imap_fetch_overview (PHP 3>= 3.0.4, PHP 4)

Read an overview of the information in the headers of the given message

array **imap_fetch_overview** (int imap_stream, string sequence [, int flags]) \linebreak

This function fetches mail headers for the given *`sequence`* and returns an overview of their contents. *`sequence`* will contain a sequence of message indices or UIDs, if *`flags`* contains FT_UID. The returned value is an array of objects describing one message header each:

- subject - the messages subject
- from - who sent it
- date - when was it sent
- message_id - Message-ID
- references - is a reference to this message id
- size - size in bytes
- uid - UID the message has in the mailbox
- msgno - message sequence number in the maibox
- recent - this message is flagged as recent
- flagged - this message is flagged
- answered - this message is flagged as answered
- deleted - this message is flagged for deletion
- seen - this message is flagged as already read
- draft - this message is flagged as being a draft

Ejemplo 1. imap_fetch_overview() example

```
$mbox = imap_open("{your.imap.host:143}","username","password")
     or die("can't connect: ".imap_last_error());

$overview = imap_fetch_overview($mbox,"2,4:6",0);
```

```
if(is_array($overview)) {
        reset($overview);
        while( list($key,$val) = each($overview)) {
                print     $val->msgno
                . " - " . $val->date
                . " - " . $val->subject
                . "\n";
        }
}

imap_close($mbox);
```

imap_fetchbody (PHP 3, PHP 4)

Localiza una sección particular en el cuerpo del mensaje

string **imap_fetchbody** (int imap_stream, int msg_number, string part_number, flags flags) \linebreak

Esta función busca una sección particular en el cuerpo de los mensajes especificados, como una cadena de texto y devuleve esa cadena. La especificación de la sección es una cadena de enteros delimitados por comas,los cuales indexan las partes del cuerpo como indica la especificación IMAP4. Partes del cuerpo no son decodificadas por esta función.

Las opciones para **imap_fetchbody ()** son una máscara de bit con una o más de las siguientes

- FT_UID - El msgono es un UID
- FT_PEEK - No activar \Seen flag si no esta ya activa
- FT_INTERNAL - La cadena devuelta está en formato "interno", sin ningún intento por canonizar CRLF

imap_fetchheader (PHP 3>= 3.0.3, PHP 4)

Devuelve la cabecera del mensaje

string **imap_fetchheader** (int imap_stream, int msgno, int flags) \linebreak

Esta función localiza el formato de la cabecera RFC 822 del mensaje especificado como una cadena de texto y devuelve esa cadena de texto.

The options are:

FT_UID El argumento msgno es un UID
FT_INTERNAL La cadena devuelta esta en formato "interno",
sin ningún intento de canonizar CRLF

FT_PREFETCHTEXT The RFC822.TEXT should be pre-fetched at the
same time. Esto evita un extra RTT en una
conexión IMAP si se desea un mensaje completo de
texto (e.g. en una operación de
"guardar a un fichero local")

imap_fetchstructure (PHP 3, PHP 4)

Lee la estructuta de un mensaje concreto

object **imap_fetchstructure** (int imap_stream, int msg_number [, int flags]) \linebreak

Esta función busca toda la información estructurada en el mensaje especificado. El parámetro opcional *`flags`* sólo tiene una opcion, *`FT_UID`*, la cual indica a la función que trate el argumento *`msg_number`* como un *`UID`*. El objeto devuelto incluye el sobre, la fecha interna, el tama¤o, flags y la estructura del cuerpo con un objeto similar por cada mime adjunto al mensaje. La estructura de los objetos devueltos es como sigue:

Tabla 1. Objetos Devueltos para imap_fetchstructure()

type	Tipo primario del cuerpo
encoding	Body transfer encoding
ifsubtype	`TRUE` si hay una cadena de subtipo
subtype	MIME subtype
ifDescripción	`TRUE` si hay una cadena de Descripción
Description	Conenido de la cadena de Descripción
ifid	`TRUE` si hay una cadena de identificación
id	Cadena de Identificación
lines	Número de lineas
bytes	Número de bytes
ifdisposition	`TRUE` si hay una cadena de configuración
disposition	Cadena de configuración
ifdparameters	`TRUE` si el array dparameters existe
dparameters a	Array de parametro de configuración
ifparameters	`TRUE` si el array de parámetros existe
parameters b	MIME parameters array

parts [c]	Array de objetos describiendo cada parte del mensaje
Notas: a. dparameters es un array de objetos donde cada objeto tiene un "atributo" y una propiedad "valor". b. parameter	

Tabla 2. Tipo primario del cuerpo

0	texto
1	multiparte
2	mensaje
3	aplicación
4	audio
5	imagen
6	video
7	otro

Tabla 3. Codificacion para tranferencia

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTRO

imap_get_quota (PHP 4 >= 4.0.5)

Retrieve the quota level settings, and usage statics per mailbox

array **imap_get_quota** (int imap_stream, string quota_root) \linebreak

Returns an array with integer values limit and usage for the given mailbox. The value of limit represents the total amount of space allowed for this mailbox. The usage value represents the mailboxes current level of capacity. Will return `FALSE` in the case of failure.

This function is currently only available to users of the c-client2000 library.

imap_stream should be the value returned from an imap_status() call. This stream is required to be opened as the mail admin user for the quota function to work. *quota_root* should normally be in the form of user.name where name is the mailbox you wish to retrieve information about.

Ejemplo 1. imap_get_quota() example

```
$mbox = imap_open("{your.imap.host}","mailadmin","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$quota_value = imap_get_quota($mbox, "user.kalowsky");
if(is_array($quota_value)) {
    print "Usage level is: " . $quota_value['usage'];
    print "Limit level is: " . $quota_value['limit'];
}

imap_close($mbox);
```

See also imap_open(), imap_set_quota().

imap_getmailboxes (PHP 3>= 3.0.12, PHP 4)

Lee la lista de buzones, devolviendo informacón detallada de cada uno

array **imap_getmailboxes** (int imap_stream, string ref, string pat) \linebreak

Devuelve un array de objetos coneniendo información del buzón. Cada objeto tiene los atributos *name*, especificando el nombre completo del buzón; *delimiter*, que es el delimitador jerárquico para la parte de la jerarquía dónde está este buzón; y *attributes*. *Attributes* es una máscara de bits contra la que se puede probar:

- LATT_NOINFERIORS - Este buzón no tiene "hijos" (No ha buzones por debajo de él)
- LATT_NOSELECT - Esto es sólo un contenedor, no un buzón - No puede abrirlo.
- LATT_MARKED - Este buzón está marcado. Unicamente usado por UW-IMAPD.
- LATT_UNMARKED - Este buzón no está marcado. Unicamente usado por UW-IMAPD.

ref normalmente debería ser solo el servidor IMAP, de la forma: {imap_server:imap_port}, y *pattern* especifica, dónde en la estructura jerárquica del buzón, para comenzar a buscar. Si quiere todo los buzones, pase el parámetro *pattern* como una cadena vacía.

Hay dos caracteres especiales que puede pasar como parte del parámetro *pattern*: '*' and '%'. '*' significa que devuelva todos los buzones. Si pasa el parámetro *pattern* como '*', obtendrá una lista con la jerarquía completa del buzón. %' significa que devuelva sólo el nivel actual. Pasar '%' en el

parámetro *pattern* devolverá sólo el nivel más alto de los buzones; '~/mail/%' en UW_IMAPD devolverá cada buzón del directorio ~/mail, pero ninguno de los subdirectorios de ese directorio.

imap_getsubscribed (PHP 3>= 3.0.12, PHP 4)

Lista todos los buzones subscritos

array **imap_getsubscribed** (int imap_stream, string ref, string pattern) \linebreak

Esta función es idéntica a imap_getmailboxes(), excepto que esta sólo devuelve los buzones a los que está subscrito el usuario.

imap_header (PHP 3, PHP 4)

Lee la cabecera del mensaje

object **imap_header** (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]]) \linebreak

Esta función devuelve un objeto con varios elementos de la cabecera.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references

message flags:
Recent - 'R' si es reciente y ha sido leido,
'N' si es reciente y no ha sido leido,
' ' si no es reciente
Unseen - 'U' si no ha sido leido Y no es reciente,
' ' si ha sido leido O no y es reciente
Answered -'A' si ha sido contestado,
' ' si no ha sido contestado
Deleted - 'D' si ha sido borrado,
' ' si no ha sido borrado
Draft - 'X' if draft,
' ' if not draft
Flagged - 'F' si esta if flagged,
' ' if not flagged

OBSERVE que el comportamiento Recent/Unseen es un poco extra¤o. Si quiere conocer si un mensaje es Unseen, debe comprobarlo asi

Unseen == 'U' || Recent == 'N'

toaddress (la linea to: al completo, hasta 1024 caracteres)

to[] (devuelve un array de objetos a partir de la linea To, conteniendo:)
  personal
  adl
  mailbox
  host

fromaddress (la linea from: al completo, hasta 1024 caracteres)

from[] (devuelve un array de objetos a partir de la linea From, conteniendo:)
  personal
  adl
  mailbox
  host

ccaddress (la linea cc: al completo, hasta 1024 caracteres)
cc[] (devuelve un array de objetos a partir de la linea Cc:, conteniendo:)
  personal
  adl
  mailbox
  host

bccaddress (la linea bcc al completo, hasta 1024 caracteres)
bcc[] (devuelve un array de objetos a partir de la linea Bcc, conteniendo:)
  personal
  adl
  mailbox
  host

reply_toaddress (la linea reply_to: al completo, hasta 1024 caracteres)
reply_to[] (devuelve un array de objetos a partir de la linea Reply_to,
conteniendo:)
  personal
  adl
  mailbox
  host

senderaddress (la linea sender: al completo, hasta 1024 caracteres)
sender[] (devuelve un array de objetos a partir de la linea sender, conteniendo:)
  personal
  adl
  mailbox
  host

return_path (la linea return-path: al completo, hasta 1024 caracteres)
return_path[] (devuelve un array de objetos a partir de la linea return_path,
conteniendo:)

personal
adl
mailbox
host

udate (fecha del mensaje en formato unix)

fetchfrom (la linea from formateada hasta ajustarse a los caracteres
indicados en `fromlength`)

fetchsubject (la linea subject formateada hasta ajustarse a los caracteres
indicados en `subjectlength`)

imap_headerinfo (PHP 3, PHP 4)

Read the header of the message

object **imap_headerinfo** (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]]) \linebreak

This function returns an object of various header elements.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references

message flags:
Recent - 'R' if recent and seen,
'N' if recent and not seen,
' ' if not recent
Unseen - 'U' if not seen AND not recent,
' ' if seen OR not seen and recent
Answered -'A' if answered,
' ' if unanswered
Deleted - 'D' if deleted,
' ' if not deleted
Draft - 'X' if draft,
' ' if not draft
Flagged - 'F' if flagged,
' ' if not flagged

NOTE that the Recent/Unseen behavior is a little odd. If you want to
know if a message is Unseen, you must check for

Unseen == ’U’ || Recent == ’N’

toaddress (full to: line, up to 1024 characters)

to[] (returns an array of objects from the To line, containing):
  personal
  adl
  mailbox
  host

fromaddress (full from: line, up to 1024 characters)

from[] (returns an array of objects from the From line, containing):
  personal
  adl
  mailbox
  host

ccaddress (full cc: line, up to 1024 characters)
cc[] (returns an array of objects from the Cc line, containing):
  personal
  adl
  mailbox
  host

bccaddress (full bcc line, up to 1024 characters)
bcc[] (returns an array of objects from the Bcc line, containing):
  personal
  adl
  mailbox
  host

reply_toaddress (full reply_to: line, up to 1024 characters)
reply_to[] (returns an array of objects from the Reply_to line,
containing):
  personal
  adl
  mailbox
  host

senderaddress (full sender: line, up to 1024 characters)
sender[] (returns an array of objects from the sender line, containing):
  personal
  adl
  mailbox
  host

return_path (full return-path: line, up to 1024 characters)
return_path[] (returns an array of objects from the return_path line,
containing):
  personal
  adl
  mailbox
  host

udate (mail message date in unix time)

fetchfrom (from line formatted to fit *`fromlength`*
characters)

fetchsubject (subject line formatted to fit *`subjectlength`* characters)

imap_headers (PHP 3, PHP 4)

Returns headers for all messages in a mailbox

array **imap_headers** (int imap_stream) \linebreak

Devuelve un array de cadenas formateadas con informacion de la cabecera. Un elemento por mensaje de correo.

imap_last_error (PHP 3>= 3.0.12, PHP 4)

Esta función devuelve el último error IMAP (si se produjo) que ocurrió durante la petición de esta página.

string **imap_last_error** (void) \linebreak

Esta función devuelve el texto completo del último error IMAP que ocurrió en la pagina actual. La plia de errores The error stack is untouched; llamando despues a la función **imap_last_error()**, sin que se produzca un error, devolverá el mismo error.

ATENCIàN: esta función no esta disponible aún en PHP4.

imap_listmailbox (PHP 3, PHP 4)

Lee la lista de buzones

array **imap_listmailbox** (int imap_stream, string ref, string pat) \linebreak

Devuelve un array que contiene los nombres de los buzones.

imap_listsubscribed (PHP 3, PHP 4)

Lista todos los buzones subscritos

array **imap_listsubscribed** (int imap_stream, string ref, string pattern) \linebreak

Devuelve un array de todos los buzones que usted tiene subscritos. Los parámetros *ref* y *pattern* especifican la localización desde donde comenzará a buscar y el patrón que el nombre del buzón debe encontrar.

imap_mail_compose (PHP 3>= 3.0.5, PHP 4)

Create a MIME message based on given envelope and body sections

string **imap_mail_compose** (array envelope, array body) \linebreak

Ejemplo 1. imap_mail_compose() example

```
<?php

$envelope["from"]="musone@afterfive.com";
$envelope["to"]="musone@darkstar";
$envelope["cc"]="musone@edgeglobal.com";

$part1["type"]=TYPEMULTIPART;
$part1["subtype"]="mixed";

$filename="/tmp/imap.c.gz";
$fp=fopen($filename,"r");
$contents=fread($fp,filesize($filename));
fclose($fp);

$part2["type"]=TYPEAPPLICATION;
$part2["encoding"]=ENCBINARY;
$part2["subtype"]="octet-stream";
$part2["description"]=basename($filename);
$part2["contents.data"]=$contents;

$part3["type"]=TYPETEXT;
$part3["subtype"]="plain";
$part3["description"]="description3";
$part3["contents.data"]="contents.data3\n\n\n\t";
```

```
$body[1]=$part1;
$body[2]=$part2;
$body[3]=$part3;

echo nl2br(imap_mail_compose($envelope,$body));

?>
```

imap_mail_copy (PHP 3, PHP 4)

Copia los mensajes especificados a un buzón

int **imap_mail_copy** (int imap_stream, string msglist, string mbox, int flags) \linebreak

Devuelve `TRUE` si no hay error y `FALSE` en caso contrario.

Copia los mensajes especificados por *`msglist`* a un buzón especificado. *`msglist`* es un rango no números de mensajes.

Flags es una máscara de bit de uno o más

- CP_UID - los números de secuencia contienen UIDS
- CP_MOVE - Elimina los mensajes del buzon actual despues de copiarlos

imap_mail_move (PHP 3, PHP 4)

Mueve los mensajes especificados a un buzón

int **imap_mail_move** (int imap_stream, string msglist, string mbox) \linebreak

Mueve los mensajes especificados por *`msglist`* al buzón especicifado. *`msglist`* es un rango no números de mensajes.

Devuelve `TRUE` si no hay error y `FALSE` en caso contrario.

imap_mail (PHP 3>= 3.0.14, PHP 4)

Send an email message

string **imap_mail** (string to, string subject, string message [, string additional_headers [, string cc [, string bcc [, string rpath]]]]) \linebreak

This function allows sending of emails with correct handling of Cc and Bcc receivers. The parameters to, cc and bcc are all strings and are all parsed as rfc822 address lists. The receivers specified in bcc will get the mail, but are excluded from the headers. Use the rpath parameter to specify return path. This is useful when using php as a mail client for multiple users.

imap_mailboxmsginfo (PHP 3>= 3.0.2, PHP 4)

Obtiene información acerca del buzón actual

object **imap_mailboxmsginfo** (int imap_stream) \linebreak

Devuelve información acerca del buzón actual. Devuelve `FALSE` en caso de fallo.

La función **imap_mailboxmsginfo()** comprueba el estado del buzón actual en el servidor y devuelve la información en un objeto con las siguientes propiedades.

Date : fecha del mensaje
Driver : driver
Mailbox : nombre del buzón
Nmsgs : número de mensajes
Recent : número de los mensajes recientes
Unread : número de los mensajes no leidos
Size : tama¤o del buzon

imap_mime_header_decode (PHP 3>= 3.0.17, PHP 4)

Decode MIME header elements

array **imap_mime_header_decode** (string text) \linebreak

imap_mime_header_decode() function decodes MIME message header extensions that are non ASCII text (see RFC2047 (http://www.faqs.org/rfcs/rfc2047.html)) The decoded elements are returned in an array of objects, where each object has two properties, "charset" & "text". If the element hasn’t been encoded, and in other words is in plain US-ASCII,the "charset" property of that element is set to "default".

Ejemplo 1. imap_mime_header_decode() example

```
$text="=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@dkuug.dk>";
```

```
$elements=imap_mime_header_decode($text);
for($i=0;$i<count($elements);$i++) {
    echo "Charset: {$elements[$i]->charset}\n";
    echo "Text: {$elements[$i]->text}\n\n";
}
```

In the above example we would have two elements, whereas the first element had previously been encoded with ISO-8859-1, and the second element would be plain US-ASCII.

imap_msgno (PHP 3>= 3.0.3, PHP 4)

Esta función devuelve el número de secuencia del mensaje para el UID dado.

int **imap_msgno** (int imap_stream, int uid) \linebreak

Esta función devuelve el número de secuencia del mensaje para el UID dado. Esta función es la inversa a imap_uid().

imap_num_msg (PHP 3, PHP 4)

Informa del número de mensajes en el buzón actual

int **imap_num_msg** (int imap_stream) \linebreak

Devuelve el número de mensajes en el buzón actual.

imap_num_recent (PHP 3, PHP 4)

Informa el número de mensajes recientes en el buzón actual

int **imap_num_recent** (int imap_stream) \linebreak

Devuleve el número de mensajes recientes en el buzón actual.

imap_open (PHP 3, PHP 4)

Abre una sesión IMAP

int **imap_open** (string mailbox, string username, string password, int flags) \linebreak

Devuleve la sesion IMAP si no hay error y `FALSE` en caso contrario. Esta función también puede ser usada para abrir sesiones con servidores POP3 y NNTP. Para conectarse a un servidor IMAP escuchando por el puerto 143 en una máquina local, haga lo siguiente:

```
$mbox = imap_open("{localhost:143}INBOX","user_id","password");
```

Para conectarse a un servidor POP3 escuchando por el puerto 110, use:

```
$mbox = imap_open("{localhost/pop3:110}INBOX","user_id","password");
```

Para conectarse a un servidor NNTP escuchando por el puerto 119, use:

```
$nntp = imap_open("{localhost/nntp:119}comp.test","","");
```

Para conectarse a un servidor remoto sustituya "localhost", por el nombre o dirección IP del servidor al cual quiere conectarse.

Las opciones son una máscara de bit con una o más de los siguientes:

- OP_READONLY - Abre el buzón en modo de sólo lectura
- OP_ANONYMOUS - No usa o actualiza un .newsrc para las noticias
- OP_HALFOPEN - Para nombres IMAP y NNTP, abre una conexión pero no abre un buzón
- CL_EXPUNGE - Purga automaticamente el buzón antes de cerrar la sesión

imap_ping (PHP 3, PHP 4)

Comprueba si la sesión IMAP está aún activa

int **imap_ping** (int imap_stream) \linebreak

Devuelve `TRUE` si la sesión está activa, `FALSE` en caso contrario.

La función **imap_ping()** pings the stream to see it is still active. Esto puede descubrir que hay correo nuevo; este es el método preferido para hacer una comprobación periodica del buzón, asi como para mantener activa sesiones en servidores que tienen inactivity timeout.

imap_popen (PHP 3>= 3.0.12, PHP 4)

Open a persistant IMAP stream to a mailbox

int **imap_popen** (string mailbox, string user, string password [, int options]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imap_qprint (PHP 3, PHP 4)

Convierte una cadena quoted-printable a una cadena de 8 bit

string **imap_qprint** (string string) \linebreak

Convierte una cadena quoted-printable a una cadena de 8 bit

Devuelve una cadena de 8 bit (binary)

imap_renamemailbox (PHP 3, PHP 4)

Renombra un buzón

int **imap_renamemailbox** (int imap_stream, string old_mbox, string new_mbox) \linebreak

Esta función renombra un buzon (ver imap_open() para el formato del parámetro *mbox*).

Devuelve `TRUE` si no hay error y `FALSE` en caso contrario.

Ver También imap_createmailbox() and imap_deletemailbox().

imap_reopen (PHP 3, PHP 4)

Reabre una sesión IMAP a un nuevo buzón

int **imap_reopen** (string imap_stream, string mailbox [, string flags]) \linebreak

Devuelve `TRUE` si no hay error y `FALSE` en caso contrario.

Esta funció reabre la sesión especificada con un nuevo buzón.

Las opciones son máscaras de bit con una o más de las siguientes:

- OP_READONLY - Abre el buzón en modo de sólo lectura
- OP_ANONYMOUS - No usa o actualiza .newsrc para noticias

- OP_HALFOPEN - Para nombres IMAP y NNTP, abre una conexión pero no abre el buzon.
- CL_EXPUNGE - Expurga automaticamente el buzon antes de cerra la sesión

imap_rfc822_parse_adrlist (PHP 3>= 3.0.2, PHP 4)

Examina la cadena dirección

string **imap_rfc822_parse_adrlist** (string address, string default_host) \linebreak

Esta función examina la cadena dirección y para cada dirección, devuelve un array de objetos. Los 4 objetos son:

mailbox - el nombre del buzón (username)
host - el nombre del ordenador
personal - el nombre personal
adl - ruta del dominio

imap_rfc822_parse_headers (PHP 4)

Parse mail headers from a string

object **imap_rfc822_parse_headers** (string headers [, string defaulthost]) \linebreak

This function returns an object of various header elements, similar to imap_header(), except without the flags and other elements that come from the IMAP server.

imap_rfc822_write_address (PHP 3>= 3.0.2, PHP 4)

Devuleve una dirección de correo correctamente formateada dado el buzón, host, e información personal.

string **imap_rfc822_write_address** (string mailbox, string host, string personal) \linebreak

Devuelve una dirección de correo correctamente formateada, dado el buzón, host, e información personal.

imap_scanmailbox (PHP 3, PHP 4)

Lee la lista de buzones y toma una cadena para buscar en el texto del buzón

array **imap_scanmailbox** (int imap_stream, string string) \linebreak

Devuelve un array que contiene los nombres de los buzones que tienen el parámetro *`string`* en el texto del buzón.

imap_search (PHP 3>= 3.0.12, PHP 4)

Esta función devuelve un array de mensajes que coinciden con el criterio de busqueda dado.

array **imap_search** (int imap_stream, string criteria, int flags) \linebreak

Esta función realiza una busqueda en el buzón actualmente abierto indicado por imap_stream. *`criteria`* es una cadena, delimitada por espacios, en la cual las siguientes palabras claves son permitidas. Cualquier argumento múltiple (ej. FROM "joey smith") debe estar entre comillas.

- ALL - devuelve todos los mensajes que coinciden con el resto del criterio
- ANSWERED - busca mensajes con el flag \\ANSWERED activado
- BCC "string" - busca mensajes con "cadena" en el campo Bcc:
- BEFORE "date" - busca mensajes con Date: antes de "date"
- BODY "string" - busca mensajes con "cadena" en el cuerpo del mensaje
- CC "string" - busca mensajes con "cadena" en el campo Cc:
- DELETED - busca mensajes eliminados
- FLAGGED - busca mensajes con el flag \\FLAGGED (sometimes referred to as Important or Urgent) activado
- FROM "string" - busca mensajes con "cadena" en el campo From:
- KEYWORD "string" - busca mensajes con "cadena" como una palabra clave
- NEW - busca mensajes nuevos
- OLD - busca mensajes viejos
- ON "date" - busca mensajes con "date" igual a Date:
- RECENT - busca mensajes con el flag \\RECENT activado
- SEEN - busca mensajes que han sido leidos (la opcion \\SEEN activada)
- SINCE "date" - busca mensajes conwith Date: after "date"
- SUBJECT "string" - busca mensajes con "string" en el campo Subject:
- TEXT "string" - busca mensajes con el texto "string"
- TO "string" - busca mensajes con "string" en el campo To:
- UNANSWERED - busca mensajes que no han sido respondidos

- UNDELETED - busca mensajes que no han sido eliminados
- UNFLAGGED - busca mensajes que no estan flagged
- UNKEYWORD "string" - busca mensajes que no coinciden con la palabra clave "string"
- UNSEEN - busca mensajes que no han sido leidos aún

Por ejemplo, para buscar todos los mensajes no contestados enviados por Mamá, usaría: "UNANSWERED FROM mamá". La busqueda parece ser no sensitiva. Esta lista de criterios está tomada del codigo fuente del UW c-client y puede que este incompleta o sea inexacta.

Valores validos para los flags son SE_UID, que provoca que el array devuelto contenga UIDs en vez de los numeros de secuencia de los mensajes

imap_set_quota (PHP 4 >= 4.0.5)

Sets a quota for a given mailbox

int **imap_set_quota** (int imap_stream, string quota_root, int quota_limit) \linebreak

Sets an upper limit quota on a per mailbox basis. This function requires the *imap_stream* to have been opened as the mail administrator account. It will not work if opened as any other user.

This function is currently only available to users of the c-client2000 library.

imap_stream is the stream pointer returned from a imap_open() call. This stream must be opened as the mail administrator, other wise this function will fail. *quota_root* is the mailbox to have a quota set. This should follow the IMAP standard format for a mailbox, 'user.name'. *quota_limit* is the maximum size (in KB) for the *quota_root*.

Returns TRUE on success and FALSE on error.

Ejemplo 1. imap_set_quota() example

```
$mbox = imap_open ("{your.imap.host:143}", "mailadmin", "password");

if(!imap_set_quota($mbox, "user.kalowsky", 3000)) {
    print "Error in setting quota\n";
    return;
}

imap_close($mbox);
```

See also imap_open(), **imap_set_quota()**.

imap_setacl (PHP 4 >= 4.1.0)

Sets the ACL for a giving mailbox

int **imap_setacl** (int stream_id, string mailbox, string id, string rights) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imap_setflag_full (PHP 3>= 3.0.3, PHP 4)

Activa flags en los mensajes

string **imap_setflag_full** (int stream, string sequence, string flag, string options) \linebreak

Esta función a¤ade el flag especificado al conjunto de flags activos para los mensajes en la secuencia especificada.

Los flags que puede seleccionar son "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", y "\\Recent" (definidos en el RFC2060)

Las opciones son una máscara de bit con uno o más de los siguientes:

ST_UID El argumento sequence contiene UIDs en vez de números secuenciales

imap_sort (PHP 3>= 3.0.3, PHP 4)

Ordena un array de cabeceras de mensajes

string **imap_sort** (int stream, int criteria, int reverse, int options) \linebreak

Devuelve un array de números de mensajes ordenados por los parametros dados

Rev es 1 para una ordenación inversa.

Criteria puede ser uno (y sólo uno) de los siguientes:

SORTDATE Fecha del mensaje
SORTARRIVAL Fecha de llegada

SORTFROM mailbox in first From address
SORTSUBJECT Asunto del mensaje
SORTTO mailbox in first To address
SORTCC mailbox in first cc address
SORTSIZE tama¤o del mensaje en bytes

Las opciones son una máscara de bit con uno o más de los siguientes:

SE_UID Devuelve UIDs en vez de números secuenciales
SE_NOPREFETCH No preselecciona los mensajes buscados.

imap_status (PHP 3>= 3.0.4, PHP 4)

Esta función devuelve el información de estado de otro buzón distinto al actual.

object **imap_status** (int imap_stream, string mailbox, int options) \linebreak

Esta función devuelve un objeto que contiene información de estado. Las opciones válidas son:

- SA_MESSAGES - activa status->messages con el número de mensajes en el buzon
- SA_RECENT - activa status->recent con el número de mensajes recientes en el buzón
- SA_UNSEEN - activa status->unseen con el número de mesajes no leidos (nuevos) en el buzón
- SA_UIDNEXT - activa status->uidnext con el próximo uid a usar en el bizón
- SA_UIDVALIDITY - activa status->uidvalidity con una constante que cambia cuando los uids del buzon ya no son válidos
- SA_ALL - activa todos los de arriba

status->flags contienen una máscara de bits la cual puede ser comprobada contra cualquiera de las propiedades de arriba.

imap_subscribe (PHP 3, PHP 4)

Subscribe to a mailbox

int **imap_subscribe** (int imap_stream, string mbox) \linebreak

Da de alta un nuevo buzón.

Devuleve `TRUE` si no hay error y `FALSE` en caso contrario.

imap_thread (PHP 4 >= 4.1.0)

Return threaded by REFERENCES tree

int **imap_thread** (int stream_id [, int flags]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

imap_uid (PHP 3>= 3.0.3, PHP 4)

Esta función devuelve el UID del número de secuencia del mensaje dado

int **imap_uid** (int imap_stream, int msgno) \linebreak

Esta función devuelve el UID del número de secuencia del mensaje dado. Esta función es la inversa a imap_msgno().

imap_undelete (PHP 3, PHP 4)

Desmarca los mensajes que están marcados como borrados

int **imap_undelete** (int imap_stream, int msg_number) \linebreak

Esta funcion elimina la marca de borrado de un mensaje especifico, puesta por la función imap_delete().

Devuleve `TRUE` si no hay error y `FALSE` en caso contrario.

imap_unsubscribe (PHP 3, PHP 4)

Unsubscribe from a mailbox

int **imap_unsubscribe** (int imap_stream, string mbox) \linebreak

Da de baja el buzón especificado.

Devuelve `TRUE` si no hay error y `FALSE` en caso contrario.

imap_utf7_decode (PHP 3>= 3.0.15, PHP 4)

Decodes a modified UTF-7 encoded string.

string **imap_utf7_decode** (string text) \linebreak

Decodes modified UTF-7 *`text`* into 8bit data.

Returns the decoded 8bit data, or `FALSE` if the input string was not valid modified UTF-7. This function is needed to decode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (original UTF-7 was defned in RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

imap_utf7_encode (PHP 3>= 3.0.15, PHP 4)

Converts 8bit data to modified UTF-7 text.

string **imap_utf7_encode** (string data) \linebreak

Converts 8bit *`data`* to modified UTF-7 text. This is needed to encode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (original UTF-7 was defned in RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

Returns the modified UTF-7 text.

imap_utf8 (PHP 3>= 3.0.13, PHP 4)

Converts text to UTF8

string **imap_utf8** (string text) \linebreak

Converts the given *`text`* to UTF8 (as defined in RFC2044 (http://www.faqs.org/rfcs/rfc2044.html)).

XLIV. Funciones para Informix

El conector para Informix Online (ODS) 7.x, SE 7.x y Universal Server (IUS) 9.x se encuentra implementado en "functions/ifx.ec" y "functions/php3_ifx.h". Para ODS 7.x está completado, con total soporte para columnas de tipo BYTE y TEXT. Para IUS 9.x está parcialmente finalizado: los tipos de datos nuevos están allí (en el IUS 9.x), pero la funcionalidad para SLOB y CLOB se encuentra bajo construcción todavía.

Notas de configuración:

Antes de ejecutar el guión (script) "configure", asegúrate que la variable "INFORMIXDIR" ha sido definida.

Si ejecutas "configure --with_informix=yes" entonces el guión de configuración detectará automáticamente las librerías y los directorios include. Puedes obviar esta detección definiendo las variables de entorno "IFX_LIBDIR", "IFX_LIBS" y "IFX_INCDIR". Definirás la variable de compilación condicional "HAVE_IFX_IUS" si la versión de Informix es 9.00 o superior.

Algunas notas sobre el uso de BLOBs (columnas de tipo TEXT y BYTE):

BLOBs son normalmente manipulados por enteros, los cuales representan identificadores de BLOB. Las consultas de selección devuelven un "blob id" para columnas de tipo BYTE y TEXT. Si eliges trabajar con los BLOBs en memoria (con: "ifx_blobinfile(0);") entonces puedes obtener el contenido con "string_var = ifx_get_blob($blob_id);". Si prefieres manipularlos en ficheros usa "ifx_blobinfile(1);" y "ifx_get_blob($blob_id);" devolverá el nombre del archivo. En este caso, utiliza las funciones habituales de entrada y salida de ficheros para obtener el contenido de los blob.

Para consultas de inserción y actualización debes crear estos identificadores de blob con "ifx_create_blob(..);". Entonces pondrás los identificadores de blob en un array y sustituirás en la cadena de la consulta las columnas de tipo blob por una interrogación (?). Para inserciones y actualizaciones eres responsable de definir el contenido de los blob con ifx_update_blob(...).

La conducta de columnas BLOB puede ser modificada mediante variables de configuración, las cuales pueden ser definidas en tiempo de ejecución mediante funciones.

variable de configuración: ifx.textasvarchar

variable de configuración: ifx.byteasvarchar

funciones en tiempo de ejecución:

ifx_textasvarchar(0): usa identificadores de blob para columnas de tipo TEXT en las consultas de selección

ifx_byteasvarchar(0): usa identificadores de blob para columnas de tipo BYTE en las consultas de selección

ifx_textasvarchar(1): devuelve columnas de tipo TEXT como si fueran de tipo VARCHAR, sin tener que usar identificadores de blob en las consultas de selección

ifx_byteasvarchar(1): devuelve columnas de tipo BYTE como si fueran de tipo VARCHAR, sin tener

que usar identificadores de blob en las consultas de selección.

variable de configuración: ifx.blobinfile

función en tiempo de ejecución:

ifx_blobinfile_mode(0): devuelve columnas de tipo BYTE en memoria, el identificador de blob te permite obtener el contenido.

ifx_blobinfile_mode(1): devuelve columnas de tipo BYTE en un fichero, el identificador te permite saber el nombre de dicho archivo.

Si defines ifx_text/byteasvarchar a 1 entonces puedes usar columnas de tipo TEXT y BYTE en las consultas de selección como campos de tipo VARCHAR, pero teniendo en cuenta que tendrán un mayor tamaño que el habitual. Ya que en PHP todas las cadenas son posibles, esto permite datos binarios. De esta forma, se pueden manejar correctamente. La información devuelta puede contener cualquier cosa, tú eres responsable del contenido.

Si defines ifx_blobinfile a 1, utiliza el nombre del archivo devuelto por ifx_get_blob(..) para acceder a los contenidos del blobs. En este caso, ERES REPONSABLE DE ELIMINAR EL ARCHIVO TEMPORAL GENERADO POR INFORMIX cuando accedas a los registros. Cada nueva fila obtenida creará un nuevo archivo temporal para cada columna de tipo BYTE.

El directorio donde se guardan los archivos temporales puede ser definido por la variable de entorno blobdir, por defecto es ".", es decir, el directorio actual. Así, putenv(blobdir=tmpblob"); definirá un directorio donde se localizarán todos los ficheros temporales y facilitará su borrado. Todos los nombres de los archivos comienzan por "blb".

Recortado (trimming) automático de datos de tipo "char" (SQLCHAR y SQLNCHAR):

Puede ser definido con la variable de configuración

ifx.charasvarchar: si se define a 1 eliminará automáticamente los espacios en blanco al final de la cadena.

Valores `NULL`:

La variable de configuración ifx.nullformat (y en tiempo de ejecución ifx_nullformat()) cuando sea definida a `TRUE` devolverá columnas `NULL` como la cadena "`NULL`", si es definida a `FALSE` entonces la cadena vacía. Esto permite distinguir entre columnas `NULL` y vacías.

ifx_affected_rows (PHP 3>= 3.0.3, PHP 4)

Obtiene el número de registros procesados por una consulta

int **ifx_affected_rows** (int result_id) \linebreak

result_id es un identificador válido del resultado de ifx_query() o ifx_prepare().

Devuelve el número de filas procesadas por una consulta representada por un *result_id* (identificador de resultado).

Para inserciones, actualizaciones y borrados el número es exactamente los registros procesados (sqlerrd[2]). Para las consultas de selección es una estimación (sqlerrd[0]). No confíes en él.

Es útil llamarla después de ejecutar ifx_prepare() pues así podemos limitar las consultas a número razonable de registros.

Examina también: ifx_num_rows()

Ejemplo 1. Número de registros procesados por una consulta

```
$rid = ifx_prepare ("select * from emp where name like " . $name, $connid);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);  // Demasia-
dos registros en el resultado
    die ("Please restrict your query<br>\n");                      // Por favor, re-
stringe tu consulta
}
```

ifx_blobinfile_mode (PHP 3>= 3.0.4, PHP 4)

Define el modo por defecto para los blob en todas las consultas de selección

void **ifx_blobinfile_mode** (int mode) \linebreak

Define el modo por defecto para los blob en todas las consultas de selección. El modo (mode) "0" quiere decir que guarda en memoria los blobs de tipo BYTE y modo "1" significa guardarlos en un archivo.

ifx_byteasvarchar (PHP 3>= 3.0.4, PHP 4)

Define el modo por defecto para los campos de tipo byte

void **ifx_byteasvarchar** (int mode) \linebreak

Define el modo por defecto para los campos de tipo byte en todas las consultas de selección. Modo (mode) "0" devolverá un identificador de blob y "1" dará el contenido en un campo de tipo varchar.

ifx_close (PHP 3>= 3.0.3, PHP 4)

Cierra una conexión con Informix

int **ifx_close** ([int link_identifier]) \linebreak

Devuelve: `TRUE` siempre.

ifx_close() cierra un enlace a una base de datos Informix que esté asociado con el identificador de enlace (link_identifier). Si el identificador de enlace no es especificado, el último enlace abierto es asumido.

Observa que esto no es necesario habitualmente ya que las conexiones no permanentes son cerradas automáticamente al finalizar el guión (script).

ifx_close() no cerrará enlaces persistentes generados por ifx_pconnect().

Examina también: ifx_connect(), y ifx_pconnect().

Ejemplo 1. Cierre de una conexión a Informix

```
$conn_id = ifx_connect (mydb@ol_srv, "itsme", "mypassword");
... algunas consultas y código ...
ifx_close($conn_id);
```

ifx_connect (PHP 3>= 3.0.3, PHP 4)

Abre una conexión con un servidor Informix

int **ifx_connect** ([string database [, string userid [, string password]]]) \linebreak

Si tuvo éxito, devuelve un identificador de conexión en otro caso `FALSE`.

ifx_connect() establece una conexión con un servidor INFORMIX. Todos los argumentos son opcionales, y si no se pasan, se toman los valores del fichero de configuración (ifx.default_host para el ordenador donde se encuentra el servidor (si no es definida, las librerías de Infomix usarán la variable de entorno INFORMIXSERVER), ifx.default_user para el usuario (userid), ifx.default_password para la contraseña (password) (ninguna, si no es definida).

Para una segunda llamada a **ifx_connect()** con los mismos argumentos, no se establecerá una nueva conexión, en vez de eso, el identificador de enlace de la conexión abierta será devuelto.

La conexión con el servidor será cerrada tan pronto como la ejecución del guión (script) finalice, a menos que anteriormente se haya llamando a ifx_close().

Examina también ifx_pconnect(), y ifx_close().

Ejemplo 1. Conexión a una base de datos Informix

```
$conn_id = ifx_pconnect (mydb@ol_srv1, "imyself", "mypassword");
```

ifx_copy_blob (PHP 3>= 3.0.4, PHP 4)

Duplica el objeto blob dado

int **ifx_copy_blob** (int bid) \linebreak

Duplica el objeto blob dado. *bid* es el identificador del objeto blob a copiar.

Devuelve FALSE si hubo error, en otro caso el identificador del nuevo objeto blob.

ifx_create_blob (PHP 3>= 3.0.4, PHP 4)

Crea un objeto blob

int **ifx_create_blob** (int type, int mode, string param) \linebreak

Crea un objeto blob.

type (tipo): 1 = TEXT, 0 = BYTE

mode (modo): 0 = el contenido del objeto blob es conservado en memoria, 1 = el contenido del objeto blob es mantenido en un archivo.

param: si mode = 0: apunta al contenido en memoria, si mode = 1: contiene el nombre del fichero.

Devuelve FALSE si hubo error, en otro caso el identificador del nuevo objeto blob.

ifx_create_char (PHP 3>= 3.0.6, PHP 4)

Crea un objeto char

int **ifx_create_char** (string param) \linebreak

Crea un objeto char. *param* será el contenido del char.

ifx_do (PHP 3>= 3.0.4, PHP 4)

Ejecuta una sentencia SQL preparada previamente

int **ifx_do** (int result_id) \linebreak

Devuelve `TRUE` si se realizó, `FALSE` si hubo algún error.

Ejecuta una consulta preparada anteriormente o abre un cursor para ella.

No libera *`result_id`* si hubo un error.

También define el número real de registros procesados para consultas que no sean de selección y se puede obtener mediante ifx_affected_rows().

Examina también: ifx_prepare() (hay un ejemplo).

ifx_error (PHP 3>= 3.0.3, PHP 4)

Devuelve el código de error de la última llamada a Informix

string **ifx_error** (void) \linebreak

Los códigos de error de Informix (SQLSTATE & SQLCODE) son representados como se especifica a continuación:

x [SQLSTATE = aa bbb SQLCODE=cccc]

donde x = un espacio : no hubo error

E : hubo error

N : no hay más datos

W : aviso

? : no definido

Si el carácter "x" es cualquier otra cosa diferente a un espacio, SQLSTATE y SQLCODE describen el error con mayor detalle.

Examina el manual de Informix para el significado de SQLSTATE y SQLCODE.

Devuelve en una cadena un caracter describiendo el resultado de una sentencia y los valores SQLSTATE y SQLCODE asociados con la última sentencia SQL ejecutada. El formato de la cadena es "(char) [SQLSTATE=(dos dígitos) (tres dígitos) SQLCODE=(un dígitos)]". El primer carácter puede ser ' ' (un espacio) (no hubo error), '`W`' (la sentencia provocó un aviso), '`E`' (la consulta produjo un error) o '`N`' (la sentencia no devolvió ningún dato).

Examina también: ifx_errormsg()

ifx_errormsg (PHP 3>= 3.0.4, PHP 4)

Devuelve el mensaje de error de la última llamada a Informix

string **ifx_errormsg** ([int errorcode]) \linebreak

Devuelve el mensaje de error asociado con el error más reciente de Informix. Si definismos el parámetro opcional "*errorcode*" (código de error), nos dará el mensaje de error correspondiente a ese código.

Examina también: ifx_error()

```
printf("%s\n<br>", ifx_errormsg(-201));
```

ifx_fetch_row (PHP 3>= 3.0.3, PHP 4)

Obtiene registros como un array (vector) enumerado

array **ifx_fetch_row** (int result_id [, mixed position]) \linebreak

Devuelve un array (vector) correspondiente a la fila leída o `FALSE` si no hay más registros.

Las columnas blob son devueltas como identificadores de blob enteros (integer) para usarlos con ifx_get_blob() a menos que hayas usado ifx_textasvarchar(1) o ifx_byteasvarchar(1), en cuyo caso los blobs son devueltos como cadenas de texto. Devuelve `FALSE` si hubo error.

result_id es un identificador válido del resultado de ifx_query() o ifx_prepare() (sólo para consultas de selección).

position es un parámetro opcional para una operación de lectura sobre un cursor de tipo "scroll": "NEXT" (siguiente), "PREVIOUS" (anterior), "CURRENT" (actual), "FIRST" (primero), "LAST" (último) o un número. Si se especifica un número, un registro concreto es leído. Este parámetro opcional es sólo válido para cursores de tipo scroll.

ifx_fetch_row() lee el contenido de un registro de la consulta representada por el identificador de resultado indicado. La fila (registro) es devuelta en un array. Cada columna es guarda en un array, empezando éste desde cero.

Las llamadas posteriores a **ifx_fetch_row()** devolverán el registro siguiente en el resultado de la consulta, o `FALSE` si no hay más filas.

Ejemplo 1. Leer registros

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
```

```
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);  // Demasia-
dos registros en el resultado
    die ("Please restrict your query<br>\n");                      // Por favor, re-
stringe tu consulta
}
if (! ifx_do ($rid)) {
   ... error ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
}
ifx_free_result ($rid);
```

ifx_fieldproperties (PHP 3>= 3.0.3, PHP 4)

Indica las propiedades de los campos de una consulta SQL

array **ifx_fieldproperties** (int result_id) \linebreak

Dada una consulta representada por *result_id* devuelve un array con los nombres de campo como llaves y las propiedades como datos. FALSE es devuelto si hubo error.

Devuelve las propiedades SQL de cada campo como un array. Las propiedades son codificadas así: "SQLTYPE;longitud;precisión;escala;ISNULLABLE" siendo SQLTYPE el tipo de dato definido en Informix como puede ser "SQLVCHAR" etc. e ISNULLABLE (puede ser nulo) igual a "Y" sí o "N" no.

Ejemplo 1. Propiedades de los campos de una consulta SQL

```
$properties = ifx_fieldtypes ($resultid);
if (! isset($properties)) {
  ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
    next ($properties);
}
```

ifx_fieldtypes (PHP 3>= 3.0.3, PHP 4)

Obtiene los campos de una consulta SQL

array **ifx_fieldtypes** (int result_id) \linebreak

Dada una consulta representada por *result_id* devuelve un array con los nombres de campo como llaves y los tipos como datos. Si no tuvo éxito da FALSE.

Ejemplo 1. Nombres y tipos de campos de una consulta SQL

```
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
  ... error ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);
    printf("%s :\t type =  %s\n", $fname, $types[$fname]);
    next($types);
}
```

ifx_free_blob (PHP 3>= 3.0.4, PHP 4)

Borra el objeto blob

int **ifx_free_blob** (int bid) \linebreak

Elimina el objeto blob representado por el identificador *bid*. Devuelve FALSE si se produjo error, en otro caso TRUE.

ifx_free_char (PHP 3>= 3.0.6, PHP 4)

Elimina un objeto char

int **ifx_free_char** (int bid) \linebreak

Borra el objeto char representado por el identificador del char *bid*. Devuelve FALSE si se produjo un error, en otro caso TRUE.

ifx_free_result (PHP 3>= 3.0.3, PHP 4)

Libera los recursos de una consulta

int **ifx_free_result** (int result_id) \linebreak

Libera los recursos representados por el identificador *result_id* de una consulta. Devuelve FALSE si hubo error.

ifx_get_blob (PHP 3>= 3.0.4, PHP 4)

Obtiene el contenido de un objeto blob

int **ifx_get_blob** (int bid) \linebreak

Devuelve el contenido de un objeto blob representado por su identificador *bid*.

ifx_get_char (PHP 3>= 3.0.6, PHP 4)

Obtiene el contenido de un objeto char

int **ifx_get_char** (int bid) \linebreak

Devuelve el contenido de un objeto char representado por su identificador *bid*.

ifx_getsqlca (PHP 3>= 3.0.8, PHP 4)

Después de una consulta, obtiene el contenido de sqlca.sqlerrd[0..5]

array **ifx_getsqlca** (int result_id) \linebreak

result_id es un identificador válido del resultado de ifx_query() o ifx_prepare().

Devuelve una seudo fila (array asociativo) con los valores de sqlca.sqlerrd[0] a sqlca.sqlerrd[5] de una consulta ejecutada, representada ésta con un identificador de resultado *result_id*.

Para inserciones, actualizaciones y borrados los valores devueltos son aquellos definidos por el servidor después de que la consulta sea ejecutada. Esto da acceso al número de registros procesados y al valor de una columna de tipo serial en una consulta de inserción. Para consultas de selección, los valores son guardados cuando se prepara la sentencia. También permite conocer el número estimado de registros procesados. El uso de esta función evita el sobrecoste de ejecutar la consulta "select dbinfo('sqlca.sqlerrdx')", como obtener los valores guardados por el conector para Informix en el momento apropiado.

Ejemplo 1. Obtener los valores sqlca.sqlerrd[x]

```
/* suponiendo que la primera columna de la tabla 'sometable' es de tipo serial */
$qid = ifx_query("insert into sometable values(0, '2nd column', 'another column' ", $connid)
if (! $qid) {
    ... error ...
```

```
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : " . $serial_value<br>\n"; // El valor de tip
rial del registro insertado es:
```

ifx_htmltbl_result (PHP 3>= 3.0.3, PHP 4)

Muestra todos los registros de una consulta en una tabla HTML

int **ifx_htmltbl_result** (int result_id [, string html_table_options]) \linebreak

Devuelve el número de registros leídos o `FALSE` si hubo error.

Muestra todas las filas de la consulta *`result_id`* dentro de una tabla html. El argumento segundo, opcional, es una cadena de parámetros del tag <table>

Ejemplo 1. Mostrar resultado como una tabla HTML

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);  // Demasia-
dos registros en el resultado
    die ("Please restrict your query<br>\n");                      // Por favor, re-
stringe tu consulta
}
if (! ifx_do($rid) {
  ... error ...
}

ifx_htmltbl_result ($rid, "border=\"2\"");

ifx_free_result($rid);
```

ifx_nullformat (PHP 3>= 3.0.4, PHP 4)

Define el valor por defecto cuando se leen valores nulos

void **ifx_nullformat** (int mode) \linebreak

Define el valor por defecto cuando se leen valores nulos. Modo (mode) "0" devuelve "", y modo "1" devuelve "NULL".

ifx_num_fields (PHP 3>= 3.0.3, PHP 4)

Devuelve el número de columnas en una consulta

int **ifx_num_fields** (int result_id) \linebreak

Dada una consulta representada por *result_id* devuelve el número de columnas o FALSE si se produjo un error.

Después de preparar o ejecutar una consulta, una llamada a esta función te da el número de columnas en la consulta.

ifx_num_rows (PHP 3>= 3.0.3, PHP 4)

Cuenta los registros ya leídos de una consulta

int **ifx_num_rows** (int result_id) \linebreak

Da el número de registros ya leídos de una consulta representada por un *result_id* después de llamar a ifx_query() o ifx_do().

ifx_pconnect (PHP 3>= 3.0.3, PHP 4)

Abre una conexión permanente con Informix

int **ifx_pconnect** ([string database [, string userid [, string password]]]) \linebreak

Devuelve un identificador positivo de enlace persistente si hubo conexión, o FALSE si se produjo un error.

ifx_pconnect() actúa muy parecido a ifx_connect() con dos principales diferencias.

Esta función se comporta exactamente igual que ifx_connect() cuando PHP no es ejecutado como un módulo de Apache. La primera diferencia es cuando se conecta, la función intentará encontrar un enlace (persistente) que exista con el mismo servidor, usuario y contraseña. Si es hallado, el identificador del enlace será devuelto en vez de abrir una nueva conexión.

Segundo, la conexión al servidor no se cerrará cuando la ejecución del guión (script) finalice. En vez de esto, la conexión permanecerá abierta para usos futuros (ifx_close() no cerrará el enlace creado por **ifx_pconnect()**).

Este tipo de enlace es, por tanto, llamado ’persistente’

Examina también: ifx_connect().

ifx_prepare (PHP 3>= 3.0.4, PHP 4)

Prepara una sentencia SQL para su ejecución

int **ifx_prepare** (string query, int conn_id [, int cursor_def, mixed blobidarray]) \linebreak

Devuelve un entero (integer) *result_id* para usarlo con ifx_do(). Es definido "affected_rows" (registros procesados) y se puede obtener mediante la función ifx_affected_rows().

Prepara una consulta (*query*) sobre una conexión (*link_identifier*). Un cursor es definido y abierto para las consultas de selección. El parámetro opcional tipo de cursor (*cursor_type*) te permite que sea un cursor de tipo "scroll" y/o "hold". Es una máscara y puede ser IFX_SCROLL, IFX_HOLD o ambos.

Para cualquier tipo de consulta el número estimado de registros afectados (procesados) es guardado y puede ser obtenido mediante ifx_affected_rows().

Si tienes columnas BLOB (BYTE o TEXT) en una consulta, puedes añadir un parámetro *blobidarray* conteniendo los identificadores de blob y sustituir los valores de esas columnas por una "?" en el texto de la consulta.

Si el contenido de la columna de tipo TEXT (o BYTE) lo permite, puedes también usar "ifx_textasvarchar(1)" y "ifx_byteasvarchar(1)". Esto supone manejar columnas de tipo TEXT (o BYTE) como si fueran columnas normales de tipo VARCHAR (pero teniendo en cuenta que tendrán un mayor tamaño que el habitual), para consultas de selección y no necesitas preocuparte por los identificadores de blob.

La opción por defecto ifx_textasvarchar(0) o ifx_byteasvarchar(0) devuelve identificadores de blob (valores enteros) para las consultas de selección. Puedes obtener el contenido del blob como una cadena o un fichero con las funciones para blob (ver más adelante).

Examina también: ifx_do().

ifx_query (PHP 3>= 3.0.3, PHP 4)

Envía una consulta a Informix

int **ifx_query** (string query [, int link_identifier [, int cursor_type [, mixed blobidarray]]]) \linebreak

Devuelve un identificador positivo de resultado si tuvo éxito, `FALSE` en otro caso.

Un entero (integer) "result_id" usado por otras funciones para obtener el resultado de la consulta. Es definido "affected_rows" (registros procesados) y se puede obtener mediante la función ifx_affected_rows().

ifx_query() envía una consulta a la base de datos activa actualmente en el servidor, la cual está representada por el identificador de enlace especificado (link_identifier). Si el identificador no es definido, el último enlace abierto es asumido. Si el enlace no se encuentra abierto, ifx_connect() es llamado y utilizado.

Ejecuta una consulta (*query*) sobre una conexión (*link_identifier*). Un cursor es definido y abierto para las consultas de selección. El parámetro opcional tipo de cursor (*cursor_type*) te

permite que sea un cursor de tipo "scroll" y/o "hold". Es una máscara y puede ser IFX_SCROLL, IFX_HOLD o ambos. Las consultas que no son de selección son ejecutadas inmediatamente.

Para cualquier tipo de consulta el número (estimado o real) de registros procesados es guardo y se puede obtener mediante ifx_affected_rows().

Si tienes columnas BLOB (BYTE o TEXT) en una consulta de actualización, puedes añadir un parámetro *blobidarray* conteniendo los identificadores de blob y sustituir los valores de esas columnas por una "?" en el texto de la consulta.

Si el contenido de la columna de tipo TEXT (o BYTE) lo permite, también puedes usar "ifx_textasvarchar(1)" y "ifx_byteasvarchar(1)". Esto supone manejar columnas de tipo TEXT (o BYTE) como si fueran columnas normales de tipo VARCHAR (pero teniendo en cuenta que tendrán un mayor tamaño que el habitual), para consultas de selección y no necesitas preocuparte por los identificadores de blob.

La opción por defecto ifx_textasvarchar(0) o ifx_byteasvarchar(0) devuelve identificadores de blob (valores enteros) para las consultas de selección. Puedes obtener el contenido del blob como una cadena o un fichero con las funciones para blob (ver más adelante).

Examina también: ifx_connect().

Ejemplo 1. Mostrar todos los registros de la tabla "orders" como una tabla html

```
ifx_textasvarchar(1);      // usa "modo texto" para blobs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
    printf("Can't select orders : %s\n<br>%s<br>\n", ifx_error());
    ifx_errormsg();
    die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);
```

Ejemplo 2. Inserta valores en la tabla "catalog"

```
                      // crea identificadores de blob para una columna de tipo byte y otra t
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");
                      // almacena los identificadores de blob en un array lla-
mado blobid
$blobidarray[] = $textid;
$blobidarray[] = $byteid;
                      // lanza la consulta
$query = "insert into catalog (stock_num, manu_code, " .
         "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
  ... error ...
}
                     // libera el resultado
ifx_free_result($res_id);
```

ifx_textasvarchar (PHP 3>= 3.0.4, PHP 4)

Define el modo por defecto para los campos de tipo text

void **ifx_textasvarchar** (int mode) \linebreak

Define el modo por defecto para los campos de tipo text en todas las consultas de selección. Modo (mode) "0" devolverá un identificador de blob y "1" dará el contenido en un campo de tipo varchar.

ifx_update_blob (PHP 3>= 3.0.4, PHP 4)

Actualiza el contenido de un objeto blob

ifx_update_blob (int bid, string content) \linebreak

Actualiza el contenido de un objeto blob representado por su identificador *`bid`*. *`content`* es una cadena con el nuevo contenido. Devuelve `FALSE` si hubo error, en otro caso `TRUE`.

ifx_update_char (PHP 3>= 3.0.6, PHP 4)

Actualiza el contenido de un objeto char

int **ifx_update_char** (int bid, string content) \linebreak

Actualiza el contenido de un objeto char representado por su identificador *`bid`*. *`content`* es una cadena con la información nueva. Devuelve `FALSE` si se produjo un error, en otro caso `TRUE`.

ifxus_close_slob (PHP 3>= 3.0.4, PHP 4)

Cierra un objeto slob

int **ifxus_close_slob** (int bid) \linebreak

Cierra un objeto slob representado por su identificador de slob *`bid`*. Devuelve `FALSE` si hubo error, `TRUE` en otro caso.

ifxus_create_slob (PHP 3>= 3.0.4, PHP 4)

Crea un objeto slob y lo abre

int **ifxus_create_slob** (int mode) \linebreak

Crea un objeto slob y lo abre. Modos: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER o una combinación de ellos. También puedes usar nombres de constantes IFX_LO_RDONLY, IFX_LO_WRONLY, etc. Devuelve `FALSE` si hubo error, en otro caso el identificador del nuevo objeto slob.

ifx_free_slob (unknown)

Elimina un objeto slob

int **ifxus_free_slob** (int bid) \linebreak

Borra un objeto slob. *`bid`* es el identificador del objeto slob. Devuelve `FALSE` si hubo error, `TRUE` en otro caso.

ifxus_open_slob (PHP 3>= 3.0.4, PHP 4)

Abre un objeto slob

int **ifxus_open_slob** (long bid, int mode) \linebreak

Abre un objeto slob. *`bid`* será un identificador de slob que válido. Modos: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER o una combinación de ellos. Devuelve `FALSE` si hubo error, en otro caso el identificador del nuevo objeto slob.

ifxus_read_slob (PHP 3>= 3.0.4, PHP 4)

Lee un número de bytes (nbytes) de un objeto slob

int **ifxus_read_slob** (long bid, long nbytes) \linebreak

Lee un número de bytes (nbytes) de un objeto slob. *`bid`* es un identificador de slob válido y *`nbytes`* es el número de bytes a leer. Devuelve `FALSE` si hubo error, sino la cadena.

ifxus_seek_slob (PHP 3>= 3.0.4, PHP 4)

Define el fichero o posición en memoria

int **ifxus_seek_blob** (long bid, int mode, long offset) \linebreak

Define el fichero o posición en memoria de un objeto slob abierto, *bid* será un identificador de slob válido. Modos (mode): 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END y *offset* es el desplazamiento en bytes. Si hubo error entonces da FALSE.

ifxus_tell_slob (PHP 3>= 3.0.4, PHP 4)

Devuelve el fichero actual o la posición en memoria

int **ifxus_tell_slob** (long bid) \linebreak

Devuelve el fichero actual o la posición en memoria de un objeto slob abierto, *bid* será un identificador de slob válido. Si hubo error entonces da FALSE.

ifxus_write_slob (PHP 3>= 3.0.4, PHP 4)

Escribe una cadena en un objeto slob

int **ifxus_write_slob** (long bid, string content) \linebreak

Escribe una cadena en un objeto slob. *bid* es un identificador de slob válido y *content* el contenido a escribir. Devuelve FALSE si hubo error, sino el número de bytes escritos.

XLV. Funciones InterBase

ibase_blob_add (PHP 3>= 3.0.7, PHP 4)

Add data into created blob

int **ibase_blob_add** (int blob_id, string data) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ibase_blob_cancel (PHP 3>= 3.0.7, PHP 4)

Cancel creating blob

int **ibase_blob_cancel** (int blob_id) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ibase_blob_close (PHP 3>= 3.0.7, PHP 4)

Close blob

int **ibase_blob_close** (int blob_id) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ibase_blob_create (PHP 3>= 3.0.7, PHP 4)

Create blob for adding data

int **ibase_blob_create** ([int link_identifier]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ibase_blob_echo (PHP 3>= 3.0.7, PHP 4)

Output blob contents to browser

int **ibase_blob_echo** (string blob_id_str) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ibase_blob_get (PHP 3>= 3.0.7, PHP 4)

Get len bytes data from open blob

string **ibase_blob_get** (int blob_id, int len) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ibase_blob_import (PHP 3>= 3.0.7, PHP 4)

Create blob, copy file in it, and close it

string **ibase_blob_import** ([int link_identifier, int file_id]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ibase_blob_info (PHP 3>= 3.0.7, PHP 4)

Return blob length and other useful info

object **ibase_blob_info** (string blob_id_str) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ibase_blob_open (PHP 3>= 3.0.7, PHP 4)

Open blob for retrieving data parts

int **ibase_blob_open** (string blob_id) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ibase_close (PHP 3>= 3.0.6, PHP 4)

ibase_close ( ) \linebreak

ibase_commit (PHP 3>= 3.0.7, PHP 4)

Commit a transaction

int **ibase_commit** ([int link_identifier, int trans_number]) \linebreak

Commits transaction *trans_number* which was created with ibase_trans().

ibase_connect (PHP 3>= 3.0.6, PHP 4)

ibase_connect ( ) \linebreak

ibase_errmsg (PHP 3>= 3.0.7, PHP 4)

Returns error messages

string **ibase_errmsg** (void) \linebreak

Returns a string containing an error message.

ibase_execute (PHP 3>= 3.0.6, PHP 4)

ibase_execute ( ) \linebreak

ibase_fetch_object (PHP 3>= 3.0.7, PHP 4)

Get an object from a InterBase database

object **ibase_fetch_object** (int result_id) \linebreak

Fetches a row as a pseudo-object from a *result_id* obtained either by ibase_query() or ibase_execute().

```
<php
    $dbh = ibase_connect ($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query ($dbh, $stmt);
    while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
    }
    ibase_close ($dbh);
?>
```

Subsequent call to **ibase_fetch_object()** would return the next row in the result set, or FALSE if there are no more rows.

See also ibase_fetch_row().

ibase_fetch_row (PHP 3>= 3.0.6, PHP 4)

ibase_fetch_row ( ) \linebreak

ibase_field_info (PHP 3>= 3.0.7, PHP 4)

Get information about a field

array **ibase_field_info** (int result, int field number) \linebreak

Returns an array with information about a field after a select query has been run. The array is in the form of name, alias, relation, length, type.

```
$rs=ibase_query("SELECT * FROM tablename");
$coln = ibase_num_fields($rs);
for ($i=0; $i < $coln; $i++) {
    $col_info = ibase_field_info($rs, $i);
    echo "name: ".$col_info['name']."\n";
    echo "alias: ".$col_info['alias']."\n";
    echo "relation: ".$col_info['relation']."\n";
    echo "length: ".$col_info['length']."\n";
    echo "type: ".$col_info['type']."\n";
    }
```

ibase_free_query (PHP 3>= 3.0.6, PHP 4)

ibase_free_query ( ) \linebreak

ibase_free_result (PHP 3>= 3.0.6, PHP 4)

ibase_free_result ( ) \linebreak

ibase_num_fields (PHP 3>= 3.0.7, PHP 4)

Get the number of fields in a result set

int **ibase_num_fields** (int result_id) \linebreak

Returns an integer containing the number of fields in a result set.

```
<?php
$dbh = ibase_connect ($host, $username, $password);
$stmt = 'SELECT * FROM tblname';
$sth = ibase_query ($dbh, $stmt);

if (ibase_num_fields($sth) > 0) {
while ($row = ibase_fetch_object ($sth)) {
print $row->email . "\n";
}
} else {
die ("No Results were found for your query");
}

ibase_close ($dbh);
?>
```

See also: ibase_field_info().

ibase_pconnect (PHP 3>= 3.0.6, PHP 4)

ibase_pconnect ( ) \linebreak

ibase_prepare (PHP 3>= 3.0.6, PHP 4)

ibase_prepare ( ) \linebreak

ibase_query (PHP 3>= 3.0.6, PHP 4)

ibase_query ( ) \linebreak

ibase_rollback (PHP 3>= 3.0.7, PHP 4)

Rolls back a transaction

int **ibase_rollback** ([int link_identifier, int trans_number]) \linebreak

Rolls back transaction *`trans_number`* which was created with ibase_trans().

ibase_timefmt (PHP 3>= 3.0.6, PHP 4)

ibase_timefmt ( ) \linebreak

ibase_trans (PHP 3>= 3.0.7, PHP 4)

Begin a transaction

int **ibase_trans** ([int trans_args [, int link_identifier]]) \linebreak

Begins a transaction.

XLVI. Ingres II functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

These functions allow you to access Ingres II database servers.

In order to have these functions available, you must compile php with Ingres support by using the `--with-ingres` option. You need the Open API library and header files included with Ingres II. If the II_SYSTEM environment variable isn't correctly set you may have to use `--with-ingres=DIR` to specify your Ingres installation directory.

When using this extension with Apache, if Apache does not start and complains with "PHP Fatal error: Unable to start ingres_ii module in Unknown on line 0" then make sure the environement variable II_SYSTEM is correctly set. Adding "export II_SYSTEM="/home/ingres/II" in the script that starts Apache, just before launching httpd, should be fine.

> **Nota:** If you already used PHP extensions to access other database servers, note that Ingres doesn't allow concurrent queries and/or transaction over one connection, thus you won't find any result or transaction handle in this extension. The result of a query must be treated before sending another query, and a transaction must be commited or rolled back before opening another transaction (which is automaticaly done when sending the first query).

ingres_autocommit (PHP 4 >= 4.0.2)

Switch autocommit on or off.

bool **ingres_autocommit** ([resource link]) \linebreak

ingres_autocommit() is called before opening a transaction (before the first call to ingres_query() or just after a call to ingres_rollback() or **ingres_autocommit()**) to switch the "autocommit" mode of the server on or off (when the script begins the autocommit mode is off).

When the autocommit mode is on, every query is automaticaly commited by the server, as if ingres_commit() was called after every call to ingres_query().

See also ingres_query(), ingres_rollback() and ingres_commit().

ingres_close (PHP 4 >= 4.0.2)

Close an Ingres II database connection

bool **ingres_close** ([resource link]) \linebreak

Returns `TRUE` on success, or `FALSE` on failure.

ingres_close() closes the connection to the Ingres server that's associated with the specified link. If the *`link`* parameter isn't specified, the last opened link is used.

ingres_close() isn't usually necessary, as it won't close persistent connections and all non-persistent connections are automatically closed at the end of the script.

See also ingres_connect(), and ingres_pconnect().

ingres_commit (PHP 4 >= 4.0.2)

Commit a transaction.

bool **ingres_commit** ([resource link]) \linebreak

ingres_commit() commits the currently open transaction, making all changes made to the database permanent.

This closes the transaction. A new one can be open by sending a query with ingres_query().

You can also have the server commit automaticaly after every query by calling ingres_autocommit() before opening the transaction.

See also ingres_query(), ingres_rollback() and ingres_autocommit().

ingres_connect (PHP 4 >= 4.0.2)

Open a connection to an Ingres II database.

resource **ingres_connect** ([string database [, string username [, string password]]]) \linebreak

Returns a Ingres II link resource on success, or `FALSE` on failure.

ingres_connect() opens a connection with the Ingres database designated by *`database`*, which follows the syntax *`[node_id::]dbname[/svr_class]`*.

If some parameters are missing, **ingres_connect()** uses the values in `php.ini` for *`ingres.default_database`*, *`ingres.default_user`* and *`ingres.default_password`*.

The connection is closed when the script ends or when ingres_close() is called on this link.

All the other ingres functions use the last opened link as a default, so you need to store the returned value only if you use more than one link at a time.

Ejemplo 1. ingres_connect() example

```
<?php
    $link = ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ($link);
?>
```

Ejemplo 2. ingres_connect() example using default link

```
<?php
    ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ();
?>
```

See also ingres_pconnect(), and ingres_close().

ingres_fetch_array (PHP 4 >= 4.0.2)

Fetch a row of result into an array.

array **ingres_fetch_array** ([int result_type [, resource link]]) \linebreak

ingres_fetch_array() Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

This function is an extended version of ingres_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use the numeric index of the column or make an alias for the column.

```
ingres_query(select t1.f1 as foo t2.f1 as bar from t1, t2);
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];
```

result_type can be II_NUM for enumerated array, II_ASSOC for associative array, or II_BOTH (default).

Speed-wise, the function is identical to ingres_fetch_object(), and almost as quick as ingres_fetch_row() (the difference is insignificant).

Ejemplo 1. ingres_fetch_array() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_array()) {
    echo $row["user_id"];  # using associative array
    echo $row["fullname"];
    echo $row[1];          # using enumerated array
    echo $row[2];
}
?>
```

See also ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_object() and ingres_fetch_row().

ingres_fetch_object (PHP 4 >= 4.0.2)

Fetch a row of result into an object.

object **ingres_fetch_object** ([int result_type [, resource link]]) \linebreak

ingres_fetch_object() Returns an object that corresponds to the fetched row, or FALSE if there are no more rows.

This function is similar to ingres_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: II_ASSOC, II_NUM, and II_BOTH.

Speed-wise, the function is identical to ingres_fetch_array(), and almost as quick as ingres_fetch_row() (the difference is insignificant).

Ejemplo 1. ingres_fetch_object() example

```
<?php
ingres_connect ($database, $user, $password);
ingres_query ("select * from table");
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

See also ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_array() and ingres_fetch_row().

ingres_fetch_row (PHP 4 >= 4.0.2)

Fetch a row of result into an enumerated array.

array **ingres_fetch_row** ([resource link]) \linebreak

ingres_fetch_row() returns an array that corresponds to the fetched row, or FALSE if there are no more rows. Each result column is stored in an array offset, starting at offset 1.

Subsequent call to **ingres_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

Ejemplo 1. ingres_fetch_row() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also ingres_num_fields(), ingres_query(), ingres_fetch_array() and ingres_fetch_object().

ingres_field_length (PHP 4 >= 4.0.2)

Get the length of a field.

int **ingres_field_length** (int index [, resource link]) \linebreak

ingres_field_length() returns the length of a field. This is the number of bytes used by the server to store the field. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_field_name (PHP 4 >= 4.0.2)

Get the name of a field in a query result.

string **ingres_field_name** (int index [, resource link]) \linebreak

ingres_field_name() returns the name of a field in a query result, or `FALSE` on failure.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_field_nullable (PHP 4 >= 4.0.2)

Test if a field is nullable.

bool **ingres_field_nullable** (int index [, resource link]) \linebreak

ingres_field_nullable() returns `TRUE` if the field can be set to the `NULL` value and `FALSE` if it can't.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_field_precision (PHP 4 >= 4.0.2)

Get the precision of a field.

int **ingres_field_precision** (int index [, resource link]) \linebreak

ingres_field_precision() returns the precision of a field. This value is used only for decimal, float and money SQL data types. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_field_scale (PHP 4 >= 4.0.2)

Get the scale of a field.

int **ingres_field_scale** (int index [, resource link]) \linebreak

ingres_field_scale() returns the scale of a field. This value is used only for the decimal SQL data type. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_field_type (PHP 4 >= 4.0.2)

Get the type of a field in a query result.

string **ingres_field_type** (int index [, resource link]) \linebreak

ingres_field_type() returns the type of a field in a query result, or `FALSE` on failure. Examples of types returned are "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE". Some of these types can map to more than one SQL type depending on the length of the field (see ingres_field_length()). For example "IIAPI_FLT_TYPE" can be a float4 or a float8. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_num_fields (PHP 4 >= 4.0.2)

Get the number of fields returned by the last query

int **ingres_num_fields** ([resource link]) \linebreak

ingres_num_fields() returns the number of fields in the results returned by the Ingres server after a call to ingres_query()

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_num_rows (PHP 4 >= 4.0.2)

Get the number of rows affected or returned by the last query

int **ingres_num_rows** ([resource link]) \linebreak

For delete, insert or update queries, **ingres_num_rows()** returns the number of rows affected by the query. For other queries, **ingres_num_rows()** returns the number of rows in the query's result.

> **Nota:** This function is mainly meant to get the number of rows modified in the database. If this function is called before using ingres_fetch_array(), ingres_fetch_object() or ingres_fetch_row() the server will delete the result's data and the script won't be able to get them.
>
> You should instead retrieve the result's data using one of these fetch functions in a loop until it returns `FALSE`, indicating that no more results are available.

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_pconnect (PHP 4 >= 4.0.2)

Open a persistent connection to an Ingres II database.

resource **ingres_pconnect** ([string database [, string username [, string password]]]) \linebreak

Returns a Ingres II link resource on success, or `FALSE` on failure.

See ingres_connect() for parameters details and examples. There are only 2 differences between **ingres_pconnect()** and ingres_connect() : First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the Ingres server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (ingres_close() will not close links established by **ingres_pconnect()**). This type of link is therefore called 'persistent'.

See also ingres_connect(), and ingres_close().

ingres_query (PHP 4 >= 4.0.2)

Send a SQL query to Ingres II

bool **ingres_query** (string query [, resource link]) \linebreak

Returns `TRUE` on success, or `FALSE` on failure.

ingres_query() sends the given *query* to the Ingres server. This query must be a valid SQL query (see the Ingres SQL reference guide)

The query becomes part of the currently open transaction. If there is no open transaction, **ingres_query()** opens a new transaction. To close the transaction, you can either call ingres_commit() to commit the changes made to the database or ingres_rollback() to cancel these changes. When the script ends, any open transaction is rolled back (by calling ingres_rollback()). You can also use ingres_autocommit() before opening a new transaction to have every SQL query immediatly commited.

Some types of SQL queries can’t be sent with this function :

- close (see ingres_close()).
- commit (see ingres_commit()).
- connect (see ingres_connect()).
- disconnect (see ingres_close()).
- get dbevent
- prepare to commit
- rollback (see ingres_rollback()).
- savepoint
- set autocommit (see ingres_autocommit()).
- all cursor related queries are unsupported

Ejemplo 1. ingres_query() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also ingres_fetch_array(), ingres_fetch_object(), ingres_fetch_row(), ingres_commit(), ingres_rollback() and ingres_autocommit().

ingres_rollback (PHP 4 >= 4.0.2)

Roll back a transaction.

bool **ingres_rollback** ([resource link]) \linebreak

ingres_rollback() rolls back the currently open transaction, actualy canceling all changes made to the database during the transaction.

This closes the transaction. A new one can be open by sending a query with ingres_query().

See also ingres_query(), ingres_commit() and ingres_autocommit().

XLVII. IRC Gateway Functions

What is ircg?

With ircg you can build powerful, fast and scalable webchat solutions in conjunction with dedicated or public IRC servers.

Platforms

IRCG runs under

- AIX
- FreeBSD
- HP-UX
- Irix
- Linux
- Solaris
- Tru64

Requirements

To install and use IRCG you need the following software:

1. IRCG-Library (http://www.schumann.cx/ircg/) from Sascha Schumann.
2. SGI Static Threads Library (http://sourceforge.net/projects/state-threads/)
3. thttpd (http://www.acme.com/software/thttpd/) webserver

Installation

A detailed installation instruction can be found here (http://lxr.php.net/source/php4/ext/ircg/README.txt).

ircg_channel_mode (PHP 4 >= 4.0.5)

Set channel mode flags for user

boolean **ircg_channel_mode** (resource connection, string channel, string mode_spec, string nick) \linebreak

Set channel mode flags for *channel* on server connected to by *connection*. Mode flags are passed in *mode_spec* and are applied to the user specified by *nick*.

Mode flags are set or cleared by specifind a mode character and prepending it with a plus or minus character respectively. E.g. operator mode is granted by '+o' and revoked by '-o' passed as *mode_spec*.

ircg_disconnect (PHP 4 >= 4.0.4)

Close connection to server

boolean **ircg_disconnect** (resource connection, string reason) \linebreak

ircg_disconnect() will close a *connection* to a server previously established with ircg-pconnect().

See also: ircg_pconnect().

ircg_fetch_error_msg (PHP 4 >= 4.1.0)

Returns the error from previous ircg operation

array **ircg_fetch_error_msg** (resource connection) \linebreak

ircg_fetch_error_msg() returns the error from the last called ircg function.

Nota: Errorcode is stored in first array element, errortext in second.

Ejemplo 1. ircg_fetch_error_msg() example

```
if (!ircg_join ($id, "#php")) {
    $error = ircg_fetch_error_msg($id);
    print ("Can't join channel #php. Errorcode:
            $error[0] Description: $error[1]");
}
```

ircg_get_username (PHP 4 >= 4.1.0)

Get username for connection

string **ircg_get_username** (int connection) \linebreak

Function **ircg_get_username()** returns the username for specified connection *connection*. Returns FALSE if *connection* died or is not valid.

ircg_html_encode (PHP 4 >= 4.0.5)

Encodes HTML preserving output

boolean **ircg_html_encode** (string html_string) \linebreak

Encodes a HTML string *html_string* for output. This feature could be usable, e.g. if someone wants to discuss about an html problem.

ircg_ignore_add (PHP 4 >= 4.0.5)

Add a user to your ignore list on a server

boolean **ircg_ignore_add** (resource connection, string nick) \linebreak

This function will add user *nick* to your ignore list on the server connected to by *connection*. By doing so you will suppress all messages from this user from being send to you.

See also: ircg_ignore_del().

ircg_ignore_del (PHP 4 >= 4.0.5)

Remove a user from your ignore list on a server

boolean **ircg_ignore_del** (resource connection, string nick) \linebreak

This function remove user *nick* from your ignore list on the server connected to by *connection*.

See also: ircg_ignore_add().

ircg_is_conn_alive (PHP 4 >= 4.0.5)

Check connection status

boolean **ircg_is_conn_alive** (resource connection) \linebreak

ircg_is_conn_alive() returns `TRUE` if *`connection`* is still alive and working or `FALSE` if the server no longer talks to us.

ircg_join (PHP 4 >= 4.0.4)

Join a channel on a connected server

boolean **ircg_join** (resource connection, string channel) \linebreak

Join the channel *`channel`* on the server connected to by *`connection`*.

ircg_kick (PHP 4 >= 4.0.5)

Kick a user out of a channel on server

boolean **ircg_kick** (resource connection, string channel, string nick, string reason) \linebreak

Kick user *`nick`* from *`channel`* on server connected to by *`connection`*. *`reason`* should give a short message describing why this action was performed.

ircg_lookup_format_messages (PHP 4 >= 4.0.5)

Select a set of format strings for display of IRC messages

boolean **ircg_lookup_format_messages** (string name) \linebreak

Select the set of format strings to use for display of IRC messages and events. Sets may be registered with ircg_register_format_messages(), a default set named `ircg` is always available.

See also: ircg_register_format_messages()

ircg_msg (PHP 4 >= 4.0.4)

Send message to channel or user on server

boolean **ircg_msg** (resource connection, string recipient, string message [, boolean suppress]) \linebreak

ircg_msg() will send the message to a channel or user on the server connected to by *`connection`*. A *`recipient`* starting with `#` or `&` will send the *`message`* to a channel, anything else will be interpreted as a username.

Setting the optional parameter *suppress* to a TRUE value will suppress output of your message to your own *connection*.

ircg_nick (PHP 4 >= 4.0.5)

Change nickname on server

boolean **ircg_nick** (resource connection, string nick) \linebreak

Change your nickname on the given *connection* to the one given in *nick* if possible.

Will return TRUE on success and FALSE on failure.

ircg_nickname_escape (PHP 4 >= 4.0.6)

Encode special characters in nickname to be IRC-compliant

string **ircg_nickname_escape** (string nick) \linebreak

Function **ircg_nickname_escape()** returns a decoded nickname specified by *nick* wich is IRC-compliant.

See also: ircg_nickname_unescape()

ircg_nickname_unescape (PHP 4 >= 4.0.6)

Decodes encoded nickname

string **ircg_nickname_unescape** (string nick) \linebreak

Function **ircg_nickname_unescape()** returns a decoded nickname, which is specified in *nick*.

See also: ircg_nickname_escape()

ircg_notice (PHP 4 >= 4.0.5)

Send a notice to a user on server

boolean **ircg_notice** (resource connection, string , string message) \linebreak

This function will send the *message* text to the user *nick* on the server connected to by *connection*. Query your IRC documentation of choice for the exact difference between a MSG and a NOTICE.

ircg_part (PHP 4 >= 4.0.4)

Leave a channel on server

boolean **ircg_part** (resource connection, string channel) \linebreak

Leave the channel *channel* on the server connected to by *connection*.

ircg_pconnect (PHP 4 >= 4.0.4)

Connect to an IRC server

resource **ircg_pconnect** (string username [, string server_ip [, int server_port [, string msg_format [, array ctcp_messages [, array user_settings]]]]]) \linebreak

ircg_pconnect() will try to establish a connection to an IRC server and return a connection resource handle for further use.

The only mandatory parameter is *username*, this will set your initial nickname on the server. *server_ip* and *server_port* are optional and default to `127.0.0.1` and `6667`.

Nota: For now parameter *server_ip* will not do any hostname lookups and will only accept IP addresses in numerical form.

Currently **ircg_pconnect()** always returns a valid handle, even if the connection failed.

You can customize the output of IRC messages and events by selection a format string set previously created with ircg_register_format_messages() by specifing the sets name in *msg_format*.

ctcp_messages

user_settings

See also: ircg_disconnect(), ircg_is_conn_alive(), ircg_register_format_messages().

ircg_register_format_messages (PHP 4 >= 4.0.5)

Register a set of format strings for display of IRC messages

boolean **ircg_register_format_messages** (string name, array messages) \linebreak

With **ircg_register_format_messages()** you can customize the way your IRC output looks like. You can even register different format string sets and switch between them on the fly with ircg_lookup_format_messages().

- Plain channel message
- Private message received
- Private message sent
- Some user leaves channel
- Some user enters channel
- Some user was kicked from the channel
- Topic has been changed
- Error
- Fatal error
- Join list end(?)
- Self part(?)
- Some user changes his nick
- Some user quits his connection
- Mass join begin
- Mass join element
- Mass join end
- Whois user
- Whois server
- Whois idle
- Whois channel
- Whois end
- Voice status change on user
- Operator status change on user
- Banlist
- Banlist end

- %f - from
- %t - to
- %c - channel
- %r - plain message
- %m - encoded message
- %j - js encoded message

- 1 - mod encode
- 2 - nickname decode

See also: ircg_lookup_format_messages().

ircg_set_current (PHP 4 >= 4.0.4)

Set current connection for output

boolean **ircg_set_current** (resource connection) \linebreak

Select the current connection for output in this execution context. Every output sent from the server connected to by *connection* will be copied to standard output while using default formatting or a format string set specified by ircg_register_format_messages() and selected by ircg_lookup_format_messages().

See also: ircg_register_format_messages() and ircg_lookup_format_messages().

ircg_set_file (PHP 4 >= 4.2.0)

Set logfile for connection

bool **ircg_set_file** (int connection, string path) \linebreak

Function **ircg_set_file()** specifies a logfile *path* in which all output from connection *connection* will be logged. Returns TRUE on success, otherwise FALSE.

ircg_set_on_die (PHP 4 >= 4.2.0)

Set hostaction to be execute when connection dies

bool **ircg_set_on_die** (int connection, string host, int port, string data) \linebreak

In case of the termination of connection *connection* IRCG will connect to *host* at *port* (Note: host must be an IPv4 address, IRCG does not resolve host-names due to blocking issues), send *data* to the new host connection and will wait until the remote part closes connection. This can be used to trigger a php script for example.

ircg_topic (PHP 4 >= 4.0.5)

Set topic for channel on server

boolean **ircg_topic** (resource connection, string channel, string new_topic) \linebreak

Change the topic for channel *channel* on the server connected to by *connection* to *new_topic*.

ircg_whois (PHP 4 >= 4.0.5)

Query user information for nick on server

boolean **ircg_whois** (resource connection, string nick) \linebreak

Sends a query to the connected server *connection* to send information for the specified user *nick*.

XLVIII. Java

There are two possible ways to bridge PHP and Java: you can either integrate PHP into a Java Servlet environment, which is the more stable and efficient solution, or integrate Java support into PHP. The former is provided by a SAPI module that interfaces with the Servlet server, the latter by the Java extension.

PHP 4 ext/java provides a simple and effective means for creating and invoking methods on Java objects from PHP. The JVM is created using JNI, and everything runs in-process. Build instructions for ext/java can be found in `php4/ext/java/README`.

Ejemplo 1. Java Example

```
<?php
  // get instance of Java class java.lang.System in PHP
  $system = new Java('java.lang.System');

  // demonstrate property access
  print 'Java version='.$system->getProperty('java.version').' <br>';
  print 'Java vendor=' .$system->getProperty('java.vendor').'  <br>';
  print 'OS='.$system->getProperty('os.name').' '.
              $system->getProperty('os.version').' on '.
              $system->getProperty('os.arch').' <br>';

  // java.util.Date example
  $formatter = new Java('java.text.SimpleDateFormat',
                        "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

  print $formatter->format(new Java('java.util.Date'));
?>
```

Ejemplo 2. AWT Example

```
<?php
  // This example is only intented to be run as a CGI.

  $frame  = new Java('java.awt.Frame', 'PHP');
  $button = new Java('java.awt.Button', 'Hello Java World!');

  $frame->add('North', $button);
  $frame->validate();
  $frame->pack();
  $frame->visible = True;

  $thread = new Java('java.lang.Thread');
  $thread->sleep(10000);
```

```
  $frame->dispose();
?>
```

Notes:

- `new Java()` will create an instance of a class if a suitable constructor is available. If no parameters are passed and the default constructor is useful as it provides access to classes like `java.lang.System` which expose most of their functionallity through static methods.
- Accessing a member of an instance will first look for bean properties then public fields. In other words, `print $date.time` will first attempt to be resolved as `$date.getTime()`, then as `$date.time`.
- Both static and instance members can be accessed on an object with the same syntax. Furthermore, if the java object is of type `java.lang.Class`, then static members of the class (fields and methods) can be accessed.
- Exceptions raised result in PHP warnings, and `NULL` results. The warnings may be eliminated by prefixing the method call with an "@" sign. The following APIs may be used to retrieve and reset the last error:
  - java_last_exception_get()
  - java_last_exception_clear()
- Overload resolution is in general a hard problem given the differences in types between the two languages. The PHP Java extension employs a simple, but fairly effective, metric for determining which overload is the best match.

  Additionally, method names in PHP are not case sensitive, potentially increasing the number of overloads to select from.

  Once a method is selected, the parameters are cooerced if necessary, possibly with a loss of data (example: double precision floating point numbers will be converted to boolean).

- In the tradition of PHP, arrays and hashtables may pretty much be used interchangably. Note that hashtables in PHP may only be indexed by integers or strings; and that arrays of primitive types in Java can not be sparse. Also note that these constructs are passed by value, so may be expensive in terms of memory and time.

sapi/servlet builds upon the mechanism defined by ext/java to enable the entire PHP processor to be run as a servlet. The primary advanatage of this from a PHP perspective is that web servers which support servlets typically take great care in pooling and reusing JVMs. Build instructions for the Servlet SAPI module can be found in `php4/sapi/README`. Notes:

- While this code is intended to be able to run on any servlet engine, it has only been tested on Apache’s Jakarta/tomcat to date. Bug reports, success stories and/or patches required to get this code

to run on other engines would be appreciated.

- PHP has a habit of changing the working directory. sapi/servlet will eventually change it back, but while PHP is running the servlet engine may not be able to load any classes from the CLASSPATH which are specified using a relative directory syntax, or find the work directory used for administration and JSP compilation tasks.

java_last_exception_clear (PHP 4 >= 4.0.2)

Clear last Java exception

void **java_last_exception_clear** (void) \linebreak

java_last_exception_get (PHP 4 >= 4.0.2)

Get last Java exception

exception **java_last_exception_get** (void) \linebreak

The following example demonstrates the usage of Java's exception handler from within PHP:

Ejemplo 1. Java exception handler

```
<?php
  $stack = new Java('java.util.Stack');
  $stack->push(1);

  // This should succeed
  $result = $stack->pop();
  $ex = java_last_exception_get();
  if (!$ex) print "$result\n";

  // This should fail (error suppressed by @)
  $result = @$stack->pop();
  $ex = java_last_exception_get();
  if ($ex) print $ex->toString();

  // Clear last exception
  java_last_exception_clear();
?>
```

XLIX. Funciones LDAP

Introducción a LDAP

LDAP es el protocolo de acceso a directorios ligero (Lightweight Directory Access Protocol), un protocolo usado para acceder a "Servidores de Directorio". El directorio es una clase especial de base de datos que contiene información estructurada en forma de árbol.

El concepto es similar a la estructura de directorios de los discos duros, pero en este caso, el directorio raiz es "El Mundo" y los subdirectorios de primer nivel son los "países". Niveles inferiores de la estructura de directorio contienen entradas para compañías, organizaciones o lugares, y en niveles aún inferiores se encuentran las entradas para la gente, y quizás de equipos informáticos y documentos.

Para referirse a un fichero en un subdirectorio del disco duro se usa algo como

```
/usr/local/misapps/docs
```

Las barras marcan cada división en la referencia al fichero, y la secuencia es leida de izquierda a derecha.

El equivalente a la referencia a un fichero en LDAP es el "distinguished name" (nombre distinguible), abreviado como "db". Un ejemplo de dn podría ser.

```
cn=Pedro Pérez,ou=Contabilidad,o=Mi Compañía,c=ES
```

Las comas marcan cada división en la referencia, y la secuencia se lee de derecha a izquierda. Este dn se leería como ..

```
country = ES
organization = Mi Compañía
organizationalUnit = Contabilidad
commonName = Pedro Pérez
```

De la misma manera que no hay reglas estrictas sobre como organizar la estructura de directorios de un disco duro, un administrador de un servidor de directorio puede establecer cualquier estructura que sea útil para sus propósitos. Sin embargo hay algunos acuerdos tácitos que siempre deben seguirse. El mensaje es que no se puede escribir código para acceder un directorio si no se conoce algo de su estructura, igual que no se puede usar una base de datos sin algún conocimiento sobre lo que está disponible en ella.

Ejemplo de código completo

Recuperar informacion para todas las entradas donde el apellido empiece por "P" de un servidor de

directorio, mostrando un extracto con el nombre y dirección de correo electrónico.

Ejemplo 1. ejemplo de búsqueda LDAP

```
<?php
// La secuencia básica para trabajar con LDAP es conectar, autentificarse,
// buscar, interpretar el resultado de la búsqueda y cerrar la conexión.

echo "<h3>Prueba de consulta LDAP</h3>";
echo "Conectando ...";
$ds=ldap_connect("localhost");  // Debe ser un servidor LDAP válido!
echo "El resultado de la conexión es ".$ds."<p>";

if ($ds) {
    echo "Autentificandose  ...";
    $r=ldap_bind($ds);     // Autentificación anónima, típicamente con
                           // acceso de lectura
    echo "El resultado de la autentificación es ".$r."<p>";

    echo "Buscando (sn=P*) ...";
    // Busqueda de entradas por apellidos
    $sr=ldap_search($ds,"o=Mi Compañia, c=ES", "sn=P*");
    echo "El resultado de la búsqueda es ".$sr."<p>";

    echo "El número de entradas devueltas es ".ldap_count_entries($ds,$sr)."<p>";

    echo "Recuperando entradas ...<p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Devueltos datos de ".$info["count"]." entradas:<p>";

    for ($i=0; $i<$info["count"]; $i++) {
        echo "dn es: ". $info[$i]["dn"] ."<br>";
        echo "La primera entrada cn es: ". $info[$i]["cn"][0] ."<br>";
        echo "La primera entrada email es: ". $info[$i]["mail"][0] ."<p>";
    }

    echo "Cerrando conexión";
    ldap_close($ds);

} else {
    echo "<h4>Ha sido imposible conectar al servidor LDAP</h4>";
}
?>
```

Usando las llamadas LDAP de PHP

Es necesario conseguir y compilar la librerías cliente de LDAP ya sea del paquete ldap-3.3 de la Universidad de Michigan o del Netscape Directory SDK. También es necesario recompilar PHP con

soporte LDAP activado para que la funciones LDAP de PHP funcionen.

Antes de usarse las llamadas LDAP se debe saber ..

- El nombre o dirección del servidor de directorio que se va a usar
- El "dn base" del servidor (la parte del directorio global contenida en ese servidor, que puede ser por ejemplo "o=Mi Compañia,c=ES")
- Si es necesaria contraseña para acceder al servidor (muchos servidores ofrecen acceso de lectura para usuarios anónimos pero requieren un password para cualquier otro acceso)

La secuencia típica de llamadas LDAP suele implementarse en aplicaciones que siguen el siguiente patrón:

```
ldap_connect()    // establecer la conexión con el servidor
  |
ldap_bind()       // login anónimo o autentificado
  |
Hacer búsquedas o actualizaciones en el directorio
y mostrar los resultados
  |
ldap_close()      // Cerrar la conexión
```

Más información

Mucha información acerca de LDAP puede ser consultada en

- Netscape (http://developer.netscape.com/tech/directory/)
- Universidad de Michigan (http://www.umich.edu/~dirsvcs/ldap/index.html)
- Proyecto OpenLDAP (http://www.openldap.org/)
- LDAP World (http://www.innosoft.com/ldapworld)

El SDK de Netscape contiene una Guía de Programación muy útil en formato html.

ldap_8859_to_t61 (PHP 4 >= 4.0.2)

Translate 8859 characters to t61 characters

string **ldap_8859_to_t61** (string value) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ldap_add (PHP 3, PHP 4)

Añade entradas a un directorio LDAP

int **ldap_add** (int identificador_de_conexion, string dn, array entrada) \linebreak

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_add()** se usa para añadir entradas o registros a un directorio LDAP. El DN ("distinguished name", nombre distinguible, la referencia de cualquier entrada LDAP) es especificado por dn. El array entrada especifica la información que quiere añadirse. Los valores del array son indexados por sus propios atributos. En caso de valores múltiples para un mismo atributo, son indexados usando enteros empezando con 0.

```
entry["atributo1"] = valor
entry["atributo2"][0] = valor1
entry["atributo2"][1] = valor2
```

Ejemplo 1. Ejemplo completo con login atentificado

```
<?php
$ds=ldap_connect("localhost");  // Asumimos que el servidor LDAP está en el
                                // servidor local

if ($ds) {
    // autentificarse con el dn apropiado para tener permisos de modificación
    $r=ldap_bind($ds,"cn=root, o=Mi Compañia, c=ES", "secreto");

    // prepare data
    $info["cn"]="Pedro Pérez";
    $info["sn"]="Pedro";
    $info["mail"]="pedro.p@algun.sitio";
    $info["objectclass"]="persona";
```

```
    // add data to directory
    $r=ldap_add($ds, "cn=Pedro Pérez, o=Mi Compañia, c=ES", $info);

    ldap_close($ds);
} else {
    echo "Ha sido imposible conectar al servidor LDAP";
}
?>
```

ldap_bind (PHP 3, PHP 4)

Autentifica en un directorio LDAP

int **ldap_bind** (int identificador_de_conexion [, string rdn_del_usuario [, string contraseña]]) \linebreak

Se conecta a un directorio LDAP con un RDN y su contraseña. Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

ldap_bind() se conecta al directorio con un determinado usuario. rdn_de_usuario y contraseña son opcionales. Si no son especificados, se intenta el acceso anónimo.

ldap_close (PHP 3, PHP 4)

Cierra una conexión a un servidor LDAP

int **ldap_close** (int identificador_de_conexion) \linebreak

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

ldap_close() cierra la conexión con el servidor LDAP asociada con el *identificador_de_conexion* especificado.

Esta llamada es idéntica internamente a ldap_unbind(). La API LDAP usa la llamada ldap_unbind(), y por lo tanto quizás deba usar esta llamada en lugar de **ldap_close()**.

ldap_compare (PHP 4 >= 4.0.2)

Compare value of attribute found in entry specified with DN

bool **ldap_compare** (resource link_identifier, string dn, string attribute, string value) \linebreak

Returns `TRUE` if *value* matches otherwise returns `FALSE`. Returns -1 on error.

ldap_compare() is used to compare *value* of *attribute* to value of same attribute in LDAP directory entry specified with *dn*.

The following example demonstrates how to check whether or not given password matches the one defined in DN specified entry.

Ejemplo 1. Complete example of password check

```
<?php

$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {

    // bind
    if(ldap_bind($ds)) {

        // prepare data
        $dn = "cn=Matti Meikku, ou=My Unit, o=My Company, c=FI";
        $value = "secretpassword";
        $attr = "password";

        // compare value
        $r=ldap_compare($ds, $dn, $attr, $value);

        if ($r === -1) {
            echo "Error: ".ldap_error($ds);
        } elseif ($r === TRUE) {
            echo "Password correct.";
        } elseif ($r === FALSE) {
            echo "Wrong guess! Password incorrect.";
        }

    } else {
        echo "Unable to bind to LDAP server.";
    }

    ldap_close($ds);

} else {
    echo "Unable to connect to LDAP server.";
}
?>
```

Aviso

ldap_compare() can NOT be used to compare BINARY values!

Nota: This function was added in 4.0.2.

ldap_connect (PHP 3, PHP 4)

Conecta con un servidor LDAP

int **ldap_connect** ([string nombre_host [, int puerto]]) \linebreak

Devuelve un identificador de conexión positivo en caso de éxito, ó falso si ocurre algún error.

ldap_connect() establece una conexión con el servidor LDAP especificado en *nombre_host* y *puerto*. Ambos argumentos son opcionales. Si no se especifican, el identificador de la conexión LDAP actualmente abierta es devuelto. Si sólo es especificado *nombre_host* el puerto tomado por defecto es el 389.

ldap_count_entries (PHP 3, PHP 4)

Cuenta el número de entradas de una búsqueda

int **ldap_count_entries** (int identificador_de_conexion, int identificador_de_resultado) \linebreak

Devuelve el número de entradas del resultado o falso si ha ocurrido algún error.

ldap_count_entries() devuelce el número de entradas almacenadas en el resultado de operaciones de búsqueda previas. *identificador_de_resultado* identifica el resultado ldap interno al que hacemos referencia.

ldap_delete (PHP 3, PHP 4)

Borra una entrada de un directorio

int **ldap_delete** (int identificador_de_conexion, string dn) \linebreak

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_delete()** borra la entrada particular dn del directorio LDAP.

ldap_dn2ufn (PHP 3, PHP 4)

Convierte un dn al formato User Friendly Naming

string **ldap_dn2ufn** (string dn) \linebreak

La función **ldap_dn2ufn()** es usada para convertir un DN en un formato más amigable para el usuario.

ldap_err2str (PHP 3>= 3.0.13, PHP 4)

Convierte un código numérico de error LDAP en un mensaje.

string **ldap_err2str** (int numerr) \linebreak

Devuelve una cadena con el mensaje de error.

Esta función devuelve una cadena con el mensaje de error explicativo del código numérico de error numerr. Aunque los códigos de error LDAP están estandarizados, diferentes librerías devuelven mensajes textuales de error diferentes o incluso localizados. Nunca se debe comprobar la existencia de un error específico por el mensaje textual, sino por el código numérico.

Var también ldap_errno() y ldap_error().

Ejemplo 1. Enumerando todos los mensajes de error LDAP

```
<?php
  for($i=0; $i<100; $i++) {
    printf("Error $i: %s<br>\n", ldap_str2err($i));
  }
?>
```

ldap_errno (PHP 3>= 3.0.12, PHP 4)

Devuelve el código numérico de error para el último comando LDAP.

int **ldap_errno** (int identificador_de_conexión) \linebreak

Devuelve el código de error del último comando LDAP para la conexión especificada.

Esta función devuelve el código numérico de error, que está estandarizado, producido por el último comando LDAP y en la conexión especificada. Este número puede ser convertido en un mensaje textual de error usando ldap_err2str().

A menos que decremente el nivel de alerta en su fichero `php3.ini` (ó `php.ini`) o anteponga a los comandos LDAP en símbolo @ (arroba) para suprimir las alertas y warnings, los errores producidos serán mostrados automáticamente en el código HTML generado.

Ejemplo 1. Generando y capturando un error

```
<?php
// Este ejemplo contiene un error, que será capturado.
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);
```

```
// error de sintaxis en la expresión del filtro (codigo
// de error 87). Debería ser "objectclass=*".
$res =  @ldap_search($ld, "o=Mi Compañía, c=ES", "objectclass");
if (!$res) {
    printf("LDAP-Código  Error: %s<br>\n", ldap_errno($ld));
    printf("LDAP-Mensaje Error: %s<br>\n", ldap_error($ld));
    die("Argh!<br>\n");
}
$info = ldap_get_entries($ld, $res);
printf("%d entradas encontradas.<br>\n", $info["count"]);
?>
```

Ver también ldap_err2str() y ldap_error().

ldap_error (PHP 3>= 3.0.12, PHP 4)

Devuelte el mensaje de error del último comando LDAP

string **ldap_error** (int identificador_de_conexión) \linebreak

Devuelve una cadena con el mensaje de error.

Esta función devuelve una cadena con el mensaje de error explicativo del error generado por el último comando LDAP en la conexión especificada. Aunque los códigos de error LDAP están estandarizados, diferentes librerías devuelven mensajes textuales de error diferentes o incluso localizados. Nunca se debe comprobar la existencia de un error específico por el mensaje textual, sino por el código numérico.

A menos que decremente el nivel de alerta en su fichero `php3.ini` (ó `php.ini`) o anteponga a los comandos LDAP en símbolo @ (arroba) para suprimir las alertas y warnings, los errores producidos serán mostrados automáticamente en el código HTML generado.

Ver también ldap_err2str() y ldap_errno().

ldap_explode_dn (PHP 3, PHP 4)

Divide un DN en las partes que le componen

array **ldap_explode_dn** (string dn, int con_atributos) \linebreak

La función **ldap_explode_dn()** es usada para dividir un DN devuelto por ldap_get_dn() en las partes que le componen. Cada parte es conocida como Relative Distinguished Name (Nombre Relativo Distinguible) abreviado como RDN. **ldap_explode_dn()** devuelve un array con todos esos componentes. *`con_atributos`* sirve para especificar si los RDN se devuelven sólo como valores o con sus atributos también (es decir, en un formato atributo=valor). Hay que poner *`with_attrib`* a 0 para obtener también los atributos y a 1 para obtener sólo los valores.

ldap_first_attribute (PHP 3, PHP 4)

Devuelte el primer atributo

string **ldap_first_attribute** (int identificador_de_conexion, int identificador_de_entrada_en_resultado, int identificador_ber) \linebreak

Devuelte el primer atributo en la entrada o falso si ocurre algún error.

De manera similar a leer entradas, los atributos también son leidos de uno en uno de una entrada en particular del directorio. **ldap_first_attribute()** devuelve el primer atributo en la entrada a la que apunta el identificador_de_entrada_en_resultado El resto de los atributos son obtenidos llamanda a la función ldap_next_attribute() sucesivamente. El parámetro *`indetificador_ber`* es el identificador del puntero interno a memoria. Es pasado por referencia. El mismo *`indetificador_ber`* es pasado a la función ldap_next_attribute() que modifica dicho puntero.

Ver también ldap_get_attributes()

ldap_first_entry (PHP 3, PHP 4)

Devuelve el identificador del primer resultado

int **ldap_first_entry** (int identificador_de_conexion, int identificador_de_resultado) \linebreak

Devuelve el identificador de la primera entrada del resultado ó falso en caso de error.

Las entradas en un resultado LDAP son leidas secuencialmente usando las funciones **ldap_first_entry()** y ldap_next_entry(). **ldap_first_entry()** devuelve el identificador de la primera entrada del resultado. Este identificador es entonces suministrado a la rutina **lap_next_entry()** pata obtener sucesivas entradas del resultado.

Ver también ldap_get_entries().

ldap_first_reference (PHP 4 >= 4.0.5)

Return first reference

resource **ldap_first_reference** (resource link, resource result) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ldap_free_result (PHP 3, PHP 4)

Libera la memoria que almacena los resultados

int **ldap_free_result** (int identificador_de_resultado) \linebreak

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

ldap_free_result() libera la memoria reservada internamente para almacenar el resultado de búsquedas LDAP asociada al identificador *identificador_de_resultado*. Toda la memoria de resultados es automáticamente liberada al finalizarse la ejecución de un script.

Normalmente la memoria reservada para resultados ldap se libera al final del script. En caso de que el script realice sucesivas búsquedas que devuelvan conjuntos de resultados grandes, puede utilizarse **ldap_free_result()** para mantener bajo el uso de memoria del script durante su ejecución.

ldap_get_attributes (PHP 3, PHP 4)

Obtiene los atributos de una entrada de un resultado de búsqueda

array **ldap_get_attributes** (int identificador_de_conexion, int identificador_de_entrada_de_resultado) \linebreak

Devuelve una completa información de la entrada en un array multidimensional o falso en caso de error.

La función **ldap_get_attributes()** es usada para simplificar el leer atributos y valores de una entrada de un resultado de búsqueda. El valor de retorno es un array multidimensional de atributos y sus valores.

Teniendo localizado una entrada específica en el directorio se puede conseguir la información que contiene dicha entrada usando esta llamada. Puede usar esta función para aplicaciones que naveguen por las entradas del directorio y/o cuando no se conoce la estructura de las entradas del directorio. En otras aplicaciones se busca un atributo específico, como la dirección de email o los apellidos y no importa el resto de información contenida..

valor_devuelto["count"] = número de atributos en la entrada
valor_devuelto[0] = primer atributo
valor_devuelto[n] = enésimo atributo

valor_devuelto["atributo"]["count"] = número de vaslores del atributo
valor_devuelto["atributo"][0] = primer valor del atributo
valor_devuelto["atributo"][i] = iésimo valor del atributo

Ejemplo 1. Mostrar la lista de atributos contenida en una entrada específica de un directorio

```
// $ds es un identificador de conexión al directorio

// $sr es un resultado de búsqueda válido de una llamada
// anterior a una de las funciones de búsqueda en directorios
// ldap.
```

```
$entrada = ldap_first_entry($ds, $sr);

$atributos = ldap_get_attributes($ds, $entrada);

echo $atributos["count"]." atributos contenidos en esta entrada:<p>";

for ($i=0; $i<$atributos["count"]; $i++)
    echo $atributos[$i]."<br>";
```

Ver también ldap_first_attribute() y ldap_next_attribute()

ldap_get_dn (PHP 3, PHP 4)

Obtiene el DN de una entrada de un resultado

string **ldap_get_dn** (int indentificador_de_conexion, int indentificador_de_entrada_de_resultado) \linebreak

Devuelve el DN de la entrada del resultado o falso en caso de error.

La función **ldap_get_dn()** se utiliza para obtener el DN de una entrada de un resultado de búsqueda.

ldap_get_entries (PHP 3, PHP 4)

Obtiene todas las entradas de un resultado

array **ldap_get_entries** (int indentificador_de_conexion, int indentificador_de_resultado) \linebreak

Devuelve una completa información de un resultado de búsqueda en un array multidimensional o falso en caso de error.

La función **ldap_get_entries()** es usada para simplificar el leer múltiples entradas de de un resultado y después leer sus atributos y multiples valores. Toda la información es devuelta por una llamada a una función en forma de array multidimensional. La estructura del array es como se muestra más abajo.

Los indices de atributos son convertidos a minúsculas. (Los atributos de servidores de directorios son indiferentes a las mayusculas/minusculas, pero no cuando son usados como indices de arrays)

valor_devuelto["count"] = número de entradas del resultado
valor_devuelto[0] : contiene los detalles de la primera entrada

valor_devuelto[i]["dn"] = DN de la entrada iésima del resultado

valor_devuelto[i]["count"] = número de atributos de la entrada iésima
valor_devuelto[i][j] = jésimo atributo de la iésima entrada del resultado

valor_devuelto[i]["atributo"]["count"] = número de valores para "atributo"

en la entrada iésima
valor_devuelto[i]["atributo"][j] = jésimo valor de "atributo" en la entrada iésima

Ver también ldap_first_entry() y ldap_next_entry()

ldap_get_option (PHP 4 >= 4.0.4)

Get the current value for given option

bool **ldap_get_option** (resource link_identifier, int option, mixed retval) \linebreak

Sets *retval* to the value of the specified option. Devuelve TRUE si todo fue bien, FALSE en caso de fallo.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN. These are described in draft-ietf-ldapext-ldap-c-api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt)

> **Nota:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4

Ejemplo 1. Check protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version))
    echo "Using protocol version $version";
else
    echo "Unable to determine protocol version";
```

See also ldap_set_option().

ldap_get_values_len (PHP 3>= 3.0.13, PHP 4)

Obtiene todos los valores binarios de un atributo de una entrada

array **ldap_get_values_len** (int indentificador_de_conexion, int indentificador_de_entrada_de_resultado, string atributo) \linebreak

Devuelve un array de valores del atributo o falso en caso de error.

La función **ldap_get_values_len()** se utiliza para obtener todos los valores de un atributo de una entrada de un resultado de búsqueda. La entrada es especificada por el *`indentificador_de_entrada_de_resultado`*. El número de valores se almacena en el índice "count" del array devuelto. Los valores individuales se almacenan con índices enteros en el array. El primer índice es 0.

Esta función se utiliza exactamente como ldap_get_values() salvo que permite manejar datos binarios y no cadenas de caracteres.

ldap_get_values (PHP 3, PHP 4)

Obtiene todos los valores de un atributo de una entrada

array **ldap_get_values** (int identificador_de_conexion, int identificador_de_entrada_de_resultado, string atributo) \linebreak

Devuelve un array de valores del atributo o falso en caso de error.

La función **ldap_get_values()** se utiliza para obtener todos los valores de un atributo de una entrada. La entrada del resultado es especificada por el *`identificador_de_entrada_de_resultado`*. El número de valores se almacena en el índice "count" del array devuelto. Los valores individuales se almacenan con índices enteros en el array. El primer índice es 0.

Esta llamada necesita un *`identificador_de_entrada_de_resultado`*, por lo que necesita ser precedida por una de las llamadas de búsqueda ldap y una llamada para obtener una entrada en particular del resultado.

La aplicación debe ser o bien programada específicamente para buscar ciertos atributos (como apellidos o email) o bien utilizar la función ldap_get_attributes() para averiguar que atributos existen para una entrada dada, antes de llamar a **ldap_get_values()**.

LDAP permite mas de un valor para cada atributo, por lo que se puede, por ejemplo, almacenar varias direcciones de email para una persona en el directorio y nombrar a ese atributo como "email"

valor_devuelto["count"] = número de valores del atributo
valor_devuelto[0] = primer valor del atributo
valor_devuelto[i] = iésimo valor del atributo

Ejemplo 1. Listar todos los valores del atributo "email" de una entrada de un directorio

```
// $ds es un identificador de conexión al directorio

// $sr es un resultado de búsqueda válido de una llamada
// anterior a una de las funciones de búsqueda en directorios
// ldap.
```

```
// $entrada es un identificador de entrada válido de una llamada
// anterior a una de las funciones que devuelven una entrada de
// directorio

$valores = ldap_get_values($ds, $entrada,"email");

echo $valores["count"]." direcciones de email para esta entrada.<p>";

for ($i=0; $i < $valores["count"]; $i++)
    echo $valores[$i]."<br>";
```

ldap_list (PHP 3, PHP 4)

Búsqueda Single-level (Nivel Único)

int **ldap_list** (int identificador_de_conexion, string dn_base, string filtro [, array atributos]) \linebreak

Devuelve un identificador de resultado de búsqueda o falso en caso de error.

ldap_list() realiza la búsqueda según el filtro especificado en el directorio con el alcance LDAP_SCOPE_ONELEVEL.

LDAP_SCOPE_ONELEVEL significa que la búsqueda solo devuelve información que se encuentre en el nivel inmediatamente inferior al dn_base especificado en la llamada a la función. (Equivalente a ejecutar "ls" en un unix y obtener un listado de ficheros y carpetas en el directorio de trabajo actual.)

Esta llamada toma un cuarto parámetro opcional, que es un array de los atributos requeridos. Consulte las notas de la función ldap_search().

Ejemplo 1. Produce una lista de todas las unidades organizativas de una compañía

```
// $ds es un identificador de conexión válido.

$dnbase = "o=Mi Compañía, c=ES";
$solonecesito = array("ou");

$sr=ldap_list($ds, $dnbase, "ou=*", $solonecesito);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++)
    echo $info[$i]["ou"][0] ;
```

ldap_mod_add (PHP 3>= 3.0.8, PHP 4)

Añade valores de atributos

int **ldap_mod_add** (int identificador_de_conexion, string dn, array entrada) \linebreak

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

Esta función añadir uno o varios atributos al dn especificado. Realiza la modificación al nivel de atributos, en vez de hacerlo al nivel de objetos. Las modificaciones a nivel de objeto son propocionadas por la función ldap_add().

ldap_mod_del (PHP 3>= 3.0.8, PHP 4)

Borra valores de atributos

int **ldap_mod_del** (int identificador_de_conexion, string dn, array entrada) \linebreak

returns `TRUE` on success and `FALSE` on error.

Esta función elimina atributos del dn especificado. Realiza la modificación a nivel de atributos, en vez de hacerlo a nivel de objetos. Las modificaciones a nivel de objeto son proporcionadas por la función **ldap_del()**.

ldap_mod_replace (PHP 3>= 3.0.8, PHP 4)

Reemplaza valores de atributos

int **ldap_mod_replace** (int identificador_de_conexion, string dn, array entrada) \linebreak

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

Esta función reemplaza atributos del dn especificado. Realiza la modificación a nivel de atributos, en vez de hacerlo a nivel de objetos. Las modificaciones a nivel de objeto son proporcionadas por la función ldap_modify().

ldap_modify (PHP 3, PHP 4)

Modifica una entrada LDAP

int **ldap_modify** (int identificador_de_conexion, string dn, array entrada) \linebreak

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_modify()** se utiliza para modificar entradas existentes en un directorio LDAP. La estructura de la entrada es igual a la de ldap_add().

ldap_next_attribute (PHP 3, PHP 4)

Obtiene el siguiente atributo de una entrada

string **ldap_next_attribute** (int identificador_de_conexion, int identificador_de_entrada_de_resultado, int identificador_ber) \linebreak

Devuelve el siguiente atributo de una entrada o falso en caso de error.

ldap_next_attribute() es llamado para recuperar los atributos de una entrada. El estado interno del puntero es mantenido por el *identificador_ber*, que es pasado por referencia a la función. La primera llamada a **ldap_next_attribute()** es realizada con el *identificador_de_entrada_de_resultado* devuelto por la función ldap_first_attribute().

Ver también ldap_get_attributes()

ldap_next_entry (PHP 3, PHP 4)

Obtiene la siguiente entrada de un resultado

int **ldap_next_entry** (int identificador_de_conexion, int identificador_de_entrada_de_resultado) \linebreak

Devuelve el identificador de la siguiente entrada del resultado. Las entradas deben haber sido leidas al principio con ldap_first_entry(). Si no hay más entradas en el resultado devuelve falso.

La función **ldap_next_entry()** se utiliza para obtener las entradas almacenadas en un resultado. Llamadas sucesivas a la función **ldap_next_entry()** devuelven las entradas una a una hasta que ya no queden más entradas. La primera llamada a **ldap_next_entry()** se realiza después de llamar a ldap_first_entry().

Ver también ldap_get_entries()

ldap_next_reference (PHP 4 >= 4.0.5)

Get next reference

resource **ldap_next_reference** (resource link, resource entry) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ldap_parse_reference (PHP 4 >= 4.0.5)

Extract information from reference entry

bool **ldap_parse_reference** (resource link, resource entry, array referrals) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ldap_parse_result (PHP 4 >= 4.0.5)

Extract information from result

bool **ldap_parse_result** (resource link, resource result, int errcode, string matcheddn, string errmsg, array referrals) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ldap_read (PHP 3, PHP 4)

Lee una entrada

int **ldap_read** (int identificador_de_conexión, string dn_base, string filtro [, array atributos]) \linebreak

Devuelve un identificador de resultado de búsqueda o falso en caso de error.

ldap_read() realiza la búsqueda según el filtro especificado con alcance LDAP_SCOPE_BASE, por lo que es equivalente a leer cualquier entrada del directorio.

No se permiten filtros vacios. Si se pretende obtener absolutamente toda la información, se debe usar un filtro del tipo "objectClass=*". Si conoce que tipos de entradas son usadas en el servidor de directorio es conveniente usar el filtro apropiado, como por ejemplo "objectClass=inetOrgPerson".

Esta llamada toma un cuarto parámetro opcional que es un array de los atributos requeridos. Consulte las notas de la función ldap_search().

ldap_rename (PHP 4 >= 4.0.5)

Modify the name of an entry

bool **ldap_rename** (resource link_identifier, string dn, string newrdn, string newparent, bool deleteoldrdn) \linebreak

The entry specified by *dn* is renamed/moved. The new RDN is specified by *newrdn* and the new parent/superior entry is specified by *newparent*. If the parameter *deleteoldrdn* is TRUE the old RDN value(s) is removed, else the old RDN value(s) is retained as non-distinguished values of the entry. Devuelve TRUE si todo fue bien, FALSE en caso de fallo.

> **Nota:** This function currently only works with LDAPv3. You may have to use ldap_set_option() prior to binding to use LDAPv3. This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.5.

ldap_search (PHP 3, PHP 4)

Busca en un arbol LDAP

int **ldap_search** (int identificador_de_conexion, string dn_base, string filtro [, array atributos]) \linebreak

Devuelve un identificador de resultado de búsqueda o falso en caso de error.

ldap_search() realiza la búsqueda según el filtro especificado con alcance LDAP_SCOPE_SUBTREE. Esto es equivalente a buscar en el directorio entero. *dn_base* especifica el DN base para el directorio.

Existe un cuarto parámetro opcional que puede ser añadido para restringir los atributos y valores devueltos por el servidor a sólo los requeridos. Es mucho más eficiente que la acción por defecto (que devolverá todos los atributos y sus valores asociados). El uso del cuarto parámetro debe ser por tanto considerado una práctica recomendable.

El cuerto parámetro es un array estándar de PHP con los atributos requeridos, por ejemplo array("email","sn","cn"). Nota: "dn" siempre es devuelto independientemente de que tipos de atributos sean solicitados.

También es necesario resaltar que algunos servidores de directorio están configurados para devolver un cierto número de entradas como máximo. Si esto ocurre, el servidor indicará que solo devuelve un conjunto de resultados parcial.

El filtro de búsqueda puede ser simple o avanzado, usando operadores booleanos en el formato descrito en la documentación sobre LDAP (Consulte el Netscape Directory SDK (http://developer.netscape.com/tech/directory/) para obtener completa información sobre filtros).

El ejemplo de abajo recupera la unidad organizativa (ou), apellidos nombre común y dirección de email para todas las personas de "Mi Compañía" donde los apellidos o el nombre común contiene la subcadena $persona. Este ejemplo usa un filtro booleano para indicar al servidor que busque la información en más de un atributo.

Ejemplo 1. Búsqueda LDAP

```
// $ds es un identificador de conexión válido

// $persona es todo o parte del nombre de una persona, por ejemplo "Pe"

$dn = "o=Mi Compañía, c=ES";
$filtro="(|(sn=$persona*)(givenname=$persona*))";
$solonecesito = array( "ou", "sn", "givenname", "mail");

$sr=ldap_search($ds, $dn, $filtro, $solonecesito);

$info = ldap_get_entries($ds, $sr);

print $info["count"]." entradas devueltas<p>";
```

ldap_set_option (PHP 4 >= 4.0.4)

Set the value of the given option

bool **ldap_set_option** (resource link_identifier, int option, mixed newval) \linebreak

Sets the value of the specified option to be *newval*. Devuelve TRUE si todo fue bien, FALSE en caso de fallo. on error.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN, LDAP_OPT_SERVER_CONTROLS, LDAP_OPT_CLIENT_CONTROLS. Here's a brief description, see draft-ietf-ldapext-ldap-c-api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt) for details.

The options LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION and LDAP_OPT_ERROR_NUMBER have integer value, LDAP_OPT_REFERRALS and LDAP_OPT_RESTART have boolean value, and the options LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING and LDAP_OPT_MATCHED_DN have string value. The first example illustrates their use. The options LDAP_OPT_SERVER_CONTROLS and LDAP_OPT_CLIENT_CONTROLS require a list of controls, this means that the value must be an array of controls. A control consists of an *oid* identifying the control, an optional *value*, and an optional flag for *criticality*. In PHP a control is given by an array containing an element with the key *oid* and string value, and two optional elements. The optional elements are key *value* with string value and key *iscritical* with boolean value. *iscritical* defaults to FALSE if not supplied. See also the second example below.

> **Nota:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4.

Ejemplo 1. Set protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3))
    echo "Using LDAPv3";
else
    echo "Failed to set protocol version to 3";
```

Ejemplo 2. Set server controls

```
// $ds is a valid link identifier for a directory server
// control with no value
$ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => TRUE);
// iscritical defaults to FALSE
$ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// try to set both controls
if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1, $ctrl2)))
    echo "Failed to set server controls";
```

See also ldap_get_option().

ldap_set_rebind_proc (PHP 4 >= 4.2.0)

Set a callback function to do re-binds on referral chasing.

bool **ldap_set_rebind_proc** (resource link, string callback) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ldap_sort (PHP 4 >= 4.2.0)

Sort LDAP result entries

bool **ldap_sort** (resource link, resource result, string sortfilter) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ldap_start_tls (PHP 4 >= 4.2.0)

Start TLS

bool **ldap_start_tls** (resource link) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ldap_t61_to_8859 (PHP 4 >= 4.0.2)

Translate t61 characters to 8859 characters

string **ldap_t61_to_8859** (string value) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ldap_unbind (PHP 3, PHP 4)

Hace logout de un directorio LDAP

int **ldap_unbind** (int identificador_de_conexion) \linebreak

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_unbind()** hace logout, desautentifica de un directorio LDAP.

L. Funciones de Correo

The mail() Funciones que permiten enviar correo.

ezmlm_hash (PHP 3>= 3.0.17, PHP 4 >= 4.0.2)

Calculate the hash value needed by EZMLM

int **ezmlm_hash** (string addr) \linebreak

ezmlm_hash() calculates the hash value needed when keeping EZMLM mailing lists in a MySQL database.

Ejemplo 1. Calculating the hash and subscribing a user

```
$user = "joecool@example.com";
$hash = ezmlm_hash ($user);
$query = sprintf ("INSERT INTO sample VALUES (%s, '%s')", $hash, $user);
$db->query($query); // using PHPLIB db interface
```

mail (PHP 3, PHP 4)

Envía correo

bool **mail** (string para, string sobre, string mensaje [, string cabeceras_adicionales]) \linebreak

mail() envía automáticamente el mensaje especificado en *mensage* al destinatario especificado en *para*. Para especificar múltiples destinatarios se puede hacer incluyendo una coma entre las direcciones en *para*.

Ejemplo 1. Enviando correo.

```
mail("pepito@loquesea.es", "Sobre este tema", "Linea 1\nLinea 2\nLinea 3");
```

Si se añadiera una cadena como cuarto argumento, esta cadena sería enviada al final de la cabecera. Esto se usa normalmente para enviar cabeceras extra en los mensajes. Si se desea enviar varias cabeceras extra el mecanismo será el mismo separándolas una linea.

Ejemplo 2. Enviando correo con varias cabeceras.

```
mail("pepito@loquesea.es", "El tema", $message,
"From: webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-Mailer: PHP/" . phpv
```

LI. mailparse functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mailparse_determine_best_xfer_encoding (4.1.0 - 4.1.2 only)

Figures out the best way of encoding the content read from the file pointer fp, which must be seek-able

int **mailparse_determine_best_xfer_encoding** (resource fp) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_msg_create (4.1.0 - 4.1.2 only)

Returns a handle that can be used to parse a message

int **mailparse_msg_create** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_msg_extract_part_file (4.1.0 - 4.1.2 only)

Extracts/decodes a message section, decoding the transfer encoding

string **mailparse_msg_extract_part_file** (resource rfc2045, string filename [, string callbackfunc]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_msg_extract_part (4.1.0 - 4.1.2 only)

Extracts/decodes a message section. If callbackfunc is not specified, the contents will be sent to "stdout"

void **mailparse_msg_extract_part** (resource rfc2045, string msgbody [, string callbackfunc]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_msg_free (4.1.0 - 4.1.2 only)

Frees a handle allocated by mailparse_msg_crea

void **mailparse_msg_free** (resource rfc2045buf) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_msg_get_part_data (4.1.0 - 4.1.2 only)

Returns an associative array of info about the message

array **mailparse_msg_get_part_data** (resource rfc2045) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_msg_get_part (4.1.0 - 4.1.2 only)

Returns a handle on a given section in a mimemessage

int **mailparse_msg_get_part** (resource rfc2045, string mimesection) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_msg_get_structure (4.1.0 - 4.1.2 only)

Returns an array of mime section names in the supplied message

array **mailparse_msg_get_structure** (resource rfc2045) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_msg_parse_file (4.1.0 - 4.1.2 only)

Parse file and return a resource representing the structure

resource **mailparse_msg_parse_file** (string filename) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_msg_parse (4.1.0 - 4.1.2 only)

Incrementally parse data into buffer

void **mailparse_msg_parse** (resource rfc2045buf, string data) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_rfc822_parse_addresses (4.1.0 - 4.1.2 only)

Parse addresses and returns a hash containing that data

array **mailparse_rfc822_parse_addresses** (string addresses) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_stream_encode (4.1.0 - 4.1.2 only)

Streams data from source file pointer, apply encoding and write to destfp

bool **mailparse_stream_encode** (resource sourcefp, resource destfp, string encoding) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mailparse_uudecode_all (unknown)

Scans the data from fp and extract each embedded uuencoded file. Returns an array listing filename information

array **mailparse_uudecode_all** (resource fp) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

LII. Funciones matemáticas

Introducción

Estas funciones matemáticas solo manejan valores dentro de los rangos de los tipos long y double de su ordenador. Si necesita manejar números mayores, pege un vistazo a funciones matemáticas de precisión arbitraria.

Constantes matemáticas

Los siguientes valores estan definidos como constantes en PHP por la extensión matemática:

Tabla 1. Constantes matemáticas

Constante	Valor	Descripción
M_PI	3.14159265358979323846	El valor de ¶ (pi)

abs (PHP 3, PHP 4)

Valor absoluto

mixed **abs** (mixed number) \linebreak

Devuelve el valor absoluto de un número. Si el número del argumento es decimal, el tipo de retorno también es decimal, de otra manera devuelve un entero.

acos (PHP 3, PHP 4)

Arco coseno

float **acos** (float arg) \linebreak

Devuelve el arco coseno de arg en radianes.

Vea también asin() y atan().

acosh (PHP 4 >= 4.1.0)

Inverse hyperbolic cosine

float **acosh** (float arg) \linebreak

Returns the inverse hyperbolic cosine of *arg*, i.e. the value whose hyperbolic cosine is *arg*.

Nota: Esta función no está implementada en plataformas Windows.

See also acos(), asinh() and atanh().

asin (PHP 3, PHP 4)

Arco seno

float **asin** (float arg) \linebreak

Devuelve el arco seno de arg en radianes.

Vea también acos() y atan().

asinh (PHP 4 >= 4.1.0)

Inverse hyperbolic sine

float **asinh** (float arg) \linebreak

Returns the inverse hyperbolic sine of `arg`, i.e. the value whose hyperbolic sine is `arg`.

Nota: Esta función no está implementada en plataformas Windows.

See also asin(), acosh() and atanh().

atan2 (PHP 3>= 3.0.5, PHP 4)

Arco tangente de dos variables

float **atan2** (float y, float x) \linebreak

Esta función calcula la arco tangente de las dos variables x e y. Es similar a el cálculo de la arco tangente de y / x, excepto que los signos de ambos argumentos son usados para determinar el cuadrante del resultado

La función devuelve el resultado en radianes, el cual está entre -PI y PI (inclusive).

Vea también acos() y atan().

atan (PHP 3, PHP 4)

Arco tangente

float **atan** (float arg) \linebreak

Devuelve la arco tangente de arg en radianes.

Vea también acos() y **atan()**.

atanh (PHP 4 >= 4.1.0)

Inverse hyperbolic tangent

float **atanh** (float arg) \linebreak

Returns the inverse hyperbolic tangent of `arg`, i.e. the value whose hyperbolic tangent is `arg`.

Nota: Esta función no está implementada en plataformas Windows.

See also atan(), asinh() and acosh().

base_convert (PHP 3>= 3.0.6, PHP 4)

Convierte un número entre bases arbitrarias

strin **base_convert** (string number, int frombase, int tobase) \linebreak

Devuelve una cadena conteniendo el *number* representado en base *tobase*. La base en la cual *number* es dado viene especificada en *frombase*. Tanto *frombase* y *tobase* tienen que estar entre 2 y 36, inclusive. Los dígitos en números con una base mayor que 10 serán representados con las letras a-z, a vale 10, b vale 11 y z vale 36.

Ejemplo 1. base_convert()

```
$binary = base_convert($hexadecimal, 16, 2);
```

BinDec (PHP 3, PHP 4)

binario a decimal

int **bindec** (string binary_string) \linebreak

Devuelve el equivalente decimal del número binario representado por el argumento binary_string.

BinDec convierte un número binario en un número decimal. El mayor número que puede ser convertido son 31 bits de unos o 2147483647 en decimal.

Vea también la función decbin() .

ceil (PHP 3, PHP 4)

Redondea fracciones hacia arriba

int **ceil** (float number) \linebreak

Devuelve el valor entero superior más cercano a *number*. El uso de **ceil()** con enteros es una perdida de tiempo absoluta.

NOTA: PHP/FI 2's **ceil()** devuelve un float. Use: `$new = (double)ceil($number);` para tener el comportamiento antiguo.

Vea también floor() y round().

cos (PHP 3, PHP 4)

coseno

float **cos** (float arg) \linebreak

Devuelve el coseno de arg en radianes.

Vea también sin() y tan().

cosh (PHP 4 >= 4.1.0)

Hyperbolic cosine

float **cosh** (float arg) \linebreak

Returns the hyperbolic cosine of *arg*, defined as `(exp(arg) + exp(-arg))/2`.

See also cos(), acosh(), sin() and tan().

DecBin (PHP 3, PHP 4)

decimal a binario

string **decbin** (int number) \linebreak

Devuelve una cadena conteniendo la representación en binario de el número dado en el argumento. El número mayor que puede ser convertido es 2147483647 en decimal que resulta una cadena de 31 unos.

Vea también la función bindec() .

DecHex (PHP 3, PHP 4)

decimal a hexadecimal

string **dechex** (int number) \linebreak

Devuelve una cadena conteniendo la representación hexadecimal del número dado en el argumento. El mayor número que puede ser convertido es 2147483647 en decimal resultando "7fffffff".

Vea también la función hexdec() .

DecOct (PHP 3, PHP 4)

decimal a octal

string **decoct** (int number) \linebreak

Devuelve una cadena conteniendo la representación octal del número dado en el argumento. Returns a string containing an octal representation of the given number argument. El mayor número que puede ser ocnvertido es 2147483647 en decimal resultando "17777777777". Vea también octdec().

deg2rad (PHP 3>= 3.0.4, PHP 4)

Converts the number in degrees to the radian equivalent

float **deg2rad** (float number) \linebreak

This function converts `number` from degrees to the radian equivalent.

See also rad2deg().

exp (PHP 3, PHP 4)

e elevado a...

float **exp** (float arg) \linebreak

Devuelve el número e elevado a la potencia de `arg`.

Vea también pow().

expm1 (PHP 4 >= 4.1.0)

Returns exp(number) - 1, computed in a way that accurate even when the value of number is close to zero

float **expm1** (float number) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

floor (PHP 3, PHP 4)

redondea fracciones hacia abajo

int **floor** (float number) \linebreak

Devuelve el valor entero inferior más cercano a *number*. El uso de **floor()** con enteros es una perdida de tiempo absoluta.

NOTA: PHP/FI 2's **floor()** devuelve un float. Use: `$new = (double)floor($number);` para tener el comportamiento antiguo.

Vea también ceil() y round().

getrandmax (PHP 3, PHP 4)

Muestra el mayor valor aleatorio posible

int **getrandmax** (void) \linebreak

Devuelve el valor máximo que puede devolver una llamada a rand().

Vea también rand(), srand() mt_rand(), mt_srand() y mt_getrandmax().

HexDec (PHP 3, PHP 4)

hexadecimal a decimal

int **hexdec** (string hex_string) \linebreak

Devuelve el equivalente decimal de el número hexadecimal representado por el argumento hex_string. HexDec convierte una cadena hexadecimal en un número decimal. El número mayor que puede ser convertido es 7fffffff o 2147483647 en decimal.

Vea también la función dechex() .

hypot (PHP 4 >= 4.1.0)

Returns sqrt(num1*num1 + num2*num2)

float **hypot** (float num1, float num2) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

is_finite (PHP 4 >= 4.2.0)

bool **is_finite** (float val) \linebreak

Returns `TRUE` if *`val`* is a legal finite number within the allowed range for a PHP float on this platform.

is_infinite (PHP 4 >= 4.2.0)

bool **is_infinite** (float val) \linebreak

Returns `TRUE` if *`val`* is infinite (positive or negative), like the result of `log(0)` or any value too big to fit into a float on this platform.

is_nan (PHP 4 >= 4.2.0)

bool **is_nan** (float val) \linebreak

Returns `TRUE` if *`val`* is 'not a number', like the result of `acos(1.01)`.

lcg_value (PHP 4)

Combined linear congruential generator

float **lcg_value** (void) \linebreak

lcg_value() returns a pseudo random number in the range of (0, 1). The function combines two CGs with periods of 2^31 - 85 and 2^31 - 249. The period of this function is equal to the product of both primes.

log10 (PHP 3, PHP 4)

Logaritmo en base-10

float **log10** (float arg) \linebreak

Devuelve el logaritmo en base-10 de arg.

log1p (PHP 4 >= 4.1.0)

Returns log(1 + number), computed in a way that accurate even when the val ue of number is close to zero

float **log1p** (float number) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

log (PHP 3, PHP 4)

Logaritmo natural

float **log** (float arg) \linebreak

Devuelve el logaritmo de arg.

max (PHP 3, PHP 4)

encuentra el valor mayor

mixed **max** (mixed arg1, mixed arg2, mixed argn) \linebreak

max() devuelve el número mayor de los valores de los parámetros.

Si el primer parámetro es un array, **max()** devuelve el mayor valor en ese array. Si el primer parámetro es un entero, string o double, necesita al menos dos parámetros y **max()** devuelve el mayor número de estos valores. Puede comparar un número ilimitado de valores.

Si uno o más de los valores es un double, todos los valores serán tratados como doubles, y devolverá un double. Si ninguno de los valores es un double, todos ellos serán tratados como enteros, y se devolverá un entero.

min (PHP 3, PHP 4)

encuentra el valor menor

mixed **min** (mixed arg1, mixed arg2, mixed argn) \linebreak

min() returns the numerically lowest of the parameter values.

Si el primer parámetro es un array, **min()** devuelve el menor valor en ese array. Si el primer parámetro es un entero, string o double, necesita al menos dos parámetros y **min()** devuelve el número menor de estos valores. Puede comparar un número ilimitado de valores.

Si uno o más de los valores es un double, todos los valores serán tratados como doubles, y devolverá un double. Si ninguno de los valores es un double, todos ellos serán tratados como enteros, y se devolverá un entero.

mt_getrandmax (PHP 3>= 3.0.6, PHP 4)

muestra el mayor valor aleatorio posible

int **mt_getrandmax** (void) \linebreak

Devuelve el valor máximo que se puede obtener con una llamada a mt_rand().

Vea también mt_rand(), mt_srand() rand(), srand() y getrandmax().

mt_rand (PHP 3>= 3.0.6, PHP 4)

genera un valor aleatorio mejorado

int **mt_rand** ([int min [, int max]]) \linebreak

Muchos generadores de números aleatorios de libcs antiguas tienen características dudosas o desconocidas y son lentas. Por defecto, PHP usa el generador de números aleatorios de libc con la función rand(). La función **mt_rand()** es un reemplazo para esta. Usa un generador de números aleatorios con características conocidas, el Tornado de Mersenne, que es capaz de producir números aleatorios que incluso se pueden emplear para propósitos criptográficos y es cuatro veces más rápido que la media de los que provee libc. La página principal del Tornado de Mersenne puede encontrarse en http://www.math.keio.ac.jp/~matumoto/emt.html, y una versión optimizada del código del TM esta disponible en http://www.scp.syr.edu/~marc/hawk/twister.html.

Si es llamada sin los parámetros opcionales min y max **mt_rand()** devuelve un valor pseudo-aleatorio entre 0 y RAND_MAX. Si quiere un número aleatorio entre 5 y 15 (inclusive), use mt_rand(5,15).

Recuerde introducir la semilla del generador de números aleatorios antes de usar la instrucción con mt_srand().

Vea también mt_srand(), mt_getrandmax(), srand(), rand() y getrandmax().

mt_srand (PHP 3>= 3.0.6, PHP 4)

Introduce la semilla del generador de números aleatorios mejorado

void **mt_srand** (int seed) \linebreak

Introduce la semilla *seed* en el generador de números aleatorios mejorado.

```
// seed son los microsegundos desde el último segundo "entero"
mt_srand((double)microtime()*1000000);
$randval = mt_rand();
```

Vea también mt_rand(), mt_getrandmax(), srand(), rand() y getrandmax().

number_format (PHP 3, PHP 4)

formatea un número en grupos de miles

string **number_format** (float number, int decimals, string dec_point, string thousands_sep) \linebreak

number_format() devuelve la versión formateada de *number*. Esta función acepta tanto uno, como dos o cuatro parámetros (tres no):

Si sólo se da un parámetro, *number* será formateado sin decimales, pero con una coma (",") entre cada grupo de miles.

Si se dan dos parámetros, *number* será formateado con *decimals* decimales con un punto (".") al principio, y una coma (",") entre cada grupo de miles.

Si se dan cuatro parámetros, *number* será formateado con *decimals* decimales, *dec_point* en vez del punto (".") antes de los decimales y *thousands_sep* en vez de la coma (",") entre cada grupo de miles.

OctDec (PHP 3, PHP 4)

octal a decimal

int **octdec** (string octal_string) \linebreak

Devuelve el equivalente decimal del número octal representado por el argumento octal_string. OctDec convierte una cadena octal en un número decimal. El mayor número que puede ser convertido es 17777777777 o 2147483647 en decimal.

Vea también decoct().

pi (PHP 3, PHP 4)

devuelve el valor de pi

double **pi** (void) \linebreak

Devuelve una aproximación de pi.

pow (PHP 3, PHP 4)

expresión exponencial

float **pow** (float base, float exp) \linebreak

Devuelve base elevado a la potencia de exp.

Vea también exp().

rad2deg (PHP 3>= 3.0.4, PHP 4)

Converts the radian number to the equivalent number in degrees

float **rad2deg** (float number) \linebreak

This function converts *number* from radian to degrees.

See also deg2rad().

rand (PHP 3, PHP 4)

genera un valor aleatorio

int **rand** ([int min [, int max]]) \linebreak

Si es llamada sin los argumentos opcionales min y max, rand() devuelve un valor pseudo-aleatorio entre 0 y RAND_MAX. Si quiere un número aleatorio entre 5 y 15 (inclusive), por ejemplo, use rand(5,15).

Recuerde introducir la semilla del generador de números aleatorios antes de usar srand().

Vea también srand(), getrandmax(), mt_rand(), mt_srand() y mt_getrandmax().

round (PHP 3, PHP 4)

Redondea un float.

double **round** (double val) \linebreak

Devuelve el valor redondeado de *val*.

```
$foo = round( 3.4 );   // $foo == 3.0
$foo = round( 3.5 );   // $foo == 4.0
$foo = round( 3.6 );   // $foo == 4.0
```

Vea también ceil() y floor().

sin (PHP 3, PHP 4)

seno

float **sin** (float arg) \linebreak

Devuelve el seno de arg en radianes.

Vea también cos() y tan().

sinh (PHP 4 >= 4.1.0)

Hyperbolic sine

float **sinh** (float arg) \linebreak

Returns the hyperbolic sine of *arg*, defined as `(exp(arg) - exp(-arg))/2`.

See also sin(), asinh(), cos() and tan().

sqrt (PHP 3, PHP 4)

Raíz cuadrada

float **sqrt** (float arg) \linebreak

Devuelve la raíz cuadrada de arg.

srand (PHP 3, PHP 4)

introduce la semilla del generador de números aleatorios

void **srand** (int seed) \linebreak

Inicializa el generador de números aleatorios con *seed*.

```
// seed son los microsegundos desde el último segundo "entero"
srand((double)microtime()*1000000);
$randval = rand();
```

Vea también rand(), getrandmax(), mt_rand(), mt_srand() y mt_getrandmax().

tan (PHP 3, PHP 4)

tangente

float **tan** (float arg) \linebreak

Devuelve la tangente de arg en radianes.

Vea también sin() y cos().

tanh (PHP 4 >= 4.1.0)

Hyperbolic tangent

float **tanh** (float arg) \linebreak

Returns the hyperbolic tangent of *arg*, defined as `sinh(arg)/cosh(arg)`.

See also tan(), atanh(), sin() and cos().

LIII. Multi-Byte String Functions

Introduction

There are many languages in which all characters can be expressed by single byte. Multi-byte character codes are used to express many characters for many languages. `mbstring` is developed to handle Japanese characters. However, many `mbstring` functions are able to handle character encoding other than Japanese.

A multi-byte character encoding represents single character with consecutive bytes. Some character encoding has shift(escape) sequences to start/end multi-byte character strings. Therefore, a multi-byte character string may be destroyed when it is divided and/or counted unless multi-byte character encoding safe method is used. This module provides multi-byte character safe string functions and other utility functions such as conversion functions.

Since PHP is basically designed for ISO-8859-1, some multi-byte character encoding does not work well with PHP. Therefore, it is important to set `mbstring.internal_encoding` to a character encoding that works with PHP.

PHP4 Character Encoding Requirements

- Per byte encoding
- Single byte characters in range of `00h-7fh` which is compatible with `ASCII`
- Multi-byte characters without `00h-7fh`

These are examples of internal character encoding that works with PHP and does NOT work with PHP.

```
Character encodings work with PHP:
ISO-8859-*, EUC-JP, UTF-8

Character encodings do NOT work with PHP:
JIS, SJIS
```

Character encoding, that does not work with PHP, may be converted with `mbstring`'s HTTP input/output conversion feature/function.

> **Nota:** SJIS should not be used for internal encoding unless the reader is familiar with parser/compiler, character encoding and character encoding issues.

> **Nota:** If you use database with PHP, it is recommended that you use the same character encoding

for both database and `internal encoding` for ease of use and better performance.

If you are using PostgreSQL, it supports character encoding that is different from backend character encoding. See the PostgreSQL manual for details.

How to Enable mbstring

`mbstring` is an extended module. You must enable module with `configure` script. Refer to the Install section for details.

The following configure options are related to `mbstring` module.

- `--enable-mbstring` : Enable `mbstring` functions. This option is required to use `mbstring` functions.
- `--enable-mbstr-enc-trans` : Enable HTTP input character encoding conversion using `mbstring` conversion engine. If this feature is enabled, HTTP input character encoding may be converted to `mbstring.internal_encoding` automatically.

HTTP Input and Output

HTTP input/output character encoding conversion may convert binary data also. Users are supposed to control character encoding conversion if binary data is used for HTTP input/output.

If `enctype` for HTML form is set to `multipart/form-data`, `mbstring` does not convert character encoding in POST data. If it is the case, strings are needed to be converted to internal character encoding.

- HTTP Input

  There is no way to control HTTP input character conversion from PHP script. To disable HTTP input character conversion, it has to be done in `php.ini`.

  Ejemplo 1. Disable HTTP input conversion in `php.ini`

```
;; Disable HTTP Input conversion
mbstring.http_input = pass
```

  When using PHP as an Apache module, it is possible to override PHP ini setting per Virtual Host in `httpd.conf` or per directory with `.htaccess`. Refer to the Configuration section and Apache

Manual for details.

- HTTP Output

  There are several ways to enable output character encoding conversion. One is using `php.ini`, another is using ob_start() with mb_output_handler() as `ob_start` callback function.

  > **Nota:** For PHP3-i18n users, `mbstring`'s output conversion differs from PHP3-i18n. Character encoding is converted using output buffer.

Ejemplo 2. `php.ini` setting example

```
;; Enable output character encoding conversion for all PHP pages

;; Enable Output Buffering
output_buffering    = On

;; Set mb_output_handler to enable output conversion
output_handler      = mb_output_handler
```

Ejemplo 3. Script example

```
<?php

// Enable output character encoding conversion only for this page

// Set HTTP output character encoding to SJIS
mb_http_output('SJIS');

// Start buffering and specify "mb_output_handler" as
// callback function
ob_start('mb_output_handler');

?>
```

Supported Character Encoding

Currently, the following character encoding is supported by `mbstring` module. Caracter encoding may be specified for `mbstring` functions' `encoding` parameter.

The following character encoding is supported in this PHP extension :

`UCS-4`, `UCS-4BE`, `UCS-4LE`, `UCS-2`, `UCS-2BE`, `UCS-2LE`, `UTF-32`, `UTF-32BE`, `UTF-32LE`, `UCS-2LE`, `UTF-16`, `UTF-16BE`, `UTF-16LE`, `UTF-8`, `UTF-7`, `ASCII`, `EUC-JP`, `SJIS`, `eucJP-win`, `SJIS-win`, `ISO-2022-JP`, `JIS`, `ISO-8859-1`, `ISO-8859-2`, `ISO-8859-3`, `ISO-8859-4`, `ISO-8859-5`, `ISO-8859-6`, `ISO-8859-7`, `ISO-8859-8`, `ISO-8859-9`, `ISO-8859-10`, `ISO-8859-13`, `ISO-8859-14`, `ISO-8859-15`, `byte2be`, `byte2le`, `byte4be`, `byte4le`, `BASE64`, `7bit`, `8bit` and `UTF7-IMAP`.

`php.ini` entry, which accepts encoding name, accepts "`auto`" and "`pass`" also. `mbstring` functions, which accepts encoding name, and accepts "`auto`".

If "`pass`" is set, no character encoding conversion is performed.

If "`auto`" is set, it is expanded to "`ASCII,JIS,UTF-8,EUC-JP,SJIS`".

See also mb_detect_order()

> **Nota:** "Supported character encoding" does not mean that it works as internal character code.

`php.ini` settings

- `mbstring.internal_encoding` defines default internal character encoding.
- `mbstring.http_input` defines default HTTP input character encoding.
- `mbstring.http_output` defines default HTTP output character encoding.
- `mbstring.detect_order` defines default character code detection order. See also mb_detect_order().
- `mbstring.substitute_character` defines character to substitute for invalid character encoding.

Web Browsers are supposed to use the same character encoding when submitting form. However, browsers may not use the same character encoding. See mb_http_input() to detect character encoding

used by browsers.

If `enctype` is set to `multipart/form-data` in HTML forms, `mbstring` does not convert character encoding in POST data. The user must convert them in the script, if conversion is needed.

Although, browsers are smart enough to detect character encoding in HTML. `charset` is better to be set in HTTP header. Change `default_charset` according to character encoding.

Ejemplo 4. `php.ini` setting example

```
;; Set default internal encoding
;; Note: Make sure to use character encoding works with PHP
mbstring.internal_encoding    = UTF-8  ; Set internal encoding to UTF-8

;; Set default HTTP input character encoding
;; Note: Script cannot change http_input setting.
mbstring.http_input           = pass    ; No conversion.
mbstring.http_input           = auto    ; Set HTTP input to auto
                                ; "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS"
mbstring.http_input           = SJIS    ; Set HTTP2 input to  SJIS
mbstring.http_input           = UTF-8,SJIS,EUC-JP ; Specify order

;; Set default HTTP output character encoding
mbstring.http_output          = pass    ; No conversion
mbstring.http_output          = UTF-8   ; Set HTTP output encoding to UTF-8

;; Set default character encoding detection order
mbstring.detect_order         = auto    ; Set detect order to auto
mbstring.detect_order         = ASCII,JIS,UTF-8,SJIS,EUC-JP ; Specify order

;; Set default substitute character
mbstring.substitute_character = 12307   ; Specify Unicode value
mbstring.substitute_character = none    ; Do not print character
mbstring.substitute_character = long    ; Long Example: U+3000,JIS+7E7E
```

Ejemplo 5. `php.ini` setting for `EUC-JP` users

```
;; Disable Output Buffering
output_buffering      = Off

;; Set HTTP header charset
default_charset       = EUC-JP

;; Set HTTP input encoding conversion to auto
mbstring.http_input   = auto
```

```
;; Convert HTTP output to EUC-JP
mbstring.http_output  = EUC-JP

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Ejemplo 6. `php.ini` setting for `SJIS` users

```
;; Enable Output Buffering
output_buffering     = On

;; Set mb_output_handler to enable output conversion
output_handler       = mb_output_handler

;; Set HTTP header charset
default_charset      = Shift_JIS

;; Set http input encoding conversion to auto
mbstring.http_input  = auto

;; Convert to SJIS
mbstring.http_output = SJIS

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Overload of PHP string functions by mbstring functions with multibyte support

Because almost PHP application written for language using single-byte character encoding, there are some difficulties for multibyte string handling including japanese. Almost PHP string functions such as

substr() do not support multibyte string.

Multibyte extension (mbstring) has some PHP string functions with multibyte support (ex. substr() supports mb_substr()).

Multibyte extension (mbstring) also supports 'function overloading' to add multibyte string functionality without code modification. Using function overloading, some PHP string functions will be oveloaded multibyte string functions. For example, mb_substr() is called instead of substr() if function overloading is enabled. Function overload makes easy to port application supporting only single-byte encoding for multibyte application.

`mbstring.func_overload` in `php.ini` should be set some positive value to use function overloading. The value should specify the category of overloading functions, sbould be set 1 to enable mail function overloading. 2 to enable string functions, 4 to regular expression functions. For example, if is set for 7, mail, strings, regex functions should be overloaded. The list of overloaded functions are shown in below.

Tabla 1. Functions to be overloaded

value of mbstring.func_overload	original function	overloaded function
1	mail()	mb_send_mail()
2	strlen()	mb_strlen()
2	strpos()	mb_strpos()
2	strrpos()	mb_strrpos()
2	substr()	mb_substr()
4	ereg()	mb_ereg()
4	eregi()	mb_eregi()
4	ereg_replace()	mb_ereg_replace()
4	eregi_replace()	mb_eregi_replace()
4	split()	mb_split()

Basics for Japanese multi-byte character

Most Japanese characters need more than 1 byte per character. In addition, several character encoding schemas are used under a Japanese environment. There are EUC-JP, Shift_JIS(SJIS) and ISO-2022-JP(JIS) character encoding. As Unicode becomes popular, UTF-8 is used also. To develop Web applications for a Japanese environment, it is important to use the character set for the task in hand, whether HTTP input/output, RDBMS and E-mail.

- Storage for a character can be up to six bytes
- A multi-byte character is usually twice of the width compared to single-byte characters. Wider characters are called "zen-kaku" - meaning full width, narrower characters are called "han-kaku" - meaning half width. "zen-kaku" characters are usually fixed width.
- Some character encoding defines shift(escape) sequence for entering/exiting multi-byte character

strings.

- ISO-2022-JP must be used for SMTP/NNTP.
- "i-mode" web site is supposed to use SJIS.

References

Multi-byte character encoding and its related issues are very complex. It is impossible to cover in sufficient detail here. Please refer to the following URLs and other resources for further readings.

- Unicode/UTF/UCS/etc

  `http://www.unicode.org/`

- Japanese/Korean/Chinese character information

  `ftp://ftp.ora.com/pub/examples/nutshell/ujip/doc/cjk.inf`

mb_convert_encoding (PHP 4 >= 4.0.6)

Convert character encoding

string **mb_convert_encoding** (string str, string to-encoding [, mixed from-encoding]) \linebreak

mb_convert_encoding() converts character encoding of string *`str`* from *`from-encoding`* to *`to-encoding`*.

`str` : String to be converted.

`from-encoding` is specified by character code name before conversion. it can be array or string - comma separated enumerated list. If it is not specified, the internal encoding will be used.

Ejemplo 1. mb_convert_encoding() example

```
/* Convert internal character encoding to SJIS */
$str = mb_convert_encoding($str, "SJIS");

/* Convert EUC-JP to UTF-7 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");

/* Auto detect encoding from JIS, eucjp-win, sjis-win, then convert str to UCS-
2LE */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
$str = mb_convert_encoding($str, "EUC-JP", "auto");
```

See also: mb_detect_order().

mb_convert_kana (PHP 4 >= 4.0.6)

Convert "kana" one from another ("zen-kaku" ,"han-kaku" and more)

string **mb_convert_kana** (string str, string option [, mixed encoding]) \linebreak

mb_convert_kana() performs "han-kaku" - "zen-kaku" conversion for string *`str`*. It returns converted string. This function is only useful for Japanese.

`option` is conversion option. Default value is `"KV"`.

`encoding` is character encoding. If it is omitted, internal character encoding is used.

```
Applicable Conversion Options
```

```
option : Specify with conversion of following options. Default "KV"
"r" :  Convert "zen-kaku" alphabets to "han-kaku"
"R" :  Convert "han-kaku" alphabets to "zen-kaku"
"n" :  Convert "zen-kaku" numbers to "han-kaku"
"N" :  Convert "han-kaku" numbers to "zen-kaku"
"a" :  Convert "zen-kaku" alphabets and numbers to "han-kaku"
"A" :  Convert "zen-kaku" alphabets and numbers to "han-kaku"
(Characters included in "a", "A" options are
U+0021 - U+007E excluding U+0022, U+0027, U+005C, U+007E)
"s" :  Convert "zen-kaku" space to "han-kaku" (U+3000 -> U+0020)
"S" :  Convert "han-kaku" space to "zen-kaku" (U+0020 -> U+3000)
"k" :  Convert "zen-kaku kata-kana" to "han-kaku kata-kana"
"K" :  Convert "han-kaku kata-kana" to "zen-kaku kata-kana"
"h" :  Convert "zen-kaku hira-gana" to "han-kaku kata-kana"
"H" :  Convert "han-kaku kata-kana" to "zen-kaku hira-gana"
"c" :  Convert "zen-kaku kata-kana" to "zen-kaku hira-gana"
"C" :  Convert "zen-kaku hira-gana" to "zen-kaku kata-kana"
"V" :  Collapse voiced sound notation and convert them into a character. Use with "K'
```

Ejemplo 1. mb_convert_kana() example

```
/* Convert all "kana" to "zen-kaku" "kata-kana" */
$str = mb_convert_kana($str, "KVC");

/* Convert "han-kaku" "kata-kana" to "zen-kaku" "kata-kana"
   and "zen-kaku" alpha-numeric to "han-kaku" */
$str = mb_convert_kana($str, "KVa");
```

mb_convert_variables (PHP 4 >= 4.0.6)

Convert character code in variable(s)

string **mb_convert_variables** (string to-encoding, mixed from-encoding, mixed vars) \linebreak

mb_convert_variables() convert character encoding of variables *`vars`* in encoding *`from-encoding`* to encoding *`to-encoding`*. It returns character encoding before conversion for success, `FALSE` for failure.

mb_convert_variables() join strings in Array or Object to detect encoding, since encoding detection tends to fail for short strings. Therefore, it is impossible to mix encoding in single array or object.

It *from-encoding* is specified by array or comma separated string, it tries to detect encoding from *from-coding*. When *encoding* is omitted, detect_order is used.

vars (3rd and larger) is reference to variable to be converted. String, Array and Object are accepted. **mb_convert_variables()** assumes all parameters have the same encoding.

Ejemplo 1. mb_convert_variables() example

```
/* Convert variables $post1, $post2 to internal encoding */
$interenc = mb_internal_encoding();
$inputenc = mb_convert_variables($interenc, "ASCII,UTF-8,SJIS-win", $post1, $post2);
```

mb_decode_mimeheader (PHP 4 >= 4.0.6)

Decode string in MIME header field

string **mb_decode_mimeheader** (string str) \linebreak

mb_decode_mimeheader() decodes encoded-word string *str* in MIME header.

It returns decoded string in internal character encoding.

See also mb_encode_mimeheader().

mb_decode_numericentity (PHP 4 >= 4.0.6)

Decode HTML numeric string reference to character

string **mb_decode_numericentity** (string str, array convmap [, string encoding]) \linebreak

Convert numeric string reference of string *str* in specified block to character. It returns converted string.

array is array to specifies code area to convert.

encoding is character encoding. If it is omitted, internal character encoding is used.

Ejemplo 1. *convmap* example

```
$convmap = array (
   int start_code1, int end_code1, int offset1, int mask1,
   int start_code2, int end_code2, int offset2, int mask2,
   ........
   int start_codeN, int end_codeN, int offsetN, int maskN );
```

```
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN,
// then convert value to numeric string reference.
```

See also: mb_encode_numericentity().

mb_detect_encoding (PHP 4 >= 4.0.6)

Detect character encoding

string **mb_detect_encoding** (string str [, mixed encoding-list]) \linebreak

mb_detect_encoding() detects character encoding in string *`str`*. It returns detected character encoding.

`encoding-list` is list of character encoding. Encoding order may be specified by array or comma separated list string.

If *`encoding_list`* is omitted, detect_order is used.

Ejemplo 1. mb_detect_encoding() example

```
/* Detect character encoding with current detect_order */
echo mb_detect_encoding($str);

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
echo mb_detect_encoding($str, "auto");

/* Specify encoding_list character encoding by comma separated list */
echo mb_detect_encoding($str, "JIS, eucjp-win, sjis-win");

/* Use array to specify encoding_list  */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
echo mb_detect_encoding($str, $ary);
```

See also: mb_detect_order().

mb_detect_order (PHP 4 >= 4.0.6)

Set/Get character encoding detection order

array **mb_detect_order** ([mixed encoding-list]) \linebreak

mb_detect_order() sets automatic character encoding detection order to *encoding-list*. It returns TRUE for success, FALSE for failure.

encoding-list is array or comma separated list of character encoding. ("auto" is expanded to "ASCII, JIS, UTF-8, EUC-JP, SJIS")

If *encoding-list* is omitted, it returns current character encoding detection order as array.

This setting affects mb_detect_encoding() and mb_send_mail().

Nota: mbstring currently implements following encoding detection filters. If there is a invalid byte sequence for following encoding, encoding detection will fail.

Nota: UTF-8, UTF-7, ASCII, EUC-JP,SJIS, eucJP-win, SJIS-win, JIS, ISO-2022-JP

For ISO-8859-*, mbstring always detects as ISO-8859-*.

For UTF-16, UTF-32, UCS2 and UCS4, encoding detection will fail always.

Ejemplo 1. Useless detect order example

```
; Always detect as ISO-8859-1
detect_order = ISO-8859-1, UTF-8

; Always detect as UTF-8, since ASCII/UTF-7 values are
; valid for UTF-8
detect_order = UTF-8, ASCII, UTF-7
```

Ejemplo 2. mb_detect_order() examples

```
/* Set detection order by enumerated list */
mb_detect_order("eucjp-win,sjis-win,UTF-8");

/* Set detection order by array */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
mb_detect_order($ary);

/* Display current detection order */
echo implode(", ", mb_detect_order());
```

See also mb_internal_encoding(), mb_http_input(), mb_http_output() mb_send_mail().

mb_encode_mimeheader (PHP 4 >= 4.0.6)

Encode string for MIME header

string **mb_encode_mimeheader** (string str [, string charset [, string transfer-encoding [, string linefeed]]]) \linebreak

mb_encode_mimeheader() converts string *`str`* to encoded-word for header field. It returns converted string in ASCII encoding.

`charset` is character encoding name. Default is `ISO-2022-JP`.

`transfer-encoding` is transfer encoding. It should be one of `"B"` (Base64) or `"Q"` (Quoted-Printable). Default is `"B"`.

`linefeed` is end of line marker. Default is `"\r\n"` (CRLF).

Ejemplo 1. mb_convert_kana() example

```
$name = ""; // kanji
$mbox = "kru";
$doma = "gtinn.mon";
$addr = mb_encode_mimeheader($name, "UTF-7", "Q") . " <" . $mbox . "@" . $doma . ">";
echo $addr;
```

See also mb_decode_mimeheader().

mb_encode_numericentity (PHP 4 >= 4.0.6)

Encode character to HTML numeric string reference

string **mb_encode_numericentity** (string str, array convmap [, string encoding]) \linebreak

mb_encode_numericentity() converts specified character codes in string *`str`* from HTML numeric character reference to character code. It returns converted string.

`array` is array specifies code area to convert.

encoding is character encoding.

Ejemplo 1. *convmap* example

```
$convmap = array (
 int start_code1, int end_code1, int offset1, int mask1,
 int start_code2, int end_code2, int offset2, int mask2,
 ........
 int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN, then
// it converts value to numeric string reference.
```

Ejemplo 2. mb_encode_numericentity() example

```
/* Convert Left side of ISO-8859-1 to HTML numeric character reference */
$convmap = array(0x80, 0xff, 0, 0xff);
$str = mb_encode_numericentity($str, $convmap, "ISO-8859-1");

/* Convert user defined SJIS-win code in block 95-104 to numeric
   string reference */
$convmap = array(
       0xe000, 0xe03e, 0x1040, 0xffff,
       0xe03f, 0xe0bb, 0x1041, 0xffff,
       0xe0bc, 0xe0fa, 0x1084, 0xffff,
       0xe0fb, 0xe177, 0x1085, 0xffff,
       0xe178, 0xe1b6, 0x10c8, 0xffff,
       0xe1b7, 0xe233, 0x10c9, 0xffff,
       0xe234, 0xe272, 0x110c, 0xffff,
       0xe273, 0xe2ef, 0x110d, 0xffff,
       0xe2f0, 0xe32e, 0x1150, 0xffff,
       0xe32f, 0xe3ab, 0x1151, 0xffff );
$str = mb_encode_numericentity($str, $convmap, "sjis-win");
```

See also: mb_decode_numericentity().

mb_ereg_match (PHP 4 >= 4.2.0)

Regular expression match for multibyte string

bool **mb_ereg_match** (string pattern, string string [, string option]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_match() returns `TRUE` if *`string`* matches regular expression *`pattern`*, `FALSE` if not.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_ereg_replace (PHP 4 >= 4.2.0)

Replace regular expression with multibyte support

string **mb_ereg_replace** (string pattern, string replacement, string string [, array option]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_replace() scans *`string`* for matches to *`pattern`*, then replaces the matched text with *`replacement`* and returns the result string or `FALSE` on error. Multibyte character can be used in *`pattern`*.

Matching condition can be set by *`option`* parameter. If `i` is specified for this parameter, the case will be ignored. If `x` is specified, white space will be ignored. If `m` is specified, match will be executed in multiline mode and line break will be included in '.'. If `p` is specified, match will be executed in POSIX mode, line break will be considered as normal character. If `e` is specified, *`replacement`* string will be evaluated as PHP expression.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_eregi_replace().

mb_ereg_search_getpos (PHP 4 >= 4.2.0)

Returns start point for next regular expression match

array **mb_ereg_search_getpos** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_search_getpos() returns the point to start regular expression match for mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). The position is represented by bytes from the head of string.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_setpos().

mb_ereg_search_getregs (PHP 4 >= 4.2.0)

Retrive the result from the last multibyte regular expression match

array **mb_ereg_search_getregs** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_search_getregs() returns an array including the sub-string of matched part by last mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). If there are some maches, the first element will have the matched sub-string, the second element will have the first part grouped with brackets, the third element will have the second part grouped with brackets, and so on. It returns `FALSE` on error;

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_init (PHP 4 >= 4.2.0)

Setup string and regular expression for multibyte regular expression match

array **mb_ereg_search_init** (string string [, string pattern [, string option]]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_search_init() sets *`string`* and *`pattern`* for multibyte regular expression. These values are used for mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). It returns `TRUE` for success, `FALSE` for error.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_regs().

mb_ereg_search_pos (PHP 4 >= 4.2.0)

Return position and length of matched part of multibyte regular expression for predefined multibyte string

array **mb_ereg_search_pos** ([string pattern [, string option]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_search_pos() returns an array including position of matched part for multibyte regular expression. The first element of the array will be the beggining of matched part, the second element will be length (bytes) of matched part. It returns `FALSE` on error.

The string for match is specified by mb_ereg_search_init(). It it is not specified, the previous one will be used.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_regs (PHP 4 >= 4.2.0)

Returns the matched part of multibyte regular expression

array **mb_ereg_search_regs** ([string pattern [, string option]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_search_regs() executes the multibyte regular expression match, and if there are some matched part, it returns an array including substring of matched part as first element, the first grouped part with brackets as second element, the second grouped part as third element, and so on. It returns `FALSE` on error.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_setpos (PHP 4 >= 4.2.0)

Set start point of next regular expression match

array **mb_ereg_search_setpos** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_search_setpos() sets the starting point of match for mb_ereg_search().

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search (PHP 4 >= 4.2.0)

Multibyte regular expression match for predefined multibyte string

bool **mb_ereg_search** ([string pattern [, string option]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_search() returns `TRUE` if the multibyte string matches with the regular expression, `FALSE` for otherwise. The string for matching is set by mb_ereg_search_init(). If *`pattern`* is not specified, the previous one is used.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg (PHP 4 >= 4.2.0)

Regular expression match with multibyte support

int **mb_ereg** (string pattern, string string [, array regs]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg() executes the regular expression match with multibyte support, and returns 1 if matches are found. If the optional third parameter was specified, the function returns the byte length of matched part, and therarray *regs* will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, FALSE will be returned.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_eregi()

mb_eregi_replace (PHP 4 >= 4.2.0)

Replace regular expression with multibyte support ignoring case

string **mb_eregi_replace** (string pattern, string replace, string string) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_ereg_replace() scans *string* for matches to *pattern*, then replaces the matched text with *replacement* and returns the result string or FALSE on error. Multibyte character can be used in *pattern*. The case will be ignored.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_replace().

mb_eregi (PHP 4 >= 4.2.0)

Regular expression match ignoring case with multibyte support

int **mb_eregi** (string pattern, string string [, array regs]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_eregi() executes the regular expression match with multibyte support, and returns 1 if matches are found. This function ignore case. If the optional third parameter was specified, the function returns the byte length of matched part, and therarray `regs` will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, `FALSE` will be returned.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_get_info (PHP 4 >= 4.2.0)

Get internal settings of mbstring

string **mb_get_info** ([string type]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_get_info() returns internal setting parameter of mbstring.

If *type* isn't specified or is specified to "all", an array having the elements "internal_encoding", "http_output", "http_input", "func_overload" will be returned.

If *type* is specified for "http_output", "http_input", "internal_encoding", "func_overload", the specified setting parameter will be returned.

See also mb_internal_encoding(), mb_http_output().

mb_http_input (PHP 4 >= 4.0.6)

Detect HTTP input character encoding

string **mb_http_input** ([string type]) \linebreak

mb_http_input() returns result of HTTP input character encoding detection.

type: Input string specifies input type. "G" for GET, "P" for POST, "C" for COOKIE. If type is omitted, it returns last input type processed.

Return Value: Character encoding name. If **mb_http_input()** does not process specified HTTP input, it returns FALSE.

See also mb_internal_encoding(), mb_http_output(), mb_detect_order().

mb_http_output (PHP 4 >= 4.0.6)

Set/Get HTTP output character encoding

string **mb_http_output** ([string encoding]) \linebreak

If *encoding* is set, **mb_http_output()** sets HTTP output character encoding to *encoding*. Output after this function is converted to *encoding*. **mb_http_output()** returns TRUE for success and FALSE for failure.

If *encoding* is omitted, **mb_http_output()** returns current HTTP output character encoding.

See also mb_internal_encoding(), mb_http_input(), mb_detect_order().

mb_internal_encoding (PHP 4 >= 4.0.6)

Set/Get internal character encoding

string **mb_internal_encoding** ([string encoding]) \linebreak

mb_internal_encoding() sets internal character encoding to *encoding* If parameter is omitted, it returns current internal encoding.

encoding is used for HTTP input character encoding conversion, HTTP output character encoding conversion and default character encoding for string functions defined by mbstring module.

encoding: Character encoding name

Return Value: If *encoding* is set,**mb_internal_encoding()** returns TRUE for success, otherwise returns FALSE. If *encoding* is omitted, it returns current character encoding name.

Ejemplo 1. mb_internal_encoding() example

```
/* Set internal character encoding to UTF-8 */
mb_internal_encoding("UTF-8");

/* Display current internal character encoding */
echo mb_internal_encoding();
```

See also mb_http_input(), mb_http_output(), mb_detect_order().

mb_language (PHP 4 >= 4.0.6)

Set/Get current language

string **mb_language** ([string language]) \linebreak

mb_language() sets language. If *language* is omitted, it returns current language as string.

language setting is used for encoding e-mail messages. Valid languages are "Japanese", "ja","English","en" and "uni" (UTF-8). mb_send_mail() uses this setting to encode e-mail.

Language and its setting is ISO-2022-JP/Base64 for Japanese, UTF-8/Base64 for uni, ISO-8859-1/quoted printable for English.

Return Value: If *language* is set and *language* is valid, it returns TRUE. Otherwise, it returns FALSE. When *language* is omitted, it returns language name as string. If no language is set previously, it returns FALSE.

See also mb_send_mail().

mb_output_handler (PHP 4 >= 4.0.6)

Callback function converts character encoding in output buffer

string **mb_output_handler** (string contents, int status) \linebreak

mb_output_handler() is ob_start() callback function. **mb_output_handler()** converts characters in output buffer from internal character encoding to HTTP output character encoding.

4.1.0 or later version, this hanlder adds charset HTTP header when following conditions are met:

- Does not set `Content-Type` by header()
- Default MIME type begins with `text/`
- `http_output` setting is other than pass

`contents` : Output buffer contents

`status` : Output buffer status

Return Value: String converted

Ejemplo 1. mb_output_handler() example

```
mb_http_output("UTF-8");
ob_start("mb_output_handler");
```

Nota: If you want to output some binary data such as image from PHP script, you must set output encoding to "pass" using mb_http_output().

See also ob_start().

mb_parse_str (PHP 4 >= 4.0.6)

Parse GET/POST/COOKIE data and set global variable

boolean **mb_parse_str** (string encoded_string [, array result]) \linebreak

mb_parse_str() parses GET/POST/COOKIE data and sets global variables. Since PHP does not provide raw POST/COOKIE data, it can only used for GET data for now. It preses URL encoded data, detects encoding, converts coding to internal encoding and set values to *`result`* array or global variables.

`encoded_string`: URL encoded data.

`result`: Array contains decoded and character encoding converted values.

Return Value: It returns `TRUE` for success or `FALSE` for failure.

See also mb_detect_order(), mb_internal_encoding().

mb_preferred_mime_name (PHP 4 >= 4.0.6)

Get MIME charset string

string **mb_preferred_mime_name** (string encoding) \linebreak

mb_preferred_mime_name() returns MIME `charset` string for character encoding *`encoding`*. It returns `charset` string.

Ejemplo 1. mb_preferred_mime_string() example

```
$outputenc = "sjis-win";
mb_http_output($outputenc);
ob_start("mb_output_handler");
header("Content-Type: text/html; charset=" . mb_preferred_mime_name($outputenc));
```

mb_regex_encoding (PHP 4 >= 4.2.0)

Returns current encoding for multibyte regex as string

string **mb_regex_encoding** ([string encoding]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_regex_encoding() returns the character encoding used by multibyte regex functions.

If the optional parameter *`encoding`* is specified, it is set to the character encoding for multibyte regex. The default value is the internal character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_internal_encoding(), mb_ereg()

mb_send_mail (PHP 4 >= 4.0.6)

Send encoded mail.

boolean **mb_send_mail** (string to, string subject, string message [, string additional_headers [, string additional_parameter]]) \linebreak

mb_send_mail() sends email. Headers and message are converted and encoded according to mb_language() setting. **mb_send_mail()** is wrapper function of mail(). See mail() for details.

to is mail addresses send to. Multiple recipients can be specified by putting a comma between each address in to. This parameter is not automatically encoded.

subject is subject of mail.

message is mail message.

additional_headers is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a newline ("\n").

additional_parameter is a MTA command line parameter. It is useful when setting the correct Return-Path header when using sendmail.

Devuelve TRUE si todo fue bien, FALSE en caso de fallo.

See also mail(), mb_encode_mimeheader(), and mb_language().

mb_split (PHP 4 >= 4.2.0)

Split multibyte string using regular expression

array **mb_split** (string pattern, string string [, int limit]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

mb_split() split multibyte *string* using regular expression *pattern* and returns the result as an array.

If optional parameter *limit* is specified, it will be split in *limit* elements as maximum.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_strcut (PHP 4 >= 4.0.6)

Get part of string

string **mb_strcut** (string str, int start [, int length [, string encoding]]) \linebreak

mb_strcut() returns the portion of *str* specified by the *start* and *length* parameters.

mb_strcut() performs equivalent operation as mb_substr() with different method. If *start* position is multi-byte character's second byte or larger, it starts from first byte of multi-byte character.

It subtracts string from *str* that is shorter than *length* AND character that is not part of multi-byte string or not being middle of shift sequence.

encoding is character encoding. If it is not set, internal character encoding is used.

See also mb_substr(), mb_internal_encoding().

mb_strimwidth (PHP 4 >= 4.0.6)

Get truncated string with specified width

string **mb_strimwidth** (string str, int start, int width, string trimmarker [, string encoding]) \linebreak

mb_strimwidth() truncates string *str* to specified *width*. It returns truncated string.

If *trimmarker* is set, *trimmarker* is appended to return value.

start is start position offset. Number of characters from the beginning of string. (First character is 0)

trimmarker is string that is added to the end of string when string is truncated.

encoding is character encoding. If it is omitted, internal encoding is used.

Ejemplo 1. mb_strimwidth() example

```
$str = mb_strimwidth($str, 0, 40, "..>");
```

See also: mb_strwidth(), mb_internal_encoding().

mb_strlen (PHP 4 >= 4.0.6)

Get string length

string **mb_strlen** (string str [, string encoding]) \linebreak

mb_strlen() returns number of characters in string *str* having character encoding *encoding*. A multi-byte character is counted as 1.

encoding is character encoding for *str*. If *encoding* is omitted, internal character encoding is used.

See also mb_internal_encoding(), strlen().

mb_strpos (PHP 4 >= 4.0.6)

Find position of first occurrence of string in a string

int **mb_strpos** (string haystack, string needle [, int offset [, string encoding]]) \linebreak

mb_strpos() returns the numeric position of the first occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns FALSE.

mb_strpos() performs multi-byte safe strpos() operation based on number of characters. *needle* position is counted from the beginning of the *haystack*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal character encoding is used. mb_strrpos() accepts string for *needle* where strrpos() accepts only character.

offset is search offset. If it is not specified, 0 is used.

encoding is character encoding name. If it is omitted, internal character encoding is used.

See also **mb_strpos()**, mb_internal_encoding(), strpos()

mb_strrpos (PHP 4 >= 4.0.6)

Find position of last occurrence of a string in a string

int **mb_strrpos** (string haystack, string needle [, string encoding]) \linebreak

mb_strrpos() returns the numeric position of the last occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns FALSE.

mb_strrpos() performs multi-byte safe strrpos() operation based on number of characters. *needle* position is counted from the beginning of *haystack*. First character's position is 0. Second character position is 1.

If *encoding* is omitted, internal encoding is assumed. **mb_strrpos()** accepts string for *needle* where strrpos() accepts only character.

encoding is character encoding. If it is not specified, internal character encoding is used.

See also mb_strpos(), mb_internal_encoding(), strrpos().

mb_strwidth (PHP 4 >= 4.0.6)

Return width of string

int **mb_strwidth** (string str [, string encoding]) \linebreak

mb_strwidth() returns width of string *str*.

Multi-byte character usually twice of width compare to single byte character.

```
Character width

U+0000 - U+0019   0
U+0020 - U+1FFF   1
U+2000 - U+FF60   2
U+FF61 - U+FF9F   1
U+FFA0 -          2
```

encoding is character encoding. If it is omitted, internal encoding is used.

See also: mb_strimwidth(), mb_internal_encoding().

mb_substitute_character (PHP 4 >= 4.0.6)

Set/Get substitution character

mixed **mb_substitute_character** ([mixed substrchar]) \linebreak

mb_substitute_character() specifies substitution character when input character encoding is invalid or character code is not exist in output character encoding. Invalid characters may be substituted NULL(no output), string or integer value (Unicode character code value).

This setting affects mb_detect_encoding() and mb_send_mail().

substchar : Specify Unicode value as integer or specify as string as follows

- "none" : no output
- "long" : Output character code value (Example: U+3000,JIS+7E7E)

Return Value: If *substchar* is set, it returns TRUE for success, otherwise returns FALSE. If *substchar* is not set, it returns Unicode value or "none"/"long".

Ejemplo 1. mb_substitute_character() example

```
/* Set with Unicode U+3013 (GETA MARK) */
mb_substitute_character(0x3013);

/* Set hex format */
mb_substitute_character("long");

/* Display current setting */
echo mb_substitute_character();
```

mb_substr (PHP 4 >= 4.0.6)

Get part of string

string **mb_substr** (string str, int start [, int length [, string encoding]]) \linebreak

mb_substr() returns the portion of *str* specified by the *start* and *length* parameters.

mb_substr() performs multi-byte safe substr() operation based on number of characters. Position is counted from the beginning of *str*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal encoding is assumed.

encoding is character encoding. If it is omitted, internal character encoding is used.

See also mb_strcut(), mb_internal_encoding().

LIV. MCAL functions

MCAL stands for Modular Calendar Access Library.

Libmcal is a C library for accessing calendars. It's written to be very modular, with plugable drivers. MCAL is the calendar equivalent of the IMAP module for mailboxes.

With mcal support, a calendar stream can be opened much like the mailbox stream with the IMAP support. Calendars can be local file stores, remote ICAP servers, or other formats that are supported by the mcal library.

Calendar events can be pulled up, queried, and stored. There is also support for calendar triggers (alarms) and reoccuring events.

With libmcal, central calendar servers can be accessed and used, removing the need for any specific database or local file programming.

To get these functions to work, you have to compile PHP with `--with-mcal`. That requires the mcal library to be installed. Grab the latest version from http://mcal.chek.com/ and compile and install it.

The following constants are defined when using the MCAL module: MCAL_SUNDAY, MCAL_MONDAY, MCAL_TUESDAY, MCAL_WEDNESDAY, MCAL_THURSDAY, MCAL_FRIDAY, MCAL_SATURDAY, MCAL_RECUR_NONE, MCAL_RECUR_DAILY, MCAL_RECUR_WEEKLY, MCAL_RECUR_MONTHLY_MDAY, MCAL_RECUR_MONTHLY_WDAY, MCAL_RECUR_YEARLY, MCAL_JANUARY, MCAL_FEBRUARY, MCAL_MARCH, MCAL_APRIL, MCAL_MAY, MCAL_JUNE, MCAL_JULY, MCAL_AUGUGT, MCAL_SEPTEMBER, MCAL_OCTOBER, MCAL_NOVEMBER, and MCAL_DECEMBER. Most of the functions use an internal event structure that is unique for each stream. This alleviates the need to pass around large objects between functions. There are convenience functions for setting, initializing, and retrieving the event structure values.

mcal_append_event (PHP 4)

Store a new event into an MCAL calendar

int **mcal_append_event** (int mcal_stream) \linebreak

mcal_append_event() Stores the global event into an MCAL calendar for the given stream.

Returns the uid of the newly inserted event.

mcal_close (PHP 3>= 3.0.13, PHP 4)

Close an MCAL stream

int **mcal_close** (int mcal_stream, int flags) \linebreak

Closes the given mcal stream.

mcal_create_calendar (PHP 3>= 3.0.13, PHP 4)

Create a new MCAL calendar

string **mcal_create_calendar** (int stream, string calendar) \linebreak

Creates a new calendar named *`calendar`*.

mcal_date_compare (PHP 3>= 3.0.13, PHP 4)

Compares two dates

int **mcal_date_compare** (int a_year, int a_month, int a_day, int b_year, int b_month, int b_day) \linebreak

mcal_date_compare() Compares the two given dates, returns <0, 0, >0 if a<b, a==b, a>b respectively

mcal_date_valid (PHP 3>= 3.0.13, PHP 4)

Returns `TRUE` if the given year, month, day is a valid date

int **mcal_date_valid** (int year, int month, int day) \linebreak

mcal_date_valid() Returns `TRUE` if the given year, month and day is a valid date, `FALSE` if not.

mcal_day_of_week (PHP 3>= 3.0.13, PHP 4)

Returns the day of the week of the given date

int **mcal_** (int year, int month, int day) \linebreak

mcal_day_of_week() returns the day of the week of the given date

mcal_day_of_year (PHP 3>= 3.0.13, PHP 4)

Returns the day of the year of the given date

int **mcal_** (int year, int month, int day) \linebreak

mcal_day_of_year() returns the day of the year of the given date

mcal_days_in_month (PHP 3>= 3.0.13, PHP 4)

Returns the number of days in the given month

int **mcal_days_in_month** (int month, int leap year) \linebreak

mcal_days_in_month() Returns the number of days in the given month, taking into account if the given year is a leap year or not.

mcal_delete_calendar (PHP 3>= 3.0.13, PHP 4)

Delete an MCAL calendar

string **mcal_delete_calendar** (int stream, string calendar) \linebreak

Deletes the calendar named *`calendar`*.

mcal_delete_event (PHP 3>= 3.0.13, PHP 4)

Delete an event from an MCAL calendar

int **mcal_delete_event** (int uid) \linebreak

mcal_delete_event() deletes the calendar event specified by the uid.

Returns `TRUE`.

mcal_event_add_attribute (PHP 3>= 3.0.15, PHP 4)

Adds an attribute and a value to the streams global event structure

void **mcal_event_add_attribute** (int stream, string attribute, string value) \linebreak

mcal_event_add_attribute() adds an attribute to the stream's global event structure with the value given by "value" .

mcal_event_init (PHP 3>= 3.0.13, PHP 4)

Initializes a streams global event structure

int **mcal_event_init** (int stream) \linebreak

mcal_event_init() initializes a streams global event structure. this effectively sets all elements of the structure to 0, or the default settings.

Returns `TRUE`.

mcal_event_set_alarm (PHP 3>= 3.0.13, PHP 4)

Sets the alarm of the streams global event structure

int **mcal_event_set_alarm** (int stream, int alarm) \linebreak

mcal_event_set_alarm() sets the streams global event structure's alarm to the given minutes before the event.

Returns `TRUE`.

mcal_event_set_category (PHP 3>= 3.0.13, PHP 4)

Sets the category of the streams global event structure

int **mcal_event_set_category** (int stream, string category) \linebreak

mcal_event_set_category() sets the streams global event structure's category to the given string.

Returns `TRUE`.

mcal_event_set_class (PHP 3>= 3.0.13, PHP 4)

Sets the class of the streams global event structure

int **mcal_event_set_class** (int stream, int class) \linebreak

mcal_event_set_class() sets the streams global event structure's class to the given value. The class is either 1 for public, or 0 for private.

Returns TRUE.

mcal_event_set_description (PHP 3>= 3.0.13, PHP 4)

Sets the description of the streams global event structure

int **mcal_event_set_description** (int stream, string description) \linebreak

mcal_event_set_description() sets the streams global event structure's description to the given string.

Returns TRUE.

mcal_event_set_end (PHP 3>= 3.0.13, PHP 4)

Sets the end date and time of the streams global event structure

int **mcal_event_set_end** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]]) \linebreak

mcal_event_set_end() sets the streams global event structure's end date and time to the given values.

Returns TRUE.

mcal_event_set_recur_daily (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_daily** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_daily() sets the streams global event structure's recurrence to the given value to be reoccuring on a daily basis, ending at the given date.

mcal_event_set_recur_monthly_mday (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_monthly_mday** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_monthly_mday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by month day basis, ending at the given date.

mcal_event_set_recur_monthly_wday (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_monthly_wday** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_monthly_wday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by week basis, ending at the given date.

mcal_event_set_recur_none (PHP 3>= 3.0.15, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_none** (int stream) \linebreak

mcal_event_set_recur_none() sets the streams global event structure to not recur (event->recur_type is set to MCAL_RECUR_NONE).

mcal_event_set_recur_weekly (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_weekly** (int stream, int year, int month, int day, int interval, int weekdays) \linebreak

mcal_event_set_recur_weekly() sets the streams global event structure's recurrence to the given value to be reoccuring on a weekly basis, ending at the given date.

mcal_event_set_recur_yearly (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_yearly** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_yearly() sets the streams global event structure's recurrence to the given value to be reoccuring on a yearly basis,ending at the given date .

mcal_event_set_start (PHP 3>= 3.0.13, PHP 4)

Sets the start date and time of the streams global event structure

int **mcal_event_set_start** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]]) \linebreak

mcal_event_set_start() sets the streams global event structure's start date and time to the given values.

Returns `TRUE`.

mcal_event_set_title (PHP 3>= 3.0.13, PHP 4)

Sets the title of the streams global event structure

int **mcal_event_set_title** (int stream, string title) \linebreak

mcal_event_set_title() sets the streams global event structure's title to the given string.

Returns `TRUE`.

mcal_expunge (unknown)

Deletes all events marked for being expunged.

int **mcal_expunge** (int stream) \linebreak

mcal_expunge() Deletes all events which have been previously marked for deletion.

mcal_fetch_current_stream_event (PHP 3>= 3.0.13, PHP 4)

Returns an object containing the current streams event structure

int **mcal_fetch_current_stream_event** (int stream) \linebreak

mcal_event_fetch_current_stream_event() returns the current stream's event structure as an object containing:

- int id - ID of that event.
- int public - `TRUE` if the event if public, `FALSE` if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.

- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_fetch_event (PHP 3>= 3.0.13, PHP 4)

Fetches an event from the calendar stream

object **mcal_fetch_event** (int mcal_stream, int event_id [, int options]) \linebreak

mcal_fetch_event() fetches an event from the calendar stream specified by *id*.

Returns an event object consisting of:

- int id - ID of that event.
- int public - `TRUE` if the event if public, `FALSE` if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date

- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_is_leap_year (PHP 3>= 3.0.13, PHP 4)

Returns if the given year is a leap year or not

int **mcal_is_leap_year** (int year) \linebreak

mcal_is_leap_year() returns 1 if the given year is a leap year, 1 if not.

mcal_list_alarms (PHP 3>= 3.0.13, PHP 4)

Return a list of events that has an alarm triggered at the given datetime

array **mcal_list_events** (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]]) \linebreak

Returns an array of event ID's that has an alarm going off between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_list_events (PHP 3>= 3.0.13, PHP 4)

Return a list of events between two given datetimes

array **mcal_list_events** (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]]) \linebreak

Returns an array of event ID's that are between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_next_recurrence (PHP 3>= 3.0.13, PHP 4)

Returns the next recurrence of the event

int **mcal_next_recurrence** (int stream, int weekstart, array next) \linebreak

mcal_next_recurrence() returns an object filled with the next date the event occurs, on or after the supplied date. Returns empty date field if event does not occur or something is invalid. Uses weekstart to determine what day is considered the beginning of the week.

mcal_open (PHP 3>= 3.0.13, PHP 4)

Opens up an MCAL connection

int **mcal_open** (string calendar, string username, string password, string options) \linebreak

Returns an MCAL stream on success, `FALSE` on error.

mcal_open() opens up an MCAL connection to the specified *`calendar`* store. If the optional *`options`* is specified, passes the *`options`* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_popen (PHP 3>= 3.0.13, PHP 4)

Opens up a persistent MCAL connection

int **mcal_popen** (string calendar, string username, string password [, int options]) \linebreak

Returns an MCAL stream on success, `FALSE` on error.

mcal_popen() opens up an MCAL connection to the specified *`calendar`* store. If the optional *`options`* is specified, passes the *`options`* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_rename_calendar (PHP 3>= 3.0.13, PHP 4)

Rename an MCAL calendar

string **mcal_rename_calendar** (int stream, string old_name, string new_name) \linebreak

Renames the calendar *old_name* to *new_name*.

mcal_reopen (PHP 3>= 3.0.13, PHP 4)

Reopens an MCAL connection

int **mcal_reopen** (string calendar [, int options]) \linebreak

Reopens an MCAL stream to a new calendar.

mcal_reopen() reopens an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also.

mcal_snooze (PHP 3>= 3.0.13, PHP 4)

Turn off an alarm for an event

int **mcal_snooze** (int uid) \linebreak

mcal_snooze() turns off an alarm for a calendar event specified by the uid.

Returns TRUE.

mcal_store_event (PHP 3>= 3.0.13, PHP 4)

Modify an existing event in an MCAL calendar

int **mcal_store_event** (int mcal_stream) \linebreak

mcal_store_event() Stores the modifications to the current global event for the given stream.

Returns TRUE on success and FALSE on error.

mcal_time_valid (PHP 3>= 3.0.13, PHP 4)

Returns TRUE if the given year, month, day is a valid time

int **mcal_time_valid** (int hour, int minutes, int seconds) \linebreak

mcal_time_valid() Returns `TRUE` if the given hour, minutes and seconds is a valid time, `FALSE` if not.

mcal_week_of_year (PHP 4)

Returns the week number of the given date

int **mcal_week_of_year** (int day, int month, int year) \linebreak

LV. Funciones Criptográficas

Estas funciones trabajan usando mcrypt (http://mcrypt.hellug.gr/).

Esta es una interfaz a la librería mcrypt, que soporta una gran variedad de algoritmos de bloque como DES, TripleDES, Blowfish (por defecto), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 y GOST en los modos de cifrado CBC, OFB, CFB y ECB. Adicionalmente, soporta RC6 e IDEA que se consideran "no-libres".

Para usarlos, descarga libmcrypt-x.x.tar.gz de aquí (http://mcrypt.hellug.gr/) y sigue las instrucciones de instalación incluidas. Necesitas compilar PHP con el parámetro `--with-mcrypt` para activar esta extensión.

mcrypt puede usarse para encriptar y desencriptar usando los cifrados mencionados arriba. Los cuatro comandos importantes de mcrypt (mcrypt_cfb(), mcrypt_cbc(), mcrypt_ecb(), y mcrypt_ofb()) pueden operar en ambos modos que se llaman MCRYPT_ENCRYPT y MCRYPT_DECRYPT, respectivamente.

Ejemplo 1. Encripta un valor de entrada con TripleDES en modo ECB

```
<?php
$key = "esta es una clave muy secreta";
$input = "Nos vemos a las 9 en punto en el lugar secreto.";

$encrypted_data = mcrypt_ecb(MCRYPT_TripleDES, $key, $input, MCRYPT_ENCRYPT);
?>
```

Este ejemplo devolverá los datos encriptados como una cadena en $encrypted_data.

mcrypt puede operar en cuatro modos de cifrado (CBC, OFB, CFB y ECB). Perfilaremos el uso normal de cada uno de estos modos. Para una mejor referencia y una discusión más completa ver Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- ECB (electronic codebook o libro de códigos electrónico) va bien para datos aleatorios, tales como encriptar otras claves. Puesto que los datos son cortos y aleatorios, las desventajas de ECB tienen un efecto negativo favorable.
- CBC (cipher block chaining o cifrado en bloque encadenado) es especialmente útil para encriptar ficheros, donde incrementa significativamente la seguridad por encima de ECB.
- CFB (cipher feedback o cifrado realimentado) es el mejor modo de encriptar flujos de bytes donde cada byte debe ser encriptado.
- OFB (output feedback o salida realimentada) es comparable al CFB, pero puede usarse en aplicaciones donde la propagación de errores no puede tolerarse.

Actualmente PHP no soporta el encriptado/desencriptado de flujos de bits. Por ahora, sólo soporta el manejo de cadenas.

Para una lista completa de los cifrados soportados, ver las definiciones al final de mcrypt.h. La regla general es que se puede acceder al cifrado desde PHP con MCRYPT_nombredelcifrado.

Aquí hay una pequeña lista de los cifrados que estan soportados actualmente por la extensión mcrypt. Si

un cifrado no está listado aquí, pero está listado por mcrypt como soportado, puedes asumir con seguridad que ésta documentación está caduca.

- MCRYPT_BLOWFISH
- MCRYPT_DES
- MCRYPT_TripleDES
- MCRYPT_ThreeWAY
- MCRYPT_GOST
- MCRYPT_CRYPT
- MCRYPT_DES_COMPAT
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_CAST128
- MCRYPT_TEAN
- MCRYPT_RC2
- MCRYPT_TWOFISH (para las antiguas versiones mcrypt 2.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx está disponible en las versiones más nuevas 2.x)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_RC6
- MCRYPT_IDEA

Debes (en los modos CFB y OFB) o puedes (en el modo CBC) suministrar un vector de inicialización (IV) a la correspondiente función de cifrado. El IV debe ser único y debe ser el mismo cuando desencriptas o encriptas. Con datos que son guardados encriptados, puedes cojer la salida de una función de índice bajo la cual los datos son almacenados (ej. la clave MD5 de un fichero). Alternativamente, puedes transmitir el IV junto con los datos encriptados (ver capítulo 9.3 de Applied Cryptography by Schneier (ISBN 0-471-11709-9) para una discusión de éste asunto).

mcrypt_cbc (PHP 3>= 3.0.8, PHP 4)

Encripta/desencripta datos en modo CBC

int **mcrypt_cbc** (int cipher, string key, string data, int mode [, string iv]) \linebreak

mcrypt_cbc() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado CBC y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

iv es el vector de inicialización opcional.

Ver también: mcrypt_cfb(), mcrypt_ecb(), mcrypt_ofb()

mcrypt_cfb (PHP 3>= 3.0.8, PHP 4)

Encripta/desencripta datos en modo CFB

int **mcrypt_cfb** (int cipher, string key, string data, int mode, string iv) \linebreak

mcrypt_cfb() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado CFB y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

iv es el vector de inicialización.

Ver también: mcrypt_cbc(), mcrypt_ecb(), mcrypt_ofb()

mcrypt_create_iv (PHP 3>= 3.0.8, PHP 4)

Crea un vector de inicialización (IV) a partir de una fuente aleatoria

string **mcrypt_create_iv** (int size, int source) \linebreak

mcrypt_create_iv() se usa para crear un IV.

mcrypt_create_iv() toma dos argumentos, *size* determina el tamaño del IV, *source* especifica la fuente del IV.

La fuente puede ser MCRYPT_RAND (generador de números aleatorios del sistema), MCRYPT_DEV_RANDOM (que lee datos de /dev/random) y MCRYPT_DEV_URANDOM (que lee datos de /dev/urandom). Si usas MCRYPT_RAND, asegurate de llamar antes a srand() para inicializar el generador de números aleatorios.

Ejemplo 1. Ejemplo de mcrypt_create_iv

```
<?php
$cipher = MCRYPT_TripleDES;
$block_size = mcrypt_get_block_size($cipher);
$iv = mcrypt_create_iv($block_size, MCRYPT_DEV_RANDOM);
?>
```

mcrypt_decrypt (PHP 4 >= 4.0.2)

Decrypts crypttext with given parameters

string **mcrypt_decrypt** (string cipher, string key, string data, string mode [, string iv]) \linebreak

mcrypt_decrypt() decrypts the data and returns the unencrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data is encrypted. If it's smaller that the required keysize, it is padded with '\0'.

Data is the data that will be decrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

mcrypt_ecb (PHP 3>= 3.0.8, PHP 4)

Encripta/desencripta datos en modo ECB

int **mcrypt_ecb** (int cipher, string key, string data, int mode) \linebreak

mcrypt_ecb() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado ECB y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

Ver también: mcrypt_cbc(), mcrypt_cfb(), mcrypt_ofb()

mcrypt_enc_get_algorithms_name (PHP 4 >= 4.0.2)

Returns the name of the opened algorithm

string **mcrypt_enc_get_algorithms_name** (resource td) \linebreak

This function returns the name of the algorithm.

Ejemplo 1. mcrypt_enc_get_algorithms_name() example

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, ”, MCRYPT_MODE_CFB, ”);
    echo mcrypt_enc_get_algorithms_name($td). "\n";

    $td = mcrypt_module_open (’cast-256’, ”, MCRYPT_MODE_CFB, ”);
    echo mcrypt_enc_get_algorithms_name($td). "\n";
?>

Prints:
CAST-256
CAST-256
```

mcrypt_enc_get_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the opened algorithm

int **mcrypt_enc_get_block_size** (resource td) \linebreak

This function returns the block size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV of the opened algorithm

int **mcrypt_enc_get_iv_size** (resource td) \linebreak

This function returns the size of the iv of the algorithm specified by the encryption descriptor in bytes. If it returns '0' then the IV is ignored in the algorithm. An IV is used in cbc, cfb and ofb modes, and in some algorithms in stream mode.

mcrypt_enc_get_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

int **mcrypt_enc_get_key_size** (resource td) \linebreak

This function returns the maximum supported key size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_modes_name (PHP 4 >= 4.0.2)

Returns the name of the opened mode

string **mcrypt_enc_get_modes_name** (resource td) \linebreak

This function returns the name of the mode.

Ejemplo 1. mcrypt_enc_get_modes_name() example

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, '', MCRYPT_MODE_CFB, '');
    echo mcrypt_enc_get_modes_name($td). "\n";

    $td = mcrypt_module_open ('cast-256', '', 'ecb', '');
    echo mcrypt_enc_get_modes_name($td). "\n";
?>

Prints:
CFB
ECB
```

mcrypt_enc_get_supported_key_sizes (PHP 4 >= 4.0.2)

Returns an array with the supported keysizes of the opened algorithm

array **mcrypt_enc_get_supported_key_sizes** (resource td) \linebreak

Returns an array with the key sizes supported by the algorithm specified by the encryption descriptor. If it returns an empty array then all key sizes between 1 and mcrypt_enc_get_key_size() are supported by the algorithm.

Ejemplo 1. mcrypt_enc_get_supported_key_sizes() example

```
<?php
    $td = mcrypt_module_open ('rijndael-256', '', 'ecb', '');
    var_dump (mcrypt_enc_get_supported_key_sizes($td));
?>

This will print:

array(3) {
  [0]=>
  int(16)
  [1]=>
  int(24)
  [2]=>
  int(32)
}
?>
```

mcrypt_enc_is_block_algorithm_mode (PHP 4 >= 4.0.2)

Checks whether the encryption of the opened mode works on blocks

bool **mcrypt_enc_is_block_algorithm_mode** (resource td) \linebreak

This function returns `TRUE` if the mode is for use with block algorithms, otherwise it returns `FALSE`. (eg. `FALSE` for stream, and `TRUE` for cbc, cfb, ofb).

mcrypt_enc_is_block_algorithm (PHP 4 >= 4.0.2)

Checks whether the algorithm of the opened mode is a block algorithm

bool **mcrypt_enc_is_block_algorithm** (resource td) \linebreak

This function returns `TRUE` if the algorithm is a block algorithm, or `FALSE` if it is a stream algorithm.

mcrypt_enc_is_block_mode (PHP 4 >= 4.0.2)

Checks whether the opened mode outputs blocks

bool **mcrypt_enc_is_block_mode** (resource td) \linebreak

This function returns `TRUE` if the mode outputs blocks of bytes or `FALSE` if it outputs bytes. (eg. `TRUE` for cbc and ecb, and `FALSE` for cfb and stream).

mcrypt_enc_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the opened module

bool **mcrypt_enc_self_test** (resource td) \linebreak

This function runs the self test on the algorithm specified by the descriptor *`td`*. If the self test succeeds it returns `FALSE`. In case of an error, it returns `TRUE`.

mcrypt_encrypt (PHP 4 >= 4.0.2)

Encrypts plaintext with given parameters

string **mcrypt_encrypt** (string cipher, string key, string data, string mode [, string iv]) \linebreak

mcrypt_encrypt() encrypts the data and returns the encrypted data.

`Cipher` is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

`Key` is the key with which the data will be encrypted. If it's smaller that the required keysize, it is padded with '`\0`'. It is better not to use ASCII strings for keys. It is recommended to use the mhash functions to create a key from a string.

`Data` is the data that will be encrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '`\0`'. The returned crypttext can be larger that the size of the data that is given by *`data`*.

`Mode` is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *`IV`* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '`\0`'.

Ejemplo 1. mcrypt_encrypt() Example

```
<?php
    $iv = mcrypt_create_iv (mcrypt_get_iv_size (MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB), MCRYP
    $key = "This is a very secret key";
    $text = "Meet me at 11 o'clock behind the monument.";
    echo strlen ($text)."\n";

    $crypttext = mcrypt_encrypt (MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_ECB, $iv);
    echo strlen ($crypttext)."\n";
?>
```

The above example will print out:

```
42
64
```

See also mcrypt_module_open() for a more advanced API and an example.

mcrypt_generic_deinit (PHP 4 >= 4.1.1)

This function deinitializes an encryption module

bool **mcrypt_generic_deinit** (resource td) \linebreak

This function terminates encryption specified by the encryption descriptor (*td*). It clears all buffers, but does not close the module. You need to call mcrypt_module_close() yourself. (But PHP does this for you at the end of the script. Returns FALSE on error, or TRUE on succes.

See for an example mcrypt_module_open() and the entry on mcrypt_generic_init().

mcrypt_generic_end (PHP 4 >= 4.0.2)

This function terminates encryption

bool **mcrypt_generic_end** (resource td) \linebreak

This function is deprecated, use mcrypt_generic_deinit() instead. It can cause crashes when used with mcrypt_module_close() due to multiple buffer frees.

This function terminates encryption specified by the encryption descriptor (*td*). Actually it clears all buffers, and closes all the modules used. Returns FALSE on error, or TRUE on succes.

mcrypt_generic_init (PHP 4 >= 4.0.2)

This function initializes all buffers needed for encryption

int **mcrypt_generic_init** (resource td, string key, string iv) \linebreak

The maximum length of the key should be the one obtained by calling mcrypt_enc_get_key_size() and every value smaller than this is legal. The IV should normally have the size of the algorithms block size, but you must obtain the size by calling mcrypt_enc_get_iv_size(). IV is ignored in ECB. IV MUST exist in CFB, CBC, STREAM, nOFB and OFB modes. It needs to be random and unique (but not secret). The same IV must be used for encryption/decryption. If you do not want to use it you should set it to zeros, but this is not recommended. The function returns a negative value on error.

You need to call this function before every call to mcrypt_generic() or mdecrypt_generic().

See for an example mcrypt_module_open() and the entry on mcrypt_generic_deinit().

mcrypt_generic (PHP 4 >= 4.0.2)

This function encrypts data

string **mcrypt_generic** (resource td, string data) \linebreak

This function encrypts data. The data is padded with "\0" to make sure the length of the data is n * blocksize. This function returns the encrypted data. Note that the length of the returned string can in fact be longer then the input, due to the padding of the data.

The encryption handle should alwayws be initialized with mcrypt_generic_init() with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling mcrypt_generic_deinit(). See mcrypt_module_open() for an example.

See also mdecrypt_generic(), mcrypt_generic_init() and mcrypt_generic_deinit().

mcrypt_get_block_size (PHP 3>= 3.0.8, PHP 4)

Obtiene el tamaño de bloque del cifrado indicado

int **mcrypt_get_block_size** (int cipher) \linebreak

mcrypt_get_block_size() se usa para obtener el tamaño de bloque del cifrado indicado en *cipher*.

mcrypt_get_block_size() toma un argumento, el cifrado *cipher* y devuelve el tamaño en bytes.

Ver también: mcrypt_get_key_size()

mcrypt_get_cipher_name (PHP 3>= 3.0.8, PHP 4)

Obtiene el nombre del cifrado especificado

string **mcrypt_get_cipher_name** (int cipher) \linebreak

mcrypt_get_cipher_name() se usa para obtener el nombre del cifrado especificado.

mcrypt_get_cipher_name() toma como argumento el número de cifrado y devuelve el nombre del cifrado o `FALSE`, si el cifrado no existe.

Ejemplo 1. Ejemplo de mcrypt_get_cipher_name

```
<?php
$cipher = MCRYPT_TripleDES;

print mcrypt_get_cipher_name($cipher);
?>
```

El ejemplo de más arriba da como resultado:

```
TripleDES
```

mcrypt_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV belonging to a specific cipher/mode combination

int **mcrypt_get_iv_size** (resource td) \linebreak int **mcrypt_get_iv_size** (string cipher, string mode) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

mcrypt_get_iv_size() returns the size of the Initialisation Vector (IV) in bytes. On error the function returns `FALSE`. If the IV is ignored in the specified cipher/mode combination zero is returned.

`cipher` is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

`mode` is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

`td` is the resource that is returned by mcrypt_module_open().

Ejemplo 1. mcrypt_create_iv() example

```
<?php
    $size = mcrypt_get_iv_size (MCRYPT_CAST_256, MCRYPT_MODE_CFB);

    $size = mcrypt_get_iv_size ('des', 'ecb');
?>
```

See also: mcrypt_create_iv()

mcrypt_get_key_size (PHP 3>= 3.0.8, PHP 4)

Obtiene el tamaño de la clave de un cifrado

int **mcrypt_get_key_size** (int cipher) \linebreak

mcrypt_get_key_size() se usa para obtener el tamaño de la clave del cifrado indicado en *cipher*.

mcrypt_get_key_size() toma un argumento, el cifrado *cipher* y devuelve el tamaño de la clave en bytes.

Ver también: mcrypt_get_block_size()

mcrypt_list_algorithms (PHP 4 >= 4.0.2)

Get an array of all supported ciphers

array **mcrypt_list_algorithms** ([string lib_dir]) \linebreak

mcrypt_list_algorithms() is used to get an array of all supported algorithms in the *lib_dir* parameter.

mcrypt_list_algorithms() takes an optional *lib_dir* parameter which specifies the directory where all algorithms are located. If not specifies, the value of the mcrypt.algorithms_dir `php.ini` directive is used.

Ejemplo 1. mcrypt_list_algorithms() Example

```
<?php
    $algorithms = mcrypt_list_algorithms ("/usr/local/lib/libmcrypt");

    foreach ($algorithms as $cipher) {
        echo "$cipher<br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the "/usr/local/lib/libmcrypt" directory.

mcrypt_list_modes (PHP 4 >= 4.0.2)

Get an array of all supported modes

array **mcrypt_list_modes** ([string lib_dir]) \linebreak

mcrypt_list_modes() is used to get an array of all supported modes in the *lib_dir*.

mcrypt_list_modes() takes as optional parameter a directory which specifies the directory where all modes are located. If not specifies, the value of the mcrypt.modes_dir `php.ini` directive is used.

Ejemplo 1. mcrypt_list_modes() Example

```
<?php
    $modes = mcrypt_list_modes ();

    foreach ($modes as $mode) {
        echo "$mode <br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the default mode directory. If it is not set with the ini directive mcrypt.modes_dir, the default directory of mcrypt is used (which is /usr/local/lib/libmcrypt).

mcrypt_module_close (PHP 4 >= 4.0.2)

Close the mcrypt module

bool **mcrypt_module_close** (resource td) \linebreak

This function closes the specified encryption handle.

See mcrypt_module_open() for an example.

mcrypt_module_get_algo_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the specified algorithm

int **mcrypt_module_get_algo_block_size** (string algorithm [, string lib_dir]) \linebreak

This function returns the block size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

int **mcrypt_module_get_algo_key_size** (string algorithm [, string lib_dir]) \linebreak

This function returns the maximum supported key size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_supported_key_sizes (PHP 4 >= 4.0.2)

Returns an array with the supported keysizes of the opened algorithm

array **mcrypt_module_get_supported_key_sizes** (string algorithm [, string lib_dir]) \linebreak

Returns an array with the key sizes supported by the specified algorithm. If it returns an empty array then all key sizes between 1 and mcrypt_module_get_algo_key_size() are supported by the algorithm. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

See also mcrypt_enc_get_supported_key_sizes() which is used on open encryption modules.

mcrypt_module_is_block_algorithm_mode (PHP 4 >= 4.0.2)

This function returns if the the specified module is a block algorithm or not

bool **mcrypt_module_is_block_algorithm_mode** (string mode [, string lib_dir]) \linebreak

This function returns `TRUE` if the mode is for use with block algorithms, otherwise it returns `FALSE`. (eg. `FALSE` for stream, and `TRUE` for cbc, cfb, ofb). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_is_block_algorithm (PHP 4 >= 4.0.2)

This function checks whether the specified algorithm is a block algorithm

bool **mcrypt_module_is_block_algorithm** (string algorithm [, string lib_dir]) \linebreak

This function returns `TRUE` if the specified algorithm is a block algorithm, or `FALSE` is it is a stream algorithm. The optional *`lib_dir`* parameter can contain the location where the algorithm module is on the system.

mcrypt_module_is_block_mode (PHP 4 >= 4.0.2)

This function returns if the the specified mode outputs blocks or not

bool **mcrypt_module_is_block_mode** (string mode [, string lib_dir]) \linebreak

This function returns `TRUE` if the mode outputs blocks of bytes or `FALSE` if it outputs just bytes. (eg. `TRUE` for cbc and ecb, and `FALSE` for cfb and stream). The optional *`lib_dir`* parameter can contain the location where the mode module is on the system.

mcrypt_module_open (PHP 4 >= 4.0.2)

This function opens the module of the algorithm and the mode to be used

resource **mcrypt_module_open** (string algorithm, string algorithm_directory, string mode, string mode_directory) \linebreak

This function opens the module of the algorithm and the mode to be used. The name of the algorithm is specified in algorithm, eg. "twofish" or is one of the MCRYPT_ciphername constants. The module is closed by calling mcrypt_module_close(). Normally it returns an encryption descriptor, or `FALSE` on error.

The *`algorithm_directory`* and *`mode_directory`* are used to locate the encryption modules. When you supply a directory name, it is used. When you set one of these to the empty string (""), the value set by the *`mcrypt.algorithms_dir`* or *`mcrypt.modes_dir`* ini-directive is used. When these are not set, the default directories that are used are the ones that were compiled in into libmcrypt (usally /usr/local/lib/libmcrypt).

Ejemplo 1. mcrypt_module_open() Example

```
<?php
    $td = mcrypt_module_open (MCRYPT_DES, '', MCRYPT_MODE_ECB, '/usr/lib/mcrypt-
modes');
    $td = mcrypt_module_open ('rijndael-256', '', 'ofb', '');
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher an dmode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

Ejemplo 2. Using mcrypt_module_open() in encryption

```
<?php
    /* Open the cipher */
    $td = mcrypt_module_open ('rijndael-256', '', 'ofb', '');

    /* Create the IV and determine the keysize length */
    $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size($td), MCRYPT_DEV_RANDOM);
    $ks = mcrypt_enc_get_key_size ($td);

    /* Create key */
    $key = substr (md5 ('very secret key'), 0, $ks);

    /* Intialize encryption */
    mcrypt_generic_init ($td, $key, $iv);

    /* Encrypt data */
    $encrypted = mcrypt_generic ($td, 'This is very important data');

    /* Terminate encryption handler */
    mcrypt_generic_deinit ($td);

    /* Initialize encryption module for decryption */
    mcrypt_generic_init ($td, $key, $iv);

    /* Decrypt encrypted string */
    $decrypted = mdecrypt_generic ($td, $encrypted);

    /* Terminate decryption handle and close module */
    mcrypt_generic_deinit ($td);
    mcrypt_module_close ($td);

    /* Show string */
    echo trim ($decrypted)."\n";
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher an dmode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

See also mcrypt_module_close(), mcrypt_generic(), mdecrypt_generic(), mcrypt_generic_init() and mcrypt_generic_deinit().

mcrypt_module_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the specified module

bool **mcrypt_module_self_test** (string algorithm [, string lib_dir]) \linebreak

This function runs the self test on the algorithm specified. The optional *lib_dir* parameter can contain the location of where the algorithm module is on the system.

The function returns TRUE if the self test succeeds, or FALSE when if fails.

mcrypt_ofb (PHP 3>= 3.0.8, PHP 4)

Encripta/desencripta datos en modo OFB

int **mcrypt_ofb** (int cipher, string key, string data, int mode, string iv) \linebreak

mcrypt_ofb() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado OFB y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

iv es el vector de inicialización.

Ver también: mcrypt_cbc(), mcrypt_cfb(), mcrypt_ecb()

mdecrypt_generic (PHP 4 >= 4.0.2)

This function decrypts data

string **mdecrypt_generic** (resource td, string data) \linebreak

This function decrypts data. Note that the length of the returned string can in fact be longer then the unencrypted string, due to the padding of the data.

Ejemplo 1. mdecrypt_generic() Example

```
<?php
    /* Data */
    $key = 'this is a very long key, even too long for the cipher';
    $plain_text = 'very important data';

    /* Open module, and create IV */
    $td = mcrypt_module_open ('des', '', 'ecb', '');
    $key = substr ($key, 0, mcrypt_enc_get_key_size ($td));
    $iv_size = mcrypt_enc_get_iv_size ($td);
    $iv = mcrypt_create_iv ($iv_size, MCRYPT_RAND);

    /* Initialize encryption handle */
    if (mcrypt_generic_init ($td, $key, $iv) != -1) {

        /* Encrypt data */
        $c_t = mcrypt_generic ($td, $plain_text);
        mcrypt_generic_deinit ($td);

        /* Reinitialize buffers for decryption */
        mcrypt_generic_init ($td, $key, $iv);
        $p_t = mdecrypt_generic ($td, $c_t);

        /* Clean up */
        mcrypt_generic_deinit ($td);
        mcrypt_module_close ($td);
    }

    if (strncmp ($p_t, $plain_text, strlen($plain_text)) == 0) {
        echo "ok\n";
    } else {
        echo "error\n";
    }
?>
```

The above example shows how to check if the data before the encryption is the same as the data after the decryption. It is very important to reinitialize the encryption buffer with mcrypt_generic_init() before you try to decrypt the data.

The decryption handle should alwayws be initialized with mcrypt_generic_init() with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling mcrypt_generic_deinit(). See mcrypt_module_open() for an example.

See also mcrypt_generic(), mcrypt_generic_init() and mcrypt_generic_deinit().

LVI. Funciones Hash

Estas funciones han sido realizadas para trabajar con mhash (http://mhash.sourceforge.net/).

Esta es una interfaz con al libreria mhash. mhash soporta una amplia variedad de algoritmos hash como MD5, SHA1, GOST, y muchos otros.

Para usarla, hay que descargar la distribucion desde su sitio web (http://mhash.sourceforge.net/) y seguir las intrucciones de instalacion. Se necesita compilar PHP con el parametr `--with-mhash` para activar esta extension.

mhash puede ser usado para crear checksums, message digests, y mas.

Ejemplo 1. Generar una clave SHA1 e imprimirla en hexadecimal

```
<?php
$input = "Let us meet at 9 o' clock at the secret place.";
$hash = mhash(MHASH_SHA1, $input);

print "The hash is ".bin2hex($hash)."\n";

?>
```

Esto generara:

```
The hash is d3b85d710d8f6e4e5efd4d5e67d041f9cecedafe
```

PAra una lista complera de hash soportados, refierase a la documentacion de mhash. La regla general es que se puede acceder a los algoritmos hash desde PHP con MHASH_HASHNAME. Como ejemplo, para acceder a HAVAL se debe usar la constante de PHP llamada MHASH_HAVAL.

Aqui hay una lista de hashes que esta actualmente soportada por mhash. Si un hash no esta en dicha lista pero aparece como soportado por mhash, entonces se asume con plena seguridad que esta

documentacion esta desfasada.

- MHASH_MD5
- MHASH_SHA1
- MHASH_HAVAL
- MHASH_RIPEMD160
- MHASH_RIPEMD128
- MHASH_SNEFRU
- MHASH_TIGER
- MHASH_GOST
- MHASH_CRC32
- MHASH_CRC32B

mhash_count (PHP 3>= 3.0.9, PHP 4)

Obtener el valor mayor del id hash disponible

int **mhash_count** (void) \linebreak

mhash_count() devuelve el valor mas alto id hash disponible. Los hash estan numerados desde 0 hasta este valor.

Ejemplo 1. Recorriendo todos los hash

```
<?php

$nr = mhash_count();

for($i = 0; $i <= $nr; $i++) {
    echo sprintf("The blocksize of %s is %d\n",
            mhash_get_hash_name($i),
            mhash_get_block_size($i));
}
?>
```

mhash_get_block_size (PHP 3>= 3.0.9, PHP 4)

Conseguir el tamaño de bloque de el hash especificado

int **mhash_get_block_size** (int hash) \linebreak

mhash_get_block_size() es usado para obtener el tamaño de un bloque de el *hash* determinado.

mhash_get_block_size() toma un argumento, el *hash* y devuelve el tamaño en bytes o FALSE, si el *hash* no existe.

mhash_get_hash_name (PHP 3>= 3.0.9, PHP 4)

Conseguir el nombre de un hash especifico

string **mhash_get_hash_name** (int hash) \linebreak

mhash_get_hash_name() es usado para conseguir el nombre de el hash determinado.

mhash_get_hash_name() toma el id del hash como un argumento y devuelve el nombre de el hash o FALSE, si el hash no existe.

Ejemplo 1. mhash_get_hash_name example

```
<?php
$hash = MHASH_MD5;

print mhash_get_hash_name($hash);
?>
```

El ejemplo anterior mostrara:

```
MD5
```

mhash_keygen_s2k (PHP 4 >= 4.0.4)

Generates a key

string **mhash_keygen_s2k** (int hash, string password, string salt, int bytes) \linebreak

mhash_keygen_s2k() generates a key that is *bytes* long, from a user given password. This is the Salted S2K algorithm as specified in the OpenPGP document (RFC 2440). That algorithm will use the specified *hash* algorithm to create the key. The *salt* must be different and random enough for every key you generate in order to create different keys. That salt must be known when you check the keys, thus it is a good idea to append the key to it. Salt has a fixed length of 8 bytes and will be padded with zeros if you supply less bytes.

Keep in mind that user supplied passwords are not really suitable to be used as keys in cryptographic algorithms, since users normally choose keys they can write on keyboard. These passwords use only 6 to 7 bits per character (or less). It is highly recommended to use some kind of tranformation (like this function) to the user supplied key.

mhash (PHP 3>= 3.0.9, PHP 4)

Calcular el hash

string **mhash** (int hash, string data) \linebreak

mhash() aplica una funcion hash especificada por *hash* a *data* y devuelve el valor hash resultante (tambien llamdo digest).

LVII. Mimetype Functions

Introducción

The functions in this module try to guess the content type and encoding of a file by looking for certain *magic* byte sequences at specific positions within the file. While this is not a bullet proof approach the heuristics used do a very good job.

This extension is derivated from Apache mod_mime_magic, which is itself based on the `file` command maintaind by Ian F. Darwin. See the source code for further historic and copyright information.

Requerimientos

Estas funciones están disponibles como parte del módulo estandar, el cual está siempre disponible.

Instalación

The extension needs a copy of the `magic.mime` as distributed with the `file` command. This file also part of most recent Linux distributions and usually stored in the `/usr/share/misc` directory.

Configuración en tiempo de ejecución

Tabla 1. MIME magic Configuration options

Name	Default	Changeable
mime_magic.magicfile	"/usr/share/misc/magic.mime"	PHP_INI_SYSTEM

Tipos de recursos

Esta extensión no define ningún tipo de recurso.

Constantes predefinidas

Esta extensión no define ninguna constante.

mime_content_type (PHP 4 CVS only)

Detect MIME Content-type for a file

string **mime_content_type** (string filename) \linebreak

Returns the MIME content type for a file as determined by using information from the `magic.mime` file. Content types are returned in MIME format, like `text/plain` or `application/octet-stream`.

LVIII. Funciones de Microsoft SQL Server

mssql_bind (PHP 4 >= 4.1.0)

Adds a parameter to a stored procedure or a remote stored procedure

int **mssql_bind** (int stmt, string param_name, mixed var, int type [, int is_output [, int is_null [, int maxlen]]]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mssql_close (PHP 3, PHP 4)

cierra una conexión con MS SQL Server

int **mssql_close** (int link_identifier) \linebreak

Devuleve: `TRUE` si se finaliza con éxito, `FALSE` si se produce un error

mssql_close() cierra la conexión con una base de datos MS SQL Server que está asociada al identificador especificado. Si el identificador no se especifica, se asume la última conexión abierta.

Observe que normalmente esto no es necesario, ya que las conexiones no-persistentes abiertas se cierran automaticamente en cuanto finaliza el script.

mssql_close() no cerrará conexiones persistentes generadas por mssql_pconnect().

Ver también: mssql_connect(), mssql_pconnect().

mssql_connect (PHP 3, PHP 4)

abre una conexión con MS SQL server

int **mssql_connect** (string servername, string username, string password) \linebreak

Devuelve: Un identificador de MSSQL si se ejecuta correctamente, o `FALSE` si se produce un error.

mssql_connect() establece una conexión con MS SQL server. El argumento servername debe ser un nombre de servidor válido, que está definido en el fichero 'interfaces'.

En caso de hacer una segunda llamada a mssql_connect() con los mismos argumentos, no se establecerá una nueva conexión, sino que se devolverá el identificador de la conexión establecida anteriormente.

La conexión con el servidor se cerrará tan pronto como finalice el script, a menos que se cierre antes, mediante una llamada explicita a la función mssql_close().

Ver también mssql_pconnect(), mssql_close().

mssql_data_seek (PHP 3, PHP 4)

mueve el puntero interno de las filas

int **mssql_data_seek** (int result_identifier, int row_number) \linebreak

Devuelve: `TRUE` si se ejecuta con éxito, `FALSE` si falla.

mssql_data_seek() mueve el puntero interno de la consulta MS SQL asociada al result_identifier especificado, para que apunte al número de fila especificada. La siguiente llamada a mssql_fetch_row() devolverá esa fila.

Ver también: **mssql_data_seek()**.

mssql_execute (PHP 4 >= 4.1.0)

Executes a stored procedure on a MS SQL server database

int **mssql_execute** (int stmt) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mssql_fetch_array (PHP 3, PHP 4)

Captura la fila en un array

int **mssql_fetch_array** (int result) \linebreak

Devuelve: Un array que corresponde a la fila capturada, o `FALSE` si no hay más filas.

mssql_fetch_array() es una versión extendida de mssql_fetch_row(). A¤ade el almacenar los datos en los índices numéricos del array resultante, también almacena los datos en índices asociativos, usando los nombres de los campos como claves.

Una observación a tener en cuenta es, que usar mssql_fetch_array() NO es más lento que usar mssql_fetch_row(), mientras que esta provee un valor a¤adido significativo.

Para más detalles, ver también mssql_fetch_row()

mssql_fetch_assoc (PHP 4 >= 4.2.0)

Returns an associative array of the current row in the result set specified by result_id

array **mssql_fetch_assoc** (int result_id [, int result_type]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mssql_fetch_batch (PHP 4 >= 4.0.4)

Returns the next batch of records

int **mssql_fetch_batch** (string result_index) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mssql_fetch_field (PHP 3, PHP 4)

obtiene la información de los campos

object **mssql_fetch_field** (int result, int field_offset) \linebreak

Devuleve un objeto que contiene información de los campos.

mssql_fetch_field() se puede usar para obtener información acerca de los campos pertenecientes al resultado de una consulta. Si el parámetro field_offset no es especificado, se devuelve la información del siguiente campo que todavía no ha sido devuelto por mssql_fetch_field().

Las propiedades de este objeto son:

- name - nombre de la columna. si la columna es el resultado de una función, esta propiedad vale #N, donde #N es un número de serie.
- column_source - la tabla de donde se tomó la columna
- max_length - longitud máxima de columna

- numeric - 1 si la columna es numérica

Ver también mssql_field_seek()

mssql_fetch_object (PHP 3)

captura la fila como un objeto

int **mssql_fetch_object** (int result) \linebreak

Devuelve: Un objeto con propiedades que se corresponden con la fila capturada, o `FALSE` si no hay más filas.

mssql_fetch_object() es parecida a mssql_fetch_array(), con una diferencia - devuelve un objeto en vez de un array. Indirectamente, esto significa que sólo se puede acceder a los datos por el nombre de los campos, y no por sus posiciones en el objeto (los números no son nombres de propiedades válidas).

La función es idéntica a mssql_fetch_array(), y casi tan rápida como mssql_fetch_row() (la diferencia es insignificante).

Ver también: mssql_fetch-array() and mssql_fetch-row().

mssql_fetch_row (PHP 3, PHP 4)

obtiene la fila como un array numerado

array **mssql_fetch_row** (int result) \linebreak

Devuleve: Un array que corresponde a la fila capturada, o `FALSE` si no hay más filas.

mssql_fetch_row() captura una fila de datos pertenecientes al resultado asociado con el identificador de resultado especificado. La fila es devuelta como un array. Cada columna de resultados es almacenada en una posición del array, comenzando en la posición 0.

Siguientes llamadas a mssql_fetch_rows() devolverían las filas siguientes del result set, o `FALSE` si no hay mas filas.

Ver también: mssql_fetch_array(), mssql_fetch_object(), mssql_data_seek(), **mssql_fetch_lengths()**, and mssql_result().

mssql_field_length (PHP 3>= 3.0.3, PHP 4)

Get the length of a field

int **mssql_field_length** (int result [, int offset]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mssql_field_name (PHP 3>= 3.0.3, PHP 4)

Get the name of a field

int **mssql_field_name** (int result [, int offset]) \linebreak

mssql_field_seek (PHP 3, PHP 4)

set field offset

int **mssql_field_seek** (int result, int field_offset) \linebreak

Se posiciona en el campo especificado por el parámetro field_offset. Si la siguiente llamada a mssql_fetch_field() no incluye el parámetro field_offset, lo que devuelve la función es el campo.

Ver también: mssql_fetch_field().

mssql_field_type (PHP 3>= 3.0.3, PHP 4)

Get the type of a field

string **mssql_field_type** (int result [, int offset]) \linebreak

mssql_free_result (PHP 3, PHP 4)

libera de la memoria el resultado de una consulta

int **mssql_free_result** (int result) \linebreak

mssql_free_result() sólo se necesita llamarla si le preocupa el estar usando mucha memoria mientras se está ejecutando el script. Toda el resultado en memoria será liberado automaticamente cuando finalice el script, puede llamar a **mssql_free_result()** con el identificador de la consulta como argumento y la consulta asociada será liberada de la memoria.

mssql_get_last_message (PHP 3, PHP 4)

Returns the last message from server (over min_message_severity?)

string **mssql_get_last_message** (void) \linebreak

mssql_guid_string (PHP 4 >= 4.1.0)

Converts a 16 byte binary GUID to a string

string **mssql_guid_string** (string binary [, int short_format]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mssql_init (PHP 4 >= 4.1.0)

Initializes a stored procedure or a remote stored procedure

int **mssql_init** (string sp_name [, int conn_id]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mssql_min_error_severity (PHP 3, PHP 4)

Sets the lower error severity

void **mssql_min_error_severity** (int severity) \linebreak

mssql_min_message_severity (PHP 3, PHP 4)

Sets the lower message severity

void **mssql_min_message_severity** (int severity) \linebreak

mssql_next_result (PHP 4 >= 4.0.5)

Move the internal result pointer to the next result

bool **mssql_next_result** (int result_id) \linebreak

When sending more than one SQL statement to the server or executing a stored procedure with multiple results, it will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the rows from the new result set. The function will return `TRUE` if an additional result set was available or `FALSE` otherwise.

Ejemplo 1. mssql_next_result() example

```
<?php
    $link = mssql_connect ("localhost", "userid", "secret");
    mssql_select_db("MyDB", $link);
    $SQL = "Select * from table1 select * from table2";
    $rs = mssql_query($SQL, $link);
    do {
        while ($row = mssql_fetch_row($rs)) {
        }
    } while (mssql_next_result($rs));
    mssql_free_result($rs);
    mssql_close ($link);
?>
```

mssql_num_fields (PHP 3, PHP 4)

obtiene el número de campos de la consulta

int **mssql_num_fields** (int result) \linebreak

mssql_num_fields() devuelve el número de campos de la consulta o result set.

Ver también: **mssql_db_query()**, mssql_query(), mssql_fetch_field(), mssql_num_rows().

mssql_num_rows (PHP 3, PHP 4)

obtiene el número de filas de la consulta

int **mssql_num_rows** (string result) \linebreak

mssql_num_rows() devuelve el número de filas de la consulta o result set.

Ver también: **mssql_db_query()**, mssql_query() and, mssql_fetch_row().

mssql_pconnect (PHP 3, PHP 4)

abre una conexión persistenete con MS SQL

int **mssql_pconnect** (string servername, string username, string password) \linebreak

Devuelve: Un identificador persistente postivo si no hay error, o `FALSE` si se produce alguno

mssql_pconnect() funciona de la misma forma que mssql_connect() aunque con dos grandes diferencias.

La primera es que cuando intenta conectar, la función intentará encontrar un enlace (persistente) que ya esté abierto en el mismo ordenador, nombre de usuario y contrase¤a. Si lo encuentra, la funcion devolverá el identificador de esta en vez de abrir una nueva conexión.

Y la segunda, la conexión con el servidor no se cerrará cuando finalice la ejecución del script. En vez de esto, el enlace permanecerá abierto para un uso futuro. (mssql_close() no cerrará enlaces establecidos por `mssql_pconnect()`).

Por consiguiente, este tipo de enlace es llamado 'persistente'.

mssql_query (PHP 3, PHP 4)

envia una consulta MS SQL

int **mssql_query** (string query, int link_identifier) \linebreak

Devuelve: Un identificado de resultado valido si no hay error, o `FALSE` en caso contrario.

mssql_query() envia una petición de consulta a la base de datos activa en el servidor asociada al identificador de enlace especificado. Si el identificador del enlace no es especificado, se asume como abierto el último enlace. Si no hay ningún enlace abierto, la función intenta establecer un enlace como si mssql_connect() hubiera sido llamada, y lo usa.

Ver también: **mssql_db_query()**, mssql_select_db(), and mssql_connect().

mssql_result (PHP 3, PHP 4)

get result data

int **mssql_result** (int result, int i, mixed field) \linebreak

Devuelve: El contenido de la celda en la fila y posición del result set especificado.

mssql_result() devuelve el contenido de una celda del result set. El parametro field puede ser la posición del campo, o el nombre del campo o bién nombretabla.nombrecampo. Si el nombre de la columna ha sido renombrado ('select foo as bar from...'), use el alias en vez del nombre de la columna.

Trabajando con result sets de gran tama¤o, debería considerar el uso de una de las funciones que capturan una fila completa (especificadas abajo). Como estas funciones devuelven el contenido de múltiples celdas en una sóla llamada, estas son MUCHO más rápidas que mssql_result(). También, observe que especificar una posición númerica para el argumento field es mucho mas rápido que especificar el nombre de un campo o utilizar la forma nombretabla.nombrecampo como argumento.

Alternativas recomendadas para mayor rendimiento : mssql_fetch_row(), mssql_fetch_array(), y mssql_fetch_object().

mssql_rows_affected (PHP 4 >= 4.0.4)

Returns the number of records affected by the query

int **mssql_rows_affected** (int conn_id) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

mssql_select_db (PHP 3, PHP 4)

selecciona una base de datos MS SQL

int **mssql_select_db** (string database_name, int link_identifier) \linebreak

Devuelve: `TRUE` si todo va bién, `FALSE` si se produce un error

mssql_select_db() selecciona como base de datos activa del servidor, la que está asociada al identificador de enlace especificado. Si no se especifica nigún identificador, se asume el último enlace. Si no hay ningún enlace abierto, la función intentará establecer un enlace como si se llamara a la función mssql_connect(), y lo usa.

Cada llamada a mssql_query() será realizada sobre la base de datos activa.

Ver también: mssql_connect(), mssql_pconnect(), y mssql_query()

LIX. Ming functions for Flash

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Introduction

Ming is an open-source (LGPL) library which allows you to create SWF ("Flash") format movies. Ming supports almost all of Flash 4's features, including: shapes, gradients, bitmaps (pngs and jpegs), morphs ("shape tweens"), text, buttons, actions, sprites ("movie clips"), streaming mp3, and color transforms--the only thing that's missing is sound events.

Ming is not an acronym.

Note that all values specifying length, distance, size, etc. are in "twips", twenty units per pixel. That's pretty much arbitrary, though, since the player scales the movie to whatever pixel size is specified in the embed/object tag, or the entire frame if not embedded.

Ming offers a number of advantages over the existing PHP/libswf module. You can use Ming anywhere you can compile the code, whereas libswf is closed-source and only available for a few platforms, Windows not one of them. Ming provides some insulation from the mundane details of the SWF file format, wrapping the movie elements in PHP objects. Also, Ming is still being maintained; if there's a feature that you want to see, just let us know ming@opaque.net (mailto:ming@opaque.net).

Ming was added in PHP 4.0.5.

Installation

To use Ming with PHP, you first need to build and install the Ming library. Source code and installation instructions are available at the Ming home page : http://www.opaque.net/ming/ along with examples, a small tutorial, and the latest news.

Download the ming archive. Unpack the archive. Go in the Ming directory. make. make install.

This will build `libming.so` and install it into `/usr/lib/`, and copy `ming.h` into `/usr/include/`. Edit the `PREFIX=` line in the `Makefile` to change the installation directory.

built into php (unix)

```
mkdir <phpdir>/ext/ming
cp php_ext/* <phpdir>/ext/ming
```

```
cd <phpdir>
./buildconf
./configure --with-ming <other config options>
```

Build and install php as usual, Restart web server if necessary

built into php (unix)

download `php_ming.so.gz`. uncompress it and copy it to your php modules directory. (you can find your php module directory by running **php-config --extension-dir**). Now either just add `extension=php_ming.so` to your `php.ini` file, or put `dl('php_ming.so');` at the head of all of your Ming scripts.

How to use Ming

Ming introduces 13 new objects in PHP, all with matching methods and attributes. To use them, you need to know about objects.

- swfmovie().
- swfshape().
- swfdisplayitem().
- swfgradient().
- swfbitmap().
- swffill().
- swfmorph().
- swftext().
- swffont().
- swftextfield().
- swfsprite().
- swfbutton().
- swfaction().

ming_setcubicthreshold (PHP 4 >= 4.0.5)

Set cubic threshold (?)

void **ming_setcubicthreshold** (int threshold) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ming_setscale (PHP 4 >= 4.0.5)

Set scale (?)

void **ming_setscale** (int scale) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

ming_useswfversion (PHP 4 >= 4.2.0)

Use SWF version (?)

void **ming_useswfversion** (int version) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

SWFAction (PHP 4 >= 4.0.5)

Creates a new Action.

new **swfaction** (string script) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfaction() creates a new Action, and compiles the given script into an SWFAction object.

The script syntax is based on the C language, but with a lot taken out- the SWF bytecode machine is just too simpleminded to do a lot of things we might like. For instance, we can't implement function calls without a tremendous amount of hackery because the jump bytecode has a hardcoded offset value. No pushing your calling address to the stack and returning- every function would have to know exactly where to return to.

So what's left? The compiler recognises the following tokens:

- break
- for
- continue
- if
- else
- do
- while

There is no typed data; all values in the SWF action machine are stored as strings. The following functions can be used in expressions:

time()

Returns the number of milliseconds (?) elapsed since the movie started.

random(seed)

Returns a pseudo-random number in the range 0-seed.

length(expr)

Returns the length of the given expression.

int(number)

Returns the given number rounded down to the nearest integer.

concat(expr, expr)

Returns the concatenation of the given expressions.

ord(expr)

Returns the ASCII code for the given character

chr(num)

Returns the character for the given ASCII code

substr(string, location, length)

Returns the substring of length length at location location of the given string string.

Additionally, the following commands may be used:

duplicateClip(clip, name, depth)

Duplicate the named movie clip (aka sprite). The new movie clip has name name and is at depth depth.

removeClip(expr)

Removes the named movie clip.

trace(expr)

Write the given expression to the trace log. Doubtful that the browser plugin does anything with this.

startDrag(target, lock, [left, top, right, bottom])

Start dragging the movie clip target. The lock argument indicates whether to lock the mouse (?)- use 0 (`FALSE`) or 1 (`TRUE`). Optional parameters define a bounding area for the dragging.

stopDrag()

Stop dragging my heart around. And this movie clip, too.

callFrame(expr)

Call the named frame as a function.

getURL(url, target, [method])

Load the given URL into the named target. The target argument corresponds to HTML document targets (such as "_top" or "_blank"). The optional method argument can be POST or GET if you want to submit variables back to the server.

loadMovie(url, target)

Load the given URL into the named target. The target argument can be a frame name (I think), or one of the magical values "_level0" (replaces current movie) or "_level1" (loads new movie on top of current movie).

nextFrame()

Go to the next frame.

prevFrame()

Go to the last (or, rather, previous) frame.

play()

Start playing the movie.

stop()

Stop playing the movie.

toggleQuality()

Toggle between high and low quality.

stopSounds()

Stop playing all sounds.

gotoFrame(num)

Go to frame number num. Frame numbers start at 0.

gotoFrame(name)

Go to the frame named name. Which does a lot of good, since I haven't added frame labels yet.

setTarget(expr)

Sets the context for action. Or so they say- I really have no idea what this does.

And there's one weird extra thing. The expression frameLoaded(num) can be used in if statements and while loops to check if the given frame number has been loaded yet. Well, it's supposed to, anyway, but I've never tested it and I seriously doubt it actually works. You can just use /:framesLoaded instead.

Movie clips (all together now- aka sprites) have properties. You can read all of them (or can you?), you can set some of them, and here they are:

- x
- y
- xScale
- yScale
- currentFrame - (read-only)
- totalFrames - (read-only)
- alpha - transparency level
- visible - 1=on, 0=off (?)
- width - (read-only)
- height - (read-only)

- rotation
- target - (read-only) (???)
- framesLoaded - (read-only)
- name
- dropTarget - (read-only) (???)
- url - (read-only) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)
- soundBufTime - (???)

So, setting a sprite's x position is as simple as `/box.x = 100;`. Why the slash in front of the box, though? That's how flash keeps track of the sprites in the movie, just like a unix filesystem- here it shows that box is at the top level. If the sprite named box had another sprite named biff inside of it, you'd set its x position with /box/biff.x = 100;. At least, I think so; correct me if I'm wrong here.

This simple example will move the red square across the window.

Ejemplo 1. swfaction() example

```
<?php
 $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for($n=0; $n<5; ++$n)
  {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000,4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
```

```
  $i->setName("box");

  $m->add(new SWFAction("/box.x += 3;"));
  $m->nextFrame();
  $m->add(new SWFAction("gotoFrame(0); play();"));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example tracks down your mouse on the screen.

Ejemplo 2. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setRate(36.0);
  $m->setDimension(1200, 800);
  $m->setBackground(0, 0, 0);

  /* mouse tracking sprite - empty, but follows mouse so we can
     get its x and y coordinates */

  $i = $m->add(new SWFSprite());
  $i->setName('mouse');

  $m->add(new SWFAction("
    startDrag('/mouse', 1); /* '1' means lock sprite to the mouse */
  "));

  /* might as well turn off antialiasing, since these are just squares. */

  $m->add(new SWFAction("
    this.quality = 0;
  "));

  /* morphing box */
  $r = new SWFMorph();
  $s = $r->getShape1();

  /* Note this is backwards from normal shapes.  No idea why. */
  $s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
  $s->movePenTo(-40, -40);
  $s->drawLine(80, 0);
  $s->drawLine(0, 80);
  $s->drawLine(-80, 0);
```

```
$s->drawLine(0, -80);

$s = $r->getShape2();

$s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
$s->movePenTo(-1, -1);
$s->drawLine(2, 0);
$s->drawLine(0, 2);
$s->drawLine(-2, 0);
$s->drawLine(0, -2);

/* sprite container for morphing box -
   this is just a timeline w/ the box morphing */

$box = new SWFSprite();
$box->add(new SWFAction("
  stop();
"));
$i = $box->add($r);

for($n=0; $n<=20; ++$n)
{
  $i->setRatio($n/20);
  $box->nextFrame();
}

/* this container sprite allows us to use the same action code many times */

$cell = new SWFSprite();
$i = $cell->add($box);
$i->setName('box');

$cell->add(new SWFAction("

  setTarget('box');

  /* ...x means the x coordinate of the parent, i.e. (..).x */
  dx = (/mouse.x + random(6)-3 - ...x)/5;
  dy = (/mouse.y + random(6)-3 - ...y)/5;
  gotoFrame(int(dx*dx + dy*dy));

"));

$cell->nextFrame();
$cell->add(new SWFAction("

  gotoFrame(0);
  play();

"));

$cell->nextFrame();
```

```
  /* finally, add a bunch of the cells to the movie */

  for($x=0; $x<12; ++$x)
  {
    for($y=0; $y<8; ++$y)
    {
      $i = $m->add($cell);
      $i->moveTo(100*$x+50, 100*$y+50);
    }
  }

  $m->nextFrame();

  $m->add(new SWFAction("

    gotoFrame(1);
    play();

  "));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Same as above, but with nice colored balls...

Ejemplo 3. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(11000, 8000);
  $m->setBackground(0x00, 0x00, 0x00);

  $m->add(new SWFAction("

this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

  "));

  // mouse tracking sprite
  $t = new SWFSprite();
  $i = $m->add($t);
  $i->setName('mouse');

  $g = new SWFGradient();
```

```
  $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
  $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

  // gradient shape thing
  $s = new SWFShape();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.03);
  $s->setRightFill($f);
  $s->movePenTo(-600, -600);
  $s->drawLine(1200, 0);
  $s->drawLine(0, 1200);
  $s->drawLine(-1200, 0);
  $s->drawLine(0, -1200);

  // need to make this a sprite so we can multColor it
  $p = new SWFSprite();
  $p->add($s);
  $p->nextFrame();

  // put the shape in here, each frame a different color
  $q = new SWFSprite();
  $q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
  $i = $q->add($p);

  $i->multColor(1.0, 1.0, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 0.75, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 0.5, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 1.0);
  $q->nextFrame();

  // finally, this one contains the action code
  $p = new SWFSprite();
  $i = $p->add($q);
  $i->setName('frames');
  $p->add(new SWFAction("

dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;
```

```
  "));
  $p->nextFrame();

  $p->add(new SWFAction("

this.x = x;
this.y = y;
this.alpha = alpha;
x += dx;
y += dy;
dy += 3;
alpha -= 8;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("prevFrame(); play();"));
  $p->nextFrame();

  $i = $m->add($p);
  $i->setName('frames');
  $m->nextFrame();

  $m->add(new SWFAction("

lastx = mousex;
lasty = mousey;
mousex = /mouse.x;
mousey = /mouse.y;

++num;

if(num == 11)
  num = 1;

removeClip('char' & num);
duplicateClip(/frames, 'char' & num, num);

  "));

  $m->nextFrame();
  $m->add(new SWFAction("prevFrame(); play();"));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will handles keyboard actions. (You'll probably have to click in the window to give it focus. And you'll probably have to leave your mouse in the frame, too. If you know how to give buttons focus programatically, feel free to share, won't you?)

Ejemplo 4. swfaction() example

```
<?php

  /* sprite has one letter per frame */

  $p = new SWFSprite();
  $p->add(new SWFAction("stop();"));

  $chars = "abcdefghijklmnopqrstuvwxyz".
       "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
           "1234567890!@#$%^&/*()_+-=/[]{}|;:,.<>¿~";

  $f = new SWFFont("_sans");

  for($n=0; $nremove($i);
    $t = new SWFTextField();
    $t->setFont($f);
    $t->setHeight(240);
    $t->setBounds(600,240);
    $t->align(SWFTEXTFIELD_ALIGN_CENTER);
    $t->addString($c);
    $i = $p->add($t);
    $p->labelFrame($c);
    $p->nextFrame();
  }

  /* hit region for button - the entire frame */

  $s = new SWFShape();
  $s->setFillStyle0($s->addSolidFill(0, 0, 0, 0));
  $s->drawLine(600, 0);
  $s->drawLine(0, 400);
  $s->drawLine(-600, 0);
  $s->drawLine(0, -400);

  /* button checks for pressed key, sends sprite to the right frame */

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT);

  for($n=0; $naddAction(new SWFAction("

setTarget('/char');
gotoFrame('$c');

    "), SWFBUTTON_KEYPRESS($c));
  }
```

```
  $m = new SWFMovie();
  $m->setDimension(600,400);
  $i = $m->add($p);
  $i->setName('char');
  $i->moveTo(0,80);

  $m->add($b);

  header('Content-type: application/x-shockwave-flash');
  $m->output();

?>
```

SWFBitmap->getHeight (unknown)

Returns the bitmap's height.

int **swfbitmap->getheight** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbitmap->getheight() returns the bitmap's height in pixels.

See also **swfbitmap->getwidth()**.

SWFBitmap->getWidth (unknown)

Returns the bitmap's width.

int **swfbitmap->getwidth** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbitmap->getwidth() returns the bitmap's width in pixels.

See also **swfbitmap->getheight()**.

SWFBitmap (PHP 4 >= 4.0.5)

Loads Bitmap object

new **swfbitmap** (string filename [, int alphafilename]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbitmap() creates a new SWFBitmap object from the Jpeg or DBL file named *`filename`*. *`alphafilename`* indicates a MSK file to be used as an alpha mask for a Jpeg image.

> **Nota:** We can only deal with baseline (frame 0) jpegs, no baseline optimized or progressive scan jpegs!

SWFBitmap has the following methods : **swfbitmap->getwidth()** and **swfbitmap->getheight()**.

You can't import png images directly, though- have to use the png2dbl utility to make a dbl ("define bits lossless") file from the png. The reason for this is that I don't want a dependency on the png library in ming- autoconf should solve this, but that's not set up yet.

Ejemplo 1. Import PNG files

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(new SWFBitmap("png.dbl"));
  $s->setRightFill($f);

  $s->drawLine(32, 0);
  $s->drawLine(0, 32);
  $s->drawLine(-32, 0);
  $s->drawLine(0, -32);

  $m = new SWFMovie();
  $m->setDimension(32, 32);
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
```

```
?>
```

And you can put an alpha mask on a jpeg fill.

Ejemplo 2. swfbitmap() example

```
<?php

  $s = new SWFShape();

  // .msk file generated with "gif2mask" utility
  $f = $s->addFill(new SWFBitmap("alphafill.jpg", "alphafill.msk"));
  $s->setRightFill($f);

  $s->drawLine(640, 0);
  $s->drawLine(0, 480);
  $s->drawLine(-640, 0);
  $s->drawLine(0, -480);

  $c = new SWFShape();
  $c->setRightFill($c->addFill(0x99, 0x99, 0x99));
  $c->drawLine(40, 0);
  $c->drawLine(0, 40);
  $c->drawLine(-40, 0);
  $c->drawLine(0, -40);

  $m = new SWFMovie();
  $m->setDimension(640, 480);
  $m->setBackground(0xcc, 0xcc, 0xcc);

  // draw checkerboard background
  for($y=0; $y<480; $y+=40)
  {
    for($x=0; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }

    $y+=40;

    for($x=40; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
  }
```

```
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

swfbutton_keypress (PHP 4 >= 4.0.5)

Returns the action flag for keyPress(char)

int **swfbutton_keypress** (string str) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

SWFbutton->addAction (unknown)

Adds an action

void **swfbutton->addaction** (ressource action, int flags) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbutton->addaction() adds the action *`action`* to this button for the given conditions. The following *`flags`* are valid: SWFBUTTON_MOUSEOVER, SWFBUTTON_MOUSEOUT, SWFBUTTON_MOUSEUP, SWFBUTTON_MOUSEUPOUTSIDE, SWFBUTTON_MOUSEDOWN, SWFBUTTON_DRAGOUT and SWFBUTTON_DRAGOVER.

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->addShape (unknown)

Adds a shape to a button

void **swfbutton->addshape** (ressource shape, int flags) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbutton->addshape() adds the shape `shape` to this button. The following `flags`' values are valid: SWFBUTTON_UP, SWFBUTTON_OVER, SWFBUTTON_DOWN or SWFBUTTON_HIT. SWFBUTTON_HIT isn't ever displayed, it defines the hit region for the button. That is, everywhere the hit shape would be drawn is considered a "touchable" part of the button.

SWFbutton->setAction (unknown)

Sets the action

void **swfbutton->setaction** (ressource action) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbutton->setaction() sets the action to be performed when the button is clicked. Alias for addAction(shape, SWFBUTTON_MOUSEUP). `action` is a swfaction().

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setdown (unknown)

Alias for addShape(shape, SWFBUTTON_DOWN))

void **swfbutton->setdown** (ressource shape) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbutton->setdown() alias for addShape(shape, SWFBUTTON_DOWN).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setHit (unknown)

Alias for addShape(shape, SWFBUTTON_HIT)

void **swfbutton->sethit** (ressource shape) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbutton->sethit() alias for addShape(shape, SWFBUTTON_HIT).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setOver (unknown)

Alias for addShape(shape, SWFBUTTON_OVER)

void **swfbutton->setover** (ressource shape) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbutton->setover() alias for addShape(shape, SWFBUTTON_OVER).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setUp (unknown)

Alias for addShape(shape, SWFBUTTON_UP)

void **swfbutton->setup** (ressource shape) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbutton->setup() alias for addShape(shape, SWFBUTTON_UP).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton (PHP 4 >= 4.0.5)

Creates a new Button.

new **swfbutton** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfbutton() creates a new Button. Roll over it, click it, see it call action code. Swank.

SWFButton has the following methods : **swfbutton->addshape()**, **swfbutton->setup()**, **swfbutton->setover() swfbutton->setdown()**, **swfbutton->sethit() swfbutton->setaction()** and **swfbutton->addaction()**.

This simple example will show your usual interactions with buttons : rollover, rollon, mouseup, mousedown, noaction.

Ejemplo 1. swfbutton() example

```
<?php

  $f = new SWFFont("_serif");

  $p = new SWFSprite();

  function label($string)
```

```
{
  global $f;

  $t = new SWFTextField();
  $t->setFont($f);
  $t->addString($string);
  $t->setHeight(200);
  $t->setBounds(3200,200);
  return $t;
}
function addLabel($string)
{
  global $p;

  $i = $p->add(label($string));
  $p->nextFrame();
  $p->remove($i);
}

$p->add(new SWFAction("stop();"));
addLabel("NO ACTION");
addLabel("SWFBUTTON_MOUSEUP");
addLabel("SWFBUTTON_MOUSEDOWN");
addLabel("SWFBUTTON_MOUSEOVER");
addLabel("SWFBUTTON_MOUSEOUT");
addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
addLabel("SWFBUTTON_DRAGOVER");
addLabel("SWFBUTTON_DRAGOUT");

function rect($r, $g, $b)
{
  $s = new SWFShape();
  $s->setRightFill($s->addFill($r, $g, $b));
  $s->drawLine(600,0);
  $s->drawLine(0,600);
  $s->drawLine(-600,0);
  $s->drawLine(0,-600);

  return $s;
}

$b = new SWFButton();
$b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
$b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
$b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
          SWFBUTTON_MOUSEUP);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
      SWFBUTTON_MOUSEDOWN);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
```

```
        SWFBUTTON_MOUSEOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
        SWFBUTTON_MOUSEOUT);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
        SWFBUTTON_MOUSEUPOUTSIDE);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
        SWFBUTTON_DRAGOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(7);"),
        SWFBUTTON_DRAGOUT);

  $m = new SWFMovie();
  $m->setDimension(4000,3000);

  $i = $m->add($p);
  $i->setName("label");
  $i->moveTo(400,1900);

  $i = $m->add($b);
  $i->moveTo(400,900);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will enables you to drag draw a big red button on the windows. No drag-and-drop, just moving around.

Ejemplo 2. swfbutton->addaction() example

```
<?php

  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->drawLine(1000,0);
  $s->drawLine(0,1000);
  $s->drawLine(-1000,0);
  $s->drawLine(0,-1000);

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN | SWFBUTTON_OVER);

  $b->addAction(new SWFAction("startDrag('/test', 0);"), // '0' means don't lock to mouse
        SWFBUTTON_MOUSEDOWN);
```

```
  $b->addAction(new SWFAction("stopDrag();"),
        SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

  $p = new SWFSprite();
  $p->add($b);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->setName('test');
  $i->moveTo(1000,1000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->addColor (unknown)

Adds the given color to this item's color transform.

void **swfdisplayitem->addcolor** ([int red [, int green [, int blue [, int a]]]]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->addcolor() adds the color to this item's color transform. The color is given in its RGB form.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->move (unknown)

Moves object in relative coordinates.

void **swfdisplayitem->move** (int dx, int dy) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->move() moves the current object by (*dx*,*dy*) from its current position.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->moveto()**.

SWFDisplayItem->moveTo (unknown)

Moves object in global coordinates.

void **swfdisplayitem->moveto** (int x, int y) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->moveto() moves the current object to (*x*,*y*) in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->move()**.

SWFDisplayItem->multColor (unknown)

Multiplies the item's color transform.

void **swfdisplayitem->multcolor** ([int red [, int green [, int blue [, int a]]]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->multcolor() multiplies the item's color transform by the given values.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This simple example will modify your picture's atmospher to Halloween (use a landscape or bright picture).

Ejemplo 1. swfdisplayitem->multcolor() example

```
<?php

  $b = new SWFBitmap("backyard.jpg");
  // note use your own picture :-)
  $s = new SWFShape();
  $s->setRightFill($s->addFill($b));
  $s->drawLine($b->getWidth(), 0);
  $s->drawLine(0, $b->getHeight());
  $s->drawLine(-$b->getWidth(), 0);
  $s->drawLine(0, -$b->getHeight());

  $m = new SWFMovie();
  $m->setDimension($b->getWidth(), $b->getHeight());

  $i = $m->add($s);

  for($n=0; $n<=20; ++$n)
  {
    $i->multColor(1.0-$n/10, 1.0, 1.0);
    $i->addColor(0xff*$n/20, 0, 0);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->remove (unknown)

Removes the object from the movie

void **swfdisplayitem->remove** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->remove() removes this object from the movie's display list.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfmovie->add()**.

SWFDisplayItem->Rotate (unknown)

Rotates in relative coordinates.

void **swfdisplayitem->rotate** (float ddegrees) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->rotate() rotates the current object by `ddegrees` degrees from its current rotation.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->rotateto()**.

SWFDisplayItem->rotateTo (unknown)

Rotates the object in global coordinates.

void **swfdisplayitem->rotateto** (float degrees) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->rotateto() set the current object rotation to *degrees* degrees in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This example bring three rotating string from the background to the foreground. Pretty nice.

Ejemplo 1. swfdisplayitem->rotateto() example

```
<?php
  $thetext =  "ming!";

  $f = new SWFFont("Bauhaus 93.fdb");

  $m = new SWFMovie();
  $m->setRate(24.0);
  $m->setDimension(2400, 1600);
  $m->setBackground(0xff, 0xff, 0xff);

  // functions with huge numbers of arbitrary
  // arguments are always a good idea!  Really!

  function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
  {
    global $f, $m;

    $t = new SWFText();
    $t->setFont($f);
    $t->setColor($r, $g, $b, $a);
    $t->setHeight(960);
    $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
    $t->addString($string);

    // we can add properties just like a normal php var,
    // as long as the names aren't already used.
    // e.g., we can't set $i->scale, because that's a function

    $i = $m->add($t);
    $i->x = $x;
    $i->y = $y;
    $i->rot = $rot;
    $i->s = $scale;
    $i->rotateTo($rot);
    $i->scale($scale, $scale);

    // but the changes are local to the function, so we have to
    // return the changed object.  kinda weird..

    return $i;
 }

  function step($i)
  {
```

```
  $oldrot = $i->rot;
  $i->rot = 19*$i->rot/20;
  $i->x = (19*$i->x + 1200)/20;
  $i->y = (19*$i->y + 800)/20;
  $i->s = (19*$i->s + 1.0)/20;

  $i->rotateTo($i->rot);
  $i->scaleTo($i->s, $i->s);
  $i->moveTo($i->x, $i->y);

  return $i;
}

// see?  it sure paid off in legibility:

$i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
$i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
$i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

for($i=1; $i<=100; ++$i)
{
  $i1 = step($i1);
  $i2 = step($i2);
  $i3 = step($i3);

  $m->nextFrame();
}

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

See also **swfdisplayitem->rotate()**.

SWFDisplayItem->scale (unknown)

Scales the object in relative coordinates.

void **swfdisplayitem->scale** (int dx, int dy) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->scale() scales the current object by (*dx*,*dy*) from its current size.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scaleto()**.

SWFDisplayItem->scaleTo (unknown)

Scales the object in global coordinates.

void **swfdisplayitem->scaleto** (int x, int y) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->scaleto() scales the current object to (*x*,*y*) in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scale()**.

SWFDisplayItem->setDepth (unknown)

Sets z-order

void **swfdisplayitem->setdepth** (float depth) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->rotate() sets the object's z-order to *depth*. Depth defaults to the order in which instances are created (by add'ing a shape/text to a movie)- newer ones are on top of older ones. If two objects are given the same depth, only the later-defined one can be moved.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setName (unknown)

Sets the object's name

void **swfdisplayitem->setname** (string name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->setname() sets the object's name to *`name`*, for targetting with action script. Only useful on sprites.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setRatio (unknown)

Sets the object's ratio.

void **swfdisplayitem->setratio** (float ratio) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->setratio() sets the object's ratio to *`ratio`*. Obviously only useful for morphs.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This simple example will morph nicely three concentric circles.

Ejemplo 1. swfdisplayitem->setname() example

```
<?php

  $p = new SWFMorph();

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0xff, 0xff);
```

```
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0xff, 0xff, 0xff);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0xff, 0xff, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape1();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0, 0);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0, 0xff, 0);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0, 0, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape2();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $f->skewXTo(1.0);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m = new SWFMovie();
  $m->setDimension(320, 240);
  $i = $m->add($p);
  $i->moveTo(160, 120);

  for($n=0; $n<=1.001; $n+=0.01)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->skewX (unknown)

Sets the X-skew.

void **swfdisplayitem->skewx** (float ddegrees) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->skewx() adds `ddegrees` to current x-skew.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewXTo (unknown)

Sets the X-skew.

void **swfdisplayitem->skewxto** (float degrees) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->skewxto() sets the x-skew to `degrees`. For `degrees` is 1.0, it means a 45-degree forward slant. More is more forward, less is more backward.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewY (unknown)

Sets the Y-skew.

void **swfdisplayitem->skewy** (float ddegrees) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->skewy() adds `ddegrees` to current y-skew.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewyto()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->skewYTo (unknown)

Sets the Y-skew.

void **swfdisplayitem->skewyto** (float degrees) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem->skewyto() sets the y-skew to `degrees`. For `degrees` is 1.0, it means a 45-degree forward slant. More is more upward, less is more downward.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewy()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem (unknown)

Creates a new displayitem object.

new **swfdisplayitem** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfdisplayitem() creates a new swfdisplayitem object.

Here's where all the animation takes place. After you define a shape, a text object, a sprite, or a button, you add it to the movie, then use the returned handle to move, rotate, scale, or skew the thing.

SWFDisplayItem has the following methods : **swfdisplayitem->move()**, **swfdisplayitem->moveto()**, **swfdisplayitem->scaleto()**, **swfdisplayitem->scale()**, **swfdisplayitem->rotate()**, **swfdisplayitem->rotateto()**, **swfdisplayitem->skewxto()**, **swfdisplayitem->skewx()**, **swfdisplayitem->skewyto() swfdisplayitem->skewyto()**, **swfdisplayitem->setdepth() swfdisplayitem->remove()**, **swfdisplayitem->setname() swfdisplayitem->setratio()**, **swfdisplayitem->addcolor()** and **swfdisplayitem->multcolor()**.

SWFFill->moveTo (unknown)

Moves fill origin

void **swffill->moveto** (int x, int y) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swffill->moveto() moves fill's origin to (x,y) in global coordinates.

SWFFill->rotateTo (unknown)

Sets fill's rotation

void **swffill->rotateto** (float degrees) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swffill->rotateto() sets fill's rotation to *degrees* degrees.

SWFFill->scaleTo (unknown)

Sets fill's scale

void **swffill->scaleto** (int x, int y) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swffill->scaleto() sets fill's scale to *x* in the x-direction, *y* in the y-direction.

SWFFill->skewXTo (unknown)

Sets fill x-skew

void **swffill->skewxto** (float x) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swffill->skewxto() sets fill x-skew to *x*. For *x* is 1.0, it is a is a 45-degree forward slant. More is more forward, less is more backward.

SWFFill->skewYTo (unknown)

Sets fill y-skew

void **swffill->skewyto** (float y) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swffill->skewyto() sets fill y-skew to `y`. For `y` is 1.0, it is a is a 45-degree upward slant. More is more upward, less is more downward.

SWFFill (PHP 4 >= 4.0.5)

Loads SWFFill object

The **swffill()** object allows you to transform (scale, skew, rotate) bitmap and gradient fills. **swffill()** objects are created by the **swfshape->addfill()** methods.

SWFFill has the following methods : **swffill->moveto()** and **swffill->scaleto()**, **swffill->rotateto()**, **swffill->skewxto()** and **swffill->skewyto()**.

swffont->getwidth (unknown)

Returns the string's width

int **swffont->getwidth** (string string) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swffont->getwidth() returns the string `string`'s width, using font's default scaling. You'll probably want to use the **SWFText()** version of this method which uses the text object's scale.

SWFFont (PHP 4 >= 4.0.5)

Loads a font definition

new **swffont** (string filename) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

If *`filename`* is the name of an FDB file (i.e., it ends in ".fdb"), load the font definition found in said file. Otherwise, create a browser-defined font reference.

FDB ("font definition block") is a very simple wrapper for the SWF DefineFont2 block which contains a full description of a font. One may create FDB files from SWT Generator template files with the included makefdb utility- look in the util directory off the main ming distribution directory.

Browser-defined fonts don't contain any information about the font other than its name. It is assumed that the font definition will be provided by the movie player. The fonts _serif, _sans, and _typewriter should always be available. For example:

```
<?php
$f = newSWFFont("_sans");
?>
```

will give you the standard sans-serif font, probably the same as what you'd get with `<font name="sans-serif">` in HTML.

swffont() returns a reference to the font definition, for use in the **SWFText->setFont()** and the **SWFTextField->setFont()** methods.

SWFFont has the following methods : **swffont->getwidth()**.

SWFGradient->addEntry (unknown)

Adds an entry to the gradient list.

void **swfgradient->addentry** (float ratio, int red, int green, int blue [, int a]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfgradient->addentry() adds an entry to the gradient list. *`ratio`* is a number between 0 and 1 indicating where in the gradient this color appears. Thou shalt add entries in order of increasing ratio.

`red`, *`green`*, *`blue`* is a color (RGB mode). Last parameter *`a`* is optional.

SWFGradient (PHP 4 >= 4.0.5)

Creates a gradient object

new **swfgradient** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfgradient() creates a new SWFGradient object.

After you've added the entries to your gradient, you can use the gradient in a shape fill with the **swfshape->addfill()** method.

SWFGradient has the following methods : **swfgradient->addentry()**.

This simple example will draw a big black-to-white gradient as background, and a redish disc in its center.

Ejemplo 1. swfgradient() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // first gradient- black to white
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);

  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.01);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // second gradient- radial gradient from red to transparent
```

```
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);

  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.005);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMorph->getshape1 (unknown)

Gets a handle to the starting shape

mixed **swfmorph->getshape1** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmorph->getshape1() gets a handle to the morph's starting shape. **swfmorph->getshape1()** returns an swfshape() object.

SWFMorph->getshape2 (unknown)

Gets a handle to the ending shape

mixed **swfmorph->getshape2** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmorph->getshape2() gets a handle to the morph's ending shape. **swfmorph->getshape2()** returns an swfshape() object.

SWFMorph (PHP 4 >= 4.0.5)

Creates a new SWFMorph object.

new **swfmorph** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmorph() creates a new SWFMorph object.

Also called a "shape tween". This thing lets you make those tacky twisting things that make your computer choke. Oh, joy!

The methods here are sort of weird. It would make more sense to just have newSWFMorph(shape1, shape2);, but as things are now, shape2 needs to know that it's the second part of a morph. (This, because it starts writing its output as soon as it gets drawing commands- if it kept its own description of its shapes and wrote on completion this and some other things would be much easier.)

SWFMorph has the following methods : **swfmorph->getshape1()** and **swfmorph->getshape1()**.

This simple example will morph a big red square into a smaller blue black-bordered square.

Ejemplo 1. swfmorph() example

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
  $s->setLine(0,0,0,0);

  /* Note that this is backwards from normal shapes (left instead of right).
     I have no idea why, but this seems to work.. */

  $s->setLeftFill($s->addFill(0xff, 0, 0));
```

```
  $s->movePenTo(-1000,-1000);
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
  $s->drawLine(-2000,0);
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,0,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
  $s->drawLine(1000,-1000);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for($r=0.0; $r<=1.0; $r+=0.1)
  {
    $i->setRatio($r);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMovie->add (unknown)

Adds any type of data to a movie.

void **swfmovie->add** (ressource instance) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->add() adds *instance* to the current movie. *instance* is any type of data : Shapes, text, fonts, etc. must all be add'ed to the movie to make this work.

For displayable types (shape, text, button, sprite), this returns an **SWFDisplayItem()**, a handle to the object in a display list. Thus, you can add the same shape to a movie multiple times and get separate handles back for each separate instance.

See also all other objects (adding this later), and **swfmovie->remove()**

See examples in : **swfdisplayitem->rotateto()** and **swfshape->addfill()**.

SWFMovie->nextframe (unknown)

Moves to the next frame of the animation.

void **swfmovie->nextframe** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->setframes() moves to the next frame of the animation.

SWFMovie->output (unknown)

Dumps your lovingly prepared movie out.

void **swfmovie->output** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->output() dumps your lovingly prepared movie out. In PHP, preceding this with the command

```
<?php
header('Content-type: application/x-shockwave-flash');
?>
```

convinces the browser to display this as a flash movie.

See also **swfmovie->save()**.

See examples in : **swfmovie->streammp3()**, **swfdisplayitem->rotateto()**, swfaction()... Any example will use this method.

SWFMovie->remove (unknown)

Removes the object instance from the display list.

void **swfmovie->remove** (resource instance) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->remove() removes the object instance `instance` from the display list.

See also **swfmovie->add()**.

SWFMovie->save (unknown)

Saves your movie in a file.

void **swfmovie->save** (string filename) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->save() saves your movie to the file named `filename`.

See also **output()**.

SWFMovie->setbackground (unknown)

Sets the background color.

void **swfmovie->setbackground** (int red, int green, int blue) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->setbackground() sets the background color. Why is there no rgba version? Think about it. (Actually, that's not such a dumb question after all- you might want to let the html background show through. There's a way to do that, but it only works on IE4. Search the http://www.macromedia.com/ site for details.)

SWFMovie->setdimension (unknown)

Sets the movie's width and height.

void **swfmovie->setdimension** (int width, int height) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->setdimension() sets the movie's width to *`width`* and height to *`height`*.

SWFMovie->setframes (unknown)

Sets the total number of frames in the animation.

void **swfmovie->setframes** (string numberofframes) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->setframes() sets the total number of frames in the animation to *`numberofframes`*.

SWFMovie->setrate (unknown)

Sets the animation's frame rate.

void **swfmovie->setrate** (int rate) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->setrate() sets the frame rate to *`rate`*, in frame per seconds. Animation will slow down if the player can't render frames fast enough- unless there's a streaming sound, in which case display frames are sacrificed to keep sound from skipping.

SWFMovie->streammp3 (unknown)

Streams a MP3 file.

void **swfmovie->streammp3** (string mp3FileName) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie->streammp3() streams the mp3 file *`mp3FileName`*. Not very robust in dealing with oddities (can skip over an initial ID3 tag, but that's about it). Like **SWFShape->addJpegFill()**, this isn't a stable function- we'll probably need to make a separate SWFSound object to contain sound types.

Note that the movie isn't smart enough to put enough frames in to contain the entire mp3 stream- you'll have to add (length of song * frames per second) frames to get the entire stream in.

Yes, now you can use ming to put that rock and roll devil worship music into your SWF files. Just don't tell the RIAA.

Ejemplo 1. swfmovie->streammp3() example

```
<?php
  $m = new SWFMovie();
  $m->setRate(12.0);
  $m->streamMp3("distortobass.mp3");
  // use your own MP3
```

```
  // 11.85 seconds at 12.0 fps = 142 frames
  $m->setFrames(142);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMovie (PHP 4 >= 4.0.5)

Creates a new movie object, representing an SWF version 4 movie.

new **swfmovie** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfmovie() creates a new movie object, representing an SWF version 4 movie.

SWFMovie has the following methods : **swfmovie->output()**,**swfmovie->save()**, **swfmovie->add()**, **swfmovie->remove()**, **swfmovie->nextframe()**, **swfmovie->setbackground()**, **swfmovie->setrate()**, **swfmovie->setdimension()**, **swfmovie->setframes()** and **swfmovie->streammp3()**.

See examples in : **swfdisplayitem->rotateto()**, **swfshape->setline()**, **swfshape->addfill()**... Any example will use this object.

SWFShape->addFill (unknown)

Adds a solid fill to the shape.

void **swfshape->addfill** (int red, int green, int blue [, int a]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

void **swfshape->addfill** (SWFbitmap bitmap [, int flags]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

void **swfshape->addfill** (SWFGradient gradient [, int flags]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfshape->addfill() adds a solid fill to the shape's list of fill styles. **swfshape->addfill()** accepts three different types of arguments.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

The *bitmap* argument is an swfbitmap() object. The *flags* argument can be one of the following values : SWFFILL_CLIPPED_BITMAP or SWFFILL_TILED_BITMAP. Default is SWFFILL_TILED_BITMAP. I think.

The *gradient* argument is an swfgradient() object. The flags argument can be one of the following values : SWFFILL_RADIAL_GRADIENT or SWFFILL_LINEAR_GRADIENT. Default is SWFFILL_LINEAR_GRADIENT. I'm sure about this one. Really.

swfshape->addfill() returns an swffill() object for use with the **swfshape->setleftfill()** and **swfshape->setrightfill()** functions described below.

See also **swfshape->setleftfill()** and **swfshape->setrightfill()**.

This simple example will draw a frame on a bitmap. Ah, here's another buglet in the flash player- it doesn't seem to care about the second shape's bitmap's transformation in a morph. According to spec, the bitmap should stretch along with the shape in this example..

Ejemplo 1. swfshape->addfill() example

```
<?php

  $p = new SWFMorph();

  $b = new SWFBitmap("alphafill.jpg");
  // use your own bitmap
  $width = $b->getWidth();
  $height = $b->getHeight();
```

```
  $s = $p->getShape1();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
  $f->moveTo(-$width/2, -$height/4);
  $f->scaleTo(1.0, 0.5);
  $s->setLeftFill($f);
  $s->movePenTo(-$width/2, -$height/4);
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);

  $s = $p->getShape2();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);

  // these two have no effect!
  $f->moveTo(-$width/4, -$height/2);
  $f->scaleTo(0.5, 1.0);

  $s->setLeftFill($f);
  $s->movePenTo(-$width/4, -$height/2);
  $s->drawLine($width/2, 0);
  $s->drawLine(0, $height);
  $s->drawLine(-$width/2, 0);
  $s->drawLine(0, -$height);

  $m = new SWFMovie();
  $m->setDimension($width, $height);
  $i = $m->add($p);
  $i->moveTo($width/2, $height/2);

  for($n=0; $n<1.001; $n+=0.03)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->drawCurve (unknown)

Draws a curve (relative).

void **swfshape->drawcurve** (int controldx, int controldy, int anchordx, int anchordy) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfshape->drawcurve() draws a quadratic curve (using the current line style,set by **swfshape->setline()**) from the current pen position to the relative position (`anchorx`,`anchory`) using relative control point (`controlx`,`controly`). That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawCurveTo (unknown)

Draws a curve.

void **swfshape->drawcurveto** (int controlx, int controly, int anchorx, int anchory) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfshape->drawcurveto() draws a quadratic curve (using the current line style, set by **swfshape->setline()**) from the current pen position to (`anchorx`,`anchory`) using (`controlx`,`controly`) as a control point. That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawLine (unknown)

Draws a line (relative).

void **swfshape->drawline** (int dx, int dy) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfshape->drawline() draws a line (using the current line style set by **swfshape->setline()**) from the current pen position to displacement (*dx*,*dy*).

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawlineto()**.

SWFShape->drawLineTo (unknown)

Draws a line.

void **swfshape->drawlineto** (int x, int y) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfshape->setrightfill() draws a line (using the current line style, set by **swfshape->setline()**) from the current pen position to point (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawline()**.

SWFShape->movePen (unknown)

Moves the shape's pen (relative).

void **swfshape->movepen** (int dx, int dy) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfshape->setrightfill() move the shape's pen from coordinates (current x,current y) to (current x + *dx*, current y + *dy*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->movePenTo (unknown)

Moves the shape's pen.

void **swfshape->movepento** (int x, int y) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfshape->setrightfill() move the shape's pen to (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepen()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->setLeftFill (unknown)

Sets left rasterizing color.

void **swfshape->setleftfill** (swfgradient fill) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

void **swfshape->setleftfill** (int red, int green, int blue [, int a]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

What this nonsense is about is, every edge segment borders at most two fills. When rasterizing the object, it's pretty handy to know what those fills are ahead of time, so the swf format requires these to be specified.

swfshape->setleftfill() sets the fill on the left side of the edge- that is, on the interior if you're defining the outline of the shape in a counter-clockwise fashion. The fill object is an SWFFill object returned from one of the addFill functions above.

This seems to be reversed when you're defining a shape in a morph, though. If your browser crashes, just try setting the fill on the other side.

Shortcut for `swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));`.

See also **swfshape->setrightfill()**.

SWFShape->setLine (unknown)

Sets the shape's line style.

void **swfshape->setline** (int width [, int red [, int green [, int blue [, int a]]]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfshape->setline() sets the shape's line style. *`width`* is the line's width. If *`width`* is 0, the line's style is removed (then, all other arguments are ignored). If *`width`* > 0, then line's color is set to *`red`*, *`green`*, *`blue`*. Last parameter *`a`* is optional.

swfshape->setline() accepts 1, 4 or 5 arguments (not 3 or 2).

You must declare all line styles before you use them (see example).

This simple example will draw a big "!#%*@", in funny colors and gracious style.

Ejemplo 1. swfshape->setline() example

```
<?php
  $s = new SWFShape();
  $f1 = $s->addFill(0xff, 0, 0);
  $f2 = $s->addFill(0xff, 0x7f, 0);
  $f3 = $s->addFill(0xff, 0xff, 0);
  $f4 = $s->addFill(0, 0xff, 0);
  $f5 = $s->addFill(0, 0, 0xff);

  // bug: have to declare all line styles before you use them
  $s->setLine(40, 0x7f, 0, 0);
  $s->setLine(40, 0x7f, 0x3f, 0);
```

```
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->setLine(40, 0, 0x7f, 0);
  $s->setLine(40, 0, 0, 0x7f);

  $f = new SWFFont('Techno.fdb');

  $s->setRightFill($f1);
  $s->setLine(40, 0x7f, 0, 0);
  $s->drawGlyph($f, '!');
  $s->movePen($f->getWidth('!'), 0);

  $s->setRightFill($f2);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->drawGlyph($f, '#');
  $s->movePen($f->getWidth('#'), 0);

  $s->setRightFill($f3);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->drawGlyph($f, '%');
  $s->movePen($f->getWidth('%'), 0);

  $s->setRightFill($f4);
  $s->setLine(40, 0, 0x7f, 0);
  $s->drawGlyph($f, '*');
  $s->movePen($f->getWidth('*'), 0);

  $s->setRightFill($f5);
  $s->setLine(40, 0, 0, 0x7f);
  $s->drawGlyph($f, '@');

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setRate(12.0);
  $i = $m->add($s);
  $i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setRightFill (unknown)

Sets right rasterizing color.

void **swfshape->setrightfill** (swfgradient fill) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

void **swfshape->setrightfill** (int red, int green, int blue [, int a]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

See also **swfshape->setleftfill()**.

Shortcut for `swfshape->setrightfill($s->addfill($r, $g, $b [, $a]));`.

SWFShape (PHP 4 >= 4.0.5)

Creates a new shape object.

new **swfshape** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfshape() creates a new shape object.

SWFShape has the following methods : **swfshape->setline()**, **swfshape->addfill()**, **swfshape->setleftfill()**, **swfshape->setrightfill()**, **swfshape->movepento()**, **swfshape->movepen()**, **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->drawcurveto()** and **swfshape->drawcurve()**.

This simple example will draw a big red elliptic quadrant.

Ejemplo 1. swfshape() example

```
<?php
  $s = new SWFShape();
  $s->setLine(40, 0x7f, 0, 0);
```

```
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(200, 200);
  $s->drawLineTo(6200, 200);
  $s->drawLineTo(6200, 4600);
  $s->drawCurveTo(200, 4600, 200, 200);

  $m = new SWFMovie();
  $m->setDimension(6400, 4800);
  $m->setRate(12.0);
  $m->add($s);
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFSprite->add (unknown)

Adds an object to a sprite

void **swfsprite->add** (resource object) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfsprite->add() adds a swfshape(), a swfbutton(), a swftext(), a swfaction() or a swfsprite() object.

For displayable types (swfshape(), swfbutton(), swftext(), swfaction() or swfsprite()), this returns a handle to the object in a display list.

SWFSprite->nextframe (unknown)

Moves to the next frame of the animation.

void **swfsprite->nextframe** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfsprite->setframes() moves to the next frame of the animation.

SWFSprite->remove (unknown)

Removes an object to a sprite

void **swfsprite->remove** (ressource object) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfsprite->remove() remove a swfshape(), a swfbutton(), a swftext(), a swfaction() or a swfsprite() object from the sprite.

SWFSprite->setframes (unknown)

Sets the total number of frames in the animation.

void **swfsprite->setframes** (int numberofframes) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfsprite->setframes() sets the total number of frames in the animation to `numberofframes`.

SWFSprite (PHP 4 >= 4.0.5)

Creates a movie clip (a sprite)

new **swfsprite** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swfsprite() are also known as a "movie clip", this allows one to create objects which are animated in their own timelines. Hence, the sprite has most of the same methods as the movie.

swfsprite() has the following methods : **swfsprite->add()**, **swfsprite->remove()**, **swfsprite->nextframe()** and **swfsprite->setframes()**.

This simple example will spin gracefully a big red square.

Ejemplo 1. swfsprite() example

```
<?php
  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->moveTo(1500,1000);
  $i->setName("blah");
```

```
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(3000,2000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFText->addString (unknown)

Draws a string

void **swftext->addstring** (string string) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftext->addstring() draws the string *`string`* at the current pen (cursor) location. Pen is at the baseline of the text; i.e., ascending text is in the -y direction.

SWFText->getWidth (unknown)

Computes string's width

void **swftext->addstring** (string string) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftext->addstring() returns the rendered width of the *`string`* string at the text object's current font, scale, and spacing settings.

SWFText->moveTo (unknown)

Moves the pen

void **swftext->moveto** (int x, int y) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftext->moveto() moves the pen (or cursor, if that makes more sense) to (x,y) in text object's coordinate space. If either is zero, though, value in that dimension stays the same. Annoying, should be fixed.

SWFText->setColor (unknown)

Sets the current font color

void **swftext->setcolor** (int red, int green, int blue [, int a]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftext->setspacing() changes the current text color. Default is black. I think. Color is represented using the RGB system.

SWFText->setFont (unknown)

Sets the current font

void **swftext->setfont** (string font) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftext->setfont() sets the current font to `font`.

SWFText->setHeight (unknown)

Sets the current font height

void **swftext->setheight** (int height) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftext->setheight() sets the current font height to `height`. Default is 240.

SWFText->setSpacing (unknown)

Sets the current font spacing

void **swftext->setspacing** (float spacing) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftext->setspacing() sets the current font spacing to `spacingspacing`. Default is 1.0. 0 is all of the letters written at the same point. This doesn't really work that well because it inflates the advance across the letter, doesn't add the same amount of spacing between the letters. I should try and explain that better, prolly. Or just fix the damn thing to do constant spacing. This was really just a way to figure out how letter advances work, anyway.. So nyah.

SWFText (PHP 4 >= 4.0.5)

Creates a new SWFText object.

new **swftext** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftext() creates a new SWFText object, fresh for manipulating.

SWFText has the following methods : **swftext->setfont()**, **swftext->setheight()**, **swftext->setspacing()**, **swftext->setcolor()**, **swftext->moveto()**, **swftext->addstring()** and **swftext->getwidth()**.

This simple example will draw a big yellow "PHP generates Flash with Ming" text, on white background.

Ejemplo 1. swftext() example

```
<?php
  $f = new SWFFont("Techno.fdb");
  $t = new SWFText();
  $t->setFont($f);
  $t->moveTo(200, 2400);
  $t->setColor(0xff, 0xff, 0);
  $t->setHeight(1200);
  $t->addString("PHP generates Flash with Ming!!");

  $m = new SWFMovie();
  $m->setDimension(5400, 3600);

  $m->add($t);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFTextField->addstring (unknown)

Concatenates the given string to the text field

void **swftextfield->addstring** (string string) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setname() concatenates the string `string` to the text field.

SWFTextField->align (unknown)

Sets the text field alignment

void **swftextfield->align** (int alignement) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->align() sets the text field alignment to `alignement`. Valid values for `alignement` are : SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER and SWFTEXTFIELD_ALIGN_JUSTIFY.

SWFTextField->setbounds (unknown)

Sets the text field width and height

void **swftextfield->setbounds** (int width, int height) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setbounds() sets the text field width to `width` and height to `height`. If you don't set the bounds yourself, Ming makes a poor guess at what the bounds are.

SWFTextField->setcolor (unknown)

Sets the color of the text field.

void **swftextfield->setcolor** (int red, int green, int blue [, int a]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setcolor() sets the color of the text field. Default is fully opaque black. Color is represented using RGB system.

SWFTextField->setFont (unknown)

Sets the text field font

void **swftextfield->setfont** (string font) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setfont() sets the text field font to the [browser-defined?] `font` font.

SWFTextField->setHeight (unknown)

Sets the font height of this text field font.

void **swftextfield->setheight** (int height) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setheight() sets the font height of this text field font to the given height `height`. Default is 240.

SWFTextField->setindentation (unknown)

Sets the indentation of the first line.

void **swftextfield->setindentation** (int width) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setindentation() sets the indentation of the first line in the text field, to `width`.

SWFTextField->setLeftMargin (unknown)

Sets the left margin width of the text field.

void **swftextfield->setleftmargin** (int width) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setleftmargin() sets the left margin width of the text field to `width`. Default is 0.

SWFTextField->setLineSpacing (unknown)

Sets the line spacing of the text field.

void **swftextfield->setlinespacing** (int height) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setlinespacing() sets the line spacing of the text field to the height of `height`. Default is 40.

SWFTextField->setMargins (unknown)

Sets the margins width of the text field.

void **swftextfield->setmargins** (int left, int right) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setmargins() set both margins at once, for the man on the go.

SWFTextField->setname (unknown)

Sets the variable name

void **swftextfield->setname** (string name) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setname() sets the variable name of this text field to `name`, for form posting and action scripting purposes.

SWFTextField->setrightMargin (unknown)

Sets the right margin width of the text field.

void **swftextfield->setrightmargin** (int width) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield->setrightmargin() sets the right margin width of the text field to `width`. Default is 0.

SWFTextField (PHP 4 >= 4.0.5)

Creates a text field object

new **swftextfield** ([int flags]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

swftextfield() creates a new text field object. Text Fields are less flexible than swftext() objects- they can't be rotated, scaled non-proportionally, or skewed, but they can be used as form entries, and they can use browser-defined fonts.

The optional flags change the text field's behavior. It has the following possibles values :

- SWFTEXTFIELD_DRAWBOX draws the outline of the textfield
- SWFTEXTFIELD_HASLENGTH
- SWFTEXTFIELD_HTML allows text markup using HTML-tags
- SWFTEXTFIELD_MULTILINE allows multiple lines
- SWFTEXTFIELD_NOEDIT indicates that the field shouldn't be user-editable
- SWFTEXTFIELD_NOSELECT makes the field non-selectable
- SWFTEXTFIELD_PASSWORD obscures the data entry
- SWFTEXTFIELD_WORDWRAP allows text to wrap

Flags are combined with the bitwise OR operation. For example,

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

creates a totally useless non-editable password field.

SWFTextField has the following methods : **swftextfield->setfont()**, **swftextfield->setbounds()**, **swftextfield->align()**, **swftextfield->setheight()**, **swftextfield->setleftmargin()**, **swftextfield->setrightmargin()**, **swftextfield->setmargins()**, **swftextfield->setindentation()**, **swftextfield->setlinespacing()**, **swftextfield->setcolor()**, **swftextfield->setname()** and **swftextfield->addstring()**.

LX. Miscelánea de funciones

Estas funciones están colacadas aquí debido a que no parecen ajustarse a ninguna otra categoría.

connection_aborted (PHP 3>= 3.0.7, PHP 4)

Devuelve TRUE si el cliente está desconectado

int **connection_aborted** (void) \linebreak

Devuelve TRUE si el cliente está desconectado. Vea la descripción de la Gestión de la Conexión en el capítulo Características para una explicación completa.

connection_status (PHP 3>= 3.0.7, PHP 4)

Devuelve el estado de la conexión en un campo de bits

int **connection_status** (void) \linebreak

Devuelve el estado de la conexión en un campo de bits. Vea la descripción de la Gestión de la Conexión en el capítulo Características para una explicación completa.

connection_timeout (PHP 3>= 3.0.7, PHP 4 <= 4.0.4)

Devuelve TRUE si el script ha alcanzado su time out

int **connection_timeout** (void) \linebreak

Devuelve TRUE si el script ha alcanzado su time out. Vea la descripción de la Gestión de la Conexión en el capítulo Características para una explicación completa.

constant (PHP 4 >= 4.0.4)

Returns the value of a constant

mixed **constant** (string name) \linebreak

constant() will return the value of the constant indicated by *name*.

constant() is useful if you need to retrieve the value of a constant, but do not know it's name. i.e. It is stored in a variable or returned by a function.

Ejemplo 1. constant() example

```
<?php

define ("MAXSIZE", 100);
```

```
echo MAXSIZE;
echo constant("MAXSIZE"); // same thing as the previous line

?>
```

See also define(), defined() and the section on Constants.

define (PHP 3, PHP 4)

Define una constante con nombre.

int **define** (string name, mixed value [, int case_insensitive]) \linebreak

Define una constante con nombre, que es similar a una variable, excepto que:

- Las constantes no tienen un símbolo dólar '$' precediéndolas;
- Las constantes son accesibles desde cualquier lugar sin tener en cuenta las reglas de ámbito de las variables.
- Las constantes no pueden ser redefinidas o iniciadas una vez que han sido establecidas, y
- Las constantes sólo pueden evaluar valores escalares

El nombre de la constante se da en *name* (nombre); el valor se da en *value* (valor).

El tercer parámetro opcional *case_insensitive* también se encuentra disponible. Si se da el valor *1*, la constante se definirá no dintinguiendo mayúsculas/minusculas. El comportamiento por defecto es si distinguir; i.e. CONSTANT y Constant representan valores diferentes.

Ejemplo 1. Definición de Constantes

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
?>
```

define() devuelve `TRUE` en caso de éxito y `FALSE` si ocurre un error.

Véase también defined() y la sección Constantes.

defined (PHP 3, PHP 4)

Comprueba que una constante con nombre dada existe.

int **defined** (string name) \linebreak

Devuelve `TRUE` si la constante con nombre dada en *`name`* (nombre) ha sido definida, `FALSE` en otro caso.

Véase también define() y la sección Constantes.

die (unknown)

Envía a la salida un mensaje y finaliza el script actual

void **die** (string message) \linebreak

Esta construcción del lenguaje envía a la salida un mensaje y finaliza la ejecución del script. No devuelve nada.

Ejemplo 1. Ejemplo die

```
<?php
$filename = '/path/to/data-file';
$file = fopen($filename, 'r')
  or die "unable to open file ($filename)";
?>
```

eval (unknown)

Evalúa una cadena de carácteres como código PHP

void **eval** (string code_str) \linebreak

eval() evalúa la cadena de carácteres dada en *`code_str`* como código PHP. Entre otras cosas, ésto puede ser útil para almacenar código en un campo de texto de base de datos para una ejecución posterior.

Hay algunos aspectos a tener en cuenta cuando se utiliza **eval()**. Recuerde que la cadena de carácteres pasada debe ser código PHP válido, incluyendo aspectos como sentencias de terminación con un punto y coma para que el parser no finalice en la línea después de **eval()**, y secuencias de formato correctas en *`code_str`*.

Recuerde también que las variables a las que se les da valor en **eval()** retendrán estos valores posteriormente en el script principal.

Ejemplo 1. Ejemplo eval() - fusión en un único texo

```
<?php
$string = 'cup';
$name = 'coffee';
$str = 'This is a $string with my $name in it.<br>';
echo $str;
eval( "\$str = \"$str\";" );
echo $str;
?>
```

El ejemplo anterior mostrará:

```
This is a $string with my $name in it.
This is a cup with my coffee in it.
```

exit (unknown)

Finaliza el script actual

void **exit** (void) \linebreak

Esta construcción del lenguaje finaliza la ejecución del script. No devuelve nada.

get_browser (PHP 3, PHP 4)

Informa sobre lo que es capaz de hacer el navegador (browser) del usuario.

object **get_browser** ([string user_agent]) \linebreak

get_browser() intenta determinar las características del navegador del usuario. Para ello consulta el fichero de información del navegador, `browscap.ini`. Por defecto, se utiliza el valor de $HTTP_USER_AGENT; sin embargo, puede alterar ésto (i.e., consultando otra información del navegador) pasando el parámetro opcional *`user_agent`* a **get_browser()**.

La información se devuelve en un objto, que contendrá varios elementos de datos que representan, por ejemplo, los números de versión (mayor y menor) del navegador y la cadena ID; valores `TRUE`/false para características como los marcos, JavaScript, y cookies; etc.

`browscap.ini` contiene información de muchos navegadores, depende de las actualizaciones del usuario para mantener la base de datos actualizada. El formato del fichero es claramente auto-explicativo.

El ejemplo siguiente muestra como se puede listar toda la información disponible recuperada del navegador del usuario.

Ejemplo 1. ejemplo get_browser()

```
<?php
function list_array( $array ) {
   while ( list( $key, $value ) = each( $array ) ) {
  $str .= "<b>$key:</b> $value<br>\n";
   }
   return $str;
}
echo "$HTTP_USER_AGENT<hr>\n";
$browser = get_browser();
echo list_array( (array) $browser );
?>
```

La salida del script anterior debería paracerse a ésto:

```
Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)<hr>
<b>browser_name_pattern:</b> Mozilla/4\.5.*<br>
<b>parent:</b> Netscape 4.0<br>
<b>platform:</b> Unknown<br>
<b>majorver:</b> 4<br>
<b>minorver:</b> 5<br>
<b>browser:</b> Netscape<br>
<b>version:</b> 4<br>
<b>frames:</b> 1<br>
<b>tables:</b> 1<br>
<b>cookies:</b> 1<br>
<b>backgroundsounds:</b> <br>
<b>vbscript:</b> <br>
<b>javascript:</b> 1<br>
<b>javaapplets:</b> 1<br>
<b>activexcontrols:</b> <br>
<b>beta:</b> <br>
<b>crawler:</b> <br>
<b>authenticodeupdate:</b> <br>
<b>msn:</b> <br>
```

Para conseguir ésto, su opción de fichero de configuración browscap debe apuntar a la correcta localización del fichero `browscap.ini`.

Para más información (incluyendo localizaciones desde las que puede obtener un fichero `browscap.ini`), consulte las FAQ sobre PHP en http://www.php.net/FAQ.html (http://www.php.net/FAQ.php).

> **Nota:** el soporte para browscap fue añadido en la versión 3.0b2 de PHP.

highlight_file (PHP 4)

Syntax highlighting of a file

mixed **highlight_file** (string filename [, bool return]) \linebreak

The **highlight_file()** function prints out a syntax higlighted version of the code contained in *filename* using the colors defined in the built-in syntax highlighter for PHP.

If the second parameter *return* is set to TRUE then **highlight_file()** will return the highlighted code as a string instead of printing it out. If the second parameter is not set to TRUE then **highlight_file()** will return TRUE on success, FALSE on failure.

Nota: The *return* parameter became available in PHP 4.2.0. Before this time it behaved like the default, which is FALSE

Nota: Care should be taken when using the show_source() and **highlight_file()** functions to make sure that you do not inadvertently reveal sensitive information such as passwords or any other type of information that might create a potential security risk.

Nota: Since PHP 4.2.1 this function is also affected by safe_mode and open_basedir.

Ejemplo 1. Creating a source highlighting URL

To setup a URL that can code hightlight any script that you pass to it, we will make use of the "ForceType" directive in Apache to generate a nice URL pattern, and use the function **highlight_file()** to show a nice looking code list.

In your httpd.conf you can add the following:

```
<Location /source>
    ForceType application/x-httpd-php
</Location>
```

And then make a file named "source" and put it in your web root directory.

```
<HTML>
<HEAD>
<TITLE>Source Display</TITLE>
```

```
</HEAD>
<BODY BGCOLOR="white">
<?php
    $script = getenv ("PATH_TRANSLATED");
    if(!$script) {
    echo "<BR><B>ERROR: Script Name needed</B><BR>";
    } else {
    if (ereg("(\.php|\.inc)$",$script)) {
    echo "<H1>Source of: $PATH_INFO</H1>\n<HR>\n";
    highlight_file($script);
    } else {
    echo "<H1>ERROR: Only PHP or include script names are allowed</H1>";
    }
    }
    echo "<HR>Processed: ".date("Y/M/d H:i:s",time());
?>
</BODY>
</HTML>
```

Then you can use an URL like the one below to display a colorized version of a script located in "/path/to/script.php" in your web site.

```
http://your.server.com/source/path/to/script.php
```

See also highlight_string(), show_source().

highlight_string (PHP 4)

Syntax highlighting of a string

mixed **highlight_string** (string str [, bool return]) \linebreak

The **highlight_string()** function outputs a syntax highlighted version of *`str`* using the colors defined in the built-in syntax highlighter for PHP.

If the second parameter *`return`* is set to `TRUE` then **highlight_string()** will return the highlighted code as a string instead of printing it out. If the second parameter is not set to `TRUE` then **highlight_string()** will return `TRUE` on success, `FALSE` on failure.

Nota: The *return* parameter became available in PHP 4.2.0. Before this time it behaved like the default, which is FALSE

See also highlight_file(), and show_source().

ignore_user_abort (PHP 3>= 3.0.7, PHP 4)

Establece si la desconexión de un cliente debe suspender la ejecución del script

int **ignore_user_abort** ([int setting]) \linebreak

Esta función establece si la desconexión de un cliente debe provocar la suspensión del script. Devolverá el valor previo y puede ser llamada sin argumentos para devolver el valor actual y no cambiarlo. Véase la sección sobre la Gestión de la Conexión en el capítulo Características para una descripción completa de la gestión de la conexión en PHP.

leak (PHP 3, PHP 4)

Filtra memoria

void **leak** (int bytes) \linebreak

leak() filtra la cantidad de memoria especificada.

Es muy útil cuando se depura el gestor de memoria, que automáticamente libera la memoria "filtrada" después de que se completa cada petición.

pack (PHP 3, PHP 4)

empaqueta datos en una cadena binaria

string **pack** (string format [, mixed args]) \linebreak

Empaqueta los argumentos dados en una cadena binaria siguiendo el formato *format*. Devuelve la cadena binaria que contiene los datos.

El concepto de esta función fue tomado de Perl y todos los códigos de formateo realizan la misma función. La cadena de formato consiste en códigos de formato seguidos por un argumento opcional de repetición. El argumento de repetición puede ser un valor entero o * para repetir hasta el fin de la entrada de datos. Para a, A, h, H la cuenta de repetición representa cuántos caracteres se toman de un argumento de datos, para @ es la posición absoluta donde poner los datos siguientes, para todo lo demás la cuenta de repetición especifica cuántos argumentos de datos se toman y empaquetan en la cadena binaria resultante. Actualmente están implementados:

- a cadena rellena de NUL
- A cadena rellena de ESPACIOS
- h cadena Hex, primero el medio byte inferior
- H cadena Hex, primero el medio byte superior
- c signed (con signo) char
- C unsigned (sin signo) char
- s signed short (siempre 16 bits, distribución de bytes de la máquina)
- S unsigned short (siempre 16 bits, distribución de bytes de la máquina)
- n unsigned short (siempre 16 bits, distribución de bytes gran endian)
- v unsigned short (siempre 16 bits, distribución de bytes pequeño endian)
- i signed integer (distribución de bytes y tamaños dependientes de la máquina)
- I unsigned integer (distribución de bytes y tamaños dependientes de la máquina)
- l signed long (siempre 32 bits, distribución de bytes de la máquina)
- L unsigned long (siempre 32 bits, distribución de bytes de la máquina)
- N unsigned long (siempre 32 bits, distribución de bytes gran endian)
- V unsigned long (siempre 32 bits, distribución de bytes pequeño endian)
- f float (representación y tamaño dependientes de la máquina)
- d double (representación y tamaño dependientes de la máquina)
- x byte NUL
- X Un byte hacia atrás
- @ relleno con NUL en la posición absoluta

Ejemplo 1. cadena de formato para pack

```
$binarydata = pack("nvc*", 0x1234, 0x5678, 65, 66);
```

La cadena binaria resultante tendrá 6 bytes de longitud y contendrá la secuencia de bytes 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Adviértase que la distinción entre valores signed (con signo) y unsigned (sin signo) sólo afecta a la función unpack(), ya que la función **pack()** da el mismo resultado para códigos de formato con signo y sin signo.

Nótese también que internamente PHP almacena valores enteros como valores con signo de un tamaño dependiente de la máquina. Si le da un valor entero sin signo demasiado grande para ser almacenado, será convertido a un double (doble), lo que a menudo produce resultados no deseados.

show_source (PHP 4)

Syntax highlighting of a file

bool **show_source** (string filename [, bool return]) \linebreak

This function is an alias to highlight_file(). For more information see the documentation there.

See also highlight_string() and highlight_file().

sleep (PHP 3, PHP 4)

Ejecución retardada

void **sleep** (int seconds) \linebreak

La función sleep retarda la ejecución del programa durante el número de *seconds* (segundos) dado.

Véase también usleep().

uniqid (PHP 3, PHP 4)

Genera un id único.

int **uniqid** (string prefix [, boolean lcg]) \linebreak

uniqid() devuelve un identificador único con un prefijo basado en la hora actual en microsegundos. El prefijo puede ser práctico por ejemplo si se generan identificadores simultáneamente en varios host que pueden haber generado el identificador en el mismo microsegundo. *prefix* (prefijo) puede ser de hasta 114 carácteres de longitud.

Si el parámetro opcional *lcg* es TRUE, **uniqid()** añadirá entropía "LCG combinada" al final del valor devuelto, que hará el resultado más único.

Con un *prefix* (prefijo) vacío, la cadena devuelta tendrá una longitud de 13 carácteres. Si *lcg* es TRUE, tendrá 23 carácteres.

Nota: El parámetro *lcg* está disponible sólo en PHP 4 y PHP 3.0.13 y posteriores.

Si necesita un identificador único o testigo, y tiene la intención de hacer público ese testigo al usuario por medio de una red (i.e. cookies de sesión) se recomienda que utilice algo parecido a estas líneas

```
$token = md5(uniqid("")); // no random portion
$better_token = md5(uniqid(rand())); // better, difficult to guess
```

Esto creará un identificador de 32 carácteres (un número hexadecimal de 128 bits) que es extremadamente difícil de predecir.

unpack (PHP 3, PHP 4)

desempaqueta datos de una cadena binaria

array **unpack** (string format, string data) \linebreak

Desempaqueta datos de una cadena binaria en un array, de acuerdo al formato *`format`*. Devuelve un array que contiene los elementos de la cadena binaria desempaquetados.

Unpack funciona de manera ligeramente diferente a Perl, ya que los datos desempaquetados se almacenan en un array asociativo. Para conseguir ésto debe nombrar los diferentes códigos de formato y separarlos por una barra inclinada /.

Ejemplo 1. cadena de formato unpack

```
$array = unpack("c2chars/nint", $binarydata);
```

El array resultante contendrá las entradas "chars1", "chars2" y "int".

Para una explicación de los códigos de formato véase también: pack()

Advierta que PHP almacena internamente los valores enteros con signo. Si desempaqueta un unsigned long (largo sin signo) demasiado grande y es del mismo tamaño tal como PHP almacena internamente los valores, el resultado será un número negativo a pesar de que se especificara desempaquetamiento sin signo.

usleep (PHP 3, PHP 4)

Retrasa la ejecución, en microsegundos

void **usleep** (int micro_seconds) \linebreak

La función usleep retrasa la ejecución del programa durante un número de *`micro_seconds`* (microsegundos) dado.

Véase también sleep().

LXI. mnoGoSearch Functions

These functions allow you to access the mnoGoSearch (former UdmSearch) free search engine. In order to have these functions available, you must compile PHP with mnogosearch support by using the `--with-mnogosearch` option. If you use this option without specifying the path to mnogosearch, PHP will look for mnogosearch under /usr/local/mnogosearch path by default. If you installed mnogosearch at other path you should specify it: `--with-mnogosearch=DIR`.

mnoGoSearch is a full-featured search engine software for intranet and internet servers, distributed under the GNU license. mnoGoSearch has number of unique features, which makes it appropriate for a wide range of application from search within your site to a specialized search system such as cooking recipes or newspaper search, FTP archive search, news articles search, etc. It offers full-text indexing and searching for HTML, PDF, and text documents. mnoGoSearch consists of two parts. The first is an indexing mechanism (indexer). The purpose of indexer is to walk through HTTP, FTP, NEWS servers or local files, recursively grabbing all the documents and storing meta-data about that documents in a SQL database in a smart and effective manner. After every document is referenced by its corresponding URL, meta-data collected by indexer is used later in a search process. The search is performed via Web interface. C CGI, PHP and Perl search front ends are included.

> **Nota:** PHP contains built-in mysql access library, which can be used to access MySQL. It is known that mnoGoSearch is not compatible with this built-in library and can work only with generic MySQL libraries. Thus, if you use mnoGoSearch with mysql, during PHP configuration you have to indicate directory of MySQL installation, that was used during mnoGoSearch configuration, i.e. for example: `--with-mnogosearch --with-mysql=/usr`.

You need at least version 3.1.10 of mnoGoSearch installed to use these functions.

More information about mnoGoSearch can be found at http://www.mnogosearch.ru/.

udm_add_search_limit (PHP 4 >= 4.0.5)

Add various search limits

int **udm_add_search_limit** (int agent, int var, string val) \linebreak

udm_add_search_limit() returns `TRUE` on success, `FALSE` on error. Adds search restrictions.

`agent` - a link to Agent, received after call to udm_alloc_agent().

`var` - defines parameter, indicating limit.

`val` - defines value of the current parameter.

Possible *`var`* values:

- UDM_LIMIT_URL - defines document URL limitations to limit search through subsection of database. It supports SQL % and _ LIKE wildcards, where % matches any number of characters, even zero characters, and _ matches exactly one character. E.g. http://my.domain.__/catalog may stand for http://my.domain.ru/catalog and http://my.domain.ua/catalog.
- UDM_LIMIT_TAG - defines site TAG limitations. In indexer-conf you can assign specific TAGs to various sites and parts of a site. Tags in mnoGoSearch 3.1.x are lines, that may contain metasymbols % and _. Metasymbols allow searching among groups of tags. E.g. there are links with tags ABCD and ABCE, and search restriction is by ABC_ - the search will be made among both of the tags.
- UDM_LIMIT_LANG - defines document language limitations.
- UDM_LIMIT_CAT - defines document category limitations. Categories are similar to tag feature, but nested. So you can have one category inside another and so on. You have to use two characters for each level. Use a hex number going from 0-F or a 36 base number going from 0-Z. Therefore a top-level category like ’Auto’ would be 01. If it has a subcategory like ’Ford’, then it would be 01 (the parent category) and then ’Ford’ which we will give 01. Put those together and you get 0101. If ’Auto’ had another subcategory named ’VW’, then it’s id would be 01 because it belongs to the ’Ford’ category and then 02 because it’s the next category. So it’s id would be 0102. If VW had a sub category called ’Engine’ then it’s id would start at 01 again and it would get the ’VW’ id 02 and ’Auto’ id of 01, making it 010201. If you want to search for sites under that category then you pass it cat=010201 in the url.
- UDM_LIMIT_DATE - defines limitation by date document was modified.

  Format of parameter value: a string with first character < or >, then with no space - date in unixtime format, for example:

  Udm_Add_Search_Limit($udm,UDM_LIMIT_DATE,"<908012006");

  If > character is used, then search will be restricted to those documents having modification date greater than entered. If <, then smaller.

udm_alloc_agent (PHP 4 >= 4.0.5)

Allocate mnoGoSearch session

int **udm_alloc_agent** (string dbaddr [, string dbmode]) \linebreak

udm_alloc_agent() returns mnogosearch agent identifier on success, `FALSE` on error. This function creates a session with database parameters.

`dbaddr` - URL-style database description. Options (type, host, database name, port, user and password) to connect to SQL database. Do not matter for built-in text files support. Format: DBAddr DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/ Currently supported DBType values are: mysql, pgsql, msql, solid, mssql, oracle, ibase. Actually, it does not matter for native libraries support. But ODBC users should specify one of supported values. If your database type is not supported, you may use "unknown" instead.

`dbmode` - You may select SQL database mode of words storage. When "single" is specified, all words are stored in the same table. If "multi" is selected, words will be located in different tables depending of their lengths. "multi" mode is usually faster but requires more tables in database. If "crc" mode is selected, mnoGoSearch will store 32 bit integer word IDs calculated by CRC32 algorythm instead of words. This mode requres less disk space and it is faster comparing with "single" and "multi" modes. "crc-multi" uses the same storage structure with the "crc" mode, but also stores words in different tables depending on words lengths like "multi" mode. Format: DBMode single/multi/crc/crc-multi

Nota: *`dbaddr`* and *`dbmode`* must match those used during indexing.

Nota: In fact this function does not open connection to database and thus does not check entered login and password. Actual connection to database and login/password verification is done by udm_find().

udm_api_version (PHP 4 >= 4.0.5)

Get mnoGoSearch API version.

int **udm_api_version** (void) \linebreak

udm_api_version() returns mnoGoSearch API version number. E.g. if mnoGoSearch 3.1.10 API is used, this function will return `30110`.

This function allows user to identify which API functions are available, e.g. udm_get_doc_count() function is only available in mnoGoSearch 3.1.11 or later.

Example:

```
if (udm_api_version() >= 30111) {
```

```
print  "Total number of urls in database: ".udm_get_doc_count($udm)."<br>\n";
}
```

udm_cat_list (PHP 4 >= 4.0.6)

Get all the categories on the same level with the current one.

array **udm_cat_list** (int agent, string category) \linebreak

udm_cat_list() returns array listing all categories of the same level as current category in the categories tree.

The function can be useful for developing categories tree browser.

Returns array with the following format:

The array consists of pairs. Elements with even index numbers contain category paths, odd elements contain corresponding category names.

$array[0] will contain ’020300’
$array[1] will contain ’Audi’
$array[2] will contain ’020301’
$array[3] will contain ’BMW’
$array[4] will contain ’020302’
$array[5] will contain ’Opel’
...
etc.

Following is an example of displaying links of the current level in format:
Audi
BMW
Opel
...

```
<?php
 $cat_list_arr = udm_cat_list($udm_agent,$cat);
 $cat_list = ”;
 for ($i=0; $i<count($cat_list_arr); $i+=2) {
   $path = $cat_list_arr[$i];
   $name = $cat_list_arr[$i+1];
   $cat_list .= "<a href=\"$PHP_SELF?cat=$path\">$name</a><br>";
 }
```

```
?>
```

udm_cat_path (PHP 4 >= 4.0.6)

Get the path to the current category.

array **udm_cat_path** (int agent, string category) \linebreak

udm_cat_path() returns array describing path in the categories tree from the tree root to the current category.

agent - agent link identifier.

category - current category - the one to get path to.

Returns array with the following format:

The array consists of pairs. Elements with even index numbers contain category paths, odd elements contain corresponding category names.

For example, the call `$array=udm_cat_path($agent, '02031D');` may return the following array:

$array[0] will contain ”
$array[1] will contain ’Root’
$array[2] will contain ’02’
$array[3] will contain ’Sport’
$array[4] will contain ’0203’
$array[5] will contain ’Auto’
$array[4] will contain ’02031D’
$array[5] will contain ’Ferrari’

Ejemplo 1. Specifying path to the current category in the following format: ’> Root > Sport > Auto > Ferrari’

```
<?php
  $cat_path_arr = udm_cat_path($udm_agent,$cat);
  $cat_path = ”;
  for ($i=0; $i<count($cat_path_arr); $i+=2) {
    $path = $cat_path_arr[$i];
    $name = $cat_path_arr[$i+1];
    $cat_path .= " > <a href=\"$PHP_SELF?cat=$path\">$name</a> ";
  }
?>
```

udm_check_charset (PHP 4 >= 4.2.0)

Check if the given charset is known to mnogosearch

int **udm_check_charset** (int agent, string charset) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

udm_check_stored (PHP 4 >= 4.2.0)

Check connection to stored

int **udm_check_stored** (int agent, int link, string doc_id) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

udm_clear_search_limits (PHP 4 >= 4.0.5)

Clear all mnoGoSearch search restrictions

int **udm_clear_search_limits** (int agent) \linebreak

udm_clear_search_limits() resets defined search limitations and returns `TRUE`.

udm_close_stored (PHP 4 >= 4.2.0)

Close connection to stored

int **udm_close_stored** (int agent, int link) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

udm_crc32 (PHP 4 >= 4.2.0)

Return CRC32 checksum of gived string

int **udm_crc32** (int agent, string str) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

udm_errno (PHP 4 >= 4.0.5)

Get mnoGoSearch error number

int **udm_errno** (int agent) \linebreak

udm_errno() returns mnoGoSearch error number, zero if no error.

`agent` - link to agent identifier, received after call to udm_alloc_agent().

Receiving numeric agent error code.

udm_error (PHP 4 >= 4.0.5)

Get mnoGoSearch error message

string **udm_error** (int agent) \linebreak

udm_error() returns mnoGoSearch error message, empty string if no error.

`agent` - link to agent identifier, received after call to udm_alloc_agent().

Receiving agent error message.

udm_find (PHP 4 >= 4.0.5)

Perform search

int **udm_find** (int agent, string query) \linebreak

udm_find() returns result link identifier on success, `FALSE` on error.

The search itself. The first argument - session, the next one - query itself. To find something just type words you want to find and press SUBMIT button. For example, "mysql odbc". You should not use quotes " in query, they are written here only to divide a query from other text. mnoGoSearch will find all documents that contain word "mysql" and/or word "odbc". Best documents having bigger weights will be displayed first. If you use search mode ALL, search will return documents that contain both (or more) words you entered. In case you use mode ANY, the search will return list of documents that contain any of the words you entered. If you want more advanced results you may use query language. You should select "bool" match mode in the search from.

mnoGoSearch understands the following boolean operators:

& - logical AND. For example, "mysql & odbc". mnoGoSearch will find any URLs that contain both "mysql" and "odbc".

| - logical OR. For example "mysql|odbc". mnoGoSearch will find any URLs, that contain word "mysql" or word "odbc".

~ - logical NOT. For example "mysql & ~odbc". mnoGoSearch will find URLs that contain word "mysql" and do not contain word "odbc" at the same time. Note that ~ just excludes given word from results. Query "~odbc" will find nothing!

() - group command to compose more complex queries. For example "(mysql | msql) & ~postgres". Query language is simple and powerful at the same time. Just consider query as usual boolean expression.

udm_free_agent (PHP 4 >= 4.0.5)

Free mnoGoSearch session

int **udm_free_agent** (int agent) \linebreak

udm_free_agent() returns `TRUE` on success, `FALSE` on error.

`agent` - link to agent identifier, received ‘ after call to udm_alloc_agent().

Freeing up memory allocated for agent session.

udm_free_ispell_data (PHP 4 >= 4.0.5)

Free memory allocated for ispell data

int **udm_free_ispell_data** (int agent) \linebreak

udm_free_ispell_data() always returns `TRUE`.

`agent` - agent link identifier, received after call to udm_alloc_agent().

> **Nota:** This function is supported beginning from version 3.1.12 of mnoGoSearch and it does not do anything in previous versions.

udm_free_res (PHP 4 >= 4.0.5)

Free mnoGoSearch result

int **udm_free_res** (int res) \linebreak

udm_free_res() returns `TRUE` on success, `FALSE` on error.

`res` - a link to result identifier, received after call to udm_find().

Freeing up memory allocated for results.

udm_get_doc_count (PHP 4 >= 4.0.5)

Get total number of documents in database.

int **udm_get_doc_count** (int agent) \linebreak

udm_get_doc_count() returns number of documents in database.

`agent` - link to agent identifier, received after call to udm_alloc_agent().

> **Nota:** This function is supported only in mnoGoSearch 3.1.11 or later.

udm_get_res_field (PHP 4 >= 4.0.5)

Fetch mnoGoSearch result field

string **udm_get_res_field** (int res, int row, int field) \linebreak

udm_get_res_field() returns result field value on success, `FALSE` on error.

`res` - a link to result identifier, received after call to udm_find().

`row` - the number of the link on the current page. May have values from 0 to *`UDM_PARAM_NUM_ROWS-1`*.

field - field identifier, may have the following values:

- UDM_FIELD_URL - document URL field
- UDM_FIELD_CONTENT - document Content-type field (for example, text/html).
- UDM_FIELD_CATEGORY - document category field. Use udm_cat_path() to get full path to current category from the categories tree root. (This parameter is available only in PHP 4.0.6 or later).
- UDM_FIELD_TITLE - document title field.
- UDM_FIELD_KEYWORDS - document keywords field (from META KEYWORDS tag).
- UDM_FIELD_DESC - document description field (from META DESCRIPTION tag).
- UDM_FIELD_TEXT - document body text (the first couple of lines to give an idea of what the document is about).
- UDM_FIELD_SIZE - document size.
- UDM_FIELD_URLID - unique URL ID of the link.
- UDM_FIELD_RATING - page rating (as calculated by mnoGoSearch).
- UDM_FIELD_MODIFIED - last-modified field in unixtime format.
- UDM_FIELD_ORDER - the number of the current document in set of found documents.
- UDM_FIELD_CRC - document CRC.

udm_get_res_param (PHP 4 >= 4.0.5)

Get mnoGoSearch result parameters

string **udm_get_res_param** (int res, int param) \linebreak

udm_get_res_param() returns result parameter value on success, `FALSE` on error.

res - a link to result identifier, received after call to udm_find().

param - parameter identifier, may have the following values:

- UDM_PARAM_NUM_ROWS - number of received found links on the current page. It is equal to UDM_PARAM_PAGE_SIZE for all search pages, on the last page - the rest of links.
- UDM_PARAM_FOUND - total number of results matching the query.
- UDM_PARAM_WORDINFO - information on the words found. E.g. search for "a good book" will return "a: stopword, good:5637, book: 120"
- UDM_PARAM_SEARCHTIME - search time in seconds.
- UDM_PARAM_FIRST_DOC - the number of the first document displayed on current page.
- UDM_PARAM_LAST_DOC - the number of the last document displayed on current page.

udm_load_ispell_data (PHP 4 >= 4.0.5)

Load ispell data

int **udm_load_ispell_data** (int agent, int var, string val1, string val2, int flag) \linebreak

udm_load_ispell_data() loads ispell data. Returns `TRUE` on success, `FALSE` on error.

`agent` - agent link identifier, received after call to udm_alloc_agent().

`var` - parameter, indicating the source for ispell data. May have the following values:

After using this function to free memory allocated for ispell data, please use udm_free_ispell_data(), even if you use UDM_ISPELL_TYPE_SERVER mode.

The fastest mode is UDM_ISPELL_TYPE_SERVER. UDM_ISPELL_TYPE_TEXT is slower and UDM_ISPELL_TYPE_DB is the slowest. The above pattern is `TRUE` for mnoGoSearch 3.1.10 - 3.1.11. It is planned to speed up DB mode in future versions and it is going to be faster than TEXT mode.

- UDM_ISPELL_TYPE_DB - indicates that ispell data should be loaded from SQL. In this case, parameters *`val1`* and *`val2`* are ignored and should be left blank. *`flag`* should be equal to `1`.

  Nota: *`flag`* indicates that after loading ispell data from defined source it sould be sorted (it is necessary for correct functioning of ispell). In case of loading ispell data from files there may be several calls to **udm_load_ispell_data()**, and there is no sense to sort data after every call, but only after the last one. Since in db mode all the data is loaded by one call, this parameter should have the value `1`. In this mode in case of error, e.g. if ispell tables are absent, the function will return `FALSE` and code and error message will be accessible through udm_error() and udm_errno().

  Example:

```
if (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_DB,"","",1)) {
  printf("Error #%d: '%s'\n", udm_errno($udm), udm_error($udm));
  exit;
}
```

- UDM_ISPELL_TYPE_AFFIX - indicates that ispell data should be loaded from file and initiates loading affixes file. In this case *`val1`* defines double letter language code for which affixes are loaded, and *`val2`* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return `FALSE`, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *`flag`* parameter description in UDM_ISPELL_TYPE_DB.

  Example:

```
if ((! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_AFFIX,'en','/opt/ispell/en.aff',0))
   (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_AFFIX,'ru','/opt/ispell/ru.aff',0)) |
   (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SPELL,'en','/opt/ispell/en.dict',0))
   (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SPELL,'ru','/opt/ispell/ru.dict',1)))
exit;
}
```

Nota: *flag* is equal to 1 only in the last call.

- UDM_ISPELL_TYPE_SPELL - indicates that ispell data should be loaded from file and initiates loading of ispell dictionary file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return FALSE, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

```
if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'en','/opt/ispell/en.aff',0))
   (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'ru','/opt/ispell/ru.aff',0)) |
   (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'en','/opt/ispell/en.dict',0))
   (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'ru','/opt/ispell/ru.dict',1)))
 exit;
 }
```

Nota: *flag* is equal to 1 only in the last call.

- UDM_ISPELL_TYPE_SERVER - enables spell server support. *val1* parameter indicates address of the host running spell server. *val2* ‘ is not used yet, but in future releases it is going to indicate number of port used by spell server. *flag* parameter in this case is not needed since ispell data is stored on spellserver already sorted.

  Spelld server reads spell-data from a separate configuration file (/usr/local/mnogosearch/etc/spelld.conf by default), sorts it and stores in memory. With clients server communicates in two ways: to indexer all the data is transferred (so that indexer starts faster), from search.cgi server receives word to normalize and then passes over to client (search.cgi) list of normalized word forms. This allows fastest, compared to db and text modes processing of search queries (by omitting loading and sorting all the spell data).

udm_load_ispell_data() function in UDM_ISPELL_TYPE_SERVER mode does not actually load ispell data, but only defines server address. In fact, server is automatically used by udm_find() function when performing search. In case of errors, e.g. if spellserver is not running or invalid host indicated, there are no messages returned and ispell conversion does not work.

Nota: This function is available in mnoGoSearch 3.1.12 or later.

Example:

```
if (!udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SERVER,”,”,1)) {
    printf("Error loading ispell data from server<br>\n");
    exit;
}
```

udm_open_stored (PHP 4 >= 4.2.0)

Open connection to stored

int **udm_open_stored** (int agent, string storedaddr) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

udm_set_agent_param (PHP 4 >= 4.0.5)

Set mnoGoSearch agent session parameters

int **udm_set_agent_param** (int agent, int var, string val) \linebreak

udm_set_agent_param() returns `TRUE` on success, `FALSE` on error. Defines mnoGoSearch session parameters.

The following parameters and their values are available:

- UDM_PARAM_PAGE_NUM - used to choose search results page number (results are returned by pages beginning from 0, with UDM_PARAM_PAGE_SIZE results per page).
- UDM_PARAM_PAGE_SIZE - number of search results displayed on one page.
- UDM_PARAM_SEARCH_MODE - search mode. The following values available: UDM_MODE_ALL - search for all words; UDM_MODE_ANY - search for any word; UDM_MODE_PHRASE - phrase search; UDM_MODE_BOOL - boolean search. See udm_find() for details on boolean search.
- UDM_PARAM_CACHE_MODE - turns on or off search result cache mode. When enabled, the search engine will store search results to disk. In case a similar search is performed later, the engine will take results from the cache for faster performance. Available values: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - turns on or off trackquery mode. Since version 3.1.2 mnoGoSearch has a query tracking support. Note that tracking is implemented in SQL version only and not available in built-in database. To use tracking, you have to create tables for tracking support. For MySQL, use create/mysql/track.txt. When doing a search, front-end uses those tables to store query words, a number of found documents and current UNIX timestamp in seconds. Available values: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - defines whether index database using phrases ("phrase" parameter in indexer.conf). Possible values: UDM_PHRASE_ENABLED and UDM_PHRASE_DISABLED. Please note, that if phrase search is enabled (UDM_PHRASE_ENABLED), it is still possible to do search in any mode (ANY, ALL, BOOL or PHRASE). In 3.1.10 version of mnoGoSearch phrase search is supported only in sql and built-in database modes, while beginning with 3.1.11 phrases are supported in cachemode as well.

  Examples of phrase search:

  "Arizona desert" - This query returns all indexed documents that contain "Arizona desert" as a phrase. Notice that you need to put double quotes around the phrase

- UDM_PARAM_CHARSET - defines local charset. Available values: set of charsets supported by mnoGoSearch, e.g. koi8-r, cp1251, ...
- UDM_PARAM_STOPFILE - Defines name and path to stopwords file. (There is a small difference with mnoGoSearch - while in mnoGoSearch if relative path or no path entered, it looks for this file in relation to UDM_CONF_DIR, the module looks for the file in relation to current path, i.e. to the path where the php script is executed.)
- UDM_PARAM_STOPTABLE - Load stop words from the given SQL table. You may use several StopwordTable commands. This command has no effect when compiled without SQL database support.
- UDM_PARAM_WEIGHT_FACTOR - represents weight factors for specific document parts. Currently body, title, keywords, description, url are supported. To activate this feature please use degrees of 2 in *Weight commands of the indexer.conf. Let’s imagine that we have these weights:

  URLWeight 1
  BodyWeight 2
  TitleWeight 4
  KeywordWeight 8

DescWeight 16

As far as indexer uses bit OR operation for word weights when some word presents several time in the same document, it is possible at search time to detect word appearance in different document parts. Word which appears only in the body will have 00000010 argegate weight (in binary notation). Word used in all document parts will have 00011111 aggregate weight.

This parameter's value is a string of hex digits ABCDE. Each digit is a factor for corresponding bit in word weight. For the given above weights configuration:

E is a factor for weight 1 (URL Weight bit)
D is a factor for weight 2 (BodyWeight bit)
C is a factor for weight 4 (TitleWeight bit)
B is a factor for weight 8 (KeywordWeight bit)
A is a factor for weight 16 (DescWeight bit)

Examples:

UDM_PARAM_WEIGHT_FACTOR=00001 will search through URLs only.

UDM_PARAM_WEIGHT_FACTOR=00100 will search through Titles only.

UDM_PARAM_WEIGHT_FACTOR=11100 will search through Title,Keywords,Description but not through URL and Body.

UDM_PARAM_WEIGHT_FACTOR=F9421 will search through:

Description with factor 15 (F hex)
Keywords with factor 9
Title with factor 4
Body with factor 2
URL with factor 1

If UDM_PARAM_WEIGHT_FACTOR variable is ommited, original weight value is taken to sort results. For a given above weight configuration it means that document description has a most big weight 16.

- UDM_PARAM_WORD_MATCH - word match. You may use this parameter to choose word match type. This feature works only in "single" and "multi" modes using SQL based and built-in database. It does not work in cachemode and other modes since they use word CRC and do not support substring search. Available values:

  UDM_MATCH_BEGIN - word beginning match;

  UDM_MATCH_END - word ending match;

  UDM_MATCH_WORD - whole word match;

  UDM_MATCH_SUBSTR - word substring match.

- UDM_PARAM_MIN_WORD_LEN - defines minimal word length. Any word shorter this limit is considered to be a stopword. Please note that this parameter value is inclusive, i.e. if

UDM_PARAM_MIN_WORD_LEN=3, a word 3 characters long will not be considered a stopword, while a word 2 characters long will be. Default value is 1.

- UDM_PARAM_ISPELL_PREFIXES - Possible values: UDM_PREFIXES_ENABLED and UDM_PREFIXES_DISABLED, that respectively enable or disable using prefixes. E.g. if a word "tested" is in search query, also words like "test", "testing", etc. Only suffixes are supported by default. Prefixes usually change word meanings, for example if somebody is searching for the word "tested" one hardly wants "untested" to be found. Prefixes support may also be found useful for site's spelling checking purposes. In order to enable ispell, you have to load ispell data with udm_load_ispell_data().
- UDM_PARAM_CROSS_WORDS - enables or disables crosswords support. Possible values: UDM_CROSS_WORDS_ENABLED and UDM_CROSS_WORDS_DISABLED.

  The corsswords feature allows to assign words between and also to a document this link leads to. It works in SQL database mode and is not supported in built-in database and Cachemode.

  Nota: Crosswords are supported only in mnoGoSearch 3.1.11 or later.

- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default `/var` directory of mnoGoSearch installation is used. Can have only string values. The parameter is available in PHP 4.1.0 or later.
- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default `/var` directory of mnoGoSearch installation is used. Can have only string values. The parameter is available in PHP 4.1.0 or later.

LXII. funciones mSQL

msql_affected_rows (PHP 3>= 3.0.6, PHP 4)

devuelve el número de filas involucradas

int **msql_affected_rows** (int query_identifier) \linebreak

Devuelve el número de filas involucradas ("tocadas") por una consulta específica (i.e. el número de filas devueltas por una SELECT, el número de filas modificadas por una actualización (update), o el número de filas suprimidas por una eliminación (delete)).

Véase también: msql_query()

msql_close (PHP 3, PHP 4)

cierra una conexión mSQL

int **msql_close** (int link_identifier) \linebreak

Devuelve `TRUE` si tiene éxito, `FALSE` en caso de error.

msql_close() cierra la conexión con una base de datos mSQL que está asociada con el identificador de conexión (link identifier) especificado. Si no se especifica el identificador de conexión, se asume la última conexión abierta.

Advierta que ésto no es necesario habitualmente, las conexiones abiertas no-persistentes son cerradas automáticamente a la conclusión de la ejecución del script.

msql_close() no cerrará las conexiones permanentes creadas por msql_pconnect().

Véase también: msql_connect() y msql_pconnect().

msql_connect (PHP 3, PHP 4)

abre una conexión mSQL

int **msql_connect** (string hostname) \linebreak

Devuelve un identificador de conexión mSQL positivo si tiene éxito, o `FALSE` en caso de error.

msql_connect() establece una conexión con un servidor mSQL. El argumento hostname es opcional, y en caso de que falte, se asume localhost.

En caso de que se haga una segunda llamada a msql_connect() con los mismos argumentos, no se establece una nueva conexión, en lugar de eso, se devuelve el identificador de conexión ya abierto.

La conexión con el servidor se cerrará tan pronto como la ejecución del script finalice, a menos que sea cerrada antes explícitamente por una llamada a msql_close().

Véase también: msql_pconnect(), msql_close().

msql_create_db (PHP 3, PHP 4)

crea una base de datos mSQL

int **msql_create_db** (string database name [, int link_identifier]) \linebreak

msql_create_db() intenta crear una base de datos nueva en el servidor asociado con el identificador de conexión (link identifier) especificado.

Véase también: msql_drop_db().

msql_createdb (PHP 3, PHP 4)

crea una base de datos mSQL

int **msql_createdb** (string database name [, int link_identifier]) \linebreak

Idéntica a msql_create_db().

msql_data_seek (PHP 3, PHP 4)

desplaza el puntero interno de fila

int **msql_data_seek** (int query_identifier, int row_number) \linebreak

Devuelve `TRUE` si tiene éxito, `FALSE` en caso de fallo.

msql_data_seek() desplaza el puntero interno de fila del resultado mSQL asociado con el identificador de consulta (query identifier) especificado para que apunte al número de fila (row number) proporcionado. La llamada posterior a msql_fetch_row() devolverá esa fila.

Véase también: msql_fetch_row().

msql_dbname (PHP 3, PHP 4)

obtiene el nombre de la base de datos mSQL actual

string **msql_dbname** (int query_identifier, int i) \linebreak

msql_dbname() devuelve el nombre de base de datos almacenado en la posición `i` del puntero devuelto desde la función msql_listdbs(). La función msql_numrows() puede utilizarse para determinar cuantos nombres de base de datos hay disponibles.

msql_drop_db (PHP 3, PHP 4)

elimina (suprime) una base de datos mSQL

int **msql_drop_db** (string database_name, int link_identifier) \linebreak

Devuelve `TRUE` si tiene éxito, `FALSE` en caso de fallo.

msql_drop_db() intenta eliminar (suprimir) una base de datos completa del servidor asociado con el identificador de conexión (link identifier) especificado.

Véase también: msql_create_db().

msql_dropdb (PHP 3, PHP 4)

elimina (suprime) una base de datos mSQL

Véase msql_drop_db().

msql_error (PHP 3, PHP 4)

devuelve el mensaje de error de la última llamada msql

string **msql_error** ( ) \linebreak

Los errores que devuelve el servidor de base de datos mSQL no dan mucha información sobre el problema. Por este motivo, utilice estas funciones para recuperar la cadena de carácteres del error.

msql_fetch_array (PHP 3, PHP 4)

recupera una fila como un array

int **msql_fetch_array** (int query_identifier [, int result_type]) \linebreak

Devuelve un array que se corresponde con la fila recuperada, o `FALSE` si no hay más filas.

msql_fetch_array() es una versión ampliada de msql_fetch_row(). Además de almacenar los datos en los índices numéricos del array resultado, también almacena los datos en índices asociativos, utilizando los nombres de los campos como claves.

El segundo argumento opcional *`result_type`* en **msql_fetch_array()** es una constante y puede tomar los valores siguientes: MSQL_ASSOC, MSQL_NUM, y MYSQL_BOTH.

Sea precavido si está recuperando resultados de una consulta que puede devolver un registro que contiene un único campo que tiene un valor de 0 (o una cadena de carácteres vacía, o `NULL`).

Un aspecto importante a tener en cuenta es que el uso de **msql_fetch_array()** NO es significativamente más lento que el uso de msql_fetch_row(), mientras que proporciona un valor añadido significativo.

Para detalles adicionales, véase también msql_fetch_row()

msql_fetch_field (PHP 3, PHP 4)

obtiene información de campo

object **msql_fetch_field** (int query_identifier, int field_offset) \linebreak

Devuelve un objeto que contiene la información del campo

msql_fetch_field() puede utilizarse para obtener información sobre campos del resultado de una consulta. Si no se especifica el desplazamiento del campo, se recupera el campo siguiente que no haya sido aún recuperado por msql_fetch_field().

Las propiedades del objeto son:

- name - nombre de la columna
- table - nombre de la tabla a la que pertenece la columna
- not_null - 1 si la columna no puede ser nula
- primary_key - 1 si la columna es una clave primaria
- unique - 1 si la columna es una clave única
- type - tipo de la columna

Véase también msql_field_seek().

msql_fetch_object (PHP 3, PHP 4)

recupera una fila como un objeto

int **msql_fetch_object** (int query_identifier [, int result_type]) \linebreak

Devuelve un objeto con las propiedades que corresponden a la fila recuperada, o `FALSE` si no hay más filas.

msql_fetch_object() es similar a msql_fetch_array(), con una diferencia - se devuelve un objeto en vez de un array. Indirectamente esto significa que sólo tiene acceso a los datos por los nombres de los campos, y no por sus desplazamientos (los números son nombres de propiedad no válidos).

El segundo parámetro opcional *`result_type`* en msql_fetch_array() es una constante y puede tomar los valores siguientes: MSQL_ASSOC, MSQL_NUM, y MSQL_BOTH.

Resumiendo, la función es indéntica a msql_fetch_array(), y casi tan rápida como msql_fetch_row() (la diferencia es insignificante).

Véase también: msql_fetch_array() y msql_fetch_row().

msql_fetch_row (PHP 3, PHP 4)

obtiene una fila como un array enumerado

array **msql_fetch_row** (int query_identifier) \linebreak

Devuelve un array que se corresponde con la fila recuperada, o `FALSE` si no hay más filas.

msql_fetch_row() recupera una fila de datos del resultado asociado con el identificador de consulta (query identifier) especificado. La fila se devuelve en un array. Cada columna devuelta se almacena en un desplazamiento del array, comenzando en el desplazamiento 0.

Una llamada posterior a msql_fetch_row() devería devolver la fila siguiente del conjunto resultado, o `FALSE` si no hay más filas.

Véase también: msql_fetch_array(), msql_fetch_object(), msql_data_seek(), y msql_result().

msql_field_seek (PHP 3, PHP 4)

establece el desplazamiento del campo

int **msql_field_seek** (int query_identifier, int field_offset) \linebreak

Se posiciona en el desplazamiento de campo (field offset) especificado. Si la siguiente llamada a msql_fetch_field() no incluye un desplazamiento de campo, este campo será el que se devuelva.

Véase también: msql_fetch_field().

msql_fieldflags (PHP 3, PHP 4)

obtiene los flags del campo

string **msql_fieldflags** (int query_identifier, int i) \linebreak

msql_fieldflags() obtiene los flags del campo (field) especificado. Actualmente pueden ser, "not `NULL`", "primary key", una combinación de ambos, o "" (cadena vacía).

msql_fieldlen (PHP 3, PHP 4)

obtiene la longitud del campo

int **msql_fieldlen** (int query_identifier, int i) \linebreak

msql_fieldlen() devuelve la longitud del campo especificado.

msql_fieldname (PHP 3, PHP 4)

obtiene el nombre del campo

string **msql_fieldname** (int query_identifier, int field) \linebreak

msql_fieldname() devuelve el nombre del campo especificado. *`query_identifier`* es el identificador de consulta, y *`field`* es el índice del campo. `msql_fieldname($result, 2);` devolverá el nombre del segundo campo del resultado asociado con el identificador result.

msql_fieldtable (PHP 3, PHP 4)

obtiene el nombre de la tabla de un campo

int **msql_fieldtable** (int query_identifier, int field) \linebreak

Devuelve el nombre de la tabla desde la que se ha recuperado el campo (*`field`*)

msql_fieldtype (PHP 3, PHP 4)

obtiene el tipo del campo

string **msql_fieldtype** (int query_identifier, int i) \linebreak

msql_fieldtype() es similar a la función msql_fieldname(). Los argumentos son idénticos, pero se devuelve el tipo del campo. Este será "int", "char" o "real".

msql_free_result (PHP 3, PHP 4)

libera la memoria del resultado

int **msql_free_result** (int query_identifier) \linebreak

msql_free_result() libera la memoria asociada con *`query_identifier`*. Cuando PHP completa una petición, esta memoria es liberada automáticamente, por este motivo solo es necesario llamar a esta función cuando se desea estar seguro de que no se utiliza demasiada memoria mientras se está ejecutando el script.

msql_freeresult (PHP 3, PHP 4)

libera la memoria del resultado

Véase msql_free_result()

msql_list_dbs (PHP 3, PHP 4)

lista las bases de datos mSQL en el servidor

int **msql_list_dbs** (void) \linebreak

msql_list_dbs() devolverá un puntero al resultado que contiene las bases de datos disponibles desde el daemon msql en uso. Utilice la función msql_dbname() para recorrer este puntero.

msql_list_fields (PHP 3, PHP 4)

lista los campos del resultado

int **msql_list_fields** (string database, string tablename) \linebreak

msql_list_fields() recupera información sobre el nombre de tabla (tablename) dado. Los argumentos son el nombre de la base de datos (database name) y el nombre de la tabla (table name). Se devuelve un puntero al resultado que puede utilizarse con msql_fieldflags(), msql_fieldlen(), msql_fieldname(), y msql_fieldtype(). Un indetificador de consulta (query identifier) es un entero positivo. La función devuelve `-1` si ocurre un error. En `$phperrmsg` se almacena una cadena de carácteres describiendo el error, y a menos que la función sea llamada como `@msql_list_fields()` esta cadena de error puede ser impresa.

Véase también msql_error().

msql_list_tables (PHP 3, PHP 4)

lista las tablas de una base de datos mSQL

int **msql_list_tables** (string database) \linebreak

msql_list_tables() toma un nombre de base de datos y devuelve un puntero similar al de la función msql() La función msql_tablename() debería utilizarse para obtener los nombres reales de las tablas del puntero devuelto.

msql_listdbs (PHP 3, PHP 4)

lista las bases de datos mSQL en el servidor

Véase msql_list_dbs().

msql_listfields (PHP 3, PHP 4)

lista los campos del resultado

Véase msql_list_fields().

msql_listtables (PHP 3, PHP 4)

lista las tablas de una base de datos mSQL

Véase msql_list_tables().

msql_num_fields (PHP 3, PHP 4)

obtiene el número de campos de un resultado

int **msql_num_fields** (int query_identifier) \linebreak

msql_num_fields() devuelve el número de campos de un conjunto resultado.

Véase también: msql(), msql_query(), msql_fetch_field(), y msql_num_rows().

msql_num_rows (PHP 3, PHP 4)

obtiene el número de filas de un resultado

int **msql_num_rows** (int query_identifier) \linebreak

msql_num_rows() devuelve el número de filas de un conjunto resultado.

Véase también: msql(), msql_query(), y msql_fetch_row().

msql_numfields (PHP 3, PHP 4)

obtiene el número de campos de un resultado

int **msql_numfields** (int query_identifier) \linebreak

Idéntica a msql_num_fields().

msql_numrows (PHP 3, PHP 4)

obtiene el número de filas en el resultado

int **msql_numrows** (void) \linebreak

Idéntica a msql_num_rows().

msql_pconnect (PHP 3, PHP 4)

abre una conexión mSQL persistente

int **msql_pconnect** (string hostname) \linebreak

En caso de éxito devuelve un identificador de conexión mSQL persistente positivo, o `FALSE` en caso de error.

msql_pconnect() se comporta de forma similar a msql_connect() con dos diferencias importantes.

Primero, cuando se conecta, la función debe intentar primero localizar una conexión (persistente) que ya esté abierta en el mismo host. Si se encuentra uno, se devuelve un identificador para el mismo en vez de abrir una conexión nueva.

Segundo, la conexión con el servidor SQL no se cerrará cuando la ejecución del script finalice. Al contrario, la conexión permanecerá abierta para un uso futuro (msql_close() no cerrará las conexiones abiertas por msql_pconnect()).

Este tipo de conexiones son por ello denominadas ’persistentes’.

msql_query (PHP 3, PHP 4)

envía una consulta mSQL

int **msql_query** (string query, int link_identifier) \linebreak

msql_query() envía una consulta a la base de datos activa actual en el servidor que está asociada con el identificador de conexión (link identifier) especificado. Si no se especifica el identificador de conexión,

se asume la última conexión abierta. Si no hay ninguna conexión abierta, la función intenta establecer una conexión como si se hubiera llamado a msql_connect(), y la utiliza.

En caso de éxito devuelve un identificador de consulta mSQL positivo, o `FALSE` en caso de error.

Véase también: msql(), msql_select_db(), y msql_connect().

msql_regcase (PHP 3, PHP 4)

construye una expresión regular para una búsqueda que no distinga mayúsculas/minúsculas

Véase sql_regcase().

msql_result (PHP 3, PHP 4)

obtiene datos resultado

int **msql_result** (int query_identifier, int i, mixed field) \linebreak

Devuelve el contenido de la celda en la fila y desplazamiento del conjunto resultado mSQL especificado.

msql_result() devuelve el contenido de una celda de un conjunto resultado mSQL. El argumento campo (field) puede ser el desplazamiento del campo, el nombre del campo, o el nombre de la tabla punto nombre del campo (nombretabla.nombrecampo). Si el nombre de la columna tiene un alias (’select foo as bar from...’), utilice el alias en vez del nombre de la columna.

Cuando se trabaja con conjuntos de resultados grandes, debería considerar el uso de las funciones que recuperen filas completas (especificadas más abajo). Como estas funciones recuperan el contenido de varias celdas en una única llamada de función, son MUCHO más rápidas que msql_result(). Advierta también que especificar un desplazamiento numérico para el argumento campo (field) es mucho más rápido que especificar un argumento nombrecampo o nombretabla.nombrecampo.

Alternativas de alto-rendimiento recomendadas: msql_fetch_row(), msql_fetch_array(), y msql_fetch_object().

msql_select_db (PHP 3, PHP 4)

selecciona una base de datos mSQL

int **msql_select_db** (string database_name, int link_identifier) \linebreak

Devuelve `TRUE` si tiene éxito, `FALSE` en caso contrario.

msql_select_db() establece la base de datos activa actual en el servidor que está asociada con el identificador de conexión (link identifier) suministrado. Si no se especifica el identificador de conexión,

se asumne la última conexión abierta. Si no hay ninguna conexión abierta la función intentará establecer una conexión como si se hubiera llamado a sql_connect(), y la utiliza.

Cada llamada posterior a msql_query() se hará en la base de datos activa.

Véase también: msql_connect(), msql_pconnect(), y msql_query().

msql_selectdb (PHP 3, PHP 4)

selecciona una base de datos mSQL

Véase msql_select_db().

msql_tablename (PHP 3, PHP 4)

obtiene el nombre de la tabla de un campo

string **msql_tablename** (int query_identifier, int field) \linebreak

msql_tablename() toma un puntero resultado devuelto por la función msql_list_tables() como un índice entero y devuelve el nombre de una tabla. La función msql_numrows() puede utilizarse para determinar el número de tablas del puntero resultado.

Ejemplo 1. msql_tablename() example

```
<?php
msql_connect ("localhost");
$result = msql_list_tables("wisconsin");
$i = 0;
while ($i < msql_numrows($result)) {
    $tb_names[$i] = msql_tablename($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

msql (PHP 3, PHP 4)

ejecuta una consulta mSQL

int **msql** (string database, string query, int link_identifier) \linebreak

Devuelve un identificador de consulta mSQL positivo en el resultado de la consulta, o `FALSE` en caso de error.

msql() selecciona una base de datos y ejecuta una consulta en ella. Si no se especifica el identificador de conexión (link identifier), la función intentará encontrar una conexión abierta en el servidor mSQL y en el caso de que no se encontrase intentará crear uno como si se llamase a msql_connect() sin parámetros (véase msql_connect()).

LXIII. Funciones MySQL

Estas funciones le permiten acceder a servidores de bases de datos MySQL.

Puede encontrar más información sobre MySQL en http://www.mysql.com/.

mysql_affected_rows (PHP 3, PHP 4)

Devuelve el número de filas afectadas de la última operación MySQL

int **mysql_affected_rows** ([int identificador_de_enlace]) \linebreak

mysql_affected_rows() devuelve el número de filas afectadas en la ultima sentencia INSERT, UPDATE o DELETE sobre el servidor asociado con el identificador de enlace especificado. Si el identificador de enlace no ha sido especificado, se asume por defecto el último enlace.

Si la última sentencia fue un DELETE sin clausula WHERE, todos los registros han sido borrados de la tabla pero esta función devolvera cero.

Este comando no es efectivo para las sentencias SELECT, sino sólo para las sentencias que modifican registros. Para conseguir el número de lineas devueltos por un SELECT, usar mysql_num_rows().

mysql_change_user (PHP 3>= 3.0.13)

Cambia el usuario conectado en la conexión activa

int **mysql_change_user** (string usuario, string password [, string base_de_datos [, int identificador_de_enlace]]) \linebreak

mysql_change_user() cambia el usuario conectado en la actual conexión activa, o si se especifica, en la conexión determinada por el identificador de enlace. Si se especifica la base de datos, esta será la base por defecto después del cambio de usuario. Si la nueva combinación de usuario/ password no esta autorizada, el usuario actualmente conectado permanece activo.

Nota: Esta función fue introducida en PHP 3.0.13 y requiere MySQL 3.23.3 o superior.

mysql_character_set_name (PHP 4 CVS only)

Returns the name of the character set

int **mysql_character_set_name** ([resource link_identifier]) \linebreak

mysql_character_set_name() returns the default character set name for the current connection.

Ejemplo 1. mysql_character_set_name() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$charset = mysql_character_set_name($link);
printf ("current character set is %s\n", $charset);
```

```
?>
```

The above example would produce the following output:

```
latin1
```

See also: mysql_real_escape_string()

mysql_close (PHP 3, PHP 4)

cierra el enlace con MySQL

int **mysql_close** ([int identificador_de_enlace]) \linebreak

Devuelve: verdadero si exito, falso si error.

mysql_close() cierra el enlace con la base MySQL que esta asociada con el identificador de enlace especificado. Si no se especifica el identificador de enlace, se asume por defecto el último enlace.

> **Nota:** Normalmente no es necesario ya que la aperturas no-persistentes son cerradas automaticamente al final de la ejecución del script.

mysql_close() no cerrará los enlaces persistentes generados con mysql_pconnect().

Ejemplo 1. Ejemplo de MySQL close

```
<?php
    $link = mysql_connect ("kraemer", "marliesle", "secret") {
        or die ("Could not connect");
    }
    print ("Connected successfully");
    mysql_close ($link);
?>
```

Ver también: mysql_connect(), y mysql_pconnect().

mysql_connect (PHP 3, PHP 4)

Abre una conexión a un servidor MySQL

int **mysql_connect** ([string server [, string usuario [, string password]]]) \linebreak

Devuelve: Un identificador de enlace positivo si tiene exito, o falso si error.

mysql_connect()establece una conexión a un servidor MySQL. Todos los argumentos son opcionales, y si no hay , se asumen los valores por defecto ('localhost', usuario propietario del proceso del servidor, password vacia).

El hostname puede incluir tambien un número de puerto . ej. "hostname:puerto" o un camino al socket ej. ":/camino/al/socket" para localhost.

Nota: Soporte para ":puerto" fue añadido en PHP 3.0B4.

Soporte para ":/camino/al/socket" fue añadido en PHP 3.0.10.

En el caso de que se haga una llamada a **mysql_connect()** con los mismos argumentos, no se establecerá un nuevo enlace, sino que se devolverá el enlace ya abierto.

El enlace al servidor sera cerrado tan pronto como la ejecución del script finalice, a menos que se cierre antes explicitamente llamando a mysql_close().

Ejemplo 1. Ejemplo de MySQL connect

```
<?php
    $link = mysql_connect ("kraemer", "marliesle", "secret") {
        or die ("Could not connect");
    }
    print ("Connected successfully");
    mysql_close ($link);
?>
```

Ver también : mysql_pconnect(), y mysql_close().

mysql_create_db (PHP 3, PHP 4)

Crea una base MySQL

int **mysql_create_db** (string base_de_datos [, int identificador_de_enlace]) \linebreak

mysql_create_db() intenta crear una base nueva en el servidor asociado al identificador de enlace.

Ejemplo 1. Ejemplo de MySQL create

```
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim") {
        or die ("Could not connect");
    }
    if (mysql_create_db ("my_db")) {
```

```
        print ("Database created successfully\n");
    } else {
        printf ("Error creating database: %s\n", mysql_error ());
    }
?>
```

Por razones de compatibilidad puede usarse **mysql_createdb()** igualmente.

Ver también: mysql_drop_db().

mysql_data_seek (PHP 3, PHP 4)

Mueve el puntero interno

int **mysql_data_seek** (int id_resultado, int numero_de_fila) \linebreak

Devuelve: verdadero si exito, falso si error.

mysql_data_seek() mueve el puntero de fila interno a la fila especificada para el identificador de resultado. La próxima llamada a mysql_fetch_row() devolverá esa fila.

numero_de_fila empieza en 0.

Ejemplo 1. Ejemplo de MySQL data seek

```
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim") {
        or die ("Could not connect");
    }

    mysql_select_db ("samp_db") {
        or die ("Could not select database");
    }

    $query = "SELECT last_name, first_name FROM friends";
    $result = mysql_query ($query) {
        or die ("Query failed");
    }

    # fetch rows in reverse order

    for ($i = mysql_num_rows ($result) - 1; $i >=0; $i--) {
        if (!mysql_data_seek ($result, $i)) {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }

        if(!($row = mysql_fetch_object ($result)))
            continue;

        printf ("%s %s<BR>\n", $row->last_name, $row->first_name);
```

```
    }

    mysql_free_result ($result);
?>
```

mysql_db_name (PHP 3>= 3.0.6, PHP 4)

Get result data

string **mysql_db_name** (resource result, int row [, mixed field]) \linebreak

mysql_db_name() takes as its first parameter the result pointer from a call to mysql_list_dbs(). The *row* parameter is an index into the result set.

If an error occurs, FALSE is returned. Use mysql_errno() and mysql_error() to determine the nature of the error.

Ejemplo 1. mysql_db_name() example

```
<?php
    error_reporting(E_ALL);

    mysql_connect('dbhost', 'username', 'password');
    $db_list = mysql_list_dbs();

    $i = 0;
    $cnt = mysql_num_rows($db_list);
    while ($i < $cnt) {
        echo mysql_db_name($db_list, $i) . "\n";
        $i++;
    }
?>
```

For backward compatibility, **mysql_dbname()** is also accepted. This is deprecated, however.

mysql_db_query (PHP 3, PHP 4)

Envia una sentencia MySQL al servidor

int **mysql_db_query** (string base_de_datos, string sentencia [, int identificador_de_enlace]) \linebreak

Devuelve: Un identificador de resultado positivo o falso si error.

mysql_db_query() selecciona una base y ejecuta una sentencia en ella. Si el identificador de enlace no ha sido especificado, la función intenta encontrar un enlace abierto al servidor MySQL y si no lo encuentra, intetará crear uno como si fuera llamado mysql_connect() sin argumentos

Ver tambiénmysql_connect().

Por razones de compatibilidad puede usarse **mysql()** igualmente.

mysql_drop_db (PHP 3, PHP 4)

Borra una base de datos MySQL

int **mysql_drop_db** (string base_de_datos [, int identificador_de_enlace]) \linebreak

Devuelve: verdadero si éxito, falso si error.

mysql_drop_db() intenta suprimir una base de datos completa del servidor asociado al identificador de enlace.

Ver también: mysql_create_db(). Por razones de compatibilidad puede usarse **mysql_dropdb()** igualmente.

mysql_errno (PHP 3, PHP 4)

Deuelve el número del mensaje de error de la última operación MySQL

int **mysql_errno** ([int identificador_de_enlace]) \linebreak

Los errores devueltos por mySQL no indican los warnings. Usar estas funciones para encontrar el número de error.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Ver también: mysql_error()

mysql_error (PHP 3, PHP 4)

Devuelve el texto del mensaje de error de la última operación MySQL

string **mysql_error** ([int identificador_de_enlace]) \linebreak

Los errores devueltos por mySQL no indican los warnings. Usar estas funciones para encontrar el número de error.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Ver también: mysql_errno()

mysql_escape_string (PHP 4 >= 4.0.3)

Escapes a string for use in a mysql_query.

string **mysql_escape_string** (string unescaped_string) \linebreak

This function will escape the *`unescaped_string`*, so that it is safe to place it in a mysql_query().

> **Nota: mysql_escape_string()** does not escape `%` and _.
>
> This function is identical to mysql_real_escape_string() except that mysql_real_escape_string() takes a connection handler and escapes the string according to the current character set. **mysql_escape_string()** does not take a connection argument and does not respect the current charset setting.

Ejemplo 1. mysql_escape_string() example

```
<?php
    $item = "Zak's Laptop";
    $escaped_item = mysql_escape_string($item);
    printf ("Escaped string: %s\n", $escaped_item);
?>
```

The above example would produce the following output:

```
Escaped string: Zak\'s Laptop
```

See also: mysql_real_escape_string(), addslashes(), and the magic_quotes_gpc directive.

mysql_fetch_array (PHP 3, PHP 4)

Extrae la fila de resultado como una matriz asociativa

array **mysql_fetch_array** (int id_resultado [, int tipo_de_resultado]) \linebreak

Devuelve una matriz que corresponde a la sentencia extraida, o falso si no quedan más filas.

mysql_fetch_array() es una versión extendida de mysql_fetch_row(). Además de guardar los datos en el indice numerico de la matriz, guarda también los datos en los indices asociativos, usando el nombre de campo como clave.

Si dos o más columnas del resultado tienen el mismo nombre de campo, la última columna toma la prioridad. Para acceder a la(s) otra(s) columna(s) con el mismo nombre, se debe escificar el indice numerico o definir un alias para la columna.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

La función **mysql_fetch_array()** no es significativemente mas lenta que mysql_fetch_row(), sin embargo tiene un valor añadido importante.

El segundo argumento opcional *`tipo_de_resultado`* en **mysql_fetch_array()** es una constante y puede tomar los valores siguientes: MYSQL_ASSOC, MYSQL_NUM, y MYSQL_BOTH. (Esta funcionalidad fue añadida en PHP 3.0.7)

Para más detalles, ver también mysql_fetch_row().

Ejemplo 1. mysql fetch array

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("database","select * from table");
while($row = mysql_fetch_array($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
mysql_free_result($result);
?>
```

mysql_fetch_assoc (PHP 4 >= 4.0.3)

Fetch a result row as an associative array

array **mysql_fetch_assoc** (resource result) \linebreak

Returns an associative array that corresponds to the fetched row, or `FALSE` if there are no more rows.

mysql_fetch_assoc() is equivalent to calling mysql_fetch_array() with MYSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way mysql_fetch_array() originally worked. If you need the numeric indices as well as the associative, use mysql_fetch_array().

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you either need to access the result with numeric indices by using mysql_fetch_row() or add alias names. See the example at the mysql_fetch_array() description about aliases.

An important thing to note is that using **mysql_fetch_assoc()** is *not significantly* slower than using mysql_fetch_row(), while it provides a significant added value.

Ejemplo 1. mysql_fetch_assoc()

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password");
    mysql_select_db("mydb");
    $query = "select * from table";
    $result = mysql_query($query);
    while ($row = mysql_fetch_assoc($result)) {
        echo $row["user_id"];
        echo $row["fullname"];
    }
    mysql_free_result($result);
?>
```

For further details, see also mysql_fetch_row(), mysql_fetch_array() and mysql_query().

mysql_fetch_field (PHP 3, PHP 4)

Extrae la información de una columna y la devuelve como un objeto.

object **mysql_fetch_field** (int id_resultado [, int salto]) \linebreak

Devuelve un objeto que contiene la información del campo.

Puede usarse **mysql_fetch_field()** para obtener información sobre campos en un resultado. Si no se especifica el salto, se extrae el siguiente campo que todavia no ha sido extraido. con **mysql_fetch_field()**.

Las propiedades del objeto son:

- name - nombre de la columna
- table - name de la tabla a la que pertenece la columna
- max_length - longitud máxima de la columna
- not_null - 1 si la columna no puede contener un valor nulo
- primary_key - 1 si la columna es clave primaria
- unique_key - 1 si la columna es clave unica
- multiple_key - 1 si la columna es clave no unica
- numeric - 1 si la columna es numerica
- blob - 1 si la columna es un BLOB
- type - el tipo de la columna
- unsigned - 1 si la columna es unsigned
- zerofill - 1 si la columna es zero-filled

Ver también mysql_field_seek()

mysql_fetch_lengths (PHP 3, PHP 4)

Devuelve la longitud de cada salida en un resultado

array **mysql_fetch_lengths** (int id_resultado) \linebreak

Devuelve: Una matriz que contiene las longitudes de cada campo de la última fila extraida por mysql_fetch_row(), o falso si error.

mysql_fetch_lengths() almacena las longitudes de cada columna en la última fila devuelta por mysql_fetch_row(), mysql_fetch_array(), y mysql_fetch_object() en una matriz, empezando por 0.

Ver también: mysql_fetch_row().

mysql_fetch_object (PHP 3, PHP 4)

Extrae una fila de resultado como un objeto

object **mysql_fetch_object** (int id_resultado [, int tipo_de_resultado]) \linebreak

Devuelve un objeto con las propiedades aur corresponden a la última fila extraida, o falso si no quedan más filas.

mysql_fetch_object() es similar a mysql_fetch_array(), con la diferencia que un objeto es devuelto en lugar de una matriz. Indirectamente, quiere decir que solo se puede aceder a los datos por el nombre del campo, y no por su posición.

El argumento opcional *tipo_de_resultado* es una constante y puede tomar los valores siguientes: MYSQL_ASSOC, MYSQL_NUM, y MYSQL_BOTH.

La función es identica a mysql_fetch_array(), y casi tan rápida como mysql_fetch_row() (la diferencia es insignificante).

Ejemplo 1. mysql fetch object

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("database","select * from table");
while($row = mysql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result($result);
?>
```

Ver también: mysql_fetch_array() y mysql_fetch_row().

mysql_fetch_row (PHP 3, PHP 4)

Devuelve una fila de resultado como matriz

array **mysql_fetch_row** (int id_resultado) \linebreak

Devuelve: Una matriz que corresponde a la fila selecionada, o falso si no quedan más lineas.

mysql_fetch_row() seleciona una fila de datos del resultado asociado al identificador de resultado especificado. La fila es devuelta como una matriz. Cada columna del resultdo es guardada en un offset de la matriz, empezando por el offset 0.

La llamada a **mysql_fetch_row()** debería devolver la próxima fila del resultado, o falso si no quedan más filas.

Ver también: mysql_fetch_array(), mysql_fetch_object(), mysql_data_seek(), mysql_fetch_lengths(), and mysql_result().

mysql_field_flags (PHP 3, PHP 4)

Devuelve los flags asociados con el campo especificado en un resultado

string **mysql_field_flags** (int id_resultado, int offset_del_campo) \linebreak

mysql_field_flags() devuelve los flags del campo especificado. Cada flag es devuelto como una palabra y estan separados un unico espacio, se puede dividir el resultado devuelto utilizando explode().

Los siguientes flags pueden ser devueltos si tu versión de MySQL los soporta: "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

Por razones de compatibilidad puede usarse tambien **mysql_fieldflags()**.

mysql_field_len (PHP 3, PHP 4)

Devuelve la longitud del campo especificado

int **mysql_field_len** (int id_resultado, int offset_del_campo) \linebreak

mysql_field_len() devuelve la longitud del campo especificado. Por razones de compatibilidad puede usarse tambien **mysql_fieldlen()**.

mysql_field_name (PHP 3, PHP 4)

Devuelve el nombre del campo especificado en un resultado

string **mysql_field_name** (int id_resultado, int indice_del_campo) \linebreak

mysql_field_name() devuelve el nombre del campo especificado. Los argumentos de la función son el identificador de resultado y el indice del campo. Por ejemplo: `mysql_field_name($result,2);`

Devolverá el nombre del segundo campo asociado al identificador de resultado.

Por razones de compatibilidad puede usarse tambien **mysql_fieldname()**.

mysql_field_seek (PHP 3, PHP 4)

Asigna el puntero del resultado al offset del campo especificado

int **mysql_field_seek** (int id_resultado, int offset_del_campo) \linebreak

Busca el offset del campo especificado. Si la próxima llamada a mysql_fetch_field() no incluye un offset de campo, se devolverá ese campo.

Ver también: mysql_fetch_field().

mysql_field_table (PHP 3, PHP 4)

Devuelve el nombre de la tabla donde esta el campo especificado

string **mysql_field_table** (int id_resultado, int offset_del_campo) \linebreak

Devuelve el nombre de la tabla del campo. Por razones de compatibilidad puede usarse tambien **mysql_fieldtable()**.

mysql_field_type (PHP 3, PHP 4)

Devuelve el tipo del campo especificado en un resultado

string **mysql_field_type** (int id_resultado, int offset_del_campo) \linebreak

mysql_field_type() es similar a la función mysql_field_name(). Los argumentos son identicos, pero se devuelve el tipo de campo. El tipo sera "int", "real", "string", "blob", o otros detallados en la documentación de MySQL.

Ejemplo 1. mysql field types

```
<?php
mysql_connect("localhost:3306");
mysql_select_db("wisconsin");
$result = mysql_query("SELECT * FROM onek");
$fields = mysql_num_fields($result);
$rows   = mysql_num_rows($result);
$i = 0;
$table = mysql_field_table($result, $i);
echo "Your '".$table."' table has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = mysql_field_type  ($result, $i);
    $name  = mysql_field_name  ($result, $i);
    $len   = mysql_field_len   ($result, $i);
    $flags = mysql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
mysql_close();
?>
```

Por razones de compatibilidad puede usarse tambien **mysql_fieldtype()**.

mysql_free_result (PHP 3, PHP 4)

Libera la memoria del resultado

int **mysql_free_result** (int id_resultado) \linebreak

mysql_free_result() solo necesita ser llamada si te preocupa usar demasiado memoria durante la ejecución de tu script. Toda la memoria del resultado especificado en el parametro del identificador de resultado sera automaticamente liberada.

Por razones de compatibilidad puede usarse tambien **mysql_freeresult()**.

mysql_get_client_info (PHP 4 >= 4.0.5)

Get MySQL client info

string **mysql_get_client_info** (void) \linebreak

mysql_get_client_info() returns a string that represents the client library version.

Ejemplo 1. mysql_get_client_info Example

```
<?php
    printf ("MySQL client info: %s\n", mysql_get_client_info());
?>
```

The above example would produce the following output:

```
MySQL client info: 3.23.39
```

See also: mysql_get_host_info(), mysql_get_proto_info() and mysql_get_server_info().

mysql_get_host_info (PHP 4 >= 4.0.5)

Get MySQL host info

string **mysql_get_host_info** ([resource link_identifier]) \linebreak

mysql_get_host_info() returns a string describing the type of connection in use for the connection *link_identifier*, including the server host name. If *link_identifier* is omitted, the last opened connection will be used.

Ejemplo 1. mysql_get_host_info Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    printf ("MySQL host info: %s\n", mysql_get_host_info());
```

```
?>
```

The above example would produce the following output:

```
MySQL host info: Localhost via UNIX socket
```

See also: mysql_get_client_info(), mysql_get_proto_info() and mysql_get_server_info().

mysql_get_proto_info (PHP 4 >= 4.0.5)

Get MySQL protocol info

int **mysql_get_proto_info** ([resource link_identifier]) \linebreak

mysql_get_proto_info() returns the protocol version used by connection *`link_identifier`*. If *`link_identifier`* is omitted, the last opened connection will be used.

Ejemplo 1. mysql_get_proto_info Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    printf ("MySQL protocol version: %s\n", mysql_get_proto_info());
?>
```

The above example would produce the following output:

```
MySQL protocol version : 10
```

See also: mysql_get_client_info(), mysql_get_host_info() and mysql_get_server_info().

mysql_get_server_info (PHP 4 >= 4.0.5)

Get MySQL server info

string **mysql_get_server_info** ([resource link_identifier]) \linebreak

mysql_get_server_info() returns the server version used by connection *`link_identifier`*. If *`link_identifier`* is omitted, the last opened connection will be used.

Ejemplo 1. mysql_get_server_info Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    printf ("MySQL server version: %s\n", mysql_get_server_info());
?>
```

The above example would produce the following output:

```
MySQL server version: 4.0.1-alpha
```

See also: mysql_get_client_info(), mysql_get_host_info() and mysql_get_proto_info().

mysql_info (PHP 4 CVS only)

Get information about the most recent query

string **mysql_info** ([resource link_identifier]) \linebreak

mysql_info() returns detailed information about the last query using the given *link_identifier*. If *link_identifier* isn't specified, the last opened link is assumed.

mysql_info() returns a string for all statements listed below. For all other FALSE. The string format depends on the given statement.

Ejemplo 1. Relevant MySQL Statements

```
INSERT INTO ... SELECT ...
String format: Records: 23 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
String format: Records: 37 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
String format: Records: 42 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
String format: Records: 60 Duplicates: 0 Warnings: 0
UPDATE
String format: Rows matched: 65 Changed: 65 Warnings: 0
```

The numbers are only for illustrating purpose; their values will correspond to the query.

Nota: mysql_info() returns a non-`FALSE` value for the INSERT ... VALUES statement only if multiple value lists are specified in the statement.

See also: mysql_affected_rows()

mysql_insert_id (PHP 3, PHP 4)

Devuelve el identificador generado en la última llamada a INSERT

int **mysql_insert_id** ([int identificador_de_enlace]) \linebreak

mysql_insert_id() devuelve el identificador generado para un campo de tipo AUTO_INCREMENTED. Se devolvera el identificador genrado por el último INSERT para el *`identificador_de_enlace`*. Si no se especifica el *`identificador_de_enlace`*, se asume por defecto el último enlace abierto.

mysql_list_dbs (PHP 3, PHP 4)

Lista las bases de datos disponibles en el servidor MySQL

int **mysql_list_dbs** ([int identificador_de_enlace]) \linebreak

mysql_list_dbs() devuelve un puntero de resultado que contiene las bases disponibles en el actual demonio mysql. Utiliza la función mysql_tablename() para explotar el puntero de resultado.

Por razones de compatibilidad puede usarse tambien **mysql_listdbs()**.

mysql_list_fields (PHP 3, PHP 4)

Lista los campos del resultado de MySQL

int **mysql_list_fields** (string base_de_datos, string tabla [, int identificador_de_enlace]) \linebreak

mysql_list_fields() lista información sobre la tabla. Los argumentos son la base de datos y el nombre de la tabla. Se devuelve un puntero que puede ser usado por las funciónes mysql_field_flags(), mysql_field_len(), mysql_field_name(), y mysql_field_type().

Un identificador de resultado es un entero positivo. La función devuelve -1 si se produce un error. Una cadena de caracteres describiendo el error sera introducida en `$phperrmsg`, y a menos que la función sea llamada como `@mysql()` el literal del error tambien sera impreso.

Por razones de compatibilidad puede usarse tambien **mysql_listfields()**.

mysql_list_processes (PHP 4 CVS only)

List MySQL processes

resource **mysql_list_processes** ([resource link_identifier]) \linebreak

mysql_list_processes() returns a result pointer describing the current server threads.

Ejemplo 1. mysql_list_processes() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');

$result = mysql_list_processes($link);
while ($row = mysql_fetch_row($result)){
    printf("%s %s %s %s %s\n", $row["Id"], $row["Host"], $row["db"],
       $row["Command"], $row["Time"]);
}
mysql_free_result ($result);
?>
```

The above example would produce the following output:

```
1 localhost test Processlist 0
4 localhost mysql sleep 5
```

See also: mysql_thread_id()

mysql_list_tables (PHP 3, PHP 4)

Lista las tablas en una base de datos MySQL

int **mysql_list_tables** (string base_de_datos [, int identificador_de_enlace]) \linebreak

mysql_list_tables() toma el nombre de la base y devuelve un puntero de resultado como la función mysql_db_query(). La función mysql_tablename() debe ser usada para extraer los nombres de las tablas del puntero.

Por razones de compatibilidad puede usarse tambien **mysql_listtables()**. can also be used.

mysql_num_fields (PHP 3, PHP 4)

devuelve el numero de campos de un resultado

int **mysql_num_fields** (int id_resultado) \linebreak

mysql_num_fields() devuelve el numero de campos de un identificador de resultado.

Ver también: mysql_db_query(), mysql_query(), mysql_fetch_field(), mysql_num_rows().

Por razones de compatibilidad puede usarse tambien **mysql_numfields()**.

mysql_num_rows (PHP 3, PHP 4)

Devuelve el numero de filas de un resultado

int **mysql_num_rows** (int id_resultado) \linebreak

mysql_num_rows() Devuelve el numero de filas de un identificador de resultado.

Ver también: mysql_db_query(), mysql_query() and, mysql_fetch_row().

Por razones de compatibilidad puede usarse tambien **mysql_numrows()**.

mysql_pconnect (PHP 3, PHP 4)

Abre una conexión persistente al servidor MySQL

int **mysql_pconnect** ([string server [, string usuario [, string password]]]) \linebreak

Devuelve: un identificador de enlace persistente, o falso si se produce un error.

mysql_pconnect() establece una conexión a un servidor MySQL. Todos los argumentos son opcionales, y si no existen, se asumen los valores por defecto ('localhost', nombre del usuario propietario del proceso, password vacia).

El hostname puede incluir un numero de puerto. ej. "hostname:port" o un camino al socket ej. ":/camino/al/socket" para el puerto para el host local.

Nota: Soporte para ":puerto" fue añadido en 3.0B4.

Soporte para ":/camino/al/socket" fue añadido en 3.0.10.

mysql_pconnect() actua como mysql_connect() con dos diferencias fundamentales.

Primero, durante la conexión, la función intenta primero encontrar un enlace persistente abierto con el mismo host, usuario y password. Si lo encuentra, devuelve el identificador de enlace en lugar de abrir otra conexión.

Segundo, la conexión no sera cerrado cuando acabe la ejecución del script. El enlace permanecera abierta para ser usado en el futuro (mysql_close() will not cierra el enlace establecido con **mysql_pconnect()**).

Este tipo de enlaces son llamados 'persistentes'.

mysql_ping (PHP 4 CVS only)

Ping a server connection or reconnect if there is no connection

bool **mysql_ping** ([resource link_identifier]) \linebreak

mysql_ping() checks whether or not the connection to the server is working. If it has gone down, an automatic reconnection is attempted. This function can be used by scripts that remain idle for a long while, to check whether or not the server has closed the connection and reconnect if necessary. **mysql_ping()** returns `TRUE` if the connection to the server is working, otherwise `FALSE`.

See also: mysql_thread_id(), mysql_list_processes().

mysql_query (PHP 3, PHP 4)

Envia una sentencia SQL a MySQL

int **mysql_query** (string sentencia [, int identificador_de_enlace]) \linebreak

mysql_query() envia una sentencia a la base activa en el servidor asociado al identificador de enlace. Si no es especificado un *`identificador_de_enlace`*, se asumira el ultilmo enlace abierto. Si no hay ningun enlace abierto, la función intenta estalecer un enlace como si se llamara función mysql_connect() sin argumentos, y lo utiliza.

La sentencia no puede terminar por punto y coma.

mysql_query() devuelve `TRUE` (no-cero) o `FALSE` para indicar si la sentencia se ha ejecutado correctamente o no. Un valor `TRUE` significa que la sentencia era correcta y pudo ser ejecutada en el servidor. No indica nada sobre el numero de fila devueltas. Es perfectamente posible que la sentencia se ejecute correctamente pero que no devuelve ninguna fila.

La siguiente sentencia es invalida sintacticamente, asi que **mysql_query()** falla y devuelve `FALSE`:

Ejemplo 1. mysql_query()

```
<?php
$result = mysql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

La siguiente sentencia es invalida semanticamente si `my_col` no es una columna de la tabla `my_tbl`, asi que **mysql_query()** falla y devuelve `FALSE`:

Ejemplo 2. mysql_query()

```
<?php
$result = mysql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

mysql_query() fallara tambien y devolvera `FALSE` si no se tiene el permiso de acceso a la tabla especificada en la sentencia.

Asumiendo la sentencia tenga exito, se puede llamar a mysql_affected_rows() para saber cuantas filas fueron afectadas (para DELETE, INSERT, REPLACE, o UPDATE) Para las sentencias SELECT, **mysql_query()** devuelve un nuevo identificador de resultado que se puede pasar a mysql_result(). Cuando se acabe de utilizar el resultado, se pueden liberar los recursos asociados utilizando mysql_free_result().

Ver también: mysql_affected_rows(), mysql_db_query(), mysql_free_result(), mysql_result(), mysql_select_db(), and mysql_connect().

mysql_real_escape_string (PHP 4 CVS only)

Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection.

string **mysql_real_escape_string** (string unescaped_string [, resource link_identifier]) \linebreak

This function will escape special characters in the *`unescaped_string`*, taking into account the current charset of the connection so that it is safe to place it in a mysql_query().

> **Nota: mysql_real_escape_string()** does not escape `%` and _.

Ejemplo 1. mysql_real_escape_string() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$item = "Zak's and Derick's Laptop";
$escaped_item = mysql_real_escape_string($item);
printf ("Escaped string: %s\n", $escaped_item);
?>
```

The above example would produce the following output:

```
Escaped string: Zak\'s and Derick\'s Laptop
```

See also: mysql_escape_string(), mysql_character_set_name().

mysql_result (PHP 3, PHP 4)

Devuelve datos de un resultado

int **mysql_result** (int id_resultado, int numero_de_fila [, mixed campo]) \linebreak

mysql_result() devuelve el contenido de una celda de un resultado MySQL. El campo argumento puede ser el nombre del campo o el offset o tabla.nombre_del_campo. Si el nombre de la columna tiene un alias ('select foo as bar from...'), utilice el alias en lugar del nombre de la columna.

Cuando se trabaja un un gran resultado, debe considerarse la utilizacion de una funcion que devuelva una fila entera ya que estas funciones son MUCHO mas rapidas que **mysql_result()**. Tambien, especificando un offset numerico en lugar del nombre del campo, la ejecucion sera mas rapida.

Las llamadas a **mysql_result()** no deben mezclarse con llamadas a las otras sentencias que trabajan con un identificador de resultado.

Alternativas recomendadas: mysql_fetch_row(), mysql_fetch_array(), y mysql_fetch_object().

mysql_select_db (PHP 3, PHP 4)

Selecciona un base de datos MySQL

int **mysql_select_db** (string base_de_datos [, int identificador_de_enlace]) \linebreak

Devuelve : `TRUE` si exito, `FALSE` si error.

mysql_select_db() establece la base activa que estara asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el ultimo enlace abierto. Si no hay ningun enlace abierto, la función intentara establecer un enlace como si se llamara a mysql_connect().

Toda llamada posterior a mysql_query() utilizara la base activada.

Ver también: mysql_connect(), mysql_pconnect(), and mysql_query().

Por razones de compatibilidad puede usarse tambien **mysql_selectdb()**.

mysql_stat (PHP 4 CVS only)

Get current system status

string **mysql_stat** ([resource link_identifier]) \linebreak

mysql_stat() returns the current server status.

> **Nota: mysql_stat()** currently only returns status for uptime, threads, queries, open tables, flush tables and queries per second. For a complete list of other status variables you have to use the SHOW STATUS SQL command.

Ejemplo 1. mysql_stat() example

```
<?php
$link = mysql_connect('localhost', "mysql_user", "mysql_password");
printf("%s\n", mysql_stat($link));
?>
```

The above example would produce the following output:

```
Uptime: 5380  Threads: 1  Questions: 1321299  Slow queries: 1  Opens: 26 Flush ta-
bles: 1  Open tables: 17  Queries per second avg: 245.595
```

mysql_tablename (PHP 3, PHP 4)

Devuelve el nombre de la tabla de un campo

string **mysql_tablename** (int id_resultado, int i) \linebreak

mysql_tablename() toma un puntero de resultado devuelto por mysql_list_tables() asi como un indice (integer) y devuelve el nomnre de una tabla. Se puede usar la función mysql_num_rows() para determinar el nombre de tablas en el puntero de resultado.

Ejemplo 1. mysql_tablename() Example

```
<?php
mysql_connect ("localhost:3306");
$result = mysql_list_tables ("wisconsin");
$i = 0;
while ($i < mysql_num_rows ($result)) {
    $tb_names[$i] = mysql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

mysql_thread_id (PHP 4 CVS only)

Return the current thread ID

int **mysql_thread_id** ([resource link_identifier]) \linebreak

mysql_thread_id() returns the current thread ID. If the connection is lost and you reconnect with mysql_ping(), the thread ID will change. This means you should not get the thread ID and store it for later. You should get it when you need it.

Ejemplo 1. mysql_thread_id() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$thread_id = mysql_thread_id($link);
if ($thread_id){
    printf ("current thread id is %d\n", $thread_id);
}
?>
```

The above example would produce the following output:

```
current thread id is 73
```

See also: mysql_ping(), mysql_list_processes().

mysql_unbuffered_query (PHP 4 >= 4.0.6)

Send an SQL query to MySQL, without fetching and buffering the result rows

resource **mysql_unbuffered_query** (string query [, resource link_identifier [, int result_mode]]) \linebreak

mysql_unbuffered_query() sends a SQL query *query* to MySQL, without fetching and buffering the result rows automatically, as mysql_query() does. On the one hand, this saves a considerable amount of memory with SQL queries that produce large result sets. On the other hand, you can start working on the result set immediately after the first row has been retrieved: you don't have to wait until the complete SQL query has been performed. When using multiple DB-connects, you have to specify the optional parameter *link_identifier*.

The optional *result_mode* parameter can be MYSQL_USE_RESULT and MYSQL_STORE_RESULT. It defaults to MYSQL_USE_RESULT, so the result is not buffered. See also mysql_query() for the counterpart of this behaviour.

> **Nota:** The benefits of **mysql_unbuffered_query()** come at a cost: You cannot use mysql_num_rows() on a result set returned from **mysql_unbuffered_query()**. You also have to fetch all result rows from an unbuffered SQL query, before you can send a new SQL query to MySQL.

See also: mysql_query().

LXIV. Mohawk Software session handler functions

msession is an interface to a high speed session daemon which can run either locally or remotely. It is designed to provide consistent session management for a PHP web farm.

The session server software can be found at http://www.mohawksoft.com/phoenix/.

msession_connect (PHP 4 >= 4.2.0)

Connect to msession server

bool **msession_connect** (string host, string port) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_count (PHP 4 >= 4.2.0)

Get session count

int **msession_count** (void) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_create (PHP 4 >= 4.2.0)

Create a session

bool **msession_create** (string session) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_destroy (PHP 4 >= 4.2.0)

Destroy a session

bool **msession_destroy** (string name) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_disconnect (PHP 4 >= 4.2.0)

Close connection to msession server

void **msession_disconnect** (void) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_find (PHP 4 >= 4.2.0)

Find value

array **msession_find** (string name, string value) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_get_array (PHP 4 >= 4.2.0)

Get array of ... ?

array **msession_get_array** (string session) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_get (PHP 4 >= 4.2.0)

Get value from session

string **msession_get** (string session, string name, string value) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_getdata (unknown)

Get data ... ?

string **msession_getdata** (string session) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_inc (PHP 4 >= 4.2.0)

Increment value in session

string **msession_inc** (string session, string name) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_list (PHP 4 >= 4.2.0)

List ... ?

array **msession_list** (void) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_listvar (PHP 4 >= 4.2.0)

List sessions with variable

array **msession_listvar** (string name) \linebreak

Returns an associative array of value, session for all sessions with a variable named *name*.

Used for searching sessions with common attributes.

msession_lock (PHP 4 >= 4.2.0)

Lock a session

int **msession_lock** (string name) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_plugin (PHP 4 >= 4.2.0)

Call an escape function within the msession personality plugin

string **msession_plugin** (string session, string val [, string param]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_randstr (PHP 4 >= 4.2.0)

Get random string

string **msession_randstr** (int param) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_set_array (PHP 4 >= 4.2.0)

Set array of ...

bool **msession_set_array** (string session, array tuples) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_set (PHP 4 >= 4.2.0)

Set value in session

bool **msession_set** (string session, string name, string value) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_setdata (unknown)

Set data ... ?

bool **msession_setdata** (string session, string value) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_timeout (PHP 4 >= 4.2.0)

Set/get session timeout

int **msession_timeout** (string session [, int param]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_uniq (PHP 4 >= 4.2.0)

Get uniq id

string **msession_uniq** (int param) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msession_unlock (PHP 4 >= 4.2.0)

Unlock a session

int **msession_unlock** (string session, int key) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

LXV. muscat functions

muscat_close (4.0.5 - 4.2.1 only)

Shuts down the muscat session and releases any memory back to PHP.

int **muscat_close** (resource muscat_handle) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

[Not back to the system, note!]

muscat_get (4.0.5 - 4.2.1 only)

Gets a line back from the core muscat API.

string **muscat_get** (resource muscat_handle) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

Returns a literal FALSE when there is no more to get (as opposed to ""). Use === FALSE or !== FALSE to check for this.

muscat_give (4.0.5 - 4.2.1 only)

Sends string to the core muscat API

int **muscat_give** (resource muscat_handle, string string) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

muscat_setup_net (4.0.5 - 4.2.1 only)

Creates a new muscat session and returns the handle.

resource **muscat_setup_net** (string muscat_host, int port) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

muscat_host is the hostname to connect to port is the port number to connect to - actually takes exactly the same args as fsockopen

muscat_setup (4.0.5 - 4.2.1 only)

Creates a new muscat session and returns the handle.

resource **muscat_setup** (int size [, string muscat_dir]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

Size is the ammount of memory in bytes to allocate for muscat muscat_dir is the muscat installation dir e.g. "/usr/local/empower", it defaults to the compile time muscat directory

LXVI. Funciones de Red

checkdnsrr (PHP 3, PHP 4)

Comprueba registros DNS correspondientes a nombres de máquinas en Internet o direcciones IP.

int **checkdnsrr** (string host [, string type]) \linebreak

Busca en DNS entradas del tipo *type* correspondientes a *host*. Devuelve verdadero si encuentra algún registro; devuelve falso si no encuentra ninguno o sucedió algún error.

type puede ser: A, MX, NS, SOA, PTR, CNAME, o ANY. Por defecto es MX.

host puede ser o la dirección IP de la forma xxxx.xxxx.xxxx.xxxx o el nombre de la máquina.

Ver también getmxrr(), gethostbyaddr(), gethostbyname(), gethostbynamel(), y named(8) en las páginas del manual.

closelog (PHP 3, PHP 4)

cierra la conexión con el logger del sistema

int **closelog** (void) \linebreak

closelog() cierra el descriptor que se está usando para escribir en el logger del sistema. El uso de **closelog()** es opcional.

debugger_off (PHP 3)

deshabilita el depurador interno de PHP

int **debugger_off** (void) \linebreak

Deshabilita el depurador interno de PHP. El depurador está aún en desarrollo.

debugger_on (PHP 3)

habilita el depurador interno de PHP

int **debugger_on** (string address) \linebreak

Habilita el depurador interno de PHP, conectándolo a *address*. El depurador esta aún en desarrollo.

define_syslog_variables (PHP 3, PHP 4)

Initializes all syslog related constants

void **define_syslog_variables** (void) \linebreak

Initializes all constants used in the syslog functions.

See also openlog(), syslog() and closelog().

fsockopen (PHP 3, PHP 4)

Abre una conexión de dominio Internet o Unix via sockets.

int **fsockopen** (string hostname, int port [, int errno [, string errstr [, double timeout]]]) \linebreak

Inicia una conexión de dominio Internet (AF_INET) o Unix (AF_UNIX). Para el domino Internet, abrirá una conexión TCP hacia el ordenador *hostname* en el puerto *port*. Para el dominio Unix, *hostname* se usará como ruta al socket, *port* debe ser 0 para este caso. El parámetro opcional *timeout* se puede usar para especificar un timeout en segundos para establecer la conexión.

fsockopen() devuelve un puntero a fichero, el cual se puede usar junto con las otras funciones de ficheros (como fgets(), fgetss(), fputs(), fclose(), feof()).

Si la llamada falla, esta devolverá falso y si los parámetros opcionales *errno* y *errstr* están presentes, indicarán el error del sistema que ocurrió en la llamada connect(). Si errno es 0 y la función devolviá falso, nos indica que el error ocurrió antes de la llamada connect(). Esto es debido principalmente a problemas inicializando el socket. Observe que los argumentos errno y errstr deben ser pasados por referencia.

Dependiendo del entorno, el dominio Unix o el parámetro opcional, timeout puede no estar disponible.

Por defecto, el socket será abierto en modo de bloqueo. Puede cambiarlo a modo de no bloqueo usando **set_socket_blocking()**.

Ejemplo 1. ejemplo con fsockopen

```
$fp = fsockopen("www.php.net", 80, &$errno, &$errstr, 30);
if(!$fp) {
echo "$errstr ($errno)<br>\n";
} else {
fputs($fp,"GET / HTTP/1.0\n\n");
while(!feof($fp)) {
echo fgets($fp,128);
}
fclose($fp);
}
```

Ver también: pfsockopen()

gethostbyaddr (PHP 3, PHP 4)

Obtiene el nombre de una máquina en Internet mediante su dirección IP.

string **gethostbyaddr** (string ip_address) \linebreak

Devuelve el nombre del ordenador conectado a Internet espeficado por el parámetro *ip_address*. Si ocurre un error, devuelve *ip_address*.

Ver también gethostbyname().

gethostbyname (PHP 3, PHP 4)

Obtiene la dirección IP correspondiente al nombre de una máquina conectada a Internet.

string **gethostbyname** (string hostname) \linebreak

Devuelve la dirección IP de una máquina conectada a Internet especificada por *hostname*.

Ver también gethostbyaddr().

gethostbynamel (PHP 3, PHP 4)

Obtiene una lista de direcciones IP correspondientea a los nombres de máquinas conectadas a Internet.

array **gethostbynamel** (string hostname) \linebreak

Devuleve una lista de direcciones IP pertenecientes a ordenadores especificados por *hostname*.

Ver también gethostbyname(), gethostbyaddr(), checkdnsrr(), getmxrr(), y named(8) en las páginas del manual.

getmxrr (PHP 3, PHP 4)

Obtiene registros MX correspondientes a una máquina conectada a Internet.

int **getmxrr** (string hostname, array mxhosts [, array weight]) \linebreak

Busca DNS de registros MX correspondientes a *hostname*. Devuelve verdadero si encuentra alg£n registro; devuelve falso si no encuentra ninguno o se produce un error.

La lista de registros MX encontrados se colocan en el array *mxhosts*. Si se proporciona el array *weight*, se rellenará con la información obtenida.

Ver también checkdnsrr(), gethostbyname(), gethostbynamel(), gethostbyaddr(), y named(8) de las páginas del manual.

getprotobyname (PHP 4)

Obtiene el n£mero asociado al nombre del protocolo

int **getprotobyname** (string name) \linebreak

getprotobyname() devuelve el n£mero asociado al nombre del protocolo *`name`* del fichero /etc/protocols. Ver también getprotobynumber().

getprotobynumber (PHP 4)

obtiene el nombre asociado al n£mero de protocolo

string **getprotobynumber** (int number) \linebreak

getprotobynumber() devuelve el nombre del protocolo asociado al *`number`* del protocolo en el fichero /etc/protocols. Ver también getprotobyname().

getservbyname (PHP 4)

obtiene el n£mero del puerto asociado al servicio Internet especificado

int **getservbyname** (string service, string protocol) \linebreak

getservbyname() devuelve el puerto que corresponde al *`service`* especificado por el *`protocol`* en /etc/services. *`protocol`* puede ser `tcp` o `udp`. Ver también getservbyport().

getservbyport (PHP 4)

obtiene el servicio Internet que correspondiente al puerto del protocolo especificado

string **getservbyport** (int port, string protocol) \linebreak

getservbyport() delvuelve el servicio Internet asociado al *`port`* para el *`protocol`* especificado en /etc/services. *`protocol`* puede ser `tcp` o `udp`. Ver también getservbyname().

ip2long (PHP 4)

Converts a string containing an (IPv4) Internet Protocol dotted address into a proper address.

int **ip2long** (string ip_address) \linebreak

The function **ip2long()** generates an IPv4 Internet network address from its Internet standard format (dotted string) representation.

Ejemplo 1. ip2long() Example

```
<?php
$ip = gethostbyname("www.example.com");
$out = "The following URLs are equivalent:<br>\n";
$out .= "http://www.example.com/, http://".$ip."/, and http://".sprintf("%u",ip2long($ip)).'
echo $out;
?>
```

Nota: Because PHP's integer type is signed, and many IP addresses will result in negative integers, you need to use the "%u" formatter of sprintf() or printf() to get the string representation of the unsigned IP address.

This second example shows how to print a converted address with the printf() function :

Ejemplo 2. Displaying an IP address

```
<?php
$ip = gethostbyname("www.example.com");
printf("%u\n", ip2long($ip));
echo $out;
?>
```

See also: long2ip()

long2ip (PHP 4)

Converts an (IPv4) Internet network address into a string in Internet standard dotted format

string **long2ip** (int proper_address) \linebreak

The function **long2ip()** generates an Internet address in dotted format (i.e.: aaa.bbb.ccc.ddd) from the proper address representation.

See also: ip2long()

openlog (PHP 3, PHP 4)

abre una conexión con el logger del sistema

int **openlog** (string ident, int option, int facility) \linebreak

openlog() abre una conexión con el logger del sistema. La cadena *ident* se a¤ade a cada mensaje. Los valores de *option* y *facility* se exponen en la siguiente sección. El uso de openlog() is opcional; Esta será llamada automaticamente por syslog() si fuera necesario, en este caso *ident* valdrá por defecto FALSE. Ver también syslog() y closelog().

pfsockopen (PHP 3>= 3.0.7, PHP 4)

Abre conexiones persistentes de dominio Internet o Unix.

int **pfsockopen** (string hostname, int port [, int errno [, string errstr [, int timeout]]]) \linebreak

Esta función se comporta exactamente como fsockopen() con la diferencia que la conexion no se cierra después de que termine el script. Esta es la versión persistente de fsockopen().

socket_get_status (PHP 4)

Returns information about an existing socket stream

array **socket_get_status** (resource socketstream) \linebreak

Returns information about an existing socket stream. This function only works on sockets created by fsockopen(), pfsockopen() or network sockets returned by fopen() when opening URLs. It does NOT work with sockets from the Socket extension. Currently returns four entries in the result array:

- *timed_out* (bool) - The socket timed out waiting for data
- *blocked* (bool) - The socket was blocked
- *eof* (bool) - Indicates EOF event
- *unread_bytes* (int) - Number of bytes left in the socket buffer

See also: Socket extension.

socket_set_blocking (PHP 4)

Set blocking/non-blocking mode on a socket

int **socket_set_blocking** (int socket descriptor, int mode) \linebreak

If *mode* is FALSE, the given socket descriptor will be switched to non-blocking mode, and if TRUE, it will be switched to blocking mode. This affects calls like fgets() that read from the socket. In non-blocking mode an fgets() call will always return right away while in blocking mode it will wait for data to become available on the socket.

This function was previously called as **set_socket_blocking()** but this usage is deprecated.

socket_set_timeout (PHP 4)

Set timeout period on a socket

bool **socket_set_timeout** (int socket descriptor, int seconds, int microseconds) \linebreak

Sets the timeout value on *socket descriptor*, expressed in the sum of *seconds* and *microseconds*.

Ejemplo 1. socket_set_timeout() Example

```
<?php
$fp = fsockopen("www.example.com", 80);
if(!$fp) {
    echo "Unable to open\n";
} else {
    fputs($fp,"GET / HTTP/1.0\n\n");
    $start = time();
    socket_set_timeout($fp, 2);
    $res = fread($fp, 2000);
    var_dump(socket_get_status($fp));
    fclose($fp);
    print $res;
}
?>
```

This function was previously called as **set_socket_timeout()** but this usage is deprecated.

See also: fsockopen() and fopen().

syslog (PHP 3, PHP 4)

genera un mensaje de sistema

int **syslog** (int priority, string message) \linebreak

syslog() genera un mensaje que será distribuido por el logger del sistema. *`priority`* es una combinación de la facility y el level, los valores se indicarán en la sección siguiente. El argumento restante es el mensaje a enviar, excepto que los dos caracteres `%m` sean reemplazados por la cadena de error (strerror) correspondiente al valor actual de errno.

Más información acerca de syslog se puede encontrar en las páginas del manual en equipos Unix.

En Windows NT, el servicio syslog es emulado usando el Log de Eventos.

LXVII. Ncurses terminal screen control functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

What is ncurses?

ncurses (new curses) is a free software emulation of curses in System V Rel 4.0 (and above). It uses terminfo format, supports pads, colors, multiple highlights, form characters and function key mapping.

Platforms

Ncurses is available for the following platforms:

- AIX
- BeOS
- Cygwin
- Digital Unix (aka OSF1)
- FreeBSD
- GNU/Linux
- HPUX
- IRIX
- OS/2
- SCO OpenServer
- Solaris
- SunOS

Requirements

You need the ncurses libraries and headerfiles. Download the latest version from the

ftp://ftp.gnu.org/pub/gnu/ncurses/ or from an other GNU-Mirror.

Installation

To get these functions to work, you have to compile the CGI version of PHP with `--with-ncurses`.

Ncurses predefined constants

Error codes

On error ncurses functions return NCURSES_ERR.

Colors

Tabla 1. ncurses color constants

constant	meaning
NCURSES_COLOR_BLACK	no color (black)
NCURSES_COLOR_WHITE	white
NCURSES_COLOR_RED	red - supported when terminal is in color mode
NCURSES_COLOR_GREEN	green - supported when terminal is in color mod
NCURSES_COLOR_YELLOW	yellow - supported when terminal is in color mod
NCURSES_COLOR_BLUE	blue - supported when terminal is in color mod
NCURSES_COLOR_CYAN	cyan - supported when terminal is in color mod
NCURSES_COLOR_MAGENTA	magenta - supported when terminal is in color mod

Keys

Tabla 2. ncurses key constants

constant	meaning
NCURSES_KEY_F0 - NCURSES_KEY_F64	function keys F1 - F64
NCURSES_KEY_DOWN	down arrow
NCURSES_KEY_UP	up arrow
NCURSES_KEY_LEFT	left arrow
NCURSES_KEY_RIGHT	right arrow
NCURSES_KEY_HOME	home key (upward+left arrow)

constant	**meaning**
NCURSES_KEY_BACKSPACE	backspace
NCURSES_KEY_DL	delete line
NCURSES_KEY_IL	insert line
NCURSES_KEY_DC	delete character
NCURSES_KEY_IC	insert char or enter insert mode
NCURSES_KEY_EIC	exit insert char mode
NCURSES_KEY_CLEAR	clear screen
NCURSES_KEY_EOS	clear to end of screen
NCURSES_KEY_EOL	clear to end of line
NCURSES_KEY_SF	scroll one line forward
NCURSES_KEY_SR	scroll one line backward
NCURSES_KEY_NPAGE	next page
NCURSES_KEY_PPAGE	previous page
NCURSES_KEY_STAB	set tab
NCURSES_KEY_CTAB	clear tab
NCURSES_KEY_CATAB	clear all tabs
NCURSES_KEY_SRESET	soft (partial) reset
NCURSES_KEY_RESET	reset or hard reset
NCURSES_KEY_PRINT	print
NCURSES_KEY_LL	lower left
NCURSES_KEY_A1	upper left of keypad
NCURSES_KEY_A3	upper right of keypad
NCURSES_KEY_B2	center of keypad
NCURSES_KEY_C1	lower left of keypad
NCURSES_KEY_C3	lower right of keypad
NCURSES_KEY_BTAB	back tab
NCURSES_KEY_BEG	beginning
NCURSES_KEY_CANCEL	cancel
NCURSES_KEY_CLOSE	close
NCURSES_KEY_COMMAND	cmd (command)
NCURSES_KEY_COPY	copy
NCURSES_KEY_CREATE	create
NCURSES_KEY_END	end
NCURSES_KEY_EXIT	exit
NCURSES_KEY_FIND	find
NCURSES_KEY_HELP	help
NCURSES_KEY_MARK	mark
NCURSES_KEY_MESSAGE	message

constant	meaning
NCURSES_KEY_MOVE	move
NCURSES_KEY_NEXT	next
NCURSES_KEY_OPEN	open
NCURSES_KEY_OPTIONS	options
NCURSES_KEY_PREVIOUS	previous
NCURSES_KEY_REDO	redo
NCURSES_KEY_REFERENCE	ref (reference)
NCURSES_KEY_REFRESH	refresh
NCURSES_KEY_REPLACE	replace
NCURSES_KEY_RESTART	restart
NCURSES_KEY_RESUME	resume
NCURSES_KEY_SAVE	save
NCURSES_KEY_SBEG	shiftet beg (beginning)
NCURSES_KEY_SCANCEL	shifted cancel
NCURSES_KEY_SCOMMAND	shifted command
NCURSES_KEY_SCOPY	shifted copy
NCURSES_KEY_SCREATE	shifted create
NCURSES_KEY_SDC	shifted delete char
NCURSES_KEY_SDL	shifted delete line
NCURSES_KEY_SELECT	select
NCURSES_KEY_SEND	shifted end
NCURSES_KEY_SEOL	shifted end of line
NCURSES_KEY_SEXIT	shifted exit
NCURSES_KEY_SFIND	shifted find
NCURSES_KEY_SHELP	shifted help
NCURSES_KEY_SHOME	shifted home
NCURSES_KEY_SIC	shifted input
NCURSES_KEY_SLEFT	shifted left arrow
NCURSES_KEY_SMESSAGE	shifted message
NCURSES_KEY_SMOVE	shifted move
NCURSES_KEY_SNEXT	shifted next
NCURSES_KEY_SOPTIONS	shifted options
NCURSES_KEY_SPREVIOUS	shifted previous
NCURSES_KEY_SPRINT	shifted print
NCURSES_KEY_SREDO	shifted redo
NCURSES_KEY_SREPLACE	shifted replace
NCURSES_KEY_SRIGHT	shifted right arrow
NCURSES_KEY_SRSUME	shifted resume

constant	meaning
NCURSES_KEY_SSAVE	shifted save
NCURSES_KEY_SSUSPEND	shifted suspend
NCURSES_KEY_UNDO	undo
NCURSES_KEY_MOUSE	mouse event has occured
NCURSES_KEY_MAX	maximum key value

Mouse

Tabla 3. mouse constants

Constant	meaning
NCURSES_BUTTON1_RELEASED - NCURSES_BUTTON4_RELEASED	button (1-4) released
NCURSES_BUTTON1_PRESSED - NCURSES_BUTTON4_PRESSED	button (1-4) pressed
NCURSES_BUTTON1_CLICKED - NCURSES_BUTTON4_CLICKED	button (1-4) clicked
NCURSES_BUTTON1_DOUBLE_CLICKED - NCURSES_BUTTON4_DOUBLE_CLICKED	button (1-4) double clicked
NCURSES_BUTTON1_TRIPLE_CLICKED - NCURSES_BUTTON4_TRIPLE_CLICKED	button (1-4) triple clicked
NCURSES_BUTTON_CTRL	ctrl pressed during click
NCURSES_BUTTON_SHIFT	shift pressed during click
NCURSES_BUTTON_ALT	alt pressed during click
NCURSES_ALL_MOUSE_EVENTS	report all mouse events
NCURSES_REPORT_MOUSE_POSITION	report mouse position

ncurses_addch (PHP 4 >= 4.1.0)

Add character at current position and advance cursor

int **ncurses_addch** (int ch) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_addchnstr (PHP 4 >= 4.2.0)

Add attributed string with specified length at current position

int **ncurses_addchnstr** (string s, int n) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_addchstr (PHP 4 >= 4.2.0)

Add attributed string at current position

int **ncurses_addchstr** (string s) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_addnstr (PHP 4 >= 4.2.0)

Add string with specified length at current position

int **ncurses_addnstr** (string s, int n) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_addstr (PHP 4 >= 4.2.0)

Output text at current position

int **ncurses_addstr** (string text) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_assume_default_colors (PHP 4 >= 4.2.0)

Define default colors for color 0

int **ncurses_assume_default_colors** (int fg, int bg) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_attroff (PHP 4 >= 4.1.0)

Turn off the given attributes

int **ncurses_attroff** (int attributes) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_attron (PHP 4 >= 4.1.0)

Turn on the given attributes

int **ncurses_attron** (int attributes) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_attrset (PHP 4 >= 4.1.0)

Set given attributes

int **ncurses_attrset** (int attributes) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_baudrate (PHP 4 >= 4.1.0)

Returns baudrate of terminal

int **ncurses_baudrate** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_beep (PHP 4 >= 4.1.0)

Let the terminal beep

int **ncurses_beep** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_beep() sends an audlible alert (bell) and if its not possible flashes the screen. Returns `FALSE` on success, otherwise `TRUE`.

See also: ncurses_flash()

ncurses_bkgd (PHP 4 >= 4.1.0)

Set background property for terminal screen

int **ncurses_bkgd** (int attrchar) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_bkgdset (PHP 4 >= 4.1.0)

Control screen background

void **ncurses_bkgdset** (int attrchar) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_border (PHP 4 >= 4.2.0)

Draw a border around the screen using attributed characters

int **ncurses_border** (int left, int right, int top, int bottom, int tl_corner, int tr_corner, int bl_corner, int br_corner) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_can_change_color (PHP 4 >= 4.1.0)

Check if we can change terminals colors

bool **ncurses_can_change_color** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The function **ncurses_can_change_color()** returns `TRUE` or `FALSE`, depending on whether the terminal has color capabilities and whether the programmer can change the colors.

ncurses_cbreak (PHP 4 >= 4.1.0)

Switch of input buffering

bool **ncurses_cbreak** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_cbreak() disables line buffering and character processing (interrupt and flow control characters are unaffected), making characters typed by the user immediately available to the program.

ncurses_cbreak() returns `TRUE` or NCURSES_ERR if any error occured.

See also: ncurses_nocbreak()

ncurses_clear (PHP 4 >= 4.1.0)

Clear screen

bool **ncurses_clear** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_clear() clears the screen completely without setting blanks. Returns `FALSE` on success, otherwise `TRUE`.

Note: **ncurses_clear()** clears the screen without setting blanks, which have the current background rendition. To clear screen with blanks, use ncurses_erase().

See also: ncurses_erase()

ncurses_clrtobot (PHP 4 >= 4.1.0)

Clear screen from current position to bottom

bool **ncurses_clrtobot** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_clrtobot() erases all lines from cursor to end of screen and creates blanks. Blanks created by **ncurses_clrtobot()** have the current background rendition. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_clear(), ncurses_clrtoeol()

ncurses_clrtoeol (PHP 4 >= 4.1.0)

Clear screen from current position to end of line

bool **ncurses_clrtoeol** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_clrtoeol() erases the current line from cursor position to the end. Blanks created by **ncurses_clrtoeol()** have the current background rendition. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_clear(), ncurses_clrtobot()

ncurses_color_set (PHP 4 >= 4.1.0)

Set fore- and background color

int **ncurses_color_set** (int pair) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_curs_set (PHP 4 >= 4.1.0)

Set cursor state

int **ncurses_curs_set** (int visibility) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_def_prog_mode (PHP 4 >= 4.1.0)

Saves terminals (program) mode

bool **ncurses_def_prog_mode** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_def_prog_mode() saves the current terminal modes for program (in curses) for use by **ncurses_reset_prog_mode()**. Returns `FALSE` on success, otherwise `TRUE`.

See also: **ncurses_reset_prog_mode()**

ncurses_def_shell_mode (PHP 4 >= 4.1.0)

Saves terminals (shell) mode

bool **ncurses_def_shell_mode** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_def_shell_mode() saves the current terminal modes for shell (not in curses) for use by **ncurses_reset_shell_mode()**. Returns `FALSE` on success, otherwise `TRUE`.

See also: **ncurses_reset_shell_mode()**

ncurses_define_key (PHP 4 >= 4.2.0)

Define a keycode

int **ncurses_define_key** (string definition, int keycode) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_delay_output (PHP 4 >= 4.1.0)

Delay output on terminal using padding characters

int **ncurses_delay_output** (int milliseconds) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_delch (PHP 4 >= 4.1.0)

Delete character at current position, move rest of line left

bool **ncurses_delch** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_delch() deletes the character under the cursor. All characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change. Returns `FALSE` on success, otherwise `TRUE`.

See also: ncurses_deleteln()

ncurses_deleteln (PHP 4 >= 4.1.0)

Delete line at current position, move rest of screen up

bool **ncurses_deleteln** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_deleteln() deletes the current line under cursorposition. All lines below the current line are moved up one line. The bottom line of window is cleared. Cursor position does not change. Returns `FALSE` on success, otherwise `TRUE`.

See also: ncurses_delch()

ncurses_delwin (PHP 4 >= 4.1.0)

Delete a ncurses window

int **ncurses_delwin** (resource window) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_doupdate (PHP 4 >= 4.1.0)

Write all prepared refreshes to terminal

bool **ncurses_doupdate** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_doupdate()() compares the virtual screen to the physical screen and updates the physical screen. This way is more effective than using multiple refresh calls. Returns `FALSE` on success, `TRUE` if any error occured.

ncurses_echo (PHP 4 >= 4.1.0)

Activate keyboard input echo

bool **ncurses_echo** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_echo() enables echo mode. All characters typed by user are echoed by ncurses_getch(). Returns `FALSE` on success, `TRUE` if any error occured.

To disable echo mode use ncurses_noecho().

ncurses_echochar (PHP 4 >= 4.1.0)

Single character output including refresh

int **ncurses_echochar** (int character) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_end (PHP 4 >= 4.1.0)

Stop using ncurses, clean up the screen

int **ncurses_end** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_erase (PHP 4 >= 4.1.0)

Erase terminal screen

bool **ncurses_erase** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_erase() fills the terminal screen with blanks. Created blanks have the current background rendition, set by ncurses_bkgd(). Returns `FALSE` on success, `TRUE` if any error occured.

See also: ncurses_bkgd(), ncurses_clear()

ncurses_erasechar (PHP 4 >= 4.1.0)

Returns current erase character

string **ncurses_erasechar** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_erasechar() returns the current erase char character.

See also: ncurses_killchar()

ncurses_filter (PHP 4 >= 4.1.0)

int **ncurses_filter** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_flash (PHP 4 >= 4.1.0)

Flash terminal screen (visual bell)

bool **ncurses_flash** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_flash() flashes the screen, and if its not possible, sends an audible alert (bell). Returns `FALSE` on success, otherwise `TRUE`.

See also: ncurses_beep()

ncurses_flushinp (PHP 4 >= 4.1.0)

Flush keyboard input buffer

bool **ncurses_flushinp** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The **ncurses_flushinp()** throws away any typeahead that has been typed and has not yet been read by your program. Returns `FALSE` on success, otherwise `TRUE`.

ncurses_getch (PHP 4 >= 4.1.0)

Read a character from keyboard

int **ncurses_getch** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_getmouse (PHP 4 >= 4.2.0)

Reads mouse event

bool **ncurses_getmouse** (array mevent) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_getmouse() reads mouse event out of queue. Function **ncurses_getmouse()** will return ;`FALSE` if a mouse event is actually visible in the given window, otherwise it will return `TRUE`. Event options will be delivered in parameter *`mevent`*, which has to be an array, passed by reference (see example below). On success an associative array with following keys will be delivered:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

Ejemplo 1. ncurses_getmouse() example

```
switch (ncurses_getch){
  case NCURSES_KEY_MOUSE:
    if (!ncurses_getmouse(&$mevent)){
      if ($mevent["mmask"] & NCURSES_MOUSE_BUTTON1_PRESSED){
        $mouse_x = $mevent["x"]; // Save mouse position
```

```
        $mouse_y = $mevent["y"];
      }
    }
  break;

  default:
    ....
}
```

See also: ncurses_ungetmouse()

ncurses_halfdelay (PHP 4 >= 4.1.0)

Put terminal into halfdelay mode

int **ncurses_halfdelay** (int tenth) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_has_colors (PHP 4 >= 4.1.0)

Check if terminal has colors

bool **ncurses_has_colors** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_has_colors() returns `TRUE` or `FALSE` depending on whether the terminal has color capacitites.

See also: ncurses_can_change_color()

ncurses_has_ic (PHP 4 >= 4.1.0)

Check for insert- and delete-capabilities

bool **ncurses_has_ic** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_has_ic() checks terminals insert- and delete capabilitites. It returns `TRUE` when terminal has insert/delete-capabilities, otherwise `FALSE`.

See also: ncurses_has_il()

ncurses_has_il (PHP 4 >= 4.1.0)

Check for line insert- and delete-capabilities

bool **ncurses_has_il** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_has_il() checks terminals insert- and delete-line-capabilities. It returns `TRUE` when terminal has insert/delete-line capabilities, otherwise `FALSE`

See also: ncurses_has_ic()

ncurses_has_key (PHP 4 >= 4.1.0)

Check for presence of a function key on terminal keyboard

int **ncurses_has_key** (int keycode) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_hline (PHP 4 >= 4.2.0)

Draw a horizontal line at current position using an attributed character and max. n characters long

int **ncurses_hline** (int charattr, int n) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_inch (PHP 4 >= 4.1.0)

Get character and attribute at current position

string **ncurses_inch** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_inch() returns the character from the current position.

ncurses_init_color (PHP 4 >= 4.2.0)

Set new RGB value for color

int **ncurses_init_color** (int color, int r, int g, int b) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_init_pair (PHP 4 >= 4.1.0)

Allocate a color pair

int **ncurses_init_pair** (int pair, int fg, int bg) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_init (PHP 4 >= 4.1.0)

Initialize ncurses

int **ncurses_init** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_insch (PHP 4 >= 4.1.0)

Insert character moving rest of line including character at current position

int **ncurses_insch** (int character) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_insdelln (PHP 4 >= 4.1.0)

Insert lines before current line scrolling down (negative numbers delete and scroll up)

int **ncurses_insdelln** (int count) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_insertln (PHP 4 >= 4.1.0)

Insert a line, move rest of screen down

bool **ncurses_insertln** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_insertln() inserts a new line above the current line. The bottom line will be lost.

ncurses_insstr (PHP 4 >= 4.2.0)

Insert string at current position, moving rest of line right

int **ncurses_insstr** (string text) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_instr (PHP 4 >= 4.2.0)

Reads string from terminal screen

int **ncurses_instr** (string buffer) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_instr() returns the number of charaters read from the current character position until end of line. `buffer` contains the characters. Atrributes are stripped from the characters.

ncurses_isendwin (PHP 4 >= 4.1.0)

Ncurses is in endwin mode, normal screen output may be performed

bool **ncurses_isendwin** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_isendwin() returns `TRUE`, if **ncurses_endwin()** has been called without any subsequent calls to ncurses_wrefresh(), otherwise `FALSE`.

See also: **ncurses_endwin() ncurses_wrefresh()()**

ncurses_keyok (PHP 4 >= 4.2.0)

Enable or disable a keycode

int **ncurses_keyok** (int keycode, bool enable) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_killchar (PHP 4 >= 4.1.0)

Returns current line kill character

bool **ncurses_killchar** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_killchar() returns the current line kill character.

See also: ncurses_erasechar()

ncurses_longname (PHP 4 >= 4.2.0)

Returns terminals description

string **ncurses_longname** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_longname() returns a verbose description of the terminal. The descritpion is truncated to 128 characters. On Error **ncurses_longname()** returns NULL.

See also: ncurses_termname()

ncurses_mouseinterval (PHP 4 >= 4.1.0)

Set timeout for mouse button clicks

int **ncurses_mouseinterval** (int milliseconds) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mousemask (PHP 4 >= 4.2.0)

Sets mouse options

int **ncurses_mousemask** (int newmask, int oldmask) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Function **ncurses_mousemask()** will set mouse events to be reported. By default no mouse events will be reported. The function **ncurses_mousemask()** will return a mask to indicated which of the in parameter `newmask` specified mouse events can be reported. On complete failure, it returns 0. In parameter `oldmask`, which is passed by reference **ncurses_mousemask()** returns the previous value of mouse event mask. Mouse events are represented bei NCURSES_KEY_MOUSE in the

ncurses_wgetch() input stream. To read the event data and pop the event of of queue, call ncurses_getmouse().

As a side effect, setting a zero mousemask in *newmask* turns off the mouse pointer. Setting a non zero value turns mouse pointer on.

mouse mask options can be set with the following predefined constants:

- NCURSES_BUTTON1_PRESSED
- NCURSES_BUTTON1_RELEASED
- NCURSES_BUTTON1_CLICKED
- NCURSES_BUTTON1_DOUBLE_CLICKED
- NCURSES_BUTTON1_TRIPLE_CLICKED
- NCURSES_BUTTON2_PRESSED
- NCURSES_BUTTON2_RELEASED
- NCURSES_BUTTON2_CLICKED
- NCURSES_BUTTON2_DOUBLE_CLICKED
- NCURSES_BUTTON2_TRIPLE_CLICKED
- NCURSES_BUTTON3_PRESSED
- NCURSES_BUTTON3_RELEASED
- NCURSES_BUTTON3_CLICKED
- NCURSES_BUTTON3_DOUBLE_CLICKED
- NCURSES_BUTTON3_TRIPLE_CLICKED
- NCURSES_BUTTON4_PRESSED
- NCURSES_BUTTON4_RELEASED
- NCURSES_BUTTON4_CLICKED
- NCURSES_BUTTON4_DOUBLE_CLICKED
- NCURSES_BUTTON4_TRIPLE_CLICKED
- NCURSES_BUTTON_SHIFT>
- NCURSES_BUTTON_CTRL
- NCURSES_BUTTON_ALT
- NCURSES_ALL_MOUSE_EVENTS
- NCURSES_REPORT_MOUSE_POSITION

See also: ncurses_getmouse(), ncurses_ungetmouse() **ncurese_getch()**

Ejemplo 1. ncurses_mousemask() example

```
$newmask = NCURSES_BUTTON1_CLICKED + NCURSES_BUTTON1_RELEASED;
$mask = ncurses_mousemask($newmask, &$oldmask);
if ($mask & $newmask){
  printf ("All specified mouse options will be supported\n");
}
```

ncurses_move (PHP 4 >= 4.1.0)

Move output position

int **ncurses_move** (int y, int x) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvaddch (PHP 4 >= 4.2.0)

Move current position and add character

int **ncurses_mvaddch** (int y, int x, int c) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvaddchnstr (PHP 4 >= 4.2.0)

Move position and add attrributed string with specified length

int **ncurses_mvaddchnstr** (int y, int x, string s, int n) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvaddchstr (PHP 4 >= 4.2.0)

Move position and add attributed string

int **ncurses_mvaddchstr** (int y, int x, string s) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvaddnstr (PHP 4 >= 4.2.0)

Move position and add string with specified length

int **ncurses_mvaddnstr** (int y, int x, string s, int n) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvaddstr (PHP 4 >= 4.2.0)

Move position and add string

int **ncurses_mvaddstr** (int y, int x, string s) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvcur (PHP 4 >= 4.2.0)

Move cursor immediately

int **ncurses_mvcur** (int old_y, int old_x, int new_y, int new_x) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvdelch (PHP 4 >= 4.2.0)

Move position and delete character, shift rest of line left

int **ncurses_mvdelch** (int y, int x) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvgetch (PHP 4 >= 4.2.0)

Move position and get character at new position

int **ncurses_mvgetch** (int y, int x) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvhline (PHP 4 >= 4.2.0)

Set new position and draw a horizontal line using an attributed character and max. n characters long

int **ncurses_mvhline** (int y, int x, int attrchar, int n) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvinch (PHP 4 >= 4.2.0)

Move position and get attributed character at new position

int **ncurses_mvinch** (int y, int x) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvvline (unknown)

Set new position and draw a vertical line using an attributed character and max. n characters long

int **ncurses_mvvline** (int y, int x, int attrchar, int n) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_mvwaddstr (PHP 4 >= 4.2.0)

Add string at new position in window

int **ncurses_mvwaddstr** (resource window, int y, int x, string text) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_napms (PHP 4 >= 4.1.0)

Sleep

int **ncurses_napms** (int milliseconds) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_newwin (PHP 4 >= 4.1.0)

Create a new window

int **ncurses_newwin** (int rows, int cols, int y, int x) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_nl (PHP 4 >= 4.1.0)

Translate newline and carriage return / line feed

bool **ncurses_nl** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_nocbreak (PHP 4 >= 4.1.0)

Switch terminal to cooked mode

bool **ncurses_nocbreak** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_nocbreak() routine returns terminal to normal (cooked) mode. Initially the terminal may or may not in cbreak mode as the mode is inherited. Therefore a program should call ncurses_cbreak() and **ncurses_nocbreak()** explicitly. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_cbreak()

ncurses_noecho (PHP 4 >= 4.1.0)

Switch off keyboard input echo

bool **ncurses_noecho** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_noecho() prevents echoing of user typed characters. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_echo(), ncurses_getch()

ncurses_nonl (PHP 4 >= 4.1.0)

Do not translate newline and carriage return / line feed

bool **ncurses_nonl** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_noqiflush (PHP 4 >= 4.1.0)

Do not flush on signal characters

int **ncurses_noqiflush** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_noraw (PHP 4 >= 4.1.0)

Switch terminal out of raw mode

bool **ncurses_noraw** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_noraw() switches the terminal out of raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_raw(), ncurses_cbreak(), ncurses_nocbreak()

ncurses_putp (PHP 4 >= 4.2.0)

int **ncurses_putp** (string text) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_qiflush (PHP 4 >= 4.1.0)

Flush on signal characters

int **ncurses_qiflush** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_raw (PHP 4 >= 4.1.0)

Switch terminal into raw mode

bool **ncurses_raw** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_raw() places the terminal in raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_noraw(), ncurses_cbreak(), ncurses_nocbreak()

ncurses_refresh (PHP 4 >= 4.1.0)

Refresh screen

int **ncurses_refresh** (int ch) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_resetty (PHP 4 >= 4.1.0)

Restores saved terminal state

bool **ncurses_resetty** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Function **ncurses_resetty()** restores the terminal state, which was previously saved by calling ncurses_savetty(). This function always returns `FALSE`.

See also: ncurses_savetty()

ncurses_savetty (PHP 4 >= 4.1.0)

Saves terminal state

bool **ncurses_savetty** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Function **ncurses_savetty()** saves the current terminal state. The saved terminal state can be restored with function ncurses_resetty(). **ncurses_savetty()** always returns `FALSE`.

See also: ncurses_resetty()

ncurses_scr_dump (PHP 4 >= 4.2.0)

Dump screen content to file

int **ncurses_scr_dump** (string filename) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_scr_init (PHP 4 >= 4.2.0)

Initialize screen from file dump

int **ncurses_scr_init** (string filename) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_scr_restore (PHP 4 >= 4.2.0)

Restore screen from file dump

int **ncurses_scr_restore** (string filename) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_scr_set (PHP 4 >= 4.2.0)

Inherit screen from file dump

int **ncurses_scr_set** (string filename) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_scrl (PHP 4 >= 4.1.0)

Scroll window content up or down without changing current position

int **ncurses_scrl** (int count) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_slk_attr (PHP 4 >= 4.1.0)

Returns current soft label key attribute

bool **ncurses_slk_attr** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_slk_attr() returns the current soft label key attribute. On error returns `TRUE`, otherwise `FALSE`.

ncurses_slk_attroff (PHP 4 >= 4.1.0)

int **ncurses_slk_attroff** (int intarg) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_slk_attron (PHP 4 >= 4.1.0)

int **ncurses_slk_attron** (int intarg) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_slk_attrset (PHP 4 >= 4.1.0)

int **ncurses_slk_attrset** (int intarg) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_slk_clear (PHP 4 >= 4.1.0)

Clears soft labels from screen

bool **ncurses_slk_clear** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The function **ncurses_slk_clear()** clears soft label keys from screen. Returns `TRUE` on error, otherwise `FALSE`.

ncurses_slk_color (PHP 4 >= 4.1.0)

Sets color for soft label keys

int **ncurses_slk_color** (int intarg) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_slk_init (PHP 4 >= 4.1.0)

Initializes soft label key functions

bool **ncurses_slk_init** (int format) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Funtion **ncurses_slk_init()** must be called before **ncurses_initscr()** or **ncurses_newterm()** is called. If **ncurses_initscr()** eventually uses a line from stdscr to emulate the soft labels, then *format* determines how the labels are arranged of the screen. Setting *format* to 0 indicates a 3-2-3 arrangement of the labels, 1 indicates a 4-4 arrangement and 2 indicates the PC like 4-4-4 mode, but in addition an index line will be created.

ncurses_slk_noutrefresh (PHP 4 >= 4.1.0)

Copies soft label keys to virtual screen

bool **ncurses_slk_noutrefresh** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_slk_refresh (PHP 4 >= 4.1.0)

Copies soft label keys to screen

bool **ncurses_slk_refresh** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_slk_refresh() copies soft label keys from virtual screen to physical screen. Returns `TRUE` on error, otherwise `FALSE`.

ncurses_slk_restore (PHP 4 >= 4.1.0)

Restores soft label keys

bool **ncurses_slk_restore** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The function **ncurses_slk_restore()** restores the soft label keys after ncurses_slk_clear() has been performed.

ncurses_slk_touch (PHP 4 >= 4.1.0)

Fources output when ncurses_slk_noutrefresh is performed

bool **ncurses_slk_touch** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The **ncurses_slk_touch()** function forces all the soft labels to be output the next time a ncurses_slk_noutrefresh() is performed.

ncurses_standend (PHP 4 >= 4.1.0)

Stop using ’standout’ attribute

int **ncurses_standend** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_standout (PHP 4 >= 4.1.0)

Start using 'standout' attribute

int **ncurses_standout** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_start_color (PHP 4 >= 4.1.0)

Start using colors

int **ncurses_start_color** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_termattrs (PHP 4 >= 4.1.0)

Returns a logical OR of all attribute flags supported by terminal

bool **ncurses_termattrs** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_termname (PHP 4 >= 4.2.0)

Returns terminals (short)-name

string **ncurses_termname** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_termname() returns terminals shortname. The shortname is truncated to 14 characters. On error **ncurses_termname()** returns NULL.

See also: ncurses_longname()

ncurses_timeout (PHP 4 >= 4.1.0)

Set timeout for special key sequences

void **ncurses_timeout** (int millisec) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_typeahead (PHP 4 >= 4.1.0)

Specify different filedescriptor for typeahead checking

int **ncurses_typeahead** (int fd) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_ungetch (PHP 4 >= 4.1.0)

Put a character back into the input stream

int **ncurses_ungetch** (int keycode) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_ungetmouse (PHP 4 >= 4.2.0)

Pushes mouse event to queue

bool **ncurses_ungetmouse** (array mevent) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

ncurses_getmouse() pushes a KEY_MOUSE event onto the unput queue and associates with this event the given state sata and screen-relative character cell coordinates, specified in `mevent`. Event options will be specified in associative array `mevent`:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

ncurses_ungetmouse() returns `FALSE` on success, otherwise `TRUE`.

See also: ncurses_getmouse()

ncurses_use_default_colors (PHP 4 >= 4.1.0)

Assign terminal default colors to color id -1

bool **ncurses_use_default_colors** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_use_env (PHP 4 >= 4.1.0)

Control use of environment information about terminal size

void **ncurses_use_env** (bool flag) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_use_extended_names (PHP 4 >= 4.1.0)

Control use of extended names in terminfo descriptions

int **ncurses_use_extended_names** (bool flag) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_vidattr (PHP 4 >= 4.1.0)

int **ncurses_vidattr** (int intarg) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_vline (PHP 4 >= 4.2.0)

Draw a vertical line at current position using an attributed character and max. n characters long

int **ncurses_vline** (int charattr, int n) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

ncurses_wrefresh (PHP 4 >= 4.2.0)

Refresh window on terminal screen

int **ncurses_wrefresh** (resource window) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

undocumented

LXVIII. Lotus Notes functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

notes_body (PHP 4 >= 4.0.5)

Open the message msg_number in the specified mailbox on the specified server (leave serv

array **notes_body** (string server, string mailbox, int msg_number) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_copy_db (PHP 4 >= 4.0.5)

Create a note using form form_name

string **notes_copy_db** (string from_database_name, string to_database_name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_create_db (PHP 4 >= 4.0.5)

Create a Lotus Notes database

bool **notes_create_db** (string database_name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_create_note (PHP 4 >= 4.0.5)

Create a note using form form_name

string **notes_create_note** (string database_name, string form_name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_drop_db (PHP 4 >= 4.0.5)

Drop a Lotus Notes database

bool **notes_drop_db** (string database_name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_find_note (PHP 4 >= 4.0.5)

Returns a note id found in database_name. Specify the name of the note. Leaving type bla

bool **notes_find_note** (string database_name, string name [, string type]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_header_info (PHP 4 >= 4.0.5)

Open the message msg_number in the specified mailbox on the specified server (leave serv

object **notes_header_info** (string server, string mailbox, int msg_number) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_list_msgs (PHP 4 >= 4.0.5)

Returns the notes from a selected database_name

bool **notes_list_msgs** (string db) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_mark_read (PHP 4 >= 4.0.5)

Mark a note_id as read for the User user_name

string **notes_mark_read** (string database_name, string user_name, string note_id) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_mark_unread (PHP 4 >= 4.0.5)

Mark a note_id as unread for the User user_name

string **notes_mark_unread** (string database_name, string user_name, string note_id) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_nav_create (PHP 4 >= 4.0.5)

Create a navigator name, in database_name

bool **notes_nav_create** (string database_name, string name) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_search (PHP 4 >= 4.0.5)

Find notes that match keywords in database_name

string **notes_search** (string database_name, string keywords) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_unread (PHP 4 >= 4.0.5)

Returns the unread note id’s for the current User user_name

string **notes_unread** (string database_name, string user_name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

notes_version (PHP 4 >= 4.0.5)

Get the version Lotus Notes

string **notes_version** (string database_name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

LXIX. ODBC functions

odbc_autocommit (PHP 3>= 3.0.6, PHP 4)

Interruptor de comportamiento de auto-entrega

int **odbc_autocommit** (int connection_id [, int OnOff]) \linebreak

Sin el parametro *`OnOff`*, esta funcion devuelve el estado de auto-entrega para *`connection_id`*. Devuelve `TRUE` si auto-entrega esta habilitado, y `FALSE` si no lo esta o ha ocurrido un error.

Si *`OnOff`* es `TRUE`, auto-entrega esta activado, si es `FALSE` auto-entrega esta desactivado. Devuelve `TRUE` cuando se cumple, `FALSE` cuando falla.

Por defecto, auto-entrega es para una conexion. Desabilitar auto-entrega es como comenzar una transaccion.

Ver tambien odbc_commit() y odbc_rollback().

odbc_binmode (PHP 3>= 3.0.6, PHP 4)

Manejo de campos de datos binarios

int **odbc_binmode** (int result_id, int mode) \linebreak

(Elementos afectados ODBC SQL: BINARY, VARBINARY, LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: Paso a traves de datos binarios
- ODBC_BINMODE_RETURN: Devuelve como es
- ODBC_BINMODE_CONVERT: Devuelve convertido en caracter

Cuando los datos binarios en SQL son convertidos a datos caracter en C, cada byte (8 bits) de datos fuente es representada como dos caracteres en ASCII. Esos caracteres son la representacion en ASCII de los numeros en su forma Hexadecimal. Por ejemplo, un 00000001 binario es convertido a `"01"` y un 11111111 binario es convertido a `"FF"`.

Tabla 1. Manejo de LONGVARBINARY

modo binario	longreadlen	resultado
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_RETURN	0	passthru
ODBC_BINMODE_CONVERT	0	passthru
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_PASSTHRU	>0	passthru
ODBC_BINMODE_RETURN	>0	Devuleve como es
ODBC_BINMODE_CONVERT	>0	Devuelve como caracter

Si usamos odbc_fetch_into(), passthru significara que una cadena vacia es devuelta por esas campos.

Si *result_id* es 0, las definiciones se aplican por defecto para nuevos resultados.

> **Nota:** Por defecto, longreadlen es 4096 y el modo binario por defecto es ODBC_BINMODE_RETURN. El manejo de campos binarias largas tambien esta afectado por odbc_longreadlen()

odbc_close_all (PHP 3>= 3.0.6, PHP 4)

Cierra todas las conexiones ODBC

void **odbc_close_all** (void) \linebreak

odbc_close_all() cerrara todas las conexiones a servidor(es) de bases de datos.

> **Nota:** Esta funcion fallara si hay transacciones abiertas sobre esta conexion. La conexion quedara abierta en ese caso.

odbc_close (PHP 3>= 3.0.6, PHP 4)

Cierra una conexion ODBC

void **odbc_close** (int connection_id) \linebreak

odbc_close() cerrara la conexion al servidor de bases datos asociado con el identificador de conexion dado.

> **Nota:** Esta funcion fallara si hay transacciones abiertas sobre esta conexion. La conexion quedara abierta en ese caso.

odbc_columnprivileges (PHP 4)

Returns a result identifier that can be used to fetch a list of columns and associated privileges

int **odbc_columnprivileges** (resource connection_id [, string qualifier [, string owner [, string table_name [, string column_name]]]]) \linebreak

Lists columns and associated privileges for the given table. Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *`column_name`* argument accepts search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_columns (PHP 4)

Lists the column names in specified tables. Returns a result identifier containing the information.

int **odbc_columns** (resource connection_id [, string qualifier [, string owner [, string table_name [, string column_name]]]]) \linebreak

Lists all columns in the requested range. Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX

- NULLABLE
- REMARKS

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner*, *table_name* and *column_name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

See also odbc_columnprivileges() to retrieve associated privileges.

odbc_commit (PHP 3>= 3.0.6, PHP 4)

Entrega una transaccion ODBC

int **odbc_commit** (int connection_id) \linebreak

Devuelve: TRUE si la operacion se realiza con exito, FALSE si falla. Todas las transacciones pendientes sobre *connection_id* son entregadas.

odbc_connect (PHP 3>= 3.0.6, PHP 4)

Conecta a una fuente de datos

int **odbc_connect** (string dsn, string user, string password [, int cursor_type]) \linebreak

Devuelve una conexion ODBC id, o 0 (FALSE) cuando ocurre un error.

La conexion id devuelta por estas funciones es necesaria para otras funciones ODBC. Se pueden tener multiples conexiones abiertas a la vez. El opcional cuarto parametro asigna el tipo de cursor que va a ser usado para esta conexion. Este parametro normalmente no es necesario, pero puede ser util para trabajar sobre problemas con algunos drivers ODBC.

Con algunos drivers ODBC, si ejecutamos un procedimiento complejo, este puede fallar con un error similar a: "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it". Usando SQL_CUR_USE_ODBC se puede evitar ese error. Algunos drivers tampoco soportan el parametro row_number en odbc_fetch_row(). SQL_CUR_USE_ODBC tambien podria ayudar en ese caso.

Las siguientes constantes son definidas por tipos de cursor:

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

Para conexiones persistentes ver odbc_pconnect().

odbc_cursor (PHP 3>= 3.0.6, PHP 4)

Toma un nombre de cursor

string **odbc_cursor** (int result_id) \linebreak

odbc_cursor devolvera un nombre de cursor para el result_id dado.

odbc_do (PHP 3>= 3.0.6, PHP 4)

sinonimo de odbc_exec()

string **odbc_do** (int conn_id, string query) \linebreak

odbc_do ejecutara una consulta (query) sobre la conexion dada

odbc_error (PHP 4 >= 4.0.5)

Get the last error code

string **odbc_error** ([resource connection_id]) \linebreak

Returns a six-digit ODBC state, or an empty string if there has been no errors. If *`connection_id`* is specified, the last state of that connection is returned, else the last state of any connection is returned.

See also: odbc_errormsg() and odbc_exec().

odbc_errormsg (PHP 4 >= 4.0.5)

Get the last error message

string **odbc_errormsg** ([resource connection_id]) \linebreak

Returns a string containing the last ODBC error message, or an empty string if there has been no errors. If *`connection_id`* is specified, the last state of that connection is returned, else the last state of any connection is returned.

See also: odbc_error() and odbc_exec().

odbc_exec (PHP 3>= 3.0.6, PHP 4)

Prepara o ejecuta una declaracion SQL

int **odbc_exec** (int connection_id, string query_string) \linebreak

Devuelve `FALSE` en caso de error. Devuelve un indetificador ODBC si el comando SQL fue ejecutado satisfactoriamente.

odbc_exec() enviara una declaracion SQL al servidor de bases de datos specificado por *`connection_id`*. Este parametro debe ser un indetificador valido devuelto por odbc_connect() o odbc_pconnect().

Ver tambien: odbc_prepare() y odbc_execute() para ejecucion multiple de declaraciones SQL.

odbc_execute (PHP 3>= 3.0.6, PHP 4)

ejecuta una declaracion preparada

int **odbc_execute** (int result_id [, array parameters_array]) \linebreak

Ejecuta uan declaracion preparada con odbc_prepare(). Devuelve `TRUE` cuando la ejecucion es satisfactoria, `FALSE` en otro caso. Introducir el vector *`arameters_array`* solo es necesario si realmente tenemos parametros en la declaracion.

odbc_fetch_array (PHP 4 >= 4.0.2)

Fetch a result row as an associative array

array **odbc_fetch_array** (resource result [, int rownumber]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

odbc_fetch_into (PHP 3>= 3.0.6, PHP 4)

Busca un registro de resutados dentro de un vector

int **odbc_fetch_into** (int result_id [, int rownumber, array result_array]) \linebreak

Devuelve el numero de campos en el resultado; `FALSE` on error. *`result_array`* debe ser pasado por referencia, pero puede ser de cualquier tipo, desde este sera convertido a tipo vector. El vector contendra el valor de campo inicial empezando en indice de vector 0.

odbc_fetch_object (PHP 4 >= 4.0.2)

Fetch a result row as an object

object **odbc_fetch_object** (resource result [, int rownumber]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

odbc_fetch_row (PHP 3>= 3.0.6, PHP 4)

Busca un registro

int **odbc_fetch_row** (int result_id [, int row_number]) \linebreak

Si **odbc_fetch_row()** fue succesful (there was a row), `TRUE` is returned. If there are no more rows, `FALSE` is returned.

odbc_fetch_row() busca un registro de datos que fue devuelta por odbc_do() / odbc_exec(). Despues de que **odbc_fetch_row()** sea llamado, se puede acceder a los campos de este registro con odbc_result().

Si no se especifica *`row_number`*, **odbc_fetch_row()** intentara buscar el siguiente registro en los resultados. Lamar a **odbc_fetch_row()** con o sin *`row_number`* puede ser mezclado.

Para pasar a traves del resultado mas de una vez, se puede llamar a **odbc_fetch_row()** con *`row_number`* 1, y despues continuar haciendo **odbc_fetch_row()** sin *`row_number`* para revisar el resultado. Si un driver no admitiese busquedas de registros por numero, el parametro *`row_number`* seria ignorado.

odbc_field_len (PHP 3>= 3.0.6, PHP 4)

Da la longitud de un campo

int **odbc_field_len** (int result_id, int field_number) \linebreak

odbc_field_len() devolvera la longitud de un campo referenciado por numero en un identificador ODBC La numeracion de campos comienza en 1.

odbc_field_name (PHP 3>= 3.0.6, PHP 4)

Devuelve el nombre de campo

string **odbc_fieldname** (int result_id, int field_number) \linebreak

odbc_field_name() devolvera el nombre del campo almacenado en el numero de campo elegido dentro del identificador ODBC. La numeracion de campos comienza en 1. En caso de error devolveria `FALSE`.

odbc_field_num (PHP 3>= 3.0.6, PHP 4)

Devuelve el numero de campo

int **odbc_field_num** (int result_id, string field_name) \linebreak

odbc_field_num() devolvera el numero de campo que corresponda con el campo llamado en el identificador ODBC. La numeracion de campos comienza en 1. En caso de error devolveria `FALSE`.

odbc_field_precision (PHP 4)

Synonym for odbc_field_len()

string **odbc_field_precision** (resource result_id, int field_number) \linebreak

odbc_field_precision() will return the precision of the field referecend by number in the given ODBC result identifier.

See also: odbc_field_scale() to get the scale of a floating point number.

odbc_field_scale (PHP 4)

Get the scale of a field

string **odbc_field_scale** (resource result_id, int field_number) \linebreak

odbc_field_precision() will return the scale of the field referecend by number in the given ODBC result identifier.

odbc_field_type (PHP 3>= 3.0.6, PHP 4)

Tipo de datos de un campo

string **odbc_field_type** (int result_id, int field_number) \linebreak

odbc_field_type() Devolvera el tipo SQL de un campo referenciado por numero en el identificador ODBC. identifier. La numeracion de campos comienza en 1.

odbc_foreignkeys (PHP 4)

Returns a list of foreign keys in the specified table or a list of foreign keys in other tables that refer to the primary key in the specified table

resource **odbc_foreignkeys** (resource connection_id, string pk_qualifier, string pk_owner, string pk_table, string fk_qualifier, string fk_owner, string fk_table) \linebreak

odbc_foreignkeys() retrieves information about foreign keys. Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- PKTABLE_QUALIFIER
- PKTABLE_OWNER
- PKTABLE_NAME
- PKCOLUMN_NAME
- FKTABLE_QUALIFIER
- FKTABLE_OWNER
- FKTABLE_NAME
- FKCOLUMN_NAME
- KEY_SEQ
- UPDATE_RULE
- DELETE_RULE
- FK_NAME
- PK_NAME

If *`pk_table`* contains a table name, **odbc_foreignkeys()** returns a result set containing the primary key of the specified table and all of the foreign keys that refer to it.

If *`fk_table`* contains a table name, **odbc_foreignkeys()** returns a result set containing all of the foreign keys in the specified table and the primary keys (in other tables) to which they refer.

If both *`pk_table`* and *`fk_table`* contain table names, **odbc_foreignkeys()** returns the foreign keys in the table specified in *`fk_table`* that refer to the primary key of the table specified in *`pk_table`*. This should be one key at most.

odbc_free_result (PHP 3>= 3.0.6, PHP 4)

recursos libres asociados con un resultado

int **odbc_free_result** (int result_id) \linebreak

Always returns `TRUE`.

odbc_free_result() solo necesita ser llamado en caso de preocupacion por demasiado uso de memoria cuando se ejecuta un script. Toda la memoria resultante quedara automaticamente liberada cuando el script finalice. Pero si es seguro que no se vaya a necesitar la informacion nada mas que en un script, se debera llamar a la funcion **odbc_free_result()**, y la memoria asociada con *`result_id`* sera liberada.

> **Nota:** Si la auto-entrega no esta activada la (ver odbc_autocommit()) y se ejecuta **odbc_free_result()** antes de la entrega, todo queda pendiente de las transaciones que esten en lista.

odbc_gettypeinfo (PHP 4)

Returns a result identifier containing information about data types supported by the data source.

int **odbc_gettypeinfo** (resource connection_id [, int data_type]) \linebreak

Retrieves information about data types supported by the data source. Returns an ODBC result identifier or `FALSE` on failure. The optional argument *`data_type`* can be used to restrict the information to a single data type.

The result set has the following columns:

- TYPE_NAME
- DATA_TYPE
- PRECISION
- LITERAL_PREFIX
- LITERAL_SUFFIX
- CREATE_PARAMS
- NULLABLE
- CASE_SENSITIVE
- SEARCHABLE
- UNSIGNED_ATTRIBUTE
- MONEY

- AUTO_INCREMENT
- LOCAL_TYPE_NAME
- MINIMUM_SCALE
- MAXIMUM_SCALE

The result set is ordered by DATA_TYPE and TYPE_NAME.

odbc_longreadlen (PHP 3>= 3.0.6, PHP 4)

manejo de LONGITUD de columnas

int **odbc_longreadlen** (int result_id, int length) \linebreak

(ODBC SQL tipos relaccionados: LONG, LONGVARBINARY) El numero de bytes devueltos para PHP es controlado por el parametro length. Si es asignado a 0, la longitud del campo es pasado al cliente.

Nota: El manejo de campos LONGVARBINARY tambien esta afectado por odbc_binmode()

odbc_next_result (PHP 4 >= 4.0.5)

Checks if multiple results are avaiable

bool **odbc_next_result** (resource result_id) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

odbc_num_fields (PHP 3>= 3.0.6, PHP 4)

numero de campos de un resultado

int **odbc_num_fields** (int result_id) \linebreak

odbc_num_fields() devolvera el numero de campos dentro de un ODBC. Esta funcion devolvera -1 en caso de error. El argumento es un identificador valido devuelto por odbc_exec().

odbc_num_rows (PHP 3>= 3.0.6, PHP 4)

Numero de campos en un resultado

int **odbc_num_rows** (int result_id) \linebreak

odbc_num_rows() devolvera el numero de registros de un ODBC. Esta funcion devolvera -1 en caso de error. Para declaraciones INSERT, UPDATE y DELETE **odbc_num_rows()** devolvera el numero de registros afectados. Para una clausula SELECT esta `puede` ser el numero de registros permitidos.

Nota: El uso de **odbc_num_rows()** para determinar el numero de registros permitidos despues de un SELECT devolvera -1.

odbc_pconnect (PHP 3>= 3.0.6, PHP 4)

Abre una conexion permanente de base de datos

int **odbc_pconnect** (string dsn, string user, string password [, int cursor_type]) \linebreak

Devuelve un identificador de conexion ODBC o 0 (`FALSE`) en caso de error. Esta funcion es odbc_connect(), excepto que la conexion no sea realmente cerrada cuando el script ha finalizado. Las respuestas futuras para una conexion con la misma combinacion *`dsn`*, *`user`*, *`password`* (via odbc_connect() y **odbc_pconnect()**) puede reusar la conexion permanente.

Nota: Las conexiones permanentes no tienen efecto si PHP es usado como programa CGI.

Para informacion acerca del paramentor opcional cursor_type ver la funcion odbc_connect(). Para mas informacion sobre conexiones permanentes, ir al apartado PHP FAQ.

odbc_prepare (PHP 3>= 3.0.6, PHP 4)

Prepara una declaracion para su ejecucion

int **odbc_prepare** (int connection_id, string query_string) \linebreak

Devuelve `FALSE` en caso de error.

Devuelve un identificador ODBC si el comando SQL esta preparado. El identificador resultante puede ser usado mas tarde para ejecutar la declaracion con odbc_execute().

odbc_primarykeys (PHP 4)

Returns a result identifier that can be used to fetch the column names that comprise the primary key for a table

resource **odbc_primarykeys** (resource connection_id, string qualifier, string owner, string table) \linebreak

Returns the column names that comprise the primary key for a table. Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- KEY_SEQ
- PK_NAME

odbc_procedurecolumns (PHP 4)

Retrieve information about parameters to procedures

resource **odbc_procedurecolumns** (resource connection_id [, string qualifier [, string owner [, string proc [, string column]]]]) \linebreak

Returns the list of input and output parameters, as well as the columns that make up the result set for the specified procedures. Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- COLUMN_NAME
- COLUMN_TYPE
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE

- RADIX
- NULLABLE
- REMARKS

The result set is ordered by PROCEDURE_QUALIFIER, PROCEDURE_OWNER, PROCEDURE_NAME and COLUMN_TYPE.

The *owner*, *proc* and *column* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_procedures (PHP 4)

Get the list of procedures stored in a specific data source. Returns a result identifier containing the information.

resource **odbc_procedures** (resource connection_id [, string qualifier [, string owner [, string name]]]) \linebreak

Lists all procedures in the requested range. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- NUM_INPUT_PARAMS
- NUM_OUTPUT_PARAMS
- NUM_RESULT_SETS
- REMARKS
- PROCEDURE_TYPE

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_result_all (PHP 3>= 3.0.6, PHP 4)

Print result as HTML table

int **odbc_result_all** (int result_id [, string format]) \linebreak

En caso de error, como resultado, devuelve FALSE.

odbc_result_all() Imprimira todos los registros de un identificador prducidoo por odbc_exec(). El resultado es impreso en una tabla formato HTML. Con el argumento de cadena opcional *`format`*, ademas, todas los formatos de tablas pueden ser realizadas.

odbc_result (PHP 3>= 3.0.6, PHP 4)

coge informacion de un campo

string **odbc_result** (int result_id, mixed field) \linebreak

Devuelve el contenido de un campo.

`field` puede ser cualquier contenido del campo que queramos; o puede ser una cadena que contenga el nombre del campo; Por ejemplo:

```
$item_3 = odbc_result($Query_ID, 3 );
$item_val = odbc_result($Query_ID, "val");
```

La primera sentencia **odbc_result()** devuelve el valor del tercer campo detro del registro actual de la cola resultante. La segunda funcion llama a **odbc_result()** y devuelve el valor de un campo cuyo nombre es "val" en el registro actual de la cola resultante. Ocurre un error si un numero de columna para un campo es menor que uno o excede el numero de campos en el registro actual. Similarmente, ocurre un error si un campo con un nombre que no sea uno de los nombres de campo de una talba o tablas que sea o sean encoladas.

Los indices de campo comienzan en 1. Recordando el metodo binario de campos con gran informacion, es devuleto con referencia a **odbc_binmode ()** y odbc_longreadlen().

odbc_rollback (PHP 3>= 3.0.6, PHP 4)

Volver a pasar una transacion

int **odbc_rollback** (int connection_id) \linebreak

Vuelve a pasar todas las declaraciones pendientes *`connection_id`*. Devuelve `TRUE` cuando el resultado es satisfactorio, `FALSE` cuando no lo es.

odbc_setoption (PHP 3>= 3.0.6, PHP 4)

Ajusta la configuracion de ODBC. Devuelve `FALSE` en caso de error, en otro caso `TRUE`.

int **odbc_setoption** (int id, int function, int option, int param) \linebreak

Esta funcion permite buscar las opciones ODBC para una conexion particular o consulta resultante. Esto esta escrito para trabajar sobre problemas en peculiaries drivers ODBC. Esta funcion Solo se deberia usar siendo un programador de ODBC y entendiendo los efectos que las opciones tendran. Debemos tener la certeza de que necesitamos una buena referencia de reference to explicar todas las diferentes opciones y valores que pueden ser usados. Las diferentes versiones de drivers soportan diferentes opciones.

Ya que los efectos pueden variar dependiendo del driver ODBC, deberiamos usar la function en scripts para ser hecho publico lo que permitira que sea fuertemente desalentado. Algunas opciones ODBC no estan permitidas para esta funcion porque debe ser configurada antes de que la conexion sea establecida o la consulta este preparada. Sin embargo, si un determinado trabajo hace la tarea de PHP, el jefe no contaria con nosotros para usar un producto comercial, esto es lo que realmente suele pasar.

`Id` es una coexion id o resultado id sobre la que cambiaremos la configuracion. Para SQLSetConnectOption(), esta es una conexion id. Para SQLSetStmtOption(), este es un resultado id.

`function` es la funcion ODBC a usar. El valor deberia ser 1 para SQLSetConnectOption() y 2 para SQLSetStmtOption().

Parmeter *`option`* es la opcion a configurar.

El parametro *`param`* es el valor para la escogida opcion *`option`*.

Ejemplo 1. Ejemplos ODBC Setoption

```
// 1. Option 102 of SQLSetConnectOption() is SQL_AUTOCOMMIT.
//    Value 1 of SQL_AUTOCOMMIT is SQL_AUTOCOMMIT_ON.
//    Este ejemplo tiene el mismo efecto que
//    odbc_autocommit($conn, true);

odbc_setoption ($conn, 1, 102, 1);

// 2. Option 0 of SQLSetStmtOption() is SQL_QUERY_TIMEOUT.
//    Este ejemplo asigna el tiempo de espera de la consulta a 30 segundos.

$result = odbc_prepare ($conn, $sql);
odbc_setoption ($result, 2, 0, 30);
odbc_execute ($result);
```

odbc_specialcolumns (PHP 4)

Returns either the optimal set of columns that uniquely identifies a row in the table or columns that are automatically updated when any value in the row is updated by a transaction

resource **odbc_specialcolumns** (resource connection_id, int type, string qualifier, string owner, string table, int scope, int nullable) \linebreak

When the type argument is SQL_BEST_ROWID, **odbc_specialcolumns()** returns the column or columns that uniquely identify each row in the table.

When the type argument is SQL_ROWVER, **odbc_specialcolumns()** returns the optimal column or set of columns that, by retrieving values from the column or columns, allows any row in the specified table to be uniquely identified.

Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- SCOPE
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- PSEUDO_COLUMN

The result set is ordered by SCOPE.

odbc_statistics (PHP 4)

Retrieve statistics about a table

resource **odbc_statistics** (resource connection_id, string qualifier, string owner, string table_name, int unique, int accuracy) \linebreak

Get statistics about a table and it’s indexes. Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- NON_UNIQUE
- INDEX_QUALIFIER
- INDEX_NAME
- TYPE
- SEQ_IN_INDEX
- COLUMN_NAME
- COLLATION

- CARDINALITY
- PAGES
- FILTER_CONDITION

The result set is ordered by NON_UNIQUE, TYPE, INDEX_QUALIFIER, INDEX_NAME and SEQ_IN_INDEX.

odbc_tableprivileges (PHP 4)

Lists tables and the privileges associated with each table

int **odbc_tableprivileges** (resource connection_id [, string qualifier [, string owner [, string name]]]) \linebreak

Lists tables in the requested range and the privileges associated with each table. Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *`owner`* and *`name`* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_tables (PHP 3>= 3.0.17, PHP 4)

Get the list of table names stored in a specific data source. Returns a result identifier containing the information.

int **odbc_tables** (resource connection_id [, string qualifier [, string owner [, string name [, string types]]]]) \linebreak

Lists all tables in the requested range. Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- TABLE_TYPE
- REMARKS

The result set is ordered by TABLE_TYPE, TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

To support enumeration of qualifiers, owners, and table types, the following special semantics for the *qualifier*, *owner*, *name*, and *table_type* are available:

- If *qualifier* is a single percent character (%) and *owner* and *name* are empty strings, then the result set contains a list of valid qualifiers for the data source. (All columns except the TABLE_QUALIFIER column contain NULLs.)
- If *owner* is a single percent character (%) and *qualifier* and *name* are empty strings, then the result set contains a list of valid owners for the data source. (All columns except the TABLE_OWNER column contain NULLs.)
- If *table_type* is a single percent character (%) and *qualifier*, *owner* and *name* are empty strings, then the result set contains a list of valid table types for the data source. (All columns except the TABLE_TYPE column contain NULLs.)

If *table_type* is not an empty string, it must contain a list of comma-separated values for the types of interest; each value may be enclosed in single quotes (') or unquoted. For example, "'TABLE','VIEW'" or "TABLE, VIEW". If the data source does not support a specified table type, **odbc_tables()** does not return any results for that type.

See also odbc_tableprivileges() to retrieve associated privileges.

LXX. Funciones de Oracle 8

Estas funciones permiten acceder a bases de datos Oracle8 y Oracle7. Estas usan la Oracle8 Call-Interface (OCI8). Necesitará las librerías clientes de Oracle8 para usar esta extensión.

Esta extensión es más flexible que las estándar de Oracle. Soporta el enlace de variables locales y globales de PHP con placeholders de Oracle, tiene soporte completo para LOB, FILE y ROWID y le permiten usar las variables definidas por el usuario.

OCIBindByName (PHP 3>= 3.0.4, PHP 4)

Enlaza una variable PHP a un Placeholder de Oracle

int **OCIBindByName** (int stmt, string ph_name, mixed & variable, int length [, int type]) \linebreak

OCIBindByName() enlaza la variable PHP *variable* a un placeholder de Oracle *ph_name*. Si esta será usada para entrada o salida se determinará en tiempo de ejecución, y sera resevado el espacio necesario de almacenamiento. El parámetro *length* establece el tamaño máximo del enlace. Si establece *length* a -1 **OCIBindByName()** usará el tamaño de la *variable* para establecer el tamaño máximo.

Si necesita enlazar tipos de datos abstractos (LOB/ROWID/BFILE) necesitará primero reservar la memoria con la función **OCINewDescriptor()**. *length* no se usa para tipos de datos abstractos y debería establecerse a -1. La variable *type* le informa a Oracle, que tipo de descriptor queremos usar. Los valores posibles son: OCI_B_FILE (Binary-File), OCI_B_CFILE (Character-File), OCI_B_CLOB (Character-LOB), OCI_B_BLOB (Binary-LOB) and OCI_B_ROWID (ROWID).

Ejemplo 1. OCIDefineByName

```
<?php
/* OCIBindByPos example thies@digicol.de (980221)

  inserts 3 resords into emp, and uses the ROWID for updating the
  records just after the insert.
*/

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"insert into emp (empno, ename) ".
                       "values (:empno,:ename) ".
                       "returning ROWID into :rid");

$data = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");

$rowid = OCINewDescriptor($conn,OCI_D_ROWID);

OCIBindByName($stmt,":empno",&$empno,32);
OCIBindByName($stmt,":ename",&$ename,32);
OCIBindByName($stmt,":rid",&$rowid,-1,OCI_B_ROWID);

$update = OCIParse($conn,"update emp set sal = :sal where ROWID = :rid");
OCIBindByName($update,":rid",&$rowid,-1,OCI_B_ROWID);
OCIBindByName($update,":sal",&$sal,32);

$sal = 10000;

while (list($empno,$ename) = each($data)) {
    OCIExecute($stmt);
    OCIExecute($update);
}
```

```
$rowid->free();

OCIFreeStatement($update);
OCIFreeStatement($stmt);

$stmt = OCIParse($conn,"select * from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
while (OCIFetchInto($stmt,&$arr,OCI_ASSOC)) {
    var_dump($arr);
}
OCIFreeStatement($stmt);

/* delete our "junk" from the emp table.... */
$stmt = OCIParse($conn,"delete from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
OCIFreeStatement($stmt);

OCILogoff($conn);
?>
```

OCICancel (PHP 3>= 3.0.8, PHP 4)

Cancel reading from cursor

int **OCICancel** (int stmt) \linebreak

If you do not want read more data from a cursor, then call **OCICancel()**.

OCICollAppend (PHP 4 >= 4.0.6)

Coming soon

string **OCICollAppend** (object collection, object object) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCICollAssign (PHP 4 >= 4.0.6)

Coming soon

string **OCICollAssign** (object collection, object object) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCICollAssignElem (PHP 4 >= 4.0.6)

Coming soon

string **OCICollAssignElem** (object collection, string ndx, string val) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCICollGetElem (PHP 4 >= 4.0.6)

Coming soon

string **OCICollGetElem** (object collection, string ndx) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCICollMax (PHP 4 >= 4.0.6)

Coming soon

string **OCICollMax** (object collection) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCICollSize (PHP 4 >= 4.0.6)

Coming soon

string **OCICollSize** (object collection) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCICollTrim (PHP 4 >= 4.0.6)

Coming soon

string **OCICollTrim** (object collection, int num) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCIColumnIsNULL (PHP 3>= 3.0.4, PHP 4)

comprueba si una una columna es `NULL`

int **OCIColumnIsNULL** (int stmt, mixed column) \linebreak

OCIColumnIsNULL() devuelve vedadero si la columna devuelta *`column`* en el resultset de la sentencia *`stmt`* es `NULL`. Puede usar el número de la columna (1-Based) o el nombre de la columa indicado por el parámetro *`col`*.

OCIColumnName (PHP 3>= 3.0.4, PHP 4)

Devuelve el nombre de una columna.

string **OCIColumnName** (int stmt, int col) \linebreak

OCIColumnName() Devuelve el nombre de la columna correspondiente al número de la columna (1-based) que es pasado.

Ejemplo 1. OCIColumnName

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vea también **OCINumCols()**, **OCIColumnType()**, y **OCIColumnSize()**.

OCIColumnPrecision (PHP 4)

Coming soon

int **OCIColumnPrecision** (int stmt, int col) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCIColumnScale (PHP 4)

Coming soon

int **OCIColumnScale** (int stmt, int col) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCIColumnSize (PHP 3>= 3.0.4, PHP 4)

devuelve el tamaño de la columna

int **OCIColumnSize** (int stmt, mixed column) \linebreak

OCIColumnSize() devuelve el tamaño de la columna indicada por Oracle Puede utilizar el número de la columna (1-Based) o el nombre indicado en el parámetro `col`.

Ejemplo 1. OCIColumnSize

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vea también **OCINumCols()**, **OCIColumnName()**, y **OCIColumnSize()**.

OCIColumnType (PHP 3>= 3.0.4, PHP 4)

Devuelve el tipo de dato de una columna.

mixed **OCIColumnType** (int stmt, int col) \linebreak

OCIColumnType() devuelve el tipo de dato de una columna correspondiente al número de la columna (1-based) que es pasado.

Ejemplo 1. OCIColumnType

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vea también **OCINumCols()**, **OCIColumnName()**, y **OCIColumnSize()**.

OCIColumnTypeRaw (PHP 4)

Coming soon

mixed **OCIColumnTypeRaw** (int stmt, int col) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCICommit (PHP 3>= 3.0.7, PHP 4)

Confirma transacciones pendientes

int **OCICommit** (int connection) \linebreak

OCICommit() confirma todas las sentencias pendientes para la conexión con Oracle *`connection`*.

OCIDefineByName (PHP 3>= 3.0.7, PHP 4)

Usa una variable de PHP para el define-step durante una sentencia SELECT

int **OCIDefineByName** (int stmt, string Column-Name, mixed & variable [, int type]) \linebreak

OCIDefineByName() busca el valor de las Columnas-SQL dentro de variables PHP definidas por el usuario. Cuidado que Oracle nombra todas las columnas en MAYUSCULAS, mientras que en su select puede usar también minúsculas write lower-case. **OCIDefineByName()** espera que *`Column-Name`* esté en mayúsculas. Si define una variable que no existe en la sentecia SELECT, no se producirá ningún error.

Si necesita definir un tipo de dato abstracto (LOB/ROWID/BFILE) tendrá que alojarlo primero usando la función **OCINewDescriptor()** function. Vea también la función **OCIBindByName()**.

Ejemplo 1. OCIDefineByName

```
<?php
/* OCIDefineByPos example thies@digicol.de (980219) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select empno, ename from emp");

/* la definición DEBE hacerse ANTES del ociexecute! */

OCIDefineByName($stmt,"EMPNO",&$empno);
OCIDefineByName($stmt,"ENAME",&$ename);

OCIExecute($stmt);

while (OCIFetch($stmt)) {
    echo "empno:".$empno."\n";
    echo "ename:".$ename."\n";
}

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIError (PHP 3>= 3.0.7, PHP 4)

Devuelve el último error de stmt|conn|global. Si no ocurre ningún error devuelve falso.

array **OCIError** ([int stmt|conn|global]) \linebreak

OCIError() devuelve el último error encontrado. Si el parámetro opcional *stmt|conn|global* no es usado, es devuelto el último error encontrado. Si no se encuentra ningún error, **OCIError()** devuelve falso. **OCIError()** devuelve el error como un array asociativo. En este array, *code* consiste en el código de error de Oracle y *message* en la cadena de descripción del error.

OCIExecute (PHP 3>= 3.0.4, PHP 4)

Ejecuta una sentencia

int **OCIExecute** (int statement [, int mode]) \linebreak

OCIExecute() ejecuta una sentencia previamente analizada. (see **OCIParse()**). El parámetro opcional *mode* le permite especificar el modo de ejecución (default is OCI_COMMIT_ON_SUCCESS). Si no desea que las sentencias se confirmen automaticamente, especifique OCI_DEFAULT como su modo.

OCIFetch (PHP 3>= 3.0.4, PHP 4)

Busca la siguiente fila en el result-buffer

int **OCIFetch** (int statement) \linebreak

OCIFetch() Busca la siguiente fila (para sentencias SELECT) dentro del result-buffer interno.

OCIFetchInto (PHP 3>= 3.0.4, PHP 4)

Busca la siguiente fila dentro del result-array

int **OCIFetchInto** (int stmt, array & result [, int mode]) \linebreak

OCIFetchInto() busca la siguiente fila (for SELECT statements) dentro del array *result*. **OCIFetchInto()** sobreescribirá el contenido previo de *result*. Por defecto *result* contendrá un array basado en todas las columnas que no son NULL.

El parámetro *mode* le permite cambiar el comportamineto por defecto. Puede especificar más de una flag simplemente añadiendolas (ej. OCI_ASSOC+OCI_RETURN_NULLS). Las flags son:

OCI_ASSOC Devuelve un array asociativo.
OCI_NUM Devuelve un array numerado empezando en 1. (POR DEFECTO)

`OCI_RETURN_NULLS` Devuelve columnas vacias.
`OCI_RETURN_LOBS` Devuelve el valor de un LOB en vez de el descriptor.

OCIFetchStatement (PHP 3>= 3.0.8, PHP 4)

Busca todas la filas de un resultset dentro de un array.

int **OCIFetchStatement** (int stmt, array & variable) \linebreak

OCIFetchStatement() busca todas las filas de un resultset dentro de un array definido por el usuario.
OCIFetchStatement() devuelve el numero de filas buscadas.

Ejemplo 1. OCIFetchStatement

```
<?php
/* OCIFetchStatement example mbritton@verinet.com (990624) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select * from emp");

OCIExecute($stmt);

$nrows = OCIFetchStatement($stmt,$results);
if ( $nrows > 0 ) {
   print "<TABLE BORDER=\"1\">\n";
   print "<TR>\n";
   while ( list( $key, $val ) = each( $results ) ) {
      print "<TH>$key</TH>\n";
   }
   print "</TR>\n";

   for ( $i = 0; $i < $nrows; $i++ ) {
      reset($results);
      print "<TR>\n";
      while ( $column = each($results) ) {
         $data = $column['value'];
         print "<TD>$data[$i]</TD>\n";
      }
      print "</TR>\n";
   }
   print "</TABLE>\n";
} else {
   echo "No data found<BR>\n";
}
print "$nrows Records Selected<BR>\n";

OCIFreeStatement($stmt);
OCILogoff($conn);
```

```
?>
```

OCIFreeCollection (PHP 4 >= 4.1.0)

Coming soon

string **OCIFreeCollection** (object lob) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCIFreeCursor (PHP 3>= 3.0.8, PHP 4)

Libera todos los recursos asociados con cursor.

int **OCIFreeCursor** (int stmt) \linebreak

OCIFreeCursor() devuelve cierto si la operacion se lleva a cabo, o falso en caso contrario.

OCIFreeDesc (PHP 4)

Deletes a large object descriptor

int **OCIFreeDesc** (object lob) \linebreak

OCIFreeDesc() returns `TRUE` if successful, or `FALSE` if unsuccessful.

OCIFreeStatement (PHP 3>= 3.0.5, PHP 4)

Libera todos los recursos asociados con una sentencia.

int **OCIFreeStatement** (int stmt) \linebreak

OCIFreeStatement() devuelve cierto si la operacion se lleva a cabo, o falso en caso contrario.

OCIInternalDebug (PHP 3>= 3.0.4, PHP 4)

Habilita o deshabilita la salida del depurador interno. Por defecto este está deshabiltado

void **OCIInternalDebug** (int onoff) \linebreak

OCIInternalDebug() habilita la salida del depurador interno. Asigne 0 a *onoff* para deshabilitar la salida y 1 para habilitarla.

OCILoadLob (PHP 4)

Coming soon

string **OCILoadLob** (object lob) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCILogOff (PHP 3>= 3.0.4, PHP 4)

Termina la conexion con Oracle

int **OCILogOff** (int connection) \linebreak

OCILogOff() cierra una conexión con Oracle.

OCILogon (PHP 3>= 3.0.4, PHP 4)

Establece la conexión con Oracle

int **OCILogon** (string username, string password [, string db]) \linebreak

OCILogon() devuelve el identificador de conexión necesario en la mayoria de las funciones OCI. El tercer parámetro, que es opcional, puede contener el nombre de la instancia a Oracle o el nombre dado en el fichero tnsnames.ora de la base de datos a la que nos queremos conectar. Si este parámetro no se especifica, PHP usa la variable de entorno ORACLE_SID (Oracle instance) o TWO_TASK (tnsnames.ora) para determinar la base de datos con la que queremos conectar.

Las conexiones son compartidas a nivel de página cuando usemos **OCILogon()**. Lo cual significa que los "commits" y "rollbacks" son aplicadas a todas las transacciones abiertas en la página, incluso si usted ha creado conexiones múltiples.

Este ejemplo demuestra como son compartidas las conexiones.

Ejemplo 1. OCILogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocilogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
            values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." commited\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
```

```
  echo $conn."----selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."----done\n\n";
}

create_table($c1);
insert_data($c1);   // Insert a row using c1
insert_data($c2);   // Insert a row using c2

select_data($c1);   // Results of both inserts are returned
select_data($c2);

rollback($c1);      // Rollback using c1

select_data($c1);   // Both inserts have been rolled back
select_data($c2);

insert_data($c2);   // Insert a row using c2
commit($c2);        // commit using c2

select_data($c1);   // result of c2 insert is returned

delete_data($c1);   // delete all rows in table using c1
select_data($c1);   // no rows returned
select_data($c2);   // no rows returned
commit($c1);        // commit using c1

select_data($c1);   // no rows returned
select_data($c2);   // no rows returned

drop_table($c1);
print "</PRE></HTML>";
?>
```

See also **OCIPLogon()** and **OCINLogon()**.

OCINewCollection (PHP 4 >= 4.0.6)

Coming soon

string **OCINewCollection** (int conn, string tdo [, string shema]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCINewCursor (PHP 3>= 3.0.8, PHP 4)

devuelve un cursor nuevo (Statement-Handle) - use esto para enlazar ref-cursors!

int **OCINewCursor** (int conn) \linebreak

OCINewCursor() allocates a new statement handle on the specified connection.

Ejemplo 1. Usando un REF CURSOR de un procedimiento almacenado

```
<?php
// suppose your stored procedure info.output returns a ref cursor in :data

$conn = OCILogon("scott","tiger");
$curs = OCINewCursor($conn);
$stmt = OCIParse($conn,"begin info.output(:data); end;");

ocibindbyname($stmt,"data",&$curs,-1,OCI_B_CURSOR);
ociexecute($stmt);
ociexecute($curs);

while (OCIFetchInto($curs,&$data)) {
    var_dump($data);
}

OCIFreeCursor($stmt);
OCIFreeStatement($curs);
OCILogoff($conn);
?>
```

Ejemplo 2. Usando un REF CURSOR en una sentencia select

```
<?php
print "<HTML><BODY>";
$conn = OCILogon("scott","tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = OCIParse($conn,"select deptno,dname,$count_cursor");

ociexecute($stmt);
print "<TABLE BORDER=\"1\">";
print "<TR>";
print "<TH>DEPT NAME</TH>";
print "<TH>DEPT #</TH>";
```

```
print "<TH># EMPLOYEES</TH>";
print "</TR>";

while (OCIFetchInto($stmt,&$data,OCI_ASSOC)) {
    print "<TR>";
    $dname  = $data["DNAME"];
    $deptno = $data["DEPTNO"];
    print "<TD>$dname</TD>";
    print "<TD>$deptno</TD>";
    ociexecute($data[ "EMPCNT" ]);
    while (OCIFetchInto($data[ "EMPCNT" ],&$subdata,OCI_ASSOC)) {
        $num_emps = $subdata["NUM_EMPS"];
        print  "<TD>$num_emps</TD>";
    }
    print "</TR>";
}
print "</TABLE>";
print "</BODY></HTML>";
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCINewDescriptor (PHP 3>= 3.0.7, PHP 4)

Inicializa un nuevo descriptor vacío LOB/FILE (LOB por defecto)

string **OCINewDescriptor** (int connection [, int type]) \linebreak

OCINewDescriptor() Reserva espacio para mantener descriptores o localizadores LOB. Los valores válidos para el tipo *type* son OCI_D_FILE, OCI_D_LOB, OCI_D_ROWID. Para descriptores LOB, los métodos load, save, y savefile están asociados con el descriptor, para BFILE sólo el método load existe. Vea el segundo ejemplo.

Ejemplo 1. OCINewDescriptor

```
<?php
    /* This script is designed to be called from a HTML form.
     * It expects $user, $password, $table, $where, and $commitsize
     * to be passed in from the form.  The script then deletes
     * the selected rows using the ROWID and commits after each
     * set of $commitsize rows. (Use with care, there is no rollback)
     */
    $conn = OCILogon($user, $password);
    $stmt = OCIParse($conn,"select rowid from $table $where");
    $rowid = OCINewDescriptor($conn,OCI_D_ROWID);
    OCIDefineByName($stmt,"ROWID",&$rowid);
    OCIExecute($stmt);
```

```
    while ( OCIFetch($stmt) ) {
       $nrows = OCIRowCount($stmt);
       $delete = OCIParse($conn,"delete from $table where ROWID = :rid");
       OCIBindByName($delete,":rid",&$rowid,-1,OCI_B_ROWID);
       OCIExecute($delete);
       print "$nrows\n";
       if ( ($nrows % $commitsize) == 0 ) {
           OCICommit($conn);
       }
    }
    $nrows = OCIRowCount($stmt);
    print "$nrows deleted...\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
?>

<?php
    /* This script demonstrates file upload to LOB columns
     * The formfield used for this example looks like this
     * <form action="upload.php3" method="post" enctype="multipart/form-data">
     * <input type="file" name="lob_upload">
     * ...
     */
  if(!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php3" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload"><br>
<input type="submit" value="Upload"> - <input type="reset">
</form>
<?php
  } else {
     // $lob_upload contains the temporary filename of the uploaded file
     $conn = OCILogon($user, $password);
     $lob = OCINewDescriptor($conn, OCI_D_LOB);
     $stmt = OCIParse($conn,"insert into $table (id, the_blob)
               values(my_seq.NEXTVAL, EMPTY_BLOB()) returning the_blob into :the_blob");
     OCIBindByName($stmt, ':the_blob', &$lob, -1, OCI_B_BLOB);
     OCIExecute($stmt);
     if($lob->savefile($lob_upload)){
        OCICommit($conn);
        echo "Blob successfully uploaded\n";
     }else{
        echo "Couldn't upload Blob\n";
     }
     OCIFreeDescriptor($lob);
     OCIFreeStatement($stmt);
     OCILogoff($conn);
  }
?>
```

OCINLogon (PHP 3>= 3.0.8, PHP 4)

Conecta con una base de datos Oracle usando una nueva conexión. Devuelve una nueva sesión.

int **OCINLogon** (string username, string password [, string db]) \linebreak

OCINLogon() crea una nueva conexión con una base de datos Oracle 8. El tercer parámetro, que es opcional, puede contener el nombre de la instancia a Oracle o el nombre dado en el fichero tnsnames.ora de la base de datos a la que nos queremos conectar. Si este parámetro no se especifica, PHP usa la variable de entorno ORACLE_SID (Oracle instance) o TWO_TASK (tnsnames.ora) para determinar la base de datos con la que queremos conectar.

OCINLogon() fuerza una nueva conexión. Se debe usar si necesita aislar un conjunto de transacciones. Por defecto, las conexiones son compartidas a nivel de página si usa **OCILogon()** o a nivel del proceso del servidor web si usa **OCIPLogon()**. Si posee múltiples conexiones abiertas usando **OCINLogon()**, todos los "commits" y "rollbacks" se aplican sólo a la conexion especificada.

Este ejemplo demuestra como las conexiones están separadas.

Ejemplo 1. OCINLogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocinlogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test
varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
            values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
```

```
}
function commit($conn)
{ ocicommit($conn);
  echo $conn." commited\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."----selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."----done\n\n";
}

create_table($c1);
insert_data($c1);

select_data($c1);
select_data($c2);

rollback($c1);

select_data($c1);
select_data($c2);

insert_data($c2);
commit($c2);

select_data($c1);

delete_data($c1);
select_data($c1);
select_data($c2);
commit($c1);

select_data($c1);
select_data($c2);

drop_table($c1);
print "</PRE></HTML>";
?>
```

See also **OCILogon()** and **OCIPLogon()**.

OCINumCols (PHP 3>= 3.0.4, PHP 4)

Devuelve el número de columnas resultantes en una sentencia

int **OCINumCols** (int stmt) \linebreak

OCINumCols() devuelve el número de columnas en una sentencia

Ejemplo 1. OCINumCols

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
        print "\n";
        $ncols = OCINumCols($stmt);
        for ( $i = 1; $i <= $ncols; $i++ ) {
            $column_name  = OCIColumnName($stmt,$i);
            $column_value = OCIResult($stmt,$i);
            print $column_name . ': ' . $column_value . "\n";
        }
        print "\n";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

OCIParse (PHP 3>= 3.0.4, PHP 4)

Analiza una consulta y devuelve una sentencia

int **OCIParse** (int conn, strint query) \linebreak

OCIParse() analiza la *query* usando *conn*. Devuelve el identificador de la sentencia si la consulta es válida, y falso si no lo es. La *query* puede ser cualquier sentencia SQL válida.

OCIPLogon (PHP 3>= 3.0.8, PHP 4)

Conecta con una base de datos Oracle usando una conexión persistente. Devuelve una nueva sesión.

int **OCIPLogon** (string username, string password [, string db]) \linebreak

OCIPLogon() crea una conexión persistente con una base de datos Oracle 8. El tercer parámetro, que es opcional, puede contener el nombre de la instancia a Oracle o el nombre dado en el fichero tnsnames.ora de la base de datos a la que nos queremos conectar. Si este parámetro no se especifica, PHP usa la variable de entorno ORACLE_SID (Oracle instance) o TWO_TASK (tnsnames.ora) para determinar la base de datos con la que queremos conectar.

Vea también **OCILogon()** y **OCINLogon()**.

OCIResult (PHP 3>= 3.0.4, PHP 4)

Devuelve el valor de una columna en la fila buscada

mixed **OCIResult** (int statement, mixed column) \linebreak

OCIResult() devuelve el valor de la columna *`column`* de la fila actual (vea **OCIFetch()**).**OCIResult()** devolverá todo como una cadena excepto para los tipo de datos abstractos (ROWIDs, LOBs and FILEs).

OCIRollback (PHP 3>= 3.0.7, PHP 4)

Restablece todas las transaciones sin confirmar

int **OCIRollback** (int connection) \linebreak

OCIRollback() restablece todas las transacciones sin confirmar para la conexión Oracle *`connection`*.

OCIRowCount (PHP 3>= 3.0.7, PHP 4)

Obtiene el número de filas afectadas

int **OCIRowCount** (int statement) \linebreak

OCIRowCount() devuelve el número de filas afectadas, por ej. en sentencias de actualización. !Esta función no indicará el número de de filas que devuelve una sentencia SELECT!

Ejemplo 1. OCIRowCount

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $stmt = OCIParse($conn,"create table emp2 as select * from emp");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows inserted.<BR>";
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"delete from emp2");
    OCIExecute($stmt);
```

```
    print OCIRowCount($stmt) . " rows deleted.<BR>";
    OCICommit($conn);
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"drop table emp2");
    OCIExecute($stmt);
    OCIFreeStatement($stmt);
    OCILogOff($conn);
    print "</PRE></HTML>";
?>
```

OCISaveLob (PHP 4)

Coming soon

string **OCISaveLob** (object lob) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCISaveLobFile (PHP 4)

Coming soon

string **OCISaveLobFile** (object lob) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

OCIServerVersion (PHP 3>= 3.0.4, PHP 4)

Devuelve una cadena conteniendo información a cerca de la version del servidor.

string **OCIServerVersion** (int conn) \linebreak

Ejemplo 1. OCIServerVersion

```
<?php
   $conn = OCILogon("scott","tiger");
   print "Server Version: " . OCIServerVersion($conn);
   OCILogOff($conn);
?>
```

OCISetPrefetch (PHP 3>= 3.0.12, PHP 4)

Sets number of rows to be prefetched

int **OCISetPrefetch** (int stmt, int rows) \linebreak

Sets the number of top level rows to be prefetched. The default value is 1 row.

OCIStatementType (PHP 3>= 3.0.5, PHP 4)

Devuelve el tipo de una sentencia OCI.

string **OCIStatementType** (int stmt) \linebreak

OCIStatementType() devuelve uno de los siguiente valores:

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"
6. "DROP"
7. "ALTER"
8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Ejemplo 1. Code examples

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $sql  = "delete from emp where deptno = 10";

    $stmt = OCIParse($conn,$sql);
    if ( OCIStatementType($stmt) == "DELETE" ) {
        die "You are not allowed to delete from this table<BR>";
    }

    OCILogoff($conn);
    print "</PRE></HTML>";
?>
```

OCIWriteLobToFile (PHP 4)

Coming soon

void **OCIWriteLobToFile** (object lob [, string filename [, int start [, int lenght]]]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

LXXI. OpenSSL functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Introduction

This module uses the functions of OpenSSL (http://www.openssl.org/) for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data. PHP-4.0.4pl1 requires OpenSSL >= 0.9.6, but PHP-4.0.5 and greater with also work with OpenSSL >= 0.9.5.

Nota: Please keep in mind that this extension is still considered experimental!

OpenSSL offers many features that this module currently doesn't support. Some of these may be added in the future.

Key/Certificate parameters

Quite a few of the openssl functions require a key or a certificate parameter. PHP 4.0.5 and earlier have to use a key or certificate resource returned by one of the openssl_get_xxx functions. Later versions may use one of the following methods:

- Certificates
  1. An X.509 resource returned from openssl_x509_read
  2. A string having the format `file://path/to/cert.pem`; the named file must contain a PEM encoded certificate
  3. A string containing the content of a certificate, PEM encoded
- Public/Private Keys
  1. A key resource returned from openssl_get_publickey() or openssl_get_privatekey()
  2. For public keys only: an X.509 resource
  3. A string having the format `file://path/to/file.pem` - the named file must contain a PEM

encoded certificate/private key (it may contain both)

4. A string containing the content of a certificate/key, PEM encoded
5. For private keys, you may also use the syntax *array($key, $passphrase)* where $key represents a key specified using the file:// or textual content notation above, and $passphrase represents a string containing the passphrase for that private key

Certificate Verification

When calling a function that will verify a signature/certificate, the *cainfo* parameter is an array containing file and directory names the specify the locations of trusted CA files. If a directory is specified, then it must be a correctly formed hashed directory as the **openssl** command would use.

PKCS7 Flags/Constants

The S/MIME functions make use of flags which are specified using a bitfield which can include one or more of the following values:

Tabla 1. PKCS7 CONSTANTS

Constant	Description
PKCS7_TEXT	adds text/plain content type headers to encrypted/signed message. If decrypting or verifying, it strips those headers from the output - if the decrypted or verified message is not of MIME type text/plain then an error will occur.
PKCS7_BINARY	normally the input message is converted to "canonical" format which is effectively using CR and LF as end of line: as required by the S/MIME specification. When this options is present, no translation occurs. This is useful when handling binary data which may not be in MIME format.
PKCS7_NOINTERN	when verifying a message, certificates (if any) included in the message are normally searched for the signing certificate. With this option only the certificates specified in the `extracerts` parameter of openssl_pkcs7_verify() are used. The supplied certificates can still be used as untrusted CAs however.
PKCS7_NOVERIFY	do not verify the signers certificate of a signed message.

Constant	Description
PKCS7_NOCHAIN	do not chain verification of signers certificates: that is don't use the certificates in the signed message as untrusted CAs.
PKCS7_NOCERTS	when signing a message the signer's certificate is normally included - with this option it is excluded. This will reduce the size of the signed message but the verifier must have a copy of the signers certificate available locally (passed using the `extracerts` to openssl_pkcs7_verify() for example.
PKCS7_NOATTR	normally when a message is signed, a set of attributes are included which include the signing time and the supported symmetric algorithms. With this option they are not included.
PKCS7_DETACHED	When signing a message, use cleartext signing with the MIME type multipart/signed. This is the default if the `flags` parameter to openssl_pkcs7_sign() if you do not specify any flags. If you turn this option off, the message will be signed using opaque signing, which is more resistant to translation by mail relays but cannot be read by mail agents that do not support S/MIME.
PKCS7_NOSIGS	Don't try and verify the signatures on a message

Nota: These constants were added in 4.0.6.

openssl_csr_export_to_file (PHP 4 >= 4.2.0)

Exports a CSR to file or a var

bool **openssl_csr_export_to_file** (resource csr, string outfilename [, bool notext]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_csr_export (PHP 4 >= 4.2.0)

Exports a CSR to file or a var

bool **openssl_csr_export** (resource csr, string out [, bool notext]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_csr_new (PHP 4 >= 4.2.0)

Generates a privkey and CSR

bool **openssl_csr_new** (array dn, resource privkey [, array extraattribs [, array configargs]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_csr_sign (PHP 4 >= 4.2.0)

Signs a cert with another CERT

resource **openssl_csr_sign** (mixed csr, mixed x509, mixed priv_key, long days) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_error_string (PHP 4 >= 4.0.6)

Return openSSL error message

mixed **openssl_error_string** (void) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Returns an error message string, or `FALSE` if there are no more error messages to return.

openssl_error_string() returns the last error from the openSSL library. Error messages are stacked, so this function should be called multiple times to collect all of the information.

The parameters/return type of this function may change before it appears in a release version of PHP

Ejemplo 1. openssl_error_string() example

```
// lets assume you just called an openssl function that failed
while($msg = openssl_error_string())
    echo $msg . "<br />\n";
```

Nota: This function was added in 4.0.6.

openssl_free_key (PHP 4 >= 4.0.4)

Free key resource

void **openssl_free_key** (resource key_identifier) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

openssl_free_key() frees the key associated with the specified `key_identifier` from memory.

openssl_get_privatekey (PHP 4 >= 4.0.4)

Prepare a PEM formatted private key for use

resource **openssl_get_privatekey** (mixed key [, string passphrase]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Returns a positive key resource identifier on success, or `FALSE` on error.

openssl_get_privatekey() parses the PEM formatted private key specified by `key` and prepares it for use by other functions. The optional parameter `passphrase` must be used if the specified key is encrypted (protected by a passphrase).

openssl_get_publickey (PHP 4 >= 4.0.4)

Extract public key from certificate and prepare it for use

resource **openssl_get_publickey** (mixed certificate) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Returns a positive key resource identifier on success, or `FALSE` on error.

openssl_get_publickey() extracts the public key from an X.509 certificate specified by `certificate` and prepares it for use by other functions.

openssl_open (PHP 4 >= 4.0.4)

Open sealed data

bool **openssl_open** (string sealed_data, string open_data, string env_key, mixed priv_key_id) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo. If successful the opened data is returned in *`open_data`*.

openssl_open() opens (decrypts) *`sealed_data`* using the private key associated with the key identifier *`priv_key_id`* and the envelope key *`env_key`*, and fills *`open_data`* with the decrypted data. The envelope key is generated when the data are sealed and can only be used by one specific private key. See openssl_seal() for more information.

Ejemplo 1. openssl_open() example

```
// $sealed and $env_key are assumed to contain the sealed data
// and our envelope key, both given to us by the sealer.

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// decrypt the data and store it in $open
if (openssl_open($sealed, $open, $env_key, $pkeyid))
    echo "here is the opened data: ", $open;
else
    echo "failed to open data";

// free the private key from memory
openssl_free_key($pkeyid);
```

See also openssl_seal().

openssl_pkcs7_decrypt (PHP 4 >= 4.0.6)

Decrypts an S/MIME encrypted message

bool **openssl_pkcs7_decrypt** (string infilename, string outfilename, mixed recipcert, mixed recipkey) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Decrypts the S/MIME encrypted message contained in the file specified by *infilename* using the certificate and it's associated private key specified by *recipcert* and *recipkey*.

The decrypted message is output to the file specified by *outfilename*

Ejemplo 1. openssl_pkcs7_decrypt() example

```
// $cert and $key are assumed to contain your personal certificate and private
// key pair, and that you are the recipient of an S/MIME message
$infilename = "encrypted.msg";  // this file holds your encrypted message
$outfilename = "decrypted.msg"; // make sure you can write to this file

if (openssl_pkcs7_decrypt($infilename, $outfilename, $cert, $key))
    echo "decrypted!";
else
    echo "failed to decrypt!";
```

Nota: This function was added in 4.0.6.

openssl_pkcs7_encrypt (PHP 4 >= 4.0.6)

Encrypt an S/MIME message

bool **openssl_pkcs7_encrypt** (string infile, string outfile, mixed recipcerts, array headers [, long flags]) \line-break

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

openssl_pkcs7_encrypt() takes the contents of the file named *infile* and encrypts them using an RC2 40-bit cipher so that they can only be read by the intended recipients specified by *recipcerts*, which is either a lone X.509 certificate, or an array of X.509 certificates. *headers* is an array of headers that will be prepended to the data after it has been encrypted. *flags* can be used to specify options that affect the encoding process - see PKCS7 constants. *headers* can be either an associative array keyed by header name, or an indexed array, where each element contains a single header line.

Ejemplo 1. openssl_pkcs7_encrypt() example

```
// the message you want to encrypt and send to your secret agent
// in the field, known as nighthawk.  You have his certificate
// in the file nighthawk.pem
$data = <<<EOD
Nighthawk,

Top secret, for your eyes only!

The enemy is closing in! Meet me at the cafe at 8.30am
to collect your forged passport!

HQ
EOD;

// load key
$key = implode("", file("nighthawk.pem"));

// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);

// encrypt it
if (openssl_pkcs7_encrypt("msg.txt", "enc.txt", $key,
    array("To" => "nighthawk@example.com", // keyed
syntax
          "From: HQ <hq@example.com>", // indexed syntax
          "Subject" => "Eyes only")))
{
    // message encrypted - send it!
    exec(ini_get("sendmail_path") . " < enc.txt");
}
```

openssl_pkcs7_sign (PHP 4 >= 4.0.6)

sign an S/MIME message

bool **openssl_pkcs7_sign** (string infilename, string outfilename, mixed signcert, mixed privkey, array headers [, long flags [, string extracertsfilename]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

openssl_pkcs7_sign() takes the contents of the file named *infilename* and signs them using the certificate and it's matching private key specified by *signcert* and *privkey* parameters.

headers is an array of headers that will be prepended to the data after it has been signed (see openssl_pkcs7_encrypt() for more information about the format of this parameter.

flags can be used to alter the output - see PKCS7 constants - if not specified, it defaults to PKCS7_DETACHED.

extracerts specifies the name of a file containing a bunch of extra certificates to include in the signature which can for example be used to help the recipient to verify the certificate that you used.

Ejemplo 1. openssl_pkcs7_sign() example

```
// the message you want to sign so that recipient can be sure it was you that
// sent it
$data = <<<EOD

You have my authorization to spend $10,000 on dinner expenses.

The CEO
EOD;
// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// encrypt it
if (openssl_pkcs7_sign("msg.txt", "signed.txt", "mycert.pem",
    array("mycert.pem", "mypassphrase"),
    array("To" => "joes@sales.com", // keyed syntax
          "From: HQ <ceo@sales.com>", // indexed syntax
          "Subject" => "Eyes only"))
{
    // message signed - send it!
    exec(ini_get("sendmail_path") . " < signed.txt");
}
```

Nota: This function was added in 4.0.6.

openssl_pkcs7_verify (PHP 4 >= 4.0.6)

Verifies the signature of an S/MIME signed message

bool **openssl_pkcs7_verify** (string filename, int flags [, string outfilename [, array cainfo [, string extracerts]]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

openssl_pkcs7_verify() reads the S/MIME message contained in the filename specified by *filename* and examines the digital signature. It returns TRUE if the signature is verified, FALSE if it is not correct (the message has been tampered with, or the signing certificate is invalid), or -1 on error.

flags can be used to affect how the signature is verified - see PKCS7 constants for more information.

If the *outfilename* is specified, it should be a string holding the name of a file into which the certificates of the persons that signed the messages will be stored in PEM format.

If the *cainfo* is specified, it should hold information about the trusted CA certificates to use in the verification process - see certificate verification for more information about this parameter.

If the *extracerts* is specified, it is the filename of a file containing a bunch of certificates to use as untrusted CAs.

Nota: This function was added in 4.0.6.

openssl_pkey_export_to_file (PHP 4 >= 4.2.0)

Gets an exportable representation of a key into a file

bool **openssl_pkey_export_to_file** (mixed key, string outfilename [, string passphrase [, array config_args]]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_pkey_export (PHP 4 >= 4.2.0)

Gets an exportable representation of a key into a string or file

bool **openssl_pkey_export** (mixed key, mixed out [, string passphrase [, array config_args]]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_pkey_new (PHP 4 >= 4.2.0)

Generates a new private key

resource **openssl_pkey_new** ([array configargs]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_private_decrypt (PHP 4 >= 4.0.6)

Decrypts data with private key

bool **openssl_private_decrypt** (string data, string crypted, mixed key [, int padding]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_private_encrypt (PHP 4 >= 4.0.6)

Encrypts data with private key

bool **openssl_private_encrypt** (string data, string crypted, mixed key [, int padding]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_public_decrypt (PHP 4 >= 4.0.6)

Decrypts data with public key

bool **openssl_public_decrypt** (string data, string crypted, resource key [, int padding]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_public_encrypt (PHP 4 >= 4.0.6)

Encrypts data with public key

bool **openssl_public_encrypt** (string data, string crypted, mixed key [, int padding]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_seal (PHP 4 >= 4.0.4)

Seal (encrypt) data

int **openssl_seal** (string data, string sealed_data, array env_keys, array pub_key_ids) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Returns the length of the sealed data on success, or `FALSE` on error. If successful the sealed data is returned in *`sealed_data`*, and the envelope keys in *`env_keys`*.

openssl_seal() seals (encrypts) *`data`* by using RC4 with a randomly generated secret key. The key is encrypted with each of the public keys associated with the identifiers in *`pub_key_ids`* and each encrypted key is returned in *`env_keys`*. This means that one can send sealed data to multiple recipients (provided one has obtained their public keys). Each recipient must receive both the sealed data and the envelope key that was encrypted with the recipient's public key.

Ejemplo 1. openssl_seal() example

```
// $data is assumed to contain the data to be sealed

// fetch public keys for our recipients, and ready them
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);
```

```
// Repeat for second recipient
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);

// seal message, only owners of $pk1 and $pk2 can decrypt $sealed with keys
// $ekeys[0] and $ekeys[1] respectively.
openssl_seal($data, $sealed, $ekeys, array($pk1,$pk2));

// free the keys from memory
openssl_free_key($pk1);
openssl_free_key($pk2);
```

See also openssl_open().

openssl_sign (PHP 4 >= 4.0.4)

Generate signature

bool **openssl_sign** (string data, string signature, mixed priv_key_id) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo. If successful the signature is returned in *`signature`*.

openssl_sign() computes a signature for the specified *`data`* by using SHA1 for hashing followed by encryption using the private key associated with *`priv_key_id`*. Note that the data itself is not encrypted.

Ejemplo 1. openssl_sign() example

```
// $data is assumed to contain the data to be signed

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
```

```
$pkeyid = openssl_get_privatekey($priv_key);

// compute signature
openssl_sign($data, $signature, $pkeyid);

// free the key from memory
openssl_free_key($pkeyid);
```

See also openssl_verify().

openssl_verify (PHP 4 >= 4.0.4)

Verify signature

int **openssl_verify** (string data, string signature, mixed pub_key_id) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Returns 1 if the signature is correct, 0 if it is incorrect, and -1 on error.

openssl_verify() verifies that the *`signature`* is correct for the specified *`data`* using the public key associated with *`pub_key_id`*. This must be the public key corresponding to the private key used for signing.

Ejemplo 1. openssl_verify() example

```
// $data and $signature are assumed to contain the data and the signature

// fetch public key from certificate and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);

// state whether signature is okay or not
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1)
    echo "good";
elseif ($ok == 0)
```

```
    echo "bad";
else
    echo "ugly, error checking signature";

// free the key from memory
openssl_free_key($pubkeyid);
```

See also openssl_sign().

openssl_x509_check_private_key (PHP 4 >= 4.2.0)

Checks if a private key corresponds to a CERT

bool **openssl_x509_check_private_key** (mixed cert, mixed key) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_x509_checkpurpose (PHP 4 >= 4.0.6)

Verifies if a certificate can be used for a particular purpose

bool **openssl_x509_checkpurpose** (mixed x509cert, int purpose, array cainfo [, string untrustedfile]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Returns `TRUE` if the certificate can be used for the intended purpose, `FALSE` if it cannot, or -1 on error.

openssl_x509_checkpurpose() examines the certificate specified by *`x509cert`* to see if it can be used for the purpose specified by *`purpose`*.

`cainfo` should be an array of trusted CA files/dirs as described in Certificate Verification.

`untrustedfile`, if specified, is the name of a PEM encoded file holding certificates that can be used to help verify the certificate, although no trust in placed in the certificates that come from that file.

Tabla 1. openssl_x509_checkpurpose() purposes

Constant	Description
X509_PURPOSE_SSL_CLIENT	Can the certificate be used for the client side of an SSL connection?
X509_PURPOSE_SSL_SERVER	Can the certificate be used for the server side of an SSL connection?
X509_PURPOSE_NS_SSL_SERVER	Can the cert be used for Netscape SSL server?
X509_PURPOSE_SMIME_SIGN	Can the cert be used to sign S/MIME email?
X509_PURPOSE_SMIME_ENCRYPT	Can the cert be used to encrypt S/MIME email?
X509_PURPOSE_CRL_SIGN	Can the cert be used to sign a certificate revocation list (CRL)?
X509_PURPOSE_ANY	Can the cert be used for Any/All purposes?

These options are not bitfields - you may specify one only!

Nota: This function was added in 4.0.6.

openssl_x509_export_to_file (PHP 4 >= 4.2.0)

Exports a CERT to file or a var

bool **openssl_x509_export_to_file** (mixed x509, string outfilename [, bool notext]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_x509_export (PHP 4 >= 4.2.0)

Exports a CERT to file or a var

bool **openssl_x509_export** (mixed x509, string outfilename [, bool notext]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

openssl_x509_free (PHP 4 >= 4.0.6)

Free certificate resource

void **openssl_x509_free** (resource x509cert) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

openssl_x509_free() frees the certificate associated with the specified `x509cert` resource from memory.

Nota: This function was added in 4.0.6.

openssl_x509_parse (PHP 4 >= 4.0.6)

Parse an X509 certificate and return the information as an array

array **openssl_x509_parse** (mixed x509cert [, bool shortnames]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

openssl_x509_parse() returns information about the supplied `x509cert`, including fields such as subject name, issuer name, purposes, valid from and valid to dates etc. `shortnames` controls how the data is indexed in the array - if `shortnames` is `TRUE` (the default) then fields will be indexed with the short name form, otherwise, the long name form will be used - e.g.: CN is the shortname form of commonName.

The structure of the returned data is (deliberately) not yet documented, as it is still subject to change.

Nota: This function was added in 4.0.6.

openssl_x509_read (PHP 4 >= 4.0.6)

Parse an X.509 certificate and return a resource identifier for it

resource **openssl_x509_read** (mixed x509certdata) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

openssl_x509_read() parses the certificate supplied by `x509certdata` and returns a resource identifier for it.

Nota: This function was added in 4.0.6.

LXXII. Funciones Oracle

Ora_Bind (PHP 3, PHP 4)

Vincula una variable PHP a un parámetro Oracle

int **ora_bind** (int cursor, string nombre de variable PHP, string nombre de parámetro SQL, int longitud [, int tipo]) \linebreak

Devuelve verdadero si el vínculo se realiza con éxito, y sino devuelve falso. Los detalles de los errores pueden examinarse usando la funciones ora_error() y ora_errorcode().

Esta función liga la variable PHP nombrada con el parámetro SQL. El parámtro SQL debe estar en la forma ":name". Con el parámetro optativo tipo, se define si el parámetro SQL se trata de un parámetro de entrada/salida (0 y por defecto), entrada (1) o salida (2). Como en PHP 3.0.1, se puede usar las constantes ORA_BIND_INOUT, ORA_BIND_IN y ORA_BIND_OUT en lugar de los números.

ora_bind debe ser llamada después de ora_parse() y antes de ora_exec(). Los valores de entrada pueden pasarse por asignación a las variables PHP vinculadas, despues de la llamada a ora_exec() dichas variables contendrán los valores de salida, si éstos estuvieran disponibles.

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind($curs, "result", ":x", $len, 2);
ora_bind($curs, "input", ":in", 5, 1);
ora_bind($curs, "output", ":out", 5, 2);
$input = 765;
ora_exec($curs);
echo "Result: $result<BR>Out: $output<BR>In: $input";
?>
```

Ora_Close (PHP 3, PHP 4)

Cierra un cursor Oracle

int **ora_close** (int cursor) \linebreak

Devuelve verdadero si el cierre fué exitoso, o falso de lo contrario. Los detalles de los errores se recuperan usando las funciones ora_error() y ora_errorcode().

Esta función cierra un cursor de datos abierto con ora_open().

Ora_ColumnName (PHP 3, PHP 4)

toma el nombre de una columna de resultados Oracle

string **Ora_ColumnName** (int cursor, int column) \linebreak

Devuelve el nomre de un campo/columna *`column`* en el cursor *`cursor`*. el nombre devuelto estará en letras mayúsculas.

Ora_ColumnSize (PHP 3, PHP 4)

get size of Oracle result column

int **Ora_ColumnSize** (int cursor, int column) \linebreak

Returns the size of the Oracle column *`column`* on the cursor *`cursor`*. Column 0 is the first column.

Ora_ColumnType (PHP 3, PHP 4)

toma el tipo de una columna de resultados Oracle

string **Ora_ColumnType** (int cursor, int column) \linebreak

Devuelve el nombre del tipo de datos del campo o columna *`column`* en el cursor *`cursor`*. Se devolverá un tipo de datos, de entre los siguientes:

```
"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"CURSOR"
```

Ora_Commit (PHP 3, PHP 4)

realiza una transacción Oracle

int **ora_commit** (int conn) \linebreak

Devuelve verdadero si es exitosa, de lo cantrario devuelve falso. Puede verse los detalles del error usando las funciones ora_error() y ora_errorcode(). Esta función realiza una transacción Oracle. Se define como transacción cualquier cambio en una conexión dada, desde la última tarea/retroceso en la ejecucion (rollback), anulación de la ejecución automática de tareas (autocommit), o cuando se ha establecido la conexión.

Ora_CommitOff (PHP 3, PHP 4)

deshabilita el modo automatico de ejecucion de tareas (autocommit)

int **ora_commitoff** (int conn) \linebreak

Devuelve verdadero si se ejecuta con éxito, sino devuelve falso. Los pormenores del error en cuestion, pueden revisarse invocando las funciones ora_error() y ora_errorcode().

Esta función deshabilita la ejecucion automatica luego de cada instancia ora_exec().

Ora_CommitOn (PHP 3, PHP 4)

Habilita la ejecucion automática de tareas (autocommit)

int **ora_commiton** (int conn) \linebreak

Esta función habilita la ejecucion automatica luego de cada instancia ora_exec() en la conexión dada.

Devuelve verdadero si se ejecuta con éxito, sino devuelve falso. Los pormenores del error en cuestion, pueden revisarse invocando las funciones ora_error() y ora_errorcode().

Ora_Do (PHP 3, PHP 4)

Parse, Exec, Fetch

int **ora_do** (int conn, string query) \linebreak

This function is quick combination of ora_parse(), ora_exec() and ora_fetch(). It will parse and execute a statement, then fetch the first result row.

Returns `TRUE` on success, `FALSE` on error. Details about the error can be retrieved using the ora_error() and ora_errorcode() functions.

See also ora_parse(),ora_exec(), and ora_fetch().

Ora_Error (PHP 3, PHP 4)

toma los mensajes de error de Oracle

string **Ora_Error** (int cursor_or_connection) \linebreak

Devuelve los mensajes de error en la forma *`XXX-NNNNN`* donde *`XXX`* es la procedencia del error y *`NNNNN`* es la identificación del mensaje de error.

Nota: El soporte para las identificaciones de conexión fue agregado en la versión 3.0.4.

En las versiones UNIX de Oracle, pueden encontrarse detalles acerca de un mensaje de error como este:

```
 $ oerr ora 00001 00001, 00000, "unique constraint (%s.%s) violated" //
*Cause: An update or insert statement attempted to insert a duplicate key //
For Trusted ORACLE configured in DBMS MAC mode, you may see // this message
if a duplicate entry exists at a different level. // *Action: Either remove
the unique restriction or do not insert the key
```

Ora_ErrorCode (PHP 3, PHP 4)

captura el código de error Oracle

int **Ora_ErrorCode** (int cursor_or_connection) \linebreak

Devuelve el codigo numérico de error de la última declaración ejecutada en el cursor o conexion espacificada.

Nota: El soporte para las identificaciones de conexión fue agregado en la versión 3.0.4.

Ora_Exec (PHP 3, PHP 4)

ejecuta las declaraciones interpretadas en un cursor Oracle

int **ora_exec** (int cursor) \linebreak

Devuelve verdadero ante la ejecución exitosa, de lo contrario, devuelve falso. Los detalles del error pueden verse invocando las funciones ora_error() y ora_errorcode().

Ora_Fetch_Into (PHP 3, PHP 4)

Fetch a row into the specified result array

int **ora_fetch_into** (int cursor, array result [, int flags]) \linebreak

Fetches a row of data into an array. The *flags* has two flag values: if the ORA_FETCHINTO_NULLS flag is set, columns with NULL values are set in the array; and if the ORA_FETCHINTO_ASSOC flag is set, an associative array is created.

Returns the number of columns fetched.

Ejemplo 1. ora_fetch_into()()

```
<?php
$results = array();
ora_fetch_into($cursor, $results);
echo $results[0];
echo $results[1];
$results = array();
ora_fetch_into($cursor, $results, ORA_FETCHINTO_NULLS|ORA_FETCHINTO_ASSOC);
echo $results['MyColumn'];
?>
```

See also ora_parse(),ora_exec(), ora_fetch(), and ora_do().

Ora_Fetch (PHP 3, PHP 4)

extrae una fila de datos a partir de un cursor

int **ora_fetch** (int cursor) \linebreak

Devuelve verdadero (se extrajo una fila) o falso (no hay mas filas, o ha ocurrido un error). Si ocurre un error, los detalles del mismo pueden verse invocando las funciones ora_error() y ora_errorcode(). Si no hubo errores, ora_errorcode() devolverá 0. Recupera una hilera de datos partiendo de un cursor especificado.

Ora_GetColumn (PHP 3, PHP 4)

toma datos de la fila extraída

mixed **ora_getcolumn** (int cursor, mixed column) \linebreak

Devuelve la columna de datos. Si hay un error, se devuelve falso y ora_errorcode() devulve un valor distinto de cero. Note, de igual manera, que la busqueda de un resultado Falso en esta función, puede resultar verdadera, aún en casos en que no ocurran errores:(resultado NULO, cadenas vacias, valor 0 o cadenas "0"). Extrae los datos para una columna o resultado de función.

Ora_Logoff (PHP 3, PHP 4)

cierra una conexión Oracle

int **ora_logoff** (int connection) \linebreak

Devuelve verdadero si es exitosa, o falso si hay errores. Los detalles del error pueden verse invocando las funciones ora_error() y ora_errorcode(). Cierra la sesión de trabajo del usuario, y lo desconecta del servidor.

Ora_Logon (PHP 3, PHP 4)

Abre una conexión Oracle

int **ora_logon** (string usuario, string contraseña) \linebreak

Establece una conexión entre PHP y una base de datos Oracle, con los datos de nombre de usuario y contraseña especificados.

Las conexiones pueden llevarse adelante usando SQL*Net indicando el nombre TNS al *usuario* de este modo:

```
$conn = Ora_Logon("usuario@TNSNAME", "contraseña");
```

Si hubiesen datos con caracteres no-ASCII, habría que asegurarse de que esté presente la variable de entorno NLS_LANG en el sistema. Para los modulos de servidor, deberían definirse en el entorno del servidor antes de iniciarlo.

Devuelve el índice de la conexión si aquella tuvo éxito, de lo cantrario devuelve falso. Los detalles del error pueden verse invocando las funciones ora_error() y ora_errorcode().

Ora_Numcols (PHP 3, PHP 4)

Returns the number of columns

int **ora_numcols** (int cursor_ind) \linebreak

ora_numcols() returns the number of columns in a result. Only returns meaningful values after an parse/exec/fetch sequence.

See also ora_parse(),ora_exec(), ora_fetch(), and ora_do().

Ora_Numrows (PHP 3, PHP 4)

Returns the number of rows

int **ora_numrows** (int cursor_ind) \linebreak

ora_numrows() returns the number of rows in a result.

Ora_Open (PHP 3, PHP 4)

abre un cursor Oracle

int **ora_open** (int connection) \linebreak

Abre un cursor asociado con la conexión.

Devuelve el índice del cursor o Falso si hay un error. Los detalles del error pueden verse invocando las funciones ora_error() y ora_errorcode().

Ora_Parse (PHP 3, PHP 4)

interpreta una declaración SQL

int **ora_parse** (int cursor_ind, string sql_statement, int defer) \linebreak

Esta función interpreta una declaración SQL o un bloque PL/SQL y los asocia con el cursor dado. Devuelve 0 si se ejecuta con exito, o -1 ante un error.

Ora_pLogon (PHP 3, PHP 4)

Open a persistent Oracle connection

int **ora_plogon** (string user, string password) \linebreak

Establishes a persistent connection between PHP and an Oracle database with the given username and password.

See also ora_logon().

Ora_Rollback (PHP 3, PHP 4)

retrocede en la lista de transacciones (hace un roll back)

int **ora_rollback** (int connection) \linebreak

Deshace una transaccion Oracle. (Ver ora_commit() para la definición de transacción.)

Devuelve verdadero si tiene éxito, o falso si hay un error. Los detalles del error pueden verse invocando las funciones ora_error() y ora_errorcode().

LXXIII. Ovrimos SQL functions

Ovrimos SQL Server, is a client/server, transactional RDBMS combined with Web capabilities and fast transactions.

Ovrimos SQL Server is available at www.ovrimos.com (http://www.ovrimos.com/). To enable ovrimos support in PHP just compile php with the '--with-ovrimos' parameter to configure script. You'll need to install the sqlcli library available in the Ovrimos SQL Server distribution.

Ejemplo 1. Connect to Ovrimos SQL Server and select from a system table

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo ("Connection ok!");
    $res = ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will just connect to SQL Server.

ovrimos_close (PHP 4 >= 4.0.3)

Closes the connection to ovrimos

void **ovrimos_close** (int connection) \linebreak

ovrimos_close() is used to close the specified connection.

ovrimos_close() closes a connection to Ovrimos. This has the effect of rolling back uncommitted transactions.

ovrimos_commit (PHP 4 >= 4.0.3)

Commits the transaction

int **ovrimos_commit** (int connection_id) \linebreak

ovrimos_commit() is used to commit the transaction.

ovrimos_commit() commits the transaction.

ovrimos_connect (PHP 4 >= 4.0.3)

Connect to the specified database

int **ovrimos_connect** (string host, string db, string user, string password) \linebreak

ovrimos_connect() is used to connect to the specified database.

ovrimos_connect() returns a connection id (greater than 0) or 0 for failure. The meaning of 'host' and 'port' are those used everywhere in Ovrimos APIs. 'Host' is a host name or IP address and 'db' is either a database name, or a string containing the port number.

Ejemplo 1. ovrimos_connect() Example

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

The above example will connect to the database and print out the specified table.

ovrimos_cursor (PHP 4 >= 4.0.3)

Returns the name of the cursor

int **ovrimos_cursor** (int result_id) \linebreak

ovrimos_cursor() is used to get the name of the cursor.

ovrimos_cursor() returns the name of the cursor. Useful when wishing to perform positioned updates or deletes.

ovrimos_exec (PHP 4 >= 4.0.3)

Executes an SQL statement

int **ovrimos_exec** (int connection_id, string query) \linebreak

ovrimos_exec() is used to execute an SQL statement.

ovrimos_exec() executes an SQL statement (query or update) and returns a result_id or `FALSE`. Evidently, the SQL statement should not contain parameters.

ovrimos_execute (PHP 4 >= 4.0.3)

Executes a prepared SQL statement

int **ovrimos_execute** (int result_id [, array parameters_array]) \linebreak

ovrimos_execute() is used to execute an SQL statement.

ovrimos_execute() executes a prepared statement. Returns `TRUE` or `FALSE`. If the prepared statement contained parameters (question marks in the statement), the correct number of parameters should be passed in an array. Notice that I don’t follow the PHP convention of placing just the name of the optional parameter inside square brackets. I couldn’t bring myself on liking it.

ovrimos_fetch_into (PHP 4 >= 4.0.3)

Fetches a row from the result set

int **ovrimos_fetch_into** (int result_id, array result_array [, string how [, int rownumber]]) \linebreak

ovrimos_fetch_into() is used to fetch a row from the result set.

ovrimos_fetch_into() fetches a row from the result set into 'result_array', which should be passed by reference. Which row is fetched is determined by the two last parameters. 'how' is one of 'Next' (default), 'Prev', 'First', 'Last', 'Absolute', corresponding to forward direction from current position, backward direction from current position, forward direction from the start, backward direction from the end and absolute position from the start (essentially equivalent to 'first' but needs 'rownumber'). Case is not significant. 'Rownumber' is optional except for absolute positioning. Returns `TRUE` or `FALSE`.

Ejemplo 1. A fetch into example

```
<?php
$conn=ovrimos_connect ("neptune", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn,"select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_into ($res, &$row)) {
            list ($table_id, $table_name) = $row;
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_into ($res, &$row)) {
                list ($table_id, $table_name) = $row;
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

This example will fetch a row.

ovrimos_fetch_row (PHP 4 >= 4.0.3)

Fetches a row from the result set

int **ovrimos_fetch_row** (int result_id [, int how [, int row_number]]) \linebreak

ovrimos_fetch_row() is used to fetch a row from the result set.

ovrimos_fetch_row() fetches a row from the result set. Column values should be retrieved with other calls. Returns `TRUE` or `FALSE`.

Ejemplo 1. A fetch row example

```
<?php
$conn = ovrimos_connect ("remote.host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_row ($res, "First")) {
            $table_id = ovrimos_result ($res, 1);
            $table_name = ovrimos_result ($res, 2);
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_row ($res, "Next")) {
                $table_id = ovrimos_result ($res, "table_id");
                $table_name = ovrimos_result ($res, "table_name");
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

This will fetch a row and print the result.

ovrimos_field_len (PHP 4 >= 4.0.3)

Returns the length of the output column

int **ovrimos_field_len** (int result_id, int field_number) \linebreak

ovrimos_field_len() is used to get the length of the output column.

ovrimos_field_len() returns the length of the output column at the (1-based) index specified.

ovrimos_field_name (PHP 4 >= 4.0.3)

Returns the output column name

int **ovrimos_field_name** (int result_id, int field_number) \linebreak

ovrimos_field_name() is used to get the output column name.

ovrimos_field_name() returns the output column name at the (1-based) index specified.

ovrimos_field_num (PHP 4 >= 4.0.3)

Returns the (1-based) index of the output column

int **ovrimos_field_num** (int result_id, string field_name) \linebreak

ovrimos_field_num() is used to get the (1-based) index of the output column.

ovrimos_field_num() returns the (1-based) index of the output column specified by name, or `FALSE`.

ovrimos_field_type (PHP 4 >= 4.0.3)

Returns the (numeric) type of the output column

int **ovrimos_field_type** (int result_id, int field_number) \linebreak

ovrimos_field_type() is used to get the (numeric) type of the output column.

ovrimos_field_type() returns the (numeric) type of the output column at the (1-based) index specified.

ovrimos_free_result (PHP 4 >= 4.0.3)

Frees the specified result_id

int **ovrimos_free_result** (int result_id) \linebreak

ovrimos_free_result() is used to free the result_id.

ovrimos_free_result() frees the specified result_id. Returns `TRUE`.

ovrimos_longreadlen (PHP 4 >= 4.0.3)

Specifies how many bytes are to be retrieved from long datatypes

int **ovrimos_longreadlen** (int result_id, int length) \linebreak

ovrimos_longreadlen() is used to specify how many bytes are to be retrieved from long datatypes.

ovrimos_longreadlen() specifies how many bytes are to be retrieved from long datatypes (long varchar and long varbinary). Default is zero. Regardless of its taking a result_id as an argument, it currently sets this parameter for all result sets. Returns `TRUE`.

ovrimos_num_fields (PHP 4 >= 4.0.3)

Returns the number of columns

int **ovrimos_num_fields** (int result_id) \linebreak

ovrimos_num_fields() is used to get the number of columns.

ovrimos_num_fields() returns the number of columns in a result_id resulting from a query.

ovrimos_num_rows (PHP 4 >= 4.0.3)

Returns the number of rows affected by update operations

int **ovrimos_num_rows** (int result_id) \linebreak

ovrimos_num_rows() is used to get the number of rows affected by update operations.

ovrimos_num_rows() returns the number of rows affected by update operations.

ovrimos_prepare (PHP 4 >= 4.0.3)

Prepares an SQL statement

int **ovrimos_prepare** (int connection_id, string query) \linebreak

ovrimos_prepare() is used to prepare an SQL statement.

ovrimos_prepare() prepares an SQL statement and returns a result_id (or `FALSE` on failure).

Ejemplo 1. Connect to Ovrimos SQL Server and prepare a statement

```
<?php
$conn=ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_prepare ($conn, "select table_id, table_name
                                  from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res)) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
```

```
        echo "Prepare not ok!\n";
    }
    ovrimos_close ($conn);
}
?>
```

This will connect to Ovrimos SQL Server, prepare a statement and the execute it.

ovrimos_result_all (PHP 4 >= 4.0.3)

Prints the whole result set as an HTML table

int **ovrimos_result_all** (int result_id [, string format]) \linebreak

ovrimos_result_all() is used to print an HTML table containing the whole result set.

ovrimos_result_all() prints the whole result set as an HTML table. Returns `TRUE` or `FALSE`.

Ejemplo 1. Prepare a statement, execute, and view the result

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_prepare ($conn, "select table_id, table_name
                                    from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res, array(3))) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close ($conn);
}
?>
```

This will execute an SQL statement and print the result in an HTML table.

Ejemplo 2. Ovrimos_result_all with meta-information

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "select table_id, table_name
                                 from sys.tables where table_id = 1")
    if ($res != 0) {
        echo "Statement ok! cursor=".ovrimos_cursor ($res)."\n";
        $colnb = ovrimos_num_fields ($res);
        echo "Output columns=".$colnb."\n";
        for ($i=1; $i<=$colnb; $i++) {
            $name = ovrimos_field_name ($res, $i);
            $type = ovrimos_field_type ($res, $i);
            $len = ovrimos_field_len ($res, $i);
            echo "Column ".$i." name=".$name." type=".$type." len=".$len."\n";
        }
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

Ejemplo 3. ovrimos_result_all example

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "update test set i=5");
    if ($res != 0) {
        echo "Statement ok!";
        echo ovrimos_num_rows ($res)." rows affected\n";
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

ovrimos_result (PHP 4 >= 4.0.3)

Retrieves the output column

int **ovrimos_result** (int result_id, mixed field) \linebreak

ovrimos_result() is used to retrieve the output column.

ovrimos_result() retrieves the output column specified by ’field’, either as a string or as an 1-based index.

ovrimos_rollback (PHP 4 >= 4.0.3)

Rolls back the transaction

int **ovrimos_rollback** (int connection_id) \linebreak

ovrimos_rollback() is used to roll back the transaction.

ovrimos_rollback() rolls back the transaction.

LXXIV. Output Control Functions

The Output Control functions allow you to control when output is sent from the script. This can be useful in several different situations, especially if you need to send headers to the browser after your script has began outputing data. The Output Control functions do not affect headers sent using header() or setcookie(), only functions such as echo() and data between blocks of PHP code.

Ejemplo 1. Output Control example

```
<?php

ob_start();
echo "Hello\n";

setcookie ("cookiename", "cookiedata");

ob_end_flush();

?>
```

In the above example, the output from echo() would be stored in the output buffer until ob_end_flush() was called. In the mean time, the call to setcookie() successfully stored a cookie without causing an error. (You can not normally send headers to the browser after data has already been sent.)

See also header() and setcookie().

flush (PHP 3, PHP 4)

Flush the output buffer

void **flush** (void) \linebreak

Flushes the output buffers of PHP and whatever backend PHP is using (CGI, a web server, etc.) This effectively tries to push all the output so far to the user's browser.

ob_clean (PHP 4 >= 4.2.0)

Clean (erase) the output buffer

void **ob_clean** (void) \linebreak

This function discards the contents of the output buffer.

This function does not destroy the output buffer like ob_end_clean() does.

See also ob_flush(), ob_end_flush() and ob_end_clean().

ob_end_clean (PHP 4)

Clean (erase) the output buffer and turn off output buffering

void **ob_end_clean** (void) \linebreak

This function discards the contents of the output buffer and turns off output buffering.

See also ob_start() and ob_end_flush().

ob_end_flush (PHP 4)

Flush (send) the output buffer and turn off output buffering

void **ob_end_flush** (void) \linebreak

This function will send the contents of the output buffer (if any) and turn output buffering off. If you want to further process the buffer's contents you have to call ob_get_contents() before **ob_end_flush()** as the buffer contents are discarded after ob_get_contents() is called.

See also ob_start(), ob_get_contents(), and ob_end_clean().

ob_flush (PHP 4 >= 4.2.0)

Flush (send) the output buffer

void **ob_flush** (void) \linebreak

This function will send the contents of the output buffer (if any). If you want to further process the buffer's contents you have to call ob_get_contents() before **ob_flush()** as the buffer contents are discarded after **ob_flush()** is called.

This function does not destroy the output buffer like ob_end_flush() does.

See also ob_get_contents(), ob_clean(), ob_end_flush() and ob_end_clean().

ob_get_contents (PHP 4)

Return the contents of the output buffer

string **ob_get_contents** (void) \linebreak

This will return the contents of the output buffer or `FALSE`, if output buffering isn't active.

See also ob_start() and ob_get_length().

ob_get_length (PHP 4 >= 4.0.2)

Return the length of the output buffer

string **ob_get_length** (void) \linebreak

This will return the length of the contents in the output buffer or `FALSE`, if output buffering isnt't active.

See also ob_start() and ob_get_contents().

ob_get_level (PHP 4 >= 4.2.0)

Return the nesting level of the output buffering mechanism

int **ob_get_level** (void) \linebreak

This will return the level of nested output buffering handlers.

See also ob_start() and ob_get_contents().

ob_get_status (PHP 4 >= 4.2.0)

Get status of output buffers

array **ob_get_status** ([bool full_status]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

This will return the current status of output buffers. It returns array contains buffer status or `FALSE` for error.

See also ob_get_level().

ob_gzhandler (PHP 4 >= 4.0.4)

ob_start callback function to gzip output buffer

string **ob_gzhandler** (string buffer [, int mode]) \linebreak

> **Nota:** *mode* was added in PHP 4.0.5.

ob_gzhandler() is intended to be used as a callback function for ob_start() to help facilitate sending gz-encoded data to web browsers that support compressed web pages. Before **ob_gzhandler()** actually sends compressed data, it determines what type of content encoding the browser will accept ("gzip", "deflate" or none at all) and will return it's output accordingly. All browsers are supported since it's up to the browser to send the correct header saying that it accepts compressed web pages.

Ejemplo 1. ob_gzhandler() Example

```
<?php

ob_start("ob_gzhandler");

?>
<html>
<body>
<p>This should be a compressed page.
</html>
<body>
```

See also ob_start() and ob_end_flush().

ob_implicit_flush (PHP 4)

Turn implicit flush on/off

void **ob_implicit_flush** ([int flag]) \linebreak

ob_implicit_flush() will turn implicit flushing on or off (if no *flag* is given, it defaults to on). Implicit flushing will result in a flush operation after every output call, so that explicit calls to flush() will no longer be needed.

Turning implicit flushing on will disable output buffering, the output buffers current output will be sent as if ob_end_flush() had been called.

See also flush(), ob_start(), and ob_end_flush().

ob_start (PHP 4)

Turn on output buffering

void **ob_start** (void) \linebreak

This function will turn output buffering on. While output buffering is active no output is sent from the script, instead the output is stored in an internal buffer.

The contents of this internal buffer may be copied into a string variable using ob_get_contents(). To output what is stored in the internal buffer, use ob_end_flush(). Alternatively, ob_end_clean() will silently discard the buffer contents.

See also ob_get_contents(), ob_end_flush(), ob_end_clean(), and ob_implicit_flush()

LXXV. Object property and method call overloading

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The purpose of this extension is to allow overloading of object property access and method calls. Only one function is defined in this extension, overload() which takes the name of the class that should have this functionality enabled. The class named has to define appropriate methods if it wants to have this functionality: `__get()`, `__set()` and `__call()` respectively for getting/setting a property, or calling a method. This way overloading can be selective. Inside these handler functions the overloading is disabled so you can access object properties normally.

Some simple examples on using the overload() function:

Ejemplo 1. Overloading a PHP class

```
<?php

class OO
{
    var $a = 111;
    var $elem = array('b' => 9, 'c' => 42);

    // Callback method for getting a property
    function __get($prop_name, &$prop_value)
    {
        if (isset($this->elem[$prop_name])) {
            $prop_value = $this->elem[$prop_name];
            return true;
        } else {
            return false;
        }
    }

    // Callback method for setting a property
    function __set($prop_name, $prop_value)
    {
        $this->elem[$prop_name] = $prop_value;
        return true;
    }
}

// Here we overload the OO object
```

```
overload('OO');

$o = new OO;
print "\$o->a: $o->a\n"; // print: $o->a:
print "\$o->b: $o->b\n"; // print: $o->b: 9
print "\$o->c: $o->c\n"; // print: $o->c: 42
print "\$o->d: $o->d\n"; // print: $o->d:

// add a new item to the $elem array in OO
$o->x = 56;

// instantiate stdclass (it is built-in in PHP 4)
// $val is not overloaded!
$val = new stdclass;
$val->prop = 555;

// Set "a" to be an array with the $val object in it
// But __set() will put this in the $elem array
$o->a = array($val);
var_dump($o->a[0]->prop);

?>
```

Aviso

As this is an experimental extension, not all things work. There is no `__call()` support currently, you can only overload the get and set operations for properties. You cannot invoke the original overloading handlers of the class, and `__set()` only works to one level of property access.

overload (PHP 4 >= 4.2.0)

Enable property and method call overloading for a class

void **overload** ([string class_name]) \linebreak

The **overload()** function will enable property and method call overloading for a class identified by *class_name*. See an example in the introductory section of this part.

LXXVI. PDF functions

You can use the PDF functions in PHP to create PDF files if you have the PDF library by Thomas Merz (available at http://www.pdflib.com/pdflib/index.html; you will also need the JPEG library (ftp://ftp.uu.net/graphics/jpeg/) and the TIFF library (http://www.libtiff.org/) to compile this. These two libs also quite often make problems when configuring php. Follow the messages of configure to fix possible problems. If you use pdflib 2.01 check how the lib was installed. There should be file or link libpdf.so. Version 2.01 just creates a lib with the name libpdf2.01.so which cannot be found when linking the test programm in configure. You will have to create a symbolic link from libpdf.so to libpdf2.01.so.).

Version 2.20 of pdflib has introduced more changes to its API and support for chinese and japanese fonts. This unfortunately causes some changes of the pdf module of php4 (not php3). If you use pdflib 2.20 handle the in memory generation of PDF documents with care. Until pdflib 3.0 is released it might be unstable. The encoding parameter of pdf_set_font() has changed to a string. This means that instead of e.g. 4 you have to use 'winansi'.

If you use pdflib 2.30 the pdf_set_text_matrix() will have gone. It is not supported any more. In general it is a good advise to consult the release notes of the used version of pdflib for possible changes.

Since version 3.0 of pdflib you should configure pdflib with the option `--enable-shared-pdflib`.

Any version of PHP4 after March, 9th 2000 do not support versions of pdflib older than 3.0. PHP3 on the other hand should not be used with version newer than 2.01.

Please consult the excellent documentation for pdflib shipped with the source distribution of pdflib. It provides a very good overview of what pdflib capable of doing. Most of the functions in pdflib and the PHP module have the same name. The parameters are also identical. You should also understand some of the concepts of PDF or Postscript to efficiently use this module. All lengths and coordinates are measured in Postscript points. There are generally 72 PostScript points to an inch, but this depends on the output resolution.

There is another PHP module for pdf document creation based on FastIO's (http://www.fastio.com/). ClibPDF. It has a slightly different API. Check the ClibPDF functions section for details.

Currently all versions of pdflib are supported. It is recommended that you use the newest version since it has more features and fixes some problems which required a patch for the old version. Unfortunately, the changes of the pdflib API in 2.x compared to 0.6 have been so severe that even some PHP functions had to be altered. Here is a list of changes:

- The Info structure does not exist anymore. Therefore the function **pdf_get_info()** is obsolete and the functions pdf_set_info_creator(), pdf_set_info_title(), pdf_set_info_author(), pdf_set_info_subject() and pdf_set_info_keywords() do not take the info structure as the first parameter but the pdf document. This also means that the pdf document must be opened before these functions can be called. The above functions can and should also be replaced by pdf_set_info()
- The way a new document is opened has changed. The function pdf_open() takes only one parameter which is the file handle of a file opened with fopen().

There were some more changes with the release 2.01 of pdflib which should be covered by PHP. Some functions are not required anymore (e.g. **pdf_put_image()**). You will get a warning so don't be shocked.

The pdf module introduces two new types of variables (if pdflib 2.x is used it is only one new type). They are called *`pdfdoc`* and *`pdfinfo`* (*`pdfinfo`* is not existent if pdflib 2.x is used. *`pdfdoc`* is a pointer

to a pdf document and almost all functions need it as its first parameter. *pdfinfo* contains meta data about the PDF document. It has to be set before pdf_open() is called.

Nota: The following is only TRUE for pdflib 0.6. Read the pdflib manual for newer version

In order to output text into a PDF document you will need to provide the afm file for each font. Afm files contain font metrics for a Postscript font. By default these afm files are searched for in a directory named 'fonts' relative to the directory where the PHP script is located. (Again, this was TRUE for pdflib 0.6, newer versions do not not neccessarily need the afm files.)

Most of the functions are fairly easy to use. The most difficult part is probably to create a very simple pdf document at all. The following example should help to get started. It uses the PHP functions for pdflib 0.6. It creates the file test.pdf with one page. The page contains the text "Times-Roman" in an outlined 30pt font. The text is also underlined.

Ejemplo 1. Creating a PDF document with pdflib 0.6

```
<?php
$fp = fopen("test.pdf", "w");
$info = PDF_get_info();
pdf_set_info_author($info, "Uwe Steinmann");
PDF_set_info_title($info, "Test for PHP wrapper of PDFlib 0.6");
PDF_set_info_author($info, "Name of Author");
pdf_set_info_creator($info, "See Author");
pdf_set_info_subject($info, "Testing");
$pdf = PDF_open($fp, $info);
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, 4);
pdf_set_text_rendering($pdf, 1);
PDF_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
PDF_end_page($pdf);
PDF_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php3>finished</A>";
?>
```

The PHP script getpdf.php3 just outputs the pdf document.

```
<?php
$fp = fopen("test.pdf", "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

Doing the same with pdflib 2.x looks like the following:

Ejemplo 2. Creating a PDF document with pdflib 2.x

```
<?php
$fp = fopen("test.pdf", "w");
$pdf = PDF_open($fp);
pdf_set_info_author($pdf, "Uwe Steinmann");
PDF_set_info_title($pdf, "Test for PHP wrapper of PDFlib 2.0");
PDF_set_info_author($pdf, "Name of Author");
pdf_set_info_creator($pdf, "See Author");
pdf_set_info_subject($pdf, "Testing");
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, 4);
pdf_set_text_rendering($pdf, 1);
PDF_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
PDF_end_page($pdf);
PDF_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php3>finished</A>";
?>
```

The PHP script getpdf.php3 is the same as above.

The pdflib distribution contains a more complex example which creates a serious of pages with an analog clock. This example converted into PHP using pdflib 2.x looks as the following (you can see the same example in the documentation for the clibpdf module):

Ejemplo 3. pdfclock example from pdflib 2.x distribution

```
<?php
$pdffilename = "clock.pdf";
$radius = 200;
$margin = 20;
$pagecount = 40;

$fp = fopen($pdffilename, "w");
$pdf = pdf_open($fp);
pdf_set_info_creator($pdf, "pdf_clock.php3");
pdf_set_info_author($pdf, "Uwe Steinmann");
pdf_set_info_title($pdf, "Analog Clock");

while($pagecount-- > 0) {
```

```
pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));

pdf_set_transition($pdf, 4);  /* wipe */
pdf_set_duration($pdf, 0.5);

pdf_translate($pdf, $radius + $margin, $radius + $margin);
pdf_save($pdf);
pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

/* minute strokes */
pdf_setlinewidth($pdf, 2.0);
for ($alpha = 0; $alpha < 360; $alpha += 6) {
    pdf_rotate($pdf, 6.0);
    pdf_moveto($pdf, $radius, 0.0);
    pdf_lineto($pdf, $radius-$margin/3, 0.0);
    pdf_stroke($pdf);
}

pdf_restore($pdf);
pdf_save($pdf);

/* 5 minute strokes */
pdf_setlinewidth($pdf, 3.0);
for ($alpha = 0; $alpha < 360; $alpha += 30) {
    pdf_rotate($pdf, 30.0);
    pdf_moveto($pdf, $radius, 0.0);
    pdf_lineto($pdf, $radius-$margin, 0.0);
    pdf_stroke($pdf);
}

$ltime = getdate();

/* draw hour hand */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius/2, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw minute hand */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius * 0.8, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw second hand */
```

```
    pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
    pdf_setlinewidth($pdf, 2);
    pdf_save($pdf);
    pdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
    pdf_moveto($pdf, -$radius/5, 0.0);
    pdf_lineto($pdf, $radius, 0.0);
    pdf_stroke($pdf);
    pdf_restore($pdf);

    /* draw little circle at center */
    pdf_circle($pdf, 0, 0, $radius/30);
    pdf_fill($pdf);

    pdf_restore($pdf);

    pdf_end_page($pdf);
}

$pdf = pdf_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php3?filename=".$pdffilename.">finished</A>";
?>
```

The PHP script getpdf.php3 just outputs the pdf document.

```
<?php
$fp = fopen($filename, "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

pdf_add_annotation (PHP 3>= 3.0.12, PHP 4)

Adds annotation

void **pdf_add_annotation** (int pdf document, double llx, double lly, double urx, double ury, string title, string content) \linebreak

The **pdf_add_annotation()** adds a note with the lower left corner at (*llx*, *lly*) and the upper right corner at (*urx*, *ury*).

pdf_add_bookmark (PHP 4 >= 4.0.1)

Adds bookmark for current page

int **pdf_add_bookmark** (int pdf object, string text [, int parent [, int open]]) \linebreak

Add a nested bookmark under *parent*, or a new top-level bookmark if *parent* = 0. Returns a bookmark descriptor which may be used as parent for subsequent nested bookmarks. If open = 1, child bookmarks will be folded out, and invisible if open = 0.

pdf_add_launchlink (PHP 4 >= 4.0.5)

Add a launch annotation for current page

int **pdf_add_launchlink** (int pdf object, float llx, float lly, float urx, float ury, string filename) \linebreak

Add a launch annotation (to a target of arbitrary file type).

pdf_add_locallink (PHP 4 >= 4.0.5)

Add a link annotation for current page

int **pdf_add_locallink** (int pdf object, float llx, float lly, float urx, float ury, int page, string dest) \linebreak

Add a link annotation to a target within the current PDF file.

pdf_add_note (PHP 4 >= 4.0.5)

Add a note annotation for current page

int **pdf_add_note** (int pdf object, float llx, float lly, float urx, float ury, string contents, string title, string icon, int open) \linebreak

Add a note annotation. icon is one of of "comment, "insert", "note", "paragraph", "newparagraph", "key", or "help".

PDF_add_outline (PHP 3>= 3.0.6, PHP 4)

Adds bookmark for current page

int **pdf_add_outline** (int pdf document, string text [, int parent [, int open]]) \linebreak

The **PDF_add_outline()** function adds a bookmark with text `text` that points to the current page. The bookmark is inserted as a child of `parent` and is by default open if `open` is not 0. The return value is an identifier for the bookmark which can be used as a parent for other bookmarks. Therefore you can build up hierarchies of bookmarks.

Unfortunately pdflib does not make a copy of the string, which forces PHP to allocate the memory. Currently this piece of memory is not been freed by any PDF function but it will be taken care of by the PHP memory manager.

pdf_add_pdflink (PHP 3>= 3.0.12, PHP 4)

Adds file link annotation for current page

int **pdf_add_pdflink** (int pdf object, float llx, float lly, float urx, float ury, string filename, int page, string dest) \linebreak

Add a file link annotation (to a PDF target).

pdf_add_thumbnail (PHP 4 >= 4.0.5)

Adds thumbnail for current page

int **pdf_add_thumbnail** (int pdf object, int image) \linebreak

Add an existing image as thumbnail for the current page.

pdf_add_weblink (PHP 3>= 3.0.12, PHP 4)

Adds weblink for current page

int **pdf_add_weblink** (int pdf object, float llx, float lly, float urx, float ury, string url) \linebreak

Add a weblink annotation to a target URL on the Web.

PDF_arc (PHP 3>= 3.0.6, PHP 4)

Draws an arc

void **pdf_arc** (int pdf document, double x-coor, double y-coor, double radius, double start, double end) \linebreak

The **PDF_arc()** function draws an arc with center at point (*x-coor*, *y-coor*) and radius *radius*, starting at angle *start* and ending at angle *end*.

See also **PDF_circle()**, **PDF_stroke()**.

pdf_arcn (PHP 4 >= 4.0.5)

Draws an arc (clockwise)

void **pdf_arc** (resource pdf object, float x, float y, float r, float alpha, float beta) \linebreak

Draw a clockwise circular arc from alpha to beta degrees

See also: pdf_arc()

pdf_attach_file (PHP 4 >= 4.0.5)

Adds a file attachement for current page

int **pdf_attach_file** (int pdf object, float llx, float lly, float urx, float ury, string filename, string description, string author, string mimetype, string icon) \linebreak

Add a file attachment annotation. icon is one of "graph, "paperclip", "pushpin", or "tag".

PDF_begin_page (PHP 3>= 3.0.6, PHP 4)

Starts new page

void **pdf_begin_page** (int pdf document, double width, double height) \linebreak

The **PDF_begin_page()** function starts a new page with height *height* and width *width*. In order to create a valid document you must call this function and **PDF_end_page()**.

See also **PDF_end_page()**.

pdf_begin_pattern (PHP 4 >= 4.0.5)

Starts new pattern

int **pdf_begin_pattern** (int pdf object, float width, float height, float xstep, float ystep, int painttype) \linebreak

Starts a new pattern definition and returns a pattern handle. *width*, and *height* define the bounding box for the pattern. *xstep* and *ystep* give the repeated pattern offsets. *painttype*=1 means that the pattern has its own colour settings whereas a value of 2 indicates that the current colour is used when the pattern is applied.

pdf_begin_template (PHP 4 >= 4.0.5)

Starts new template

void **pdf_begin_template** (int pdf object, float width, float height) \linebreak

Start a new template definition.

PDF_circle (PHP 3>= 3.0.6, PHP 4)

Draws a circle

void **pdf_circle** (int pdf document, double x-coor, double y-coor, double radius) \linebreak

The **PDF_circle()** function draws a circle with center at point (*x-coor*, *y-coor*) and radius *radius*.

See also **PDF_arc()**, **PDF_stroke()**.

PDF_clip (PHP 3>= 3.0.6, PHP 4)

Clips to current path

void **pdf_clip** (int pdf document) \linebreak

The **PDF_clip()** function clips all drawing to the current path.

PDF_close_image (PHP 3>= 3.0.7, PHP 4)

Closes an image

void **pdf_close_image** (int image) \linebreak

The **PDF_close_image()** function closes an image which has been opened with any of the **PDF_open_xxx()** functions.

See also **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_memory_image()**.

pdf_close_pdi_page (PHP 4 >= 4.0.5)

Close the page handle

void **pdf_close_pdi_page** (int pdf object, int pagehandle) \linebreak

Close the page handle, and free all page-related resources.

pdf_close_pdi (PHP 4 >= 4.0.5)

Close the input PDF document

void **pdf_close_pdi** (int pdf object, int dochandle) \linebreak

Close all open page handles, and close the input PDF document.

PDF_close (PHP 3>= 3.0.6, PHP 4)

Closes a pdf document

void **pdf_close** (int pdf document) \linebreak

The **PDF_close()** function closes the pdf document.

Nota: Due to an unclean implementation of the pdflib 0.6 the internal closing of the document also closes the file. This should not be done because pdflib did not open the file, but expects an already open file when **PDF_open()** is called. Consequently it shouldn't close the file. In order to fix this just take out line 190 of the file p_basic.c in the pdflib 0.6 source distribution until the next release of pdflib will fix this.

Nota: This function works properly without any patches to pdflib if pdflib 2.0 support is activated.

See also **PDF_open()**, fclose().

PDF_closepath_fill_stroke (PHP 3>= 3.0.6, PHP 4)

Closes, fills and strokes current path

void **pdf_closepath_fill_stroke** (int pdf document) \linebreak

The **PDF_closepath_fill_stroke()** function closes, fills the interior of the current path with the current fill color and draws current path.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_closepath_stroke (PHP 3>= 3.0.6, PHP 4)

Closes path and draws line along path

void **pdf_closepath_stroke** (int pdf document) \linebreak

The **PDF_closepath_stroke()** function is a combination of **PDF_closepath()** and **PDF_stroke()**. It also clears the path.

See also **PDF_closepath()**, **PDF_stroke()**.

PDF_closepath (PHP 3>= 3.0.6, PHP 4)

Closes path

void **pdf_closepath** (int pdf document) \linebreak

The **PDF_closepath()** function closes the current path. This means, it draws a line from current point to the point where the first line was started. Many functions like **PDF_moveto()**, **PDF_circle()** and **PDF_rect()** start a new path.

pdf_concat (PHP 4 >= 4.0.5)

Concatenate a matrix to the CTM

void **pdf_concat** (int pdf object, float a, float b, float c, float d, float e, float f) \linebreak

Concatenate a matrix to the CTM.

PDF_continue_text (PHP 3>= 3.0.6, PHP 4)

Outputs text in next line

void **pdf_continue_text** (int pdf document, string text) \linebreak

The **PDF_continue_text()** function outputs the string in `text` in the next line. The distance between the lines can be set with **PDF_set_leading()**.

See also **PDF_show_xy()**, **PDF_set_leading()**, **PDF_set_text_pos()**.

PDF_curveto (PHP 3>= 3.0.6, PHP 4)

Draws a curve

void **pdf_curveto** (int pdf document, double x1, double y1, double x2, double y2, double x3, double y3) \linebreak

The **PDF_curveto()** function draws a Bezier curve from the current point to the point (`x3`, `y3`) using (`x1`, `y1`) and (`x2`, `y2`) as control points.

See also **PDF_moveto()**, **PDF_lineto()**, **PDF_stroke()**.

pdf_delete (PHP 4 >= 4.0.5)

Deletes a PDF object

void **pdf_delete** (int pdf object) \linebreak

Delete the PDF object, and free all internal resources.

PDF_end_page (PHP 3>= 3.0.6, PHP 4)

Ends a page

void **pdf_end_page** (int pdf document) \linebreak

The **PDF_end_page()** function ends a page. Once a page is ended it cannot be modified anymore.

See also **PDF_begin_page()**.

pdf_end_pattern (PHP 4 >= 4.0.5)

Finish pattern

void **pdf_end_pattern** (int pdf object) \linebreak

Finish the pattern definition.

pdf_end_template (PHP 4 >= 4.0.5)

Finish template

void **pdf_end_template** (int pdf object) \linebreak

Finish the template definition.

PDF_endpath (PHP 3>= 3.0.6, PHP 4)

Ends current path

void **pdf_endpath** (int pdf document) \linebreak

The **PDF_endpath()** function ends the current path but does not close it.

See also **PDF_closepath()**.

PDF_fill_stroke (PHP 3>= 3.0.6, PHP 4)

Fills and strokes current path

void **pdf_fill_stroke** (int pdf document) \linebreak

The **PDF_fill_stroke()** function fills the interior of the current path with the current fill color and draws current path.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_fill (PHP 3>= 3.0.6, PHP 4)

Fills current path

void **pdf_fill** (int pdf document) \linebreak

The **PDF_fill()** function fills the interior of the current path with the current fill color.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

pdf_findfont (PHP 4 >= 4.0.5)

Prepare font for later use with pdf_setfont().

int **pdf_findfont** (int pdf object, string fontname, string encoding, int embed) \linebreak

Prepare a font for later use with pdf_setfont(). The metrics will be loaded, and if embed is nonzero, the font file will be checked, but not yet used. *encoding* is one of "builtin", "macroman", "winansi", "host", or a user-defined encoding name, or the name of a CMap.

pdf_findfont() returns a font handle or FALSE on error.

Ejemplo 1. pdf_findfont() example

```
<?php

$font = pdf_findfont($pdf, "Times New Roman", "winansi", 1);
if ($font) {
    pdf_setfont($pdf, $font, 10);
}

?>
```

pdf_get_buffer (PHP 4 >= 4.0.5)

Fetch the buffer containig the generated PDF data.

string **pdf_get_buffer** (int pdf object) \linebreak

Get the contents of the PDF output buffer. The result must be used by the client before calling any other PDFlib function.

pdf_get_font (PHP 4)

Deprecated: font handling

Deprecated.

See pdf_get_value().

pdf_get_fontname (PHP 4)

Deprecated: font handling

Deprecated.

See pdf_get_parameter().

pdf_get_fontsize (PHP 4)

Deprecated: font handling

Deprecated.

See pdf_get_value().

pdf_get_image_height (PHP 3>= 3.0.12, PHP 4)

Returns height of an image

string **pdf_get_image_height** (int pdf object, int image) \linebreak

pdf_get_image_height() is deprecated, use pdf_get_value() instead.

pdf_get_image_width (PHP 3>= 3.0.12, PHP 4)

Returns width of an image

string **pdf_get_image_width** (int pdf object, int image) \linebreak

The **pdf_get_image_width()** is deprecated, use pdf_get_value() instead.

pdf_get_majorversion (PHP 4 >= 4.2.0)

Returns the major version number of the PDFlib

int **pdf_get_majorversion** (void) \linebreak

Returns the major version number of the PDFlib.

pdf_get_minorversion (PHP 4 >= 4.2.0)

Returns the minor version number of the PDFlib

int **pdf_get_majorversion** (void) \linebreak

Returns the minor version number of the PDFlib.

PDF_get_parameter (PHP 4 >= 4.0.1)

Gets certain parameters

string **pdf_get_parameter** (int pdf document, string name, double modifier) \linebreak

The **PDF_get_parameter()** function gets several parameters of pdflib which are of the type string. The function parameter *modifier* characterizes the parameter to get. If the modifier is not needed it has to be 0.

See also **PDF_get_value()**, **PDF_set_value()**, **PDF_set_parameter()**.

pdf_get_pdi_parameter (PHP 4 >= 4.0.5)

Get some PDI string parameters

string **pdf_get_pdi_parameter** (int pdf object, string key, int doc, int page, int index) \linebreak

Get the contents of some PDI document parameter with string type.

pdf_get_pdi_value (PHP 4 >= 4.0.5)

Gets some PDI numerical parameters

string **pdf_get_pdi_value** (int pdf object, string key, int doc, int page, int index) \linebreak

Get the contents of some PDI document parameter with numerical type.

PDF_get_value (PHP 4 >= 4.0.1)

Gets certain numerical value

double **pdf_get_value** (int pdf document, string name, double modifier) \linebreak

The **PDF_get_value()** function gets several numerical parameters of pdflib. The function parameter *modifier* characterizes the parameter to get. If the modifier is not needed it has to be 0.

See also **PDF_set_value()**, **PDF_get_parameter()**, **PDF_set_parameter()**.

pdf_initgraphics (PHP 4 >= 4.0.5)

Resets graphic state

void **pdf_initgraphics** (int pdf object) \linebreak

Reset all implicit color and graphics state parameters to their defaults.

PDF_lineto (PHP 3>= 3.0.6, PHP 4)

Draws a line

void **pdf_lineto** (int pdf document, double x-coor, double y-coor) \linebreak

The **PDF_lineto()** function draws a line from the current point to the point with coordinates (*x-coor*, *y-coor*).

See also **PDF_moveto()**, **PDF_curveto()**, **PDF_stroke()**.

pdf_makespotcolor (PHP 4 >= 4.0.5)

Makes a spotcolor

void **pdf_makespotcolor** (int pdf object, string spotname) \linebreak

Make a named spot color from the current color.

PDF_moveto (PHP 3>= 3.0.6, PHP 4)

Sets current point

void **pdf_moveto** (int pdf document, double x-coor, double y-coor) \linebreak

The **PDF_moveto()** function sets the current point to the coordinates *x-coor* and *y-coor*.

pdf_new (PHP 4 >= 4.0.5)

Creates a new pdf object

int **pdf_new** ( ) \linebreak

Create a new PDF object, using default error handling and memory management.

pdf_open_CCITT (PHP 4 >= 4.0.5)

Opens a new image file with raw CCITT data

int **pdf_open_CCITT** (int pdf object, string filename, int width, int height, int BitReverse, int k, int Blackls1) \linebreak

Open a raw CCITT image.

pdf_open_file (PHP 4 >= 4.0.5)

Opens a new pdf object

int **pdf_open_file** (int pdf object [, string filename]) \linebreak

Create a new PDF file using the supplied file name. If *filename* is empty the PDF document will be generated in memory instead of on file. The result must be fetched by the client with the pdf_get_buffer() function.

The following example shows how to create a pdf document in memory and how to output it correctly.

Ejemplo 1. Creating a PDF document in memory

```
<?php

$pdf = pdf_new();

pdf_open_file($pdf);
pdf_begin_page($pdf, 595, 842);
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "A PDF document created in memory!", 50, 750);
pdf_end_page($pdf);
pdf_close($pdf);
```

```
$data = pdf_get_buffer($pdf);

header("Content-type: application/pdf");
header("Content-disposition: inline; filename=test.pdf");
header("Content-length: " . strlen($data));

echo $data;

?>
```

PDF_open_gif (PHP 3>= 3.0.7, PHP 4)

Opens a GIF image

int **pdf_open_gif** (int pdf document, string filename) \linebreak

The **PDF_open_gif()** function opens an image stored in the file with the name *filename*. The format of the image has to be gif. The function returns a pdf image identifier.

Ejemplo 1. Including a gif image

```
<?php
$im = PDF_open_gif($pdf, "test.gif");
pdf_place_image($pdf, $im, 100, 100, 1);
pdf_close_image($pdf, $im);
?>
```

See also **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

pdf_open_image_file (PHP 3 CVS only, PHP 4)

Reads an image from a file

int **pdf_open_image_file** (int PDF-document, string imagetype, string filename [, string stringparam [, string intparam]]) \linebreak

Open an image file. Supported types are "jpeg", "tiff", "gif", and "png". *stringparam* is either "", "mask", "masked", or "page". *intparam*is either 0, the image id of the applied mask, or the page.

pdf_open_image (PHP 4 >= 4.0.5)

Versatile function for images

int **pdf_open_image** (int PDF-document, string imagetype, string source, string data, long length, int width, int height, int components, int bpc, string params) \linebreak

Use image data from a variety of data sources. Supported types are "jpeg", "ccitt", "raw". Supported sources are "memory", "fileref", "url". len is only used for type="raw", params is only used for type="ccitt".

PDF_open_jpeg (PHP 3>= 3.0.7, PHP 4)

Opens a JPEG image

int **pdf_open_jpeg** (int pdf document, string filename) \linebreak

The **PDF_open_jpeg()** function opens an image stored in the file with the name *filename*. The format of the image has to be jpeg. The function returns a pdf image identifier.

See also **PDF_close_image()**, **PDF_open_gif()**, **PDF_open_png()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_open_memory_image (PHP 3>= 3.0.10, PHP 4)

Opens an image created with PHP's image functions

int **pdf_open_memory_image** (int pdf document, int image) \linebreak

The **PDF_open_memory_image()** function takes an image created with the PHP's image functions and makes it available for the pdf document. The function returns a pdf image identifier.

Ejemplo 1. Including a memory image

```
<?php
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$pim = PDF_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

See also **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_png()** **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

pdf_open_pdi_page (PHP 4 >= 4.0.5)

Prepare a page

int **pdf_open_pdi_page** (int pdf object, int dochandle, int pagenumber, string pagelabel) \linebreak

Prepare a page for later use with pdf_place_image()

pdf_open_pdi (PHP 4 >= 4.0.5)

Opens a PDF file

int **pdf_open_pdi** (int pdf object, string filename, string stringparam, int intparam) \linebreak

Open an existing PDF document for later use.

PDF_open_png (PHP 4)

Opens a PNG image

int **pdf_open_png** (int pdf, string png_file) \linebreak

The **PDF_open_png()** function opens an image stored in the file with the name *filename*. The format of the image has to be png. The function returns a pdf image identifier.

Ejemplo 1. Including a PNG image

```
<?php
$im = PDF_open_png ($pdf, "test.png");
pdf_place_image ($pdf, $im, 100, 100, 1);
pdf_close_image ($pdf, $im);
?>
```

See also **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

pdf_open_tiff (PHP 4)

Deprecated: Opens a TIFF image

int **pdf_open_tiff** (int PDF-document, string filename) \linebreak

Deprecated.

See also pdf_open_image(),

PDF_open (PHP 3>= 3.0.6, PHP 4)

Opens a new pdf document

int **pdf_open** (int file, int info) \linebreak

The **PDF_open()** function opens a new pdf document. The corresponding file has to be opened with fopen() and the file descriptor passed as argument *`file`*. *`info`* is the info structure that has to be created with **pdf_get_info()**. The info structure will be deleted within this function.

Nota: The return value is needed as the first parameter in all other functions writing to the pdf document.

Nota: This function does not allow the second parameter if pdflib 2.0 support is activated.

See also fopen(), **PDF_get_info()**, **PDF_close()**.

PDF_place_image (PHP 3>= 3.0.7, PHP 4)

Places an image on the page

void **pdf_place_image** (int pdf document, int image, double x-coor, double y-coor, double scale) \linebreak

The **PDF_place_image()** function places an image on the page at postion (*`x-coor`*, *`x-coor`*). The image can be scaled at the same time.

See also **PDF_put_image()**.

pdf_place_pdi_page (PHP 4 >= 4.0.6)

Places an image on the page

void **pdf_place_pdi_page** (int pdf object, int page, float x, float y, float sx, float sy) \linebreak

Place a PDF page with the lower left corner at (*x*, *y*), and scale it.

PDF_rect (PHP 3>= 3.0.6, PHP 4)

Draws a rectangle

void **pdf_rect** (int pdf document, double x-coor, double y-coor, double width, double height) \linebreak

The **PDF_rect()** function draws a rectangle with its lower left corner at point (*x-coor*, *y-coor*). This width is set to *widgth*. This height is set to *height*.

See also **PDF_stroke()**.

PDF_restore (PHP 3>= 3.0.6, PHP 4)

Restores formerly saved environment

void **pdf_restore** (int pdf document) \linebreak

The **PDF_restore()** function restores the environment saved with **PDF_save()**. It works like the postscript command grestore.

Ejemplo 1. Save and Restore

```
<?php PDF_save($pdf);
// do all kinds of rotations, transformations, ...
PDF_restore($pdf) ?>
```

See also **PDF_save()**.

PDF_rotate (PHP 3>= 3.0.6, PHP 4)

Sets rotation

void **pdf_rotate** (int pdf document, double angle) \linebreak

The **PDF_rotate()** function sets the rotation in degress to *angle*.

PDF_save (PHP 3>= 3.0.6, PHP 4)

Saves the current environment

void **pdf_save** (int pdf document) \linebreak

The **PDF_save()** function saves the current environment. It works like the postscript command gsave. Very useful if you want to translate or rotate an object without effecting other objects. **PDF_save()** should always be followed by **PDF_restore()** to restore the environment before **PDF_save()**.

See also **PDF_restore()**.

PDF_scale (PHP 3>= 3.0.6, PHP 4)

Sets scaling

void **pdf_scale** (int pdf document, double x-scale, double y-scale) \linebreak

The **PDF_scale()** function sets the scaling factor in both directions. The following example scales x and y direction by 72. The following line will therefore be drawn one inch in both directions.

Ejemplo 1. Scaling

```
<?php PDF_scale($pdf, 72.0, 72.0);
PDF_lineto($pdf, 1, 1);
PDF_stroke($pdf);
?>
```

PDF_set_border_color (PHP 3>= 3.0.12, PHP 4)

Sets color of border around links and annotations

void **pdf_set_border_color** (int pdf document, double red, double green, double blue) \linebreak

The **PDF_set_border_color()** function sets the color of the suroundig box of links and annotations. The three color components have to have a value between 0.0 and 1.0.

See also **PDF_set_border_style()**, **PDF_set_border_dash()**.

PDF_set_border_dash (PHP 4 >= 4.0.1)

Sets dash style of border around links and annotations

void **pdf_set_border_dash** (int pdf document, double black, double white) \linebreak

The **PDF_set_border_dash()** function sets the lenght of black and white areas of a dashed line of the suroundig box of links and annotations.

See also **PDF_set_border_style()**, **PDF_set_border_color()**.

PDF_set_border_style (PHP 3>= 3.0.12, PHP 4)

Sets style of border around links and annotations

void **pdf_set_border_style** (int pdf document, string style, double width) \linebreak

The **PDF_set_border_style()** function sets the style and width of the suroundig box of links and annotations. The parameter `style` can be 'solid' or 'dashed'.

See also **PDF_set_border_color()**, **PDF_set_border_dash()**.

PDF_set_char_spacing (PHP 3>= 3.0.6, PHP 4)

Sets character spacing

void **pdf_set_char_spacing** (int pdf document, double space) \linebreak

The **PDF_set_char_spacing()** function sets the spacing between characters.

See also **PDF_set_word_spacing()**, **PDF_set_leading()**.

PDF_set_duration (PHP 3>= 3.0.6, PHP 4)

Sets duration between pages

void **pdf_set_duration** (int pdf document, double duration) \linebreak

The **PDF_set_duration()** function set the duration between following pages in seconds.

See also **PDF_set_transition()**.

PDF_set_font (PHP 3>= 3.0.6, PHP 4)

Selects a font face and size

void **pdf_set_font** (int pdf document, string font name, double size, string encoding [, int embed]) \linebreak

The **PDF_set_font()** function sets the current font face, font size and encoding. If you use pdflib 0.6 you will need to provide the Adobe Font Metrics (afm-files) for the font in the font path (default is ./fonts). If you use php3 or a version of pdflib older than 2.20 the fourth parameter *encoding* can take the following values: 0 = builtin, 1 = pdfdoc, 2 = macroman, 3 = macexpert, 4 = winansi. An encoding greater than 4 and less than 0 will default to winansi. winansi is often a good choice. If you use php4 and a version of pdflib >= 2.20 the encoding parameter has changed to a string. Use 'winansi', 'builtin', 'host', 'macroman' etc. instead. If the last parameter is set to 1 the font is embedded into the pdf document otherwise it is not. To embed a font is usually a good idea if the font is not widely spread and you cannot ensure that the person watching your document has access on fonts in the document. I font is only embedded once even if you call **PDF_set_font()** several times.

Nota: This function has to be called after **PDF_begin_page()** in order to create a valid pdf document.

Nota: If you reference a font in a .upr file make sure the name in the afm file and the font name are the same. Otherwise, the font will be embedded several times (Thanks to Paul Haddon for finding this.)

PDF_set_horiz_scaling (PHP 3>= 3.0.6, PHP 4)

Sets horizontal scaling of text

void **pdf_set_horiz_scaling** (int pdf document, double scale) \linebreak

The **PDF_set_horiz_scaling()** function sets the horizontal scaling to *scale* percent.

pdf_set_info_author (PHP 3>= 3.0.6, PHP 4)

Fills the author field of the document

bool **pdf_set_info_author** (int pdfdoc, string author) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_creator (PHP 3>= 3.0.6, PHP 4)

Fills the creator field of the document

bool **pdf_set_info_creator** (int pdfdoc, string creator) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_keywords (PHP 3>= 3.0.6, PHP 4)

Fills the keywords field of the document

bool **pdf_set_info_keywords** (int pdfdoc, string keywords) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_subject (PHP 3>= 3.0.6, PHP 4)

Fills the subject field of the document

bool **pdf_set_info_subject** (int pdfdoc, string subject) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_title (PHP 3>= 3.0.6, PHP 4)

Fills the title field of the document

bool **pdf_set_info_title** (int pdfdoc, string title) \linebreak

This function is deprecate, use pdf_set_info() instead.

PDF_set_info (PHP 4 >= 4.0.1)

Fills a field of the document information

void **pdf_set_info** (int pdf document, string fieldname, string value) \linebreak

The **PDF_set_info()** function sets an information field of a pdf document. Possible values for the fieldname are 'Subject', 'Title', 'Creator', 'Author', 'Keywords' and one user-defined name. It can be called before beginning a page.

Ejemplo 1. Setting document information

```
<?php
$fd = fopen("test.pdf", "w");
$pdfdoc = pdf_open($fd);
pdf_set_info($pdfdoc, "Author", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Creator", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Title", "Testing Info Fields");
pdf_set_info($pdfdoc, "Subject", "Test");
pdf_set_info($pdfdoc, "Keywords", "Test, Fields");
pdf_set_info($pdfdoc, "CustomField", "What ever makes sense");
```

```
pdf_begin_page($pdfdoc, 595, 842);
pdf_end_page($pdfdoc);
pdf_close($pdfdoc);
?>
```

Nota: This function replaces **PDF_set_info_keywords()**, **PDF_set_info_title()**, **PDF_set_info_subject()**, **PDF_set_info_creator()**, **PDF_set_info_sybject()**.

PDF_set_leading (PHP 3>= 3.0.6, PHP 4)

Sets distance between text lines

void **pdf_set_leading** (int pdf document, double distance) \linebreak

The **PDF_set_leading()** function sets the distance between text lines. This will be used if text is output by **PDF_continue_text()**.

See also **PDF_continue_text()**.

PDF_set_parameter (PHP 4)

Sets certain parameters

void **pdf_set_parameter** (int pdf document, string name, string value) \linebreak

The **PDF_set_parameter()** function sets several parameters of pdflib which are of the type string.

See also **PDF_get_value()**, **PDF_set_value()**, **PDF_get_parameter()**.

PDF_set_text_matrix (PHP 3>= 3.0.6)

Sets the text matrix

void **pdf_set_text_matrix** (int pdf document, array matrix) \linebreak

The **PDF_set_text_matrix()** function sets a matrix which describes a transformation applied on the current text font. The matrix has to passed as an array with six elements.

PDF_set_text_pos (PHP 3>= 3.0.6, PHP 4)

Sets text position

void **pdf_set_text_pos** (int pdf document, double x-coor, double y-coor) \linebreak

The **PDF_set_text_pos()** function sets the position of text for the next pdf_show() function call.

See also **PDF_show()**, **PDF_show_xy()**.

PDF_set_text_rendering (PHP 3>= 3.0.6, PHP 4)

Determines how text is rendered

void **pdf_set_text_rendering** (int pdf document, int mode) \linebreak

The **PDF_set_text_rendering()** function determines how text is rendered. The possible values for *mode* are 0=fill text, 1=stroke text, 2=fill and stroke text, 3=invisible, 4=fill text and add it to cliping path, 5=stroke text and add it to clipping path, 6=fill and stroke text and add it to cliping path, 7=add it to clipping path.

PDF_set_text_rise (PHP 3>= 3.0.6, PHP 4)

Sets the text rise

void **pdf_set_text_rise** (int pdf document, double rise) \linebreak

The **PDF_set_text_rise()** function sets the text rising to *rise* points.

PDF_set_value (PHP 4 >= 4.0.1)

Sets certain numerical value

void **pdf_set_value** (int pdf document, string name, double value) \linebreak

The **PDF_set_value()** function sets several numerical parameters of pdflib.

See also **PDF_get_value()**, **PDF_get_parameter()**, **PDF_set_parameter()**.

PDF_set_word_spacing (PHP 3>= 3.0.6, PHP 4)

Sets spacing between words

void **pdf_set_word_spacing** (int pdf document, double space) \linebreak

The **PDF_set_word_spacing()** function sets the spacing between words.

See also **PDF_set_char_spacing()**, **PDF_set_leading()**.

pdf_setcolor (PHP 4 >= 4.0.5)

Sets fill and stroke color

void **pdf_setcolor** (int pdf object, string type, string colorspace, float c1 [, float c2 [, float c3 [, float c4]]]) \linebreak

Set the current color space and color. The parameter *type* can be "fill", "stroke", or "both" to specify that the color is set for filling, stroking or both filling and stroking. The parameter *colorspace* can be gray, rgb, cmyk, spot or pattern. The parameters *c1*, *c2*, *c3* and *c4* represent the color components for the color space specified by *colorspace*. Except as otherwise noted, the color components are floating-point values that range from 0 to 1.

For gray only *c1* is used.

For rgb parameters *c1*, *c2*, and *c3* specify the red, green and blue values respectively.

```
// Set fill and stroke colors to white.
pdf_setcolor($pdf, "both", "rgb", 1, 1, 1);
```

For cmyk, parameters *c1*, *c2*, *c3*, and *c4* are the cyan, magenta, yellow and black values, respectively.

```
// Set fill and stroke colors to white.
pdf_setcolor($pdf, "both", "cmyk", 0, 0, 0, 1);
```

For spot, *c1* should be a spot color handles returned by pdf_makespotcolor() and *c2* is a tint value between 0 and 1.

For pattern, *c1* should be a pattern handle returned by pdf_begin_pattern().

PDF_setdash (PHP 3>= 3.0.6, PHP 4)

Sets dash pattern

void **pdf_setdash** (int pdf document, double white, double black) \linebreak

The **PDF_setdash()** function sets the dash pattern *white* white points and *black* black points. If both are 0 a solid line is set.

PDF_setflat (PHP 3>= 3.0.6, PHP 4)

Sets flatness

void **pdf_setflat** (int pdf document, double value) \linebreak

The **PDF_setflat()** function sets the flatness to a value between 0 and 100.

pdf_setfont (PHP 4 >= 4.0.5)

Set the current font

void **pdf_setfont** (int pdf object, int font, float size) \linebreak

Set the current font in the given size, using a *font* handle returned by pdf_findfont()

See Also: pdf_findfont().

PDF_setgray_fill (PHP 3>= 3.0.6, PHP 4)

Sets filling color to gray value

void **pdf_setgray_fill** (int pdf document, double gray value) \linebreak

The **PDF_setgray_fill()** function sets the current gray value to fill a path.

See also **PDF_setrgbcolor_fill()**.

PDF_setgray_stroke (PHP 3>= 3.0.6, PHP 4)

Sets drawing color to gray value

void **pdf_setgray_stroke** (int pdf document, double gray value) \linebreak

The **PDF_setgray_stroke()** function sets the current drawing color to the given gray value.

See also **PDF_setrgbcolor_stroke()**.

PDF_setgray (PHP 3>= 3.0.6, PHP 4)

Sets drawing and filling color to gray value

void **pdf_setgray** (int pdf document, double gray value) \linebreak

The **PDF_setgray()** function sets the current drawing and filling color to the given gray value.

See also **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

PDF_setlinecap (PHP 3>= 3.0.6, PHP 4)

Sets linecap parameter

void **pdf_setlinecap** (int pdf document, int value) \linebreak

The **PDF_setlinecap()** function sets the linecap parameter between a value of 0 and 2.

PDF_setlinejoin (PHP 3>= 3.0.6, PHP 4)

Sets linejoin parameter

void **pdf_setlinejoin** (int pdf document, long value) \linebreak

The **PDF_setlinejoin()** function sets the linejoin parameter between a value of 0 and 2.

PDF_setlinewidth (PHP 3>= 3.0.6, PHP 4)

Sets line width

void **pdf_setlinewidth** (int pdf document, double width) \linebreak

The **PDF_setlinewidth()** function sets the line width to *`width`*.

pdf_setmatrix (PHP 4 >= 4.0.5)

Sets current transformation matrix

void **pdf_setmatrix** (int pdf object, float a, float b, float c, float d, float e, float f) \linebreak

Explicitly set the current transformation matrix.

PDF_setmiterlimit (PHP 3>= 3.0.6, PHP 4)

Sets miter limit

void **pdf_setmiterlimit** (int pdf document, double value) \linebreak

The **PDF_setmiterlimit()** function sets the miter limit to a value greater of equal than 1.

pdf_setpolydash (PHP 4 >= 4.0.5)

Sets complicated dash pattern

void **pdf_setpolydash** (int pdf object, float * dasharray) \linebreak

Set a more complicated dash pattern defined by an array.

PDF_setrgbcolor_fill (PHP 3>= 3.0.6, PHP 4)

Sets filling color to rgb color value

void **pdf_setrgbcolor_fill** (int pdf document, double red value, double green value, double blue value) \linebreak

The **PDF_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

See also **PDF_setrgbcolor_fill()**.

PDF_setrgbcolor_stroke (PHP 3>= 3.0.6, PHP 4)

Sets drawing color to rgb color value

void **pdf_setrgbcolor_stroke** (int pdf document, double red value, double green value, double blue value) \linebreak

The **PDF_setrgbcolor_stroke()** function sets the current drawing color to the given rgb color value.

See also **PDF_setrgbcolor_stroke()**.

PDF_setrgbcolor (PHP 3>= 3.0.6, PHP 4)

Sets drawing and filling color to rgb color value

void **pdf_setrgbcolor** (int pdf document, double red value, double green value, double blue value) \linebreak

The **PDF_setrgbcolor_stroke()** function sets the current drawing and filling color to the given rgb color value.

See also **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

PDF_show_boxed (PHP 4)

Output text in a box

int **pdf_show_boxed** (int pdf document, string text, double x-coor, double y-coor, double width, double height, string mode) \linebreak

The **PDF_show_boxed()** function outputs the string *text* in a box with its lower left position at (*x-coor*, *y-coor*). The boxes dimension is *height* by *width*. The parameter *mode* determines how the text is type set. If *width* and *height* are zero, the *mode* can be "left", "right" or "center". If *width* or *height* is unequal zero it can also be "justify" and "fulljustify".

Returns the number of characters that could not be processed because they did not fit into the box.

See also **PDF_show()**, **PDF_show_xy()**.

PDF_show_xy (PHP 3>= 3.0.6, PHP 4)

Output text at given position

void **pdf_show_xy** (int pdf document, string text, double x-coor, double y-coor) \linebreak

The **PDF_show_xy()** function outputs the string *text* at position (*x-coor*, *y-coor*).

See also **PDF_show()**.

PDF_show (PHP 3>= 3.0.6, PHP 4)

Output text at current position

void **pdf_show** (int pdf document, string text) \linebreak

The **PDF_show()** function outputs the string *text* at the current position using the current font.

See also **PDF_show_xy()**, **PDF_set_text_pos()**, **PDF_set_font()**.

PDF_skew (PHP 4)

Skews the coordinate system

void **pdf_skew** (int pdf document, double alpha, double beta) \linebreak

The **PDF_skew()** function skew the coordinate system by *alpha* (x) and *beta* (y) degrees. *alpha* and *beta* may not be 90 or 270 degrees.

PDF_stringwidth (PHP 3>= 3.0.6, PHP 4)

Returns width of text using current font

double **pdf_stringwidth** (int pdf document, string text) \linebreak

The **PDF_stringwidth()** function returns the width of the string in *text* by using the current font. It requires a font to be set before with **PDF_set_font()**.

See also **PDF_set_font()**.

PDF_stroke (PHP 3>= 3.0.6, PHP 4)

Draws line along path

void **pdf_stroke** (int pdf document) \linebreak

The **PDF_stroke()** function draws a line along current path. The current path is the sum of all line drawing. Without this function the line would not be drawn.

See also **PDF_closepath()**, **PDF_closepath_stroke()**.

PDF_translate (PHP 3>= 3.0.6, PHP 4)

Sets origin of coordinate system

void **pdf_translate** (int pdf document, double x-coor, double y-coor) \linebreak

The **PDF_translate()** function sets the origin of coordinate system to the point (*x-coor*, *y-coor*) relativ the current origin. The following example draws a line from (0, 0) to (200, 200) relative to the initial coordinate system. You have to set the current point after **PDF_translate()** and before you start drawing more objects.

Ejemplo 1. Translation

```
<?php PDF_moveto($pdf, 0, 0);
PDF_lineto($pdf, 100, 100);
PDF_stroke($pdf);
PDF_translate($pdf, 100, 100);
PDF_moveto($pdf, 0, 0);
PDF_lineto($pdf, 100, 100);
```

```
PDF_stroke($pdf);
?>
```

LXXVII. Verisign Payflow Pro functions

This extension allows you to process credit cards and other financial transactions using Verisign Payment Services, formerly known as Signio (http://www.verisign.com/products/payflow/pro/index.html).

These functions are only available if PHP has been compiled with the `--with-pfpro[=DIR]` option. You will require the appropriate SDK for your platform, which may be downloaded from within the manager interface (https://manager.verisign.com/) once you have registered.

Once you have downloaded the SDK you should copy the files from the `lib` directory of the distribution. Copy the header file `pfpro.h` to `/usr/local/include` and the library file `libpfpro.so` to `/usr/local/lib`.

When using these functions, you may omit calls to pfpro_init() and pfpro_cleanup() as this extension will do so automatically if required. However the functions are still available in case you are processing a number of transactions and require fine control over the library. You may perform any number of transactions using pfpro_process() between the two.

These functions have been added in PHP 4.0.2.

> **Nota:** These functions only provide a link to Verisign Payment Services. Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

pfpro_cleanup (PHP 4 >= 4.0.2)

Shuts down the Payflow Pro library

void **pfpro_cleanup** (void) \linebreak

pfpro_cleanup() is used to shutdown the Payflow Pro library cleanly. It should be called after you have processed any transactions and before the end of your script. However you may omit this call, in which case this extension will automatically call **pfpro_cleanup()** after your script terminates.

See also pfpro_init().

pfpro_init (PHP 4 >= 4.0.2)

Initialises the Payflow Pro library

void **pfpro_init** (void) \linebreak

pfpro_init() is used to initialise the Payflow Pro library. You may omit this call, in which case this extension will automatically call **pfpro_init()** before the first transaction.

See also pfpro_cleanup().

pfpro_process_raw (PHP 4 >= 4.0.2)

Process a raw transaction with Payflow Pro

string **pfpro_process_raw** (string parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]]) \linebreak

Returns: A string containing the response.

pfpro_process_raw() processes a raw transaction string with Payflow Pro. You should really use pfpro_process() instead, as the encoding rules of these transactions are non-standard.

The first parameter in this case is a string containing the raw transaction request. All other parameters are the same as with pfpro_process(). The return value is a string containing the raw response.

> **Nota:** Be sure to read the Payflow Pro Developers Guide for full details of the required parameters and encoding rules. You would be well advised to use pfpro_process() instead.

Ejemplo 1. Payflow Pro raw example

```
<?php

pfpro_init();
```

```
$response = pfpro_process("USER=mylogin&PWD[5]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=411111

if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign raw response was ".$response;

pfpro_cleanup();

?>
```

pfpro_process (PHP 4 >= 4.0.2)

Process a transaction with Payflow Pro

array **pfpro_process** (array parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]]) \linebreak

Returns: An associative array containing the response

pfpro_process() processes a transaction with Payflow Pro. The first parameter is an associative array containing keys and values that will be encoded and passed to the processor.

The second parameter is optional and specifies the host to connect to. By default this is "test.signio.com", so you will certainly want to change this to "connect.signio.com" in order to process live transactions.

The third parameter specifies the port to connect on. It defaults to 443, the standard SSL port.

The fourth parameter specifies the timeout to be used, in seconds. This defaults to 30 seconds. Note that this timeout appears to only begin once a link to the processor has been established and so your script could potentially continue for a very long time in the event of DNS or network problems.

The fifth parameter, if required, specifies the hostname of your SSL proxy. The sixth parameter specifies the port to use.

The seventh and eighth parameters specify the logon identity and password to use on the proxy.

The function returns an associative array of the keys and values in the response.

Nota: Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

Ejemplo 1. Payflow Pro example

```
<?php

pfpro_init();
```

```
$transaction = array(USER => 'mylogin',
     PWD => 'mypassword',
     TRXTYPE => 'S',
     TENDER => 'C',
     AMT => 1.50,
     ACCT => '4111111111111111',
     EXPDATE => '0904'
     );

$response = pfpro_process($transaction);

if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign response code was ".$response[RESULT];
echo ", which means: ".$response[RESPMSG]."\n";

echo "\nThe transaction request: ";
print_r($transaction);

echo "\nThe response: ";
print_r($response);

pfpro_cleanup();

?>
```

pfpro_version (PHP 4 >= 4.0.2)

Returns the version of the Payflow Pro software

string **pfpro_version** (void) \linebreak

pfpro_version() returns the version string of the Payflow Pro library. At the time of writing, this was L211.

LXXVIII. opciones e información de PHP

assert_options (PHP 4)

Set/get the various assert flags

mixed **assert_options** (int what [, mixed value]) \linebreak

Using **assert_options()** you may set the various assert() control options or just query their current settings.

Tabla 1. Assert Options

option	ini-parameter	default	description
ASSERT_ACTIVE	assert.active	1	enable assert() evaluation
ASSERT_WARNING	assert.warning	1	issue a PHP warning for each failed assertion
ASSERT_BAIL	assert.bail	0	terminate execution on failed assertions
ASSERT_QUIET_EVAL	assert.quiet_eval	0	disable error_reporting during assertion expression evaluation
ASSERT_CALLBACK	assert_callback	(NULL)	user function to call on failed assertions

assert_options() will return the original setting of any option or FALSE on errors.

assert (PHP 4)

Checks if assertion is FALSE

int **assert** (string|bool assertion) \linebreak

assert() will check the given *assertion* and take appropriate action if its result is FALSE.

If the *assertion* is given as a string it will be evaluated as PHP code by **assert()**. The advantages of a string *assertion* are less overhead when assertion checking is off and messages containing the *assertion* expression when an assertion fails.

Assertions should be used as a debugging feature only. You may use them for sanity-checks that test for conditions that should always be TRUE and that indicate some programming errors if not or to check for the presence of certain features like extension functions or certain system limits and features.

Assertions should not be used for normal runtime operations like input parameter checks. As a rule of thumb your code should always be able to work correctly if assertion checking is not activated.

The behavior of **assert()** may be configured by assert_options() or by .ini-settings described in that functions manual page.

The assert_options() function and/or ASSERT_CALLBACK configuration directive allow a callback function to be set to handle failed assertions.

assert() callbacks are particularly useful for building automated test suites because they allow you to easily capture the code passed to the assertion, along with information on where the assertion was made. While this information can be captured via other methods, using assertions makes it much faster and easier!

The callback function should accept three arguments. The first argument will contain the file the assertion failed in. The second argument will contain the line the assertion failed on and the third argument will contain the expression that failed (if any - literal values such as 1 or "two" will not be passed via this argument)

Ejemplo 1. Handle a failed assertion with a custom handler

```
<?php
// Active assert and make it quiet
assert_options (ASSERT_ACTIVE, 1);
assert_options (ASSERT_WARNING, 0);
assert_options (ASSERT_QUIET_EVAL, 1);

// Create a handler function
function my_assert_handler ($file, $line, $code) {
    echo "<hr>Assertion Failed:
        File '$file'<br>
        Line '$line'<br>
        Code '$code'<br><hr>";
}

// Set up the callback
assert_options (ASSERT_CALLBACK, 'my_assert_handler');

// Make an assertion that should fail
assert ('mysql_query ("")');
?>
```

dl (PHP 3, PHP 4)

Loads a PHP extension at runtime

bool **dl** (string library) \linebreak

Loads the PHP extension given by the parameter *`library`*. The *`library`* parameter is *only* the filename of the extension to load which also depends on your platform. For example, the sockets extension (if compiled as a shared module, not the default!) would be called `sockets.so` on unix platforms whereas it is called `php_sockets.dll` on the windows platform.

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo. If the functionality of loading modules is not available (see Note) or has been disabled (either by turning it off `enable_dl` or by enabling `safe_mode` in `php.ini`) an `E_ERROR` is emitted and execution is stopped. If **dl()** fails because the specified library couldn't be loaded, in addition to `FALSE` an `E_WARNING` message is emitted.

Use extension_loaded() to test whether a given extension is already available or not. This works on both built-in extensions and dynamically loaded ones (either through `php.ini` or **dl()**).

Example:

```
if (!extension_loaded('gd')) {
    if (!dl('gd.so')) {
        exit;
    }
}
```

The directory where the extension is loaded from depends on your platform:

Windows - If not explicitly set in the `php.ini`, the extension is loaded from `c:\php4\extensions\` by default.

Unix - If not explicitly set in the `php.ini`, the default extension directory depends on

- whether PHP has been built with `--enable-debug` or not
- whether PHP has been built with (experimental) ZTS (Zend Thread Safety) support or not
- the current internal `ZEND_MODULE_API_NO` (Zend internal module API number, which is basically the date on which a major module API change happened, e.g. `20010901`)

Taking into account the above, the directory then defaults to `<php-install-directory>/lib/php/extension/<debug-or-not>-<zts-or-not>-ZEND_MODULE_API_NO`, e.g. `/usr/local/php/lib/php/extensions/debug-non-zts-20010901` or `/usr/local/php/lib/php/extensions/no-debug-zts-20010901`.

> **Nota: dl()** is *not* supported in multithreaded Web servers. Use the `extensions` statement in your `php.ini` when operating under such an environment. However, the `CGI` and `CLI` build are **not** affected !

> **Nota: dl()** is case sensitive on unix platforms.

See also Extension Loading Directives and extension_loaded().

extension_loaded (PHP 3>= 3.0.10, PHP 4)

averigua si una extensión ha sido cargada

bool **extension_loaded** (string name) \linebreak

Devuelve `TRUE` si la extensión identificada por *`name`* (nombre) está cargada. Puede ver el nombre de varias extensiones utilizando phpinfo().

Véase también phpinfo().

Nota: Esta función fue añadida en 3.0.10.

get_cfg_var (PHP 3, PHP 4)

Obtiene el valor de una opción de configuración de PHP.

string **get_cfg_var** (string varname) \linebreak

Devuelve el valor actual de una variable de configuración de PHP especificada en *`varname`*, o `FALSE` si ocurre un error.

No devolverá información de la configuracion cuando el PHP fue compilado, o leído desde un fichero de configuración Apache (utilizando las directivas php3_configuration_option directives).

Para comprobar si el sistema está utilizando un fichero de configuración, intente recuperar el valor de cfg_file_path. Si está disponible, se está utilizando un fichero de configuración.

get_current_user (PHP 3, PHP 4)

Obtiene el nombre del propietario del script PHP actual.

string **get_current_user** (void) \linebreak

Devuelve el nombre del propietario del script PHP actual.

Véase también getmyuid(), getmypid(), getmyinode(), y getlastmod().

get_defined_constants (PHP 4 >= 4.1.0)

Returns an associative array with the names of all the constants and their values

array **get_defined_constants** (void) \linebreak

This function returns the names and values of all the constants currently defined. This includes those created by extensions as well as those created with the define() function.

For example the line below

```
print_r (get_defined_constants());
```

will print a list like:

```
Array
(
    [E_ERROR] => 1
    [E_WARNING] => 2
    [E_PARSE] => 4
    [E_NOTICE] => 8
    [E_CORE_ERROR] => 16
    [E_CORE_WARNING] => 32
    [E_COMPILE_ERROR] => 64
    [E_COMPILE_WARNING] => 128
    [E_USER_ERROR] => 256
    [E_USER_WARNING] => 512
    [E_USER_NOTICE] => 1024
    [E_ALL] => 2047
    [TRUE] => 1
)
```

See also get_loaded_extensions(), get_defined_functions() and get_defined_vars().

get_extension_funcs (PHP 4)

Returns an array with the names of the functions of a module

array **get_extension_funcs** (string module_name) \linebreak

This function returns the names of all the functions defined in the module indicated by *module_name*.

For example the lines below

```
print_r (get_extension_funcs ("xml"));
print_r (get_extension_funcs ("gd"));
```

will print a list of the functions in the modules `xml` and `gd` respectively.

See also: get_loaded_extensions()

get_included_files (PHP 4)

Returns an array with the names of included or required files

array **get_included_files** (void) \linebreak

Returns an array of the names of all files that have been included using include(), include_once(), require() or require_once().

Files that are included or required multiple times only show up once in the returned array.

Nota: Files included using the `auto_prepend_file` configuration directive are not included in the returned array.

Ejemplo 1. get_included_files() Example

```
<?php

include("test1.php");
include_once("test2.php");
require("test3.php");
require_once("test4.php");

$included_files = get_included_files();

foreach($included_files as $filename) {
    echo "$filename\n";
}

?>
```

will generate the following output:

```
test1.php
test2.php
test3.php
test4.php
```

Nota: In PHP 4.0.1pl2 and previous versions **get_included_files()** assumed that the required files ended in the extension `.php`; other extensions would not be returned. The array returned by

get_included_files() was an associative array and only listed files included by include() and include_once().

See also: include(), include_once(), require(), require_once(), and get_required_files().

get_loaded_extensions (PHP 4)

Returns an array with the names of all modules compiled and loaded

array **get_loaded_extensions** (void) \linebreak

This function returns the names of all the modules compiled and loaded in the PHP interpreter.

For example the line below

```
print_r (get_loaded_extensions());
```

will print a list like:

```
Array
(
    [0] => xml
    [1] => wddx
    [2] => standard
    [3] => session
    [4] => posix
    [5] => pgsql
    [6] => pcre
    [7] => gd
    [8] => ftp
    [9] => db
    [10] => calendar
    [11] => bcmath
)
```

See also: get_extension_funcs().

get_magic_quotes_gpc (PHP 3>= 3.0.6, PHP 4)

Obtiene el valor de la configuración activa actual de las comillas mágicas gpc.

long **get_magic_quotes_gpc** (void) \linebreak

Devuelve el valor de la configuración activa actual de magic_quotes_gpc. (0 desactivado, 1 activado)

Véase también get_magic_quotes_runtime(), set_magic_quotes_runtime().

get_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4)

Obtiene el valor de la configuración activa actual de magic_quotes_runtime.

long **get_magic_quotes_runtime** (void) \linebreak

Devuelve el valor de la configuración activa actual de magic_quotes_runtime. (0 desactivado, 1 activado)

Véase también get_magic_quotes_gpc(), set_magic_quotes_runtime().

get_required_files (PHP 4)

Returns an array with the names of included or required files

array **get_required_files** (void) \linebreak

As of PHP 4.0.4, this function is an alias for get_included_files()

In PHP 4.0.1pl2 and previous versions **get_required_files()** assumed that the required files ended in the extension `.php`, other extensions would not be returned. The array returned by **get_required_files()** was an associative array and only listed files included by require() and require_once().

See also: require(), require_once(), include(), include_once(), and get_included_files().

getenv (PHP 3, PHP 4)

Obtiene el valor de una variable de entorno

string **getenv** (string varname) \linebreak

Devuelve el valor de la variable de entorno *`varname`*, o `FALSE` en caso de error.

```
$ip = getenv("REMOTE_ADDR"); // get the ip number of the user
```

Puede ver una lista de todas las variables de entorno utilizando phpinfo(). Puede encontrar el significado de la mayoría echando un vistazo en CGI specification (especificación CGI) (http://hoohoo.ncsa.uiuc.edu/cgi/), especialmente en page on environmental variables (página de variables de entorno) (http://hoohoo.ncsa.uiuc.edu/cgi/env.html).

getlastmod (PHP 3, PHP 4)

Recupera la fecha/hora de la última modificación de la página.

int **getlastmod** (void) \linebreak

Devuelve la fecha/hora de la última modificación de la página actual. El valor devuelto está en formato de fecha/hora Unix, adecuado para que sirva a date(). Devuelve `FALSE` en caso de error.

Ejemplo 1. ejemplo getlastmod()

```
// outputs e.g. 'Last modified: March 04 1998 20:43:59.'
echo "Last modified: ".date( "F d Y H:i:s.", getlastmod() );
```

Véase también date(), getmyuid(), get_current_user(), getmyinode(), y getmypid().

getmygid (PHP 4 >= 4.1.0)

Get PHP script owner's GID

int **getmygid** (void) \linebreak

Returns the group ID of the current script, or `FALSE` on error.

See also getmyuid(), getmypid(), get_current_user(), getmyinode(), and getlastmod().

getmyinode (PHP 3, PHP 4)

Recupera el inodo del script actual.

int **getmyinode** (void) \linebreak

Devuelve el inodo del script actual, o `FALSE` en caso de error.

Véase también getmyuid(), get_current_user(), getmypid(), y getlastmod().

getmypid (PHP 3, PHP 4)

Obtiene el ID de proceso de PHP.

int **getmypid** (void) \linebreak

Devuelve el ID del proceso PHP actual, o `FALSE` en caso de error.

Advierta que cuando se ejecuta como un módulo de servidor, diferentes llamadas del script no garantizan que tengan distintos pids.

Véase también getmyuid(), get_current_user(), getmyinode(), y getlastmod().

getmyuid (PHP 3, PHP 4)

Obtiene el UID del propietario del script PHP.

int **getmyuid** (void) \linebreak

Devuelve el ID de usuario del script actual, o `FALSE` en caso de error.

Véase también getmypid(), get_current_user(), getmyinode(), y getlastmod().

getrusage (PHP 3>= 3.0.7, PHP 4)

Obtiene el consumo actual de recursos.

array **getrusage** ([int who]) \linebreak

Es un interface a getrusage(2). Devuelve un array asociativo que contiene los datos devueltos de la llamada del sistema. Si who (quien) es 1, getrusage debería llamarse con RUSAGE_CHILDREN. Todas las entradas son accesibles utilizando sus nombres de campo documentados.

Ejemplo 1. Ejemplo Getrusage

```
$dat = getrusage();
echo $dat["ru_nswap"];         # number of swaps
echo $dat["ru_majflt"];        # number of page faults
echo $dat["ru_utime.tv_sec"];  # user time used (seconds)
echo $dat["ru_utime.tv_usec"]; # user time used (microseconds)
```

Vea la página man de system para más detalles.

ini_alter (PHP 4)

Changes the value of a configuration option

string **ini_alter** (string varname, string newvalue) \linebreak

Changes the value of a configuration option, returns `FALSE` on failure, and the previous value of the configuration option on success.

Nota: This is an alias of ini_set()

See also ini_get(), ini_get_all(), ini_restore(), and ini_set()

ini_get_all (PHP 4 >= 4.2.0)

Gets all configuration options

array **ini_get_all** ([string extension]) \linebreak

Returns all the registered configuration options as an associative array. If optional *`extension`* parameter is set, returns only options specific for that extension.

See also: ini_alter(), ini_restore(), ini_get(), and ini_set()

ini_get (PHP 4)

Gets the value of a configuration option

string **ini_get** (string varname) \linebreak

Returns the value of the configuration option on success, an empty string on failure.

See also get_cfg_var(), ini_get_all(), ini_alter(), ini_restore(), and ini_set()

ini_restore (PHP 4)

Restores the value of a configuration option

string **ini_restore** (string varname) \linebreak

Restores a given configuration option to its original value.

See also: ini_alter(), ini_get(), ini_get_all(), and ini_set()

ini_set (PHP 4)

Sets the value of a configuration option

string **ini_set** (string varname, string newvalue) \linebreak

Sets the value of the given configuration option. Returns the old value on success, `FALSE` on failure. The configuration option will keep this new value during the script's execution, and will be restored at the script's ending.

Not all the available options can be changed using **ini_set()**. Below is a table with a list of all PHP options (as of PHP 4.2.0), indicating which ones can be changed/set and at what level.

Tabla 1. Configuration options

Name	Default	Changeable
com.allow_dcom	"0"	PHP_INI_SYSTEM
com.autoregister_typelib	"0"	PHP_INI_SYSTEM
com.autoregister_verbose	"0"	PHP_INI_SYSTEM
com.autoregister_casesensitive	"1"	PHP_INI_SYSTEM
com.typelib_file	""	PHP_INI_SYSTEM
crack.default_dictionary	NULL	PHP_INI_SYSTEM
exif.encode_unicode	"ISO-8859-15"	PHP_INI_ALL
exif.decode_unicode_motorola	"UCS-2BE"	PHP_INI_ALL
exif.decode_unicode_intel	"UCS-2LE"	PHP_INI_ALL
exif.encode_jis	""	PHP_INI_ALL
exif.decode_jis_motorola	"JIS"	PHP_INI_ALL
exif.decode_jis_intel	"JIS"	PHP_INI_ALL
fbsql.allow_persistent	"1"	PHP_INI_SYSTEM
fbsql.generate_warnings	"0"	PHP_INI_SYSTEM
fbsql.autocommit	"1"	PHP_INI_SYSTEM
fbsql.max_persistent	"-1"	PHP_INI_SYSTEM
fbsql.max_links	"128"	PHP_INI_SYSTEM
fbsql.max_connections	"128"	PHP_INI_SYSTEM
fbsql.max_results	"128"	PHP_INI_SYSTEM
fbsql.batchSize	"1000"	PHP_INI_SYSTEM
fbsql.default_host	NULL	PHP_INI_SYSTEM
fbsql.default_user	"_SYSTEM"	PHP_INI_SYSTEM
fbsql.default_password	""	PHP_INI_SYSTEM
fbsql.default_database	""	PHP_INI_SYSTEM
fbsql.default_database_password	""	PHP_INI_SYSTEM
hwapi.allow_persistent	"0"	PHP_INI_SYSTEM

Name	Default	Changeable
hyerwave.allow_persistent	"0"	PHP_INI_SYSTEM
hyperwave.default_port	"418"	PHP_INI_ALL
iconv.input_encoding	ICONV_INPUT_ENCODING	PHP_INI_ALL
iconv.output_encoding	ICONV_OUTPUT_ENCODING	PHP_INI_ALL
iconv.internal_encoding	ICONV_INTERNAL_ENCODING	PHP_INI_ALL
ifx.allow_persistent	"1"	PHP_INI_SYSTEM
ifx.max_persistent	"-1"	PHP_INI_SYSTEM
ifx.max_links	"-1"	PHP_INI_SYSTEM
ifx.default_host	NULL	PHP_INI_SYSTEM
ifx.default_user	NULL	PHP_INI_SYSTEM
ifx.default_password	NULL	PHP_INI_SYSTEM
ifx.blobinfile	"1"	PHP_INI_ALL
ifx.textasvarchar	"0"	PHP_INI_ALL
ifx.byteasvarchar	"0"	PHP_INI_ALL
ifx.charasvarchar	"0"	PHP_INI_ALL
ifx.nullformat	"0"	PHP_INI_ALL
ingres.allow_persistent	"1"	PHP_INI_SYSTEM
ingres.max_persistent	"-1"	PHP_INI_SYSTEM
ingres.max_links	"-1"	PHP_INI_SYSTEM
ingres.default_database	NULL	PHP_INI_ALL
ingres.default_user	NULL	PHP_INI_ALL
ingres.default_password	NULL	PHP_INI_ALL
ibase.allow_persistent	"1"	PHP_INI_SYSTEM
ibase.max_persistent	"-1"	PHP_INI_SYSTEM
ibase.max_links	"-1"	PHP_INI_SYSTEM
ibase.default_user	NULL	PHP_INI_ALL
ibase.default_password	NULL	PHP_INI_ALL
ibase.timestampformat	"%m/%d/%Y%H:%M:%S"	PHP_INI_ALL
ibase.dateformat	"%m/%d/%Y"	PHP_INI_ALL
ibase.timeformat	"%H:%M:%S"	PHP_INI_ALL
java.class.path	NULL	PHP_INI_ALL
java.home	NULL	PHP_INI_ALL
java.library.path	NULL	PHP_INI_ALL
java.library	JAVALIB	PHP_INI_ALL
java.library	NULL	PHP_INI_ALL
ldap.max_links	"-1"	PHP_INI_SYSTEM
mbstring.detect_order	NULL	PHP_INI_ALL
mbstring.http_input	NULL	PHP_INI_ALL

Name	Default	Changeable
mbstring.http_output	NULL	PHP_INI_ALL
mbstring.internal_encoding	NULL	PHP_INI_ALL
mbstring.substitute_character	NULL	PHP_INI_ALL
mbstring.func_overload	"0"	PHP_INI_SYSTEM
mcrypt.algorithms_dir	NULL	PHP_INI_ALL
mcrypt.modes_dir	NULL	PHP_INI_ALL
mime_magic.magicfile	"/usr/share/misc/magic.mime"	PHP_INI_SYSTEM
mssql.allow_persistent	"1"	PHP_INI_SYSTEM
mssql.max_persistent	"-1"	PHP_INI_SYSTEM
mssql.max_links	"-1"	PHP_INI_SYSTEM
mssql.max_procs	"25"	PHP_INI_ALL
mssql.min_error_severity	"10"	PHP_INI_ALL
mssql.min_message_severity	"10"	PHP_INI_ALL
mssql.compatability_mode	"0"	PHP_INI_ALL
mssql.connect_timeout	"5"	PHP_INI_ALL
mssql.timeout	"60"	PHP_INI_ALL
mssql.textsize	"-1"	PHP_INI_ALL
mssql.textlimit	"-1"	PHP_INI_ALL
mssql.batchsize	"0"	PHP_INI_ALL
mssql.datetimeconvert	"1"	PHP_INI_ALL
mysql.allow_persistent	"1"	PHP_INI_SYSTEM
mysql.max_persistent	"-1"	PHP_INI_SYSTEM
mysql.max_links	"-1"	PHP_INI_SYSTEM
mysql.default_host	NULL	PHP_INI_ALL
mysql.default_user	NULL	PHP_INI_ALL
mysql.default_password	NULL	PHP_INI_ALL
mysql.default_port	NULL	PHP_INI_ALL
mysql.default_socket	NULL	PHP_INI_ALL
ncurses.value	"42"	PHP_INI_ALL
ncurses.string	"foobar"	PHP_INI_ALL
odbc.allow_persistent	"1"	PHP_INI_SYSTEM
odbc.max_persistent	"-1"	PHP_INI_SYSTEM
odbc.max_links	"-1"	PHP_INI_SYSTEM
odbc.default_db	NULL	PHP_INI_ALL
odbc.default_user	NULL	PHP_INI_ALL
odbc.default_pw	NULL	PHP_INI_ALL
odbc.defaultlrl	"4096"	PHP_INI_ALL
odbc.defaultbinmode	"1"	PHP_INI_ALL

Name	Default	Changeable
odbc.check_persistent	"1"	PHP_INI_SYSTEM
pfpro.defaulthost	"test.signio.com"	
pfpro.defaulthost	"test-payflow.verisign.com"	
pfpro.defaultport	"443"	PHP_INI_ALL
pfpro.defaulttimeout	"30"	PHP_INI_ALL
pfpro.proxyaddress	""	PHP_INI_ALL
pfpro.proxyport	""	PHP_INI_ALL
pfpro.proxylogon	""	PHP_INI_ALL
pfpro.proxypassword	""	PHP_INI_ALL
pgsql.allow_persistent	"1"	PHP_INI_SYSTEM
pgsql.max_persistent	"-1"	PHP_INI_SYSTEM
pgsql.max_links	"-1"	PHP_INI_SYSTEM
pgsql.auto_reset_persistent	"0"	PHP_INI_SYSTEM
pgsql.ignore_notice	"0"	PHP_INI_ALL
pgsql.log_notice	"0"	PHP_INI_ALL
session.save_path	"/tmp"	PHP_INI_ALL
session.name	"PHPSESSID"	PHP_INI_ALL
session.save_handler	"files"	PHP_INI_ALL
session.auto_start	"0"	PHP_INI_ALL
session.gc_probability	"1"	PHP_INI_ALL
session.gc_maxlifetime	"1440"	PHP_INI_ALL
session.serialize_handler	"php"	PHP_INI_ALL
session.cookie_lifetime	"0"	PHP_INI_ALL
session.cookie_path	"/"	PHP_INI_ALL
session.cookie_domain	""	PHP_INI_ALL
session.cookie_secure	""	PHP_INI_ALL
session.use_cookies	"1"	PHP_INI_ALL
session.use_only_cookies	"0"	PHP_INI_ALL
session.referer_check	""	PHP_INI_ALL
session.entropy_file	""	PHP_INI_ALL
session.entropy_length	"0"	PHP_INI_ALL
session.cache_limiter	"nocache"	PHP_INI_ALL
session.cache_expire	"180"	PHP_INI_ALL
session.use_trans_sid	"1"	PHP_INI_ALL
session.encode_sources	"globals	track"
extname.global_value	"42"	PHP_INI_ALL
extname.global_string	"foobar"	PHP_INI_ALL
assert.active	"1"	PHP_INI_ALL

Name	Default	Changeable
assert.bail	"0"	PHP_INI_ALL
assert.warning	"1"	PHP_INI_ALL
assert.callback	NULL	PHP_INI_ALL
assert.quiet_eval	"0"	PHP_INI_ALL
safe_mode_protected_env_vars	SAFE_MODE_PROTECTED_ENV_VARS	PHP_INI_SYSTEM
safe_mode_allowed_env_vars	SAFE_MODE_ALLOWED_ENV_VARS	PHP_INI_SYSTEM
url_rewriter.tags	"a=href	area=href
url_rewriter.tags	"a=href	area=href
sybct.allow_persistent	"1"	PHP_INI_SYSTEM
sybct.max_persistent	"-1"	PHP_INI_SYSTEM
sybct.max_links	"-1"	PHP_INI_SYSTEM
sybct.min_server_severity	"10"	PHP_INI_ALL
sybct.min_client_severity	"10"	PHP_INI_ALL
sybct.hostname	NULL	PHP_INI_ALL
tokenizer.global_value	"42"	PHP_INI_ALL
tokenizer.global_string	"foobar"	PHP_INI_ALL
vpopmail.directory	""	PHP_INI_ALL
zlib.output_compression	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
zlib.output_compression_level	"-1"	PHP_INI_ALL
define_syslog_variables	"0"	PHP_INI_ALL
highlight.bg	HL_BG_COLOR	PHP_INI_ALL
highlight.comment	HL_COMMENT_COLOR	PHP_INI_ALL
highlight.default	HL_DEFAULT_COLOR	PHP_INI_ALL
highlight.html	HL_HTML_COLOR	PHP_INI_ALL
highlight.keyword	HL_KEYWORD_COLOR	PHP_INI_ALL
highlight.string	HL_STRING_COLOR	PHP_INI_ALL
allow_call_time_pass_reference	"1"	PHP_INI_SYSTEM|PHP_INI_PERDIR
asp_tags	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
display_errors	"1"	PHP_INI_ALL
display_startup_errors	"0"	PHP_INI_ALL
enable_dl	"1"	PHP_INI_SYSTEM
expose_php	"1"	PHP_INI_SYSTEM
html_errors	"1"	PHP_INI_SYSTEM
xmlrpc_errors	"0"	PHP_INI_SYSTEM

Name	Default	Changeable
xmlrpc_error_number	"0"	PHP_INI_ALL
ignore_user_abort	"0"	PHP_INI_ALL
implicit_flush	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
log_errors	"0"	PHP_INI_ALL
log_errors_max_len	"1024"	PHP_INI_ALL
ignore_repeated_errors	"0"	PHP_INI_ALL
ignore_repeated_source	"0"	PHP_INI_ALL
magic_quotes_gpc	"1"	PHP_INI_ALL
magic_quotes_runtime	"0"	PHP_INI_ALL
magic_quotes_sybase	"0"	PHP_INI_ALL
output_buffering	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
output_handler	NULL	PHP_INI_PERDIR|PHP_INI_SYSTEM
register_argc_argv	"1"	PHP_INI_ALL
register_globals	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
safe_mode	"1"	PHP_INI_SYSTEM
safe_mode	"0"	PHP_INI_SYSTEM
safe_mode_include_dir	NULL	PHP_INI_SYSTEM
safe_mode_gid	"0"	PHP_INI_SYSTEM
short_open_tag	DEFAULT_SHORT_OPEN_TAG	PHP_INI_SYSTEM|PHP_INI_PERDIR
sql.safe_mode	"0"	PHP_INI_SYSTEM
track_errors	"0"	PHP_INI_ALL
y2k_compliance	"0"	PHP_INI_ALL
unserialize_callback_func	NULL	PHP_INI_ALL
arg_separator.output	"&"	PHP_INI_ALL
arg_separator.input	"&"	PHP_INI_SYSTEM|PHP_INI_PERDIR
auto_append_file	NULL	PHP_INI_ALL
auto_prepend_file	NULL	PHP_INI_ALL
doc_root	NULL	PHP_INI_SYSTEM
default_charset	SAPI_DEFAULT_CHARSET	PHP_INI_ALL
default_mimetype	SAPI_DEFAULT_MIMETYPE	PHP_INI_ALL
error_log	NULL	PHP_INI_ALL
extension_dir	PHP_EXTENSION_DIR	PHP_INI_SYSTEM
gpc_order	"GPC"	PHP_INI_ALL

Name	Default	Changeable
include_path	PHP_INCLUDE_PATH	PHP_INI_ALL
max_execution_time	"30"	PHP_INI_ALL
open_basedir	NULL	PHP_INI_SYSTEM
safe_mode_exec_dir	"1"	PHP_INI_SYSTEM
upload_max_filesize	"2M"	PHP_INI_SYSTEM
file_uploads	"1"	PHP_INI_ALL
post_max_size	"8M"	PHP_INI_SYSTEM
upload_tmp_dir	NULL	PHP_INI_SYSTEM
user_dir	NULL	PHP_INI_SYSTEM
variables_order	NULL	PHP_INI_ALL
error_append_string	NULL	PHP_INI_ALL
error_prepend_string	NULL	PHP_INI_ALL
SMTP	"localhost"	PHP_INI_ALL
smtp_port	25	PHP_INI_ALL
browscap	NULL	PHP_INI_SYSTEM
error_reporting	NULL	PHP_INI_ALL
memory_limit	"8M"	PHP_INI_ALL
precision	"14"	PHP_INI_ALL
sendmail_from	NULL	PHP_INI_ALL
sendmail_path	DEFAULT_SENDMAIL_PATH	PHP_INI_SYSTEM
disable_functions	""	PHP_INI_SYSTEM
allow_url_fopen	"1"	PHP_INI_ALL
always_populate_raw_post_data	"0"	PHP_INI_ALL
xbithack	"0"	PHP_INI_ALL
engine	"1"	PHP_INI_ALL
last_modified	"0"	PHP_INI_ALL
child_terminate	"0"	PHP_INI_ALL
async_send	"0"	PHP_INI_ALL

Tabla 2. Definition of PHP_INI_* constants

Constant	Value	Meaning
PHP_INI_USER	1	Entry can be set in user scripts
PHP_INI_PERDIR	2	Entry can be set in `.htaccess`
PHP_INI_SYSTEM	4	Entry can be set in `php.ini` or `httpd.conf`
PHP_INI_ALL	7	Entry can be set anywhere

See also: ini_alter(), ini_get(), and ini_restore()

php_logo_guid (PHP 4)

Obtiene el guid logo

string **php_logo_guid** (void) \linebreak

Nota: Esta funcionalidad fue añadida en PHP4 Beta 4.

php_sapi_name (PHP 4 >= 4.0.1)

Returns the type of interface between web server and PHP

string **php_sapi_name** (void) \linebreak

php_sapi_name() returns a lowercase string which describes the type of interface between web server and PHP (Server API, SAPI). In CGI PHP, this string is "cgi", in mod_php for Apache, this string is "apache" and so on.

Ejemplo 1. php_sapi_name() Example

```
$sapi_type = php_sapi_name();
if ($sapi_type == "cgi")
    print "You are using CGI PHP\n";
else
    print "You are not using CGI PHP\n";
```

php_uname (PHP 4 >= 4.0.2)

Returns information about the operating system PHP was built on

string **php_uname** (void) \linebreak

php_uname() returns a string with a description of the operating system PHP is built on.

Ejemplo 1. php_uname() Example

```
if (substr(php_uname(), 0, 7) == "Windows") {
   die ("Sorry, this script doesn't run on Windows.\n");
}
```

phpcredits (PHP 4)

Prints out the credits for PHP

void **phpcredits** ([int flag]) \linebreak

This function prints out the credits listing the PHP developers, modules, etc. It generates the appropriate HTML codes to insert the information in a page. *flag* is optional, and it defaults to CREDITS_ALL. To generate a custom credits page, you may want to use the *flag* parameter. For example to print the general credits, you will use somewhere in your code:

```
...
phpcredits(CREDITS_GENERAL);
...
```

And if you want to print the core developers and the documentation group, in a page of its own, you will use:

```
<?php
phpcredits(CREDITS_GROUP + CREDITS_DOCS + CREDITS_FULLPAGE);
?>
```

And if you feel like embedding all the credits in your page, then code like the one below will do it:

```
<html>
 <head>
  <title>My credits page</title>
 </head>
```

```
 <body>
  <?php
 // some code of your own
 phpcredits(CREDITS_ALL);
 // some more code
 ?>
 </body>
</html>
```

Tabla 1. Pre-defined phpcredits() flags

name	description
CREDITS_ALL	All the credits, equivalent to using: CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE. It generates a complete stand-alone HTML page with the appropriate tags.
CREDITS_DOCS	The credits for the documentation team
CREDITS_FULLPAGE	Usually used in combination with the other flags. Indicates that the a complete stand-alone HTML page needs to be printed including the information indicated by the other flags.
CREDITS_GENERAL	General credits: Language design and concept, PHP 4.0 authors and SAPI module.
CREDITS_GROUP	A list of the core developers
CREDITS_MODULES	A list of the extension modules for PHP, and their authors
CREDITS_SAPI	A list of the server API modules for PHP, and their authors

See also: phpinfo(), phpversion(), and php_logo_guid().

phpinfo (PHP 3, PHP 4)

Recupera gran cantidad de información de PHP.

int **phpinfo** (void) \linebreak

Obtiene gran cantidad de información sobre el estado actual de PHP. Esto incluye información sobre las opciones de compilación y extensiones de PHP, la versión PHP, información y entorno del servidor (si

está compilado como un módulo), el entorno PHP, información sobre la versión del SO, rutas, opciones de configuración maestras y locales, cabeceras HTTP, y la Licencia Pública GNU.

Véase también phpversion().

phpversion (PHP 3, PHP 4)

Obtiene la versión actual de PHP.

string **phpversion** (void) \linebreak

Devuelve una cadena de carácteres que contiene la versión del parser PHP que está ejecutándose actualmente.

Ejemplo 1. ejemplo phpversion()

```
// prints e.g. 'Current PHP version: 3.0rel-dev'
echo "Current PHP version: ".phpversion();
```

Véase también phpinfo().

putenv (PHP 3, PHP 4)

Establece el valor de una variable de entorno.

void **putenv** (string setting) \linebreak

Añade *`setting (valor)`* al entorno.

Ejemplo 1. Establecer una Variable de Entorno

```
putenv("UNIQID=$uniqid");
```

set_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4)

Establece el valor de la configuración activa actual de magic_quotes_runtime.

long **set_magic_quotes_runtime** (int new_setting) \linebreak

Establece el valor de la configuración activa actual de magic_quotes_runtime. (0 desactivado, 1 activado)

Véase también get_magic_quotes_gpc(), get_magic_quotes_runtime().

set_time_limit (PHP 3, PHP 4)

limita el tiempo máximo de ejecución

void **set_time_limit** (int seconds) \linebreak

Establece el número de segundos que se le permite a un script ejecutarse. Si éste es alcanzado, el script devuelve un error de tipo fatal. El límite por defecto es 30 segundos o, si existe, el valor max_execution_time definido en el fichero de configuración. Si seconds (segundos) se establece a cero, no se impone ningún límite.

Cuando se llama, **set_time_limit()** reinicia el contador del timeout a cero. En otras palabras, si el timeout es el de por defecto de 30 segundos, y después de 25 segundos de ejecución del script se realiza una llamada set_time_limit(20), el script se ejecutará durante un total de 45 segundos antes de alcanzar su límite.

Advierta que **set_time_limit()** no tiene efecto cuando PHP se ejecuta en modo seguro (safe mode). No hay otra opción que que desactivar el modo seguro o cambiar el límite de tiempo en el fichero de configuración.

version_compare (PHP 4 >= 4.1.0)

Compares two "PHP-standardized" version number strings

int **version_compare** (string version1, string version2 [, string operator]) \linebreak

version_compare() compares two "PHP-standardized" version number strings. This is useful if you would like to write programs working only on some versions of PHP.

version_compare() returns -1 if the first version is lower than the second, 0 if they are equal, and +1 if the second is lower.

If you specify the third optional *operator* argument, you can test for a particular relationship. The possible operators are: `<`, `lt`, `<=`, `le`, `>`, `gt`, `>=`, `ge`, `==`, `=`, `eq`, `!=`, `<>`, `ne` respectively. Using this argument, the function will return 1 if the relationship is the one specified by the operator, 0 otherwise.

Ejemplo 1. version_compare() Example

```
// prints -1
echo version_compare("4.0.4", "4.0.6");

// these all print 1
echo version_compare("4.0.4", "4.0.6", "<");
echo version_compare("4.0.6", "4.0.6", "eq");
```

zend_logo_guid (PHP 4)

Obtiene el guid zend

string **zend_logo_guid** (void) \linebreak

Nota: Esta funcionalidad fue añadida en PHP4 Beta 4.

zend_version (PHP 4)

Gets the version of the current Zend engine

string **zend_version** (void) \linebreak

Returns a string containing the version of the currently running PHP parser.

Ejemplo 1. zend_version() Example

```
// prints e.g. 'Zend engine version: 1.0.4'
echo "Zend engine version: " . zend_version();
```

See also phpinfo(), phpcredits(), php_logo_guid(), and phpversion().

LXXIX. Funciones POSIX

Este módulo contiene una interfaz a aquellas funciones definidas en el documento estandar IEEE 1003.1 (POSIX.1) que no son accesibles de otra manera. POSIX.1 por ejemplo definió las funciones open(),read(), write() y close(), las cuales han sido parte de PHP durante mucho tiempo.Algunas funciones específicas del sistema no habian estado disponibles antes, aunque con este módulo se intenta remediar esto ofreciendo un acceso fácil a esas funciones.

posix_ctermid (PHP 3>= 3.0.13, PHP 4)

Recoge el nombre de ruta de la terminal de control

string **posix_ctermid** (void) \linebreak

Necesita ser escrito.

posix_getcwd (PHP 3>= 3.0.13, PHP 4)

Nombre de ruta del directorio actual

string **posix_getcwd** (void) \linebreak

Necesita ser escrito cuanto antes.

posix_getegid (PHP 3>= 3.0.10, PHP 4)

Devuelve el ID de grupo efectivo del proceso actual

int **posix_getegid** (void) \linebreak

Devuelve el valor numérico ID de grupo efectivo del proceso actual. Vea también posix_getgrgid() para información sobre como convertir este número en un nombre de grupo manejable.

posix_geteuid (PHP 3>= 3.0.10, PHP 4)

Devuelve el ID de usuario efectivo del proceso actual

int **posix_geteuid** (void) \linebreak

Devuelve el valor numérico ID de usuario efectivo del proceso actual. Vea también posix_getpwuid() para información sobre como convertir este número en un nombre de usuario manejable.

posix_getgid (PHP 3>= 3.0.10, PHP 4)

Devuelve el ID de grupo real del proceso actual

int **posix_getgid** (void) \linebreak

Devuelve el valor númerico ID de grupo real del proceso actual. Vea también posix_getgrgid() para información sobre como convertir esto en un nombre de grupo manejable.

posix_getgrgid (PHP 3>= 3.0.13, PHP 4)

Devuelve información sobre un grupo a trave del id de grupo

array **posix_getgrgid** (int gid) \linebreak

Necesita ser escrito.

posix_getgrnam (PHP 3>= 3.0.13, PHP 4)

Devuelve información sobre un grupo a traves del nombre

array **posix_getgrnam** (string name) \linebreak

Necesita ser escrito.

posix_getgroups (PHP 3>= 3.0.10, PHP 4)

Devuelve el conjunto de grupos del proceso actual

array **posix_getgroups** (void) \linebreak

Devuelve un vector de enteros que contiene los ids numéricos de grupo de el conjunto de grupos del proceso actual. Vea también posix_getgrgid() para información sobre como convertir esto en nombres de grupo manejables.

posix_getlogin (PHP 3>= 3.0.13, PHP 4)

Devuelve el nombre de usuario

string **posix_getlogin** (void) \linebreak

Devuelve el nombre de usuario (login) que es dueño del proceso actual. Vea posix_getpwnam() para información sobre como conseguir mas datos de este usuario.

posix_getpgid (PHP 3>= 3.0.10, PHP 4)

Recoge el id del grupo de procesos para el control de trabajo

int **posix_getpgid** (int pid) \linebreak

Devuelve el identificador de grupo de procesos del proceso `pid`.

Esta no es una función POSIX, pero es normal en sistemas BSD y System V. Si su sistema no soporta esta función a nivel de sistema, esta función PHP devolverá siempre `FALSE`.

posix_getpgrp (PHP 3>= 3.0.10, PHP 4)

Devuelve el identificador de grupo del proceso actual

int **posix_getpgrp** (void) \linebreak

Devuelve el identificador de grupo de proceso del proceso actual. Vea POSIX.1 y la página de manual getpgrp(2) de su sistema POSIX para más información de grupos de procesos.

posix_getpid (PHP 3>= 3.0.10, PHP 4)

Devuelve el identificador del proceso actual

int **posix_getpid** (void) \linebreak

Devuelve el identificador de proceso del proceso actual.

posix_getppid (PHP 3>= 3.0.10, PHP 4)

Devuelve el identificador del proceso padre

int **posix_getppid** (void) \linebreak

Devuelve el identificador de proceso del proceso padre del proceso actual.

posix_getpwnam (PHP 3>= 3.0.13, PHP 4)

Devuelve información sobre un usuario a traves del nombre de usuario

array **posix_getpwnam** (string username) \linebreak

Devuelve un vector asociativo conteniendo información sobre un usuario referenciado por un nombre alfanumérico, pasado a la función en el parametro *`username`*.

Los elementos del vector devuelto son:

Elemento	Descripción

Tabla 1. El vector de información de usuario

Elemento	Descripción
name	El elemento name contiene el nombre de usuario del usuario. Este es un nombre, normalmente menor de 16 caracteres, que no es su nombre completo, pero identifica al usario. Este debe ser el mismo que el parámetro *username* usado en la llamada a la función y por lo tanto es redundante.
passwd	El elemento passwd contiene la contraseña del usuario en un formato encriptado. Normalmente, por ejemplo en un sistema que este utilizando contraseñas "shadow", devolverá un asterisco.
uid	El ID de usuario del usuario en formato numérico.
gid	El ID de grupo del usuario. Utiliza la función posix_getgrgid() para resolver el nombre del grupo y una lista de sus miembros.
gecos	GECOS es un término obsoleto que se refiere al campo apuntado de información en un sistema de procesamiento batch Honeywell. El campo y sus contenidos han sido formalizado por POSIX y contiene una lista separada por comas con el nombre completo del usuario, teléfono del trabajo, número de oficina y télefono de casa. En muchos sistemas solo está disponible el nombre completo del usuario.
dir	Este elemento contiene la ruta absoluta al directorio del usuario (directorio home).
shell	El elemento shell contiene la ruta absoluta al ejecutable del shell por defecto del usuario.

posix_getpwuid (PHP 3>= 3.0.13, PHP 4)

Devuelve información sobre un usuario a traves de su id

array **posix_getpwuid** (int uid) \linebreak

Devuelve un vector asociativo que contiene información sobre un usuario referenciado con un ID de usuario, pasado por el parámetro *uid*.

Los elementos del array son:

Tabla 1. El vector de información del usuario

Elemento	Descripción
name	El elemento name contiene el nombre de usuario del usuario. Este es un nombre, normalmente menor de 16 caracteres, que no es su verdadero nombre.
passwd	El elemento passwd contiene la contraseña del usuario en un formato encriptado. Normalmente, por ejemplo en un sistema con contraseñas "shadow", devolverá un asterisco.
uid	ID del usuario, debe ser el mismo que el parametro *`uid`* usado en la llamada a la función, y por lo tanto redundante.
gid	El ID del grupo del usuario. Utiliza la función posix_getgrgid() para resolver el nombre del grupo y una lista de sus miembros.
gecos	GECOS es un término obsoleto que se refiere al campo apuntado de de información en un sistema de procesamiento batch Honeywell. El campo y sus contenidos han sido formalizados por POSIX y contiene una lista separada por comas con el nombre completo del usuario, teléfono del trabajo, número de oficina y teléfono de casa. En muchos sistemas solo está disponible el nombre completo del usuario.
dir	Este elemento contiene la ruta absoluta al directorio del usuario (directorio home).
shell	El elemento shell contiene la ruta absoluta al ejecutable del shell por defecto del usuario.

posix_getrlimit (PHP 3>= 3.0.10, PHP 4)

Devuelve información sobre los limites de recursos del sistema

array **posix_getrlimit** (void) \linebreak

Necesita ser escrita tan pronto como sea posible.

posix_getsid (PHP 3>= 3.0.10, PHP 4)

Consigue el sid actual del proceso

int **posix_getsid** (int pid) \linebreak

Devuelve el sid del proceso *pid*. Si *pid* es 0, se devolverá el sid del proceso actual.

Esta no es una función POSIX, pero es normal en sistemas System V. Si su sistema no soporta esta función a nivel de sistema, esta función PHP devolverá siempre FALSE.

posix_getuid (PHP 3>= 3.0.10, PHP 4)

Devuelve el ID de usuario real del proceso actual

int **posix_getuid** (void) \linebreak

Devuelve el valor numerico ID de usuario real del proceso actual. Vea también posix_getpwuid() para información sobre como convertir este ID en un nombre de usuario manejable.

posix_isatty (PHP 3>= 3.0.13, PHP 4)

Determina si un descriptor de fichero esta en una terminal interactiva

bool **posix_isatty** (int fd) \linebreak

Necesita ser escrito.

posix_kill (PHP 3>= 3.0.13, PHP 4)

Manda una señal a un proceso

bool **posix_kill** (int pid, int sig) \linebreak

Manda la señal *sig* al proceso con el identificador de proceso *pid*. Devuelve FALSE, si no puede enviar la señal.Si sí la envia devuelve TRUE .

Vea también la página de manual kill(2) de su sistema POSIX, la cual contiene información adicional sobre los identificadores de proceso negativos, el pid especial 0, el pid especial -1, y la señal numero 0.

posix_mkfifo (PHP 3>= 3.0.13, PHP 4)

Crea un fichero especial fifo (los llamados pipe o tuberias)

bool **posix_getcwd** (string pathname, int mode) \linebreak

Necesita ser escrito lo más pronto posible.

posix_setegid (PHP 4 >= 4.0.2)

Set the effective GID of the current process

bool **posix_setegid** (int gid) \linebreak

Set the effective group ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns `TRUE` on success, `FALSE` otherwise.

posix_seteuid (PHP 4 >= 4.0.2)

Set the effective UID of the current process

bool **posix_seteuid** (int uid) \linebreak

Set the real user ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns `TRUE` on success, `FALSE` otherwise. See also posix_setgid().

posix_setgid (PHP 3>= 3.0.13, PHP 4)

Asigna el GID efectivo del proceso actual

bool **posix_setgid** (int gid) \linebreak

Asigna el ID de grupo real del proceso actual. Esta es una función privilegiada y necesitas los privilegion apropiados (normalmente root) en tu sistema para realizar esta función. El orden apropiado de llamada es **posix_setgid()** primero, posix_setuid() después.

Devuelve `TRUE` si tiene éxito, `FALSE` en caso contrario.

posix_setpgid (PHP 3>= 3.0.13, PHP 4)

Asigna el id de grupo de procesos para el control de trabajos

int **posix_setpgid** (int pid, int pgid) \linebreak

Inserta el proceso *`pid`* en el grupo de procesos *`pgid`*. Vea POSIX.1 y la página de manual setsid(2) de su sistema POSIX para más información sobre grupos de procesos y control de trabajo. Devuelve `TRUE` si tiene éxito y `FALSE` en caso contrario.

posix_setsid (PHP 3>= 3.0.13, PHP 4)

Convierte el proceso actual en lider de sesión

int **posix_setsid** (void) \linebreak

Convierte el proceso actual en lider de sesión. Vea POSIX.1 y la página de manual setsid(2) en su sistema POSIX para más informacion en grupos de proceso y control de trabajos. Devuelve el id de sesión.

posix_setuid (PHP 3>= 3.0.13, PHP 4)

Asigna el UID efectivo del proceso actual

bool **posix_setuid** (int uid) \linebreak

Asigna el ID de usuario real al proceso actual. Esta es una función privilegiada y necesitas los privilegios apropiados (normalmente root) en tu sistema para realizar esta función.

Devuelve `TRUE` si tiene exito, `FALSE` en caso contrario. Vea también posix_setgid().

posix_times (PHP 3>= 3.0.13, PHP 4)

Recoge el tiempo de los procesos

array **posix_times** (void) \linebreak

Devuelve un hash de cadenas con información sobre el uso de CPU del proceso actual. Los índices del hash son

- ticks - el numero de ticks de reloj que han pasado desde el reinicio.
- utime - tiempo de usuario usado por el proceso actual.
- stime - tiempo de sistema usado por el proceso actual.
- cutime - tiempo de usuario usado por el proceso actual e hijos.
- cstime - tiempo de sistema usado por el proceso actual e hijos.

posix_ttyname (PHP 3>= 3.0.13, PHP 4)

Determina el nombre del dispositivo terminal

string **posix_ttyname** (int fd) \linebreak

Necesita ser escrito.

posix_uname (PHP 3>= 3.0.10, PHP 4)

Consigue el nombre del sistema

array **posix_uname** (void) \linebreak

Devuelve un hash de cadenas con información sobre el sistema. Los índices del hash son

- sysname - nombre del sistema operativo (e.g. Linux)
- nodename - nombre del sistema (e.g. valiant)
- release - release del sistema operativo (e.g. 2.2.10)
- version - versión del sistema operativo (e.g. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - arquitectura del sistema (e.g. i586)

Posix requiere que usted no debe hacer ninguna suposición sobre el formato de los valores, por ejemplo usted no puede confiar en los tres dígitos de la version o cualquier cosa devuelta por esta función.

LXXX. Funciones de PostgreSQL

Postgres, desarrollado originalmente en el UC Berkeley Computer Science Department, ha sido pionero en muchos de los conceptos relacionales/orientados a objeto que ahora están empezando a estar disponibles en algunas bases de datos comperciales. Tiene soporte de lenguaje SQL92/SQL3, integridad transaccional, y extensibilidad de tipos. PostgreSQL es un descendiente de dominio público, más concretamente open source, del código original de Berkeley.

PostgreSQL se encuentra disponible sin coste alguno. La versión actual la tienes a tu disposición en www.PostgreSQL.org (http://www.postgresql.org/).

Desde la versión 6.3 (02/03/1998) PostgreSQL usa sockets tipo Unix. Abajo se da una tabla con las diferentes posibilidades. El socket se encuentra en el ficheero `/tmp/.s.PGSQL.5432`. Esta opción se controla mediante el flag ’-i’ del **postmaster** y cuando se incluye significa "escuchar sockets TCP/IP además de los de dominio Unix" ya que si no se le dice nada solo escucha sockets tipo Unix.

Tabla 1. Postmaster y PHP

Postmaster	PHP	Estado
postmaster &	pg_connect("", "", "", "", "dbname");	OK
postmaster -i &	pg_connect("", "", "", "", "dbname");	OK
postmaster &	pg_connect("localhost", "", "", "", "dbname");	Unable to connect to PostgreSQL server: connectDB() failed: Is the postmaster running and accepting TCP/IP (with -i) connection at ’localhost’ on port ’5432’? in /path/to/file.php3 on line 20. (Imposible conectar al servidor PostgreSQL, la llamada connectDB() ha fallado: ¿Está funcionando el postmaster aceptando conextiones TCP/IP (con -i) en ’localhost’ en el puerto ’5432’? en /path/to/file.php3 en linea 20.
postmaster -i &	pg_connect("localhost", "", "", "", "dbname");	OK

Uno puede establecer una conexión con el siguiente comando:

Para usar el interface de objetos grandes (large object o lo), es necesario encapsularlo en un bloque de transacción. Un bloque de transacción empieza con un **begin** y si la transacción fue valida termina con

commit y **end**. Si la transacción falla debe ser cerrada con **abort** y **rollback**.

Ejemplo 1. Usando Objetos Grandes (lo)

```
<?php
$database = pg_Connect ("", "", "", "", "jacarta");
pg_exec ($database, "begin");
    $oid = pg_locreate ($database);
    echo ("$oid\n");
    $handle = pg_loopen ($database, $oid, "w");
    echo ("$handle\n");
    pg_lowrite ($handle, "gaga");
    pg_loclose ($handle);
pg_exec ($database, "commit")
pg_exec ($database, "end")
?>
```

pg_affected_rows (PHP 4 >= 4.2.0)

Returns number of affected records(tuples)

int **pg_affected_rows** (resource result) \linebreak

pg_affected_rows() returns the number of tuples (instances/records/rows) affected by INSERT, UPDATE, and DELETE queries executed by pg_query(). If no tuple is affected by this function, it will return 0.

Ejemplo 1. pg_affected_rows()

```
<?php
     $result = pg_query ($conn, "INSERT INTO publisher VALUES ('Author')");
     $cmdtuples = pg_affected_rows ($result);
     echo $cmdtuples . " tuples are affected.";
?>
```

Nota: This function used to be called `pg_cmdtuples()`.

See also pg_query() and pg_num_rows().

pg_cancel_query (PHP 4 >= 4.2.0)

Cancel async query

bool **pg_cancel_query** (resource connection) \linebreak

pg_cancel_query() cancel asynchronous query sent by pg_send_query(). You cannot cancel query executed by pg_query().

See also pg_send_query() and pg_connection_busy()

pg_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Get the client encoding

string **pg_client_encoding** ([resource connection]) \linebreak

pg_client_encoding() returns the client encoding as the string. The returned string should be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

> **Nota:** This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. If libpq is compiled without multibyte encoding support, pg_set_client_encoding() always return "SQL_ASCII". Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details to enable multibyte support and encoding supported.
>
> The function used to be called **pg_clientencoding()**.

See also pg_set_client_encoding().

pg_Close (PHP 3, PHP 4)

Cierra una conexión PostgreSQL

bool **pg_close** (int connection) \linebreak

Devuelve `FALSE` si connection no es un indice de conexión valido y `TRUE` en cualquier otro caso. Cierra la conexión a la base de datos PostgreSQL asociada con el indice de conexión pasado como parámetro.

pg_Connect (PHP 3, PHP 4)

Abre una conexión

int **pg_connect** (string host, string port, string options, string tty, string dbname) \linebreak

Devuelve un índice de conexión en caso de éxito, o falso si la conexión no se puede realizar. Esta función abre una conexión a una base de datos PostgreSQL. Cada uno de los argumentos debe ser una cadena entrecomillada, incluyendo el número de puerto. Los parámetros options y tty son opcionales y pueden ser omitidos. Esta función devuelve un índice de conexión que se necesitará para otras funciones PostgreSQL. Puedes tener multiples conexiones abiertas al mismo tiempo.

Una conexión también se puede establecer con el siguiente comando: **$conn = pg_connect("dbname=marliese port=5432");** Otros parámetros aparte de *`dbname`* y *`port`* son *`host`*, *`tty`*, *`options`*, *`user`* y *`password`*.

Ver también **pg_pConnect()**.

pg_connection_busy (PHP 4 >= 4.2.0)

Get connection is busy or not

bool **pg_connection_busy** (resource connection) \linebreak

pg_connection_busy() returns `TRUE` if the connection is busy. If it is busy, a previous query is still executing. If pg_get_result() is called, it will be blocked.

See also pg_connection_status() and pg_get_result()

pg_connection_reset (PHP 4 >= 4.2.0)

Reset connection (reconnect)

bool **pg_connection_reset** (resource connection) \linebreak

pg_connection_reset() resets the connection. It is useful for error recovery. Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

See also pg_connect(), pg_pconnect() and pg_connection_status()

pg_connection_status (PHP 4 >= 4.2.0)

Get connection status

int **pg_connection_status** (resource connection) \linebreak

pg_connection_status() returns a connection status. Possible statuses are `PGSQL_CONNECTION_OK` and `PGSQL_CONNECTION_BAD`.

See also pg_connection_busy().

pg_convert (PHP 4 CVS only)

Convert associative array value into suitable for SQL statement.

array **pg_convert** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_convert() check and convert `assoc_array` suitable for SQL statement.

Nota: This function is experimental.

See also pg_metadata()

pg_copy_from (PHP 4 >= 4.2.0)

Insert records into a table from an array

int **pg_copy_from** (int connection, string table_name, array rows [, string delimiter [, string null_as]]) \linebreak

pg_copy_from() insert records into a table from *rows*. It issues `COPY` command internally to insert records. Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

See also pg_copy_to()

pg_copy_to (PHP 4 >= 4.2.0)

Copy a table to an array

int **pg_copy_to** (int connection, string table_name [, string delimiter [, string null_as]]) \linebreak

pg_copy_to() copies a table to an array. The resulting array is returned. It returns `FALSE` on failure.

See also pg_copy_from()

pg_DBname (PHP 3, PHP 4)

Nombre de la base de datos

string **pg_dbname** (int connection) \linebreak

Devuelve el nombre de la base de datos a la cual es el índice de conexión con PostgreSQL está conectado, o `FALSE` si connection no es un índice de conexión válido.

pg_delete (PHP 4 CVS only)

Delete records.

long **pg_delete** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_delete() deletes record condition specified by `assoc_array` which has `field=>value`. If `option` is specified, pg_convert() is applied to `assoc_array` with specified option.

Ejemplo 1. pg_delete

```
<?php
    $db = pg_connect ('dbname=foo');
    // This is safe, since $_POST is converted automatically
```

```
    $res = pg_delete($db, 'post_log', $_POST);
    if ($res) {
        echo "POST data is deleted: $res\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Nota: This function is experimental.

See also pg_convert()

pg_end_copy (PHP 4 >= 4.0.3)

Sync with PostgreSQL backend

bool **pg_end_copy** ([resource connection]) \linebreak

pg_end_copy() syncs the PostgreSQL frontend (usually a web server process) with the PostgreSQL server after doing a copy operation performed by pg_put_line(). **pg_end_copy()** must be issued, otherwise the PostgreSQL server may get out of sync with the frontend and will report an error. Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

For further details and an example, see also pg_put_line().

pg_escape_bytea (PHP 4 >= 4.2.0)

Escape binary for bytea type

string **pg_escape_bytea** (string data) \linebreak

pg_escape_bytea() escapes string for bytea datatype. It returns escaped string.

Nota: When you SELECT bytea type, PostgreSQL returns octal byte value prefixed by \ (e.g. \032). Users are supposed to convert back to binary format by yourself.

This function requires PostgreSQL 7.2 or later. With PostgreSQL 7.2.0 and 7.2.1, bytea type must be casted when you enable multi-byte support. i.e. `INSERT INTO test_table (image) VALUES ('$image_escaped'::bytea);` PostgreSQL 7.2.2 or later does not need cast. Exception is when client and backend character encoding does not match, there may be multi-byte stream error. User must cast to bytea to avoid this error.

Newer PostgreSQL will support unescape function. Support for built-in unescape function will be added when it's available.

See also pg_escape_string()

pg_escape_string (PHP 4 >= 4.2.0)

Escape string for text/char type

string **pg_escape_string** (string data) \linebreak

pg_escape_string() escapes string for text/char datatype. It returns escaped string for PostgreSQL. Use of this functon is recommended instead of addslashes().

Nota: This function requires PostgreSQL 7.2 or later.

See also pg_escape_bytea()

pg_Fetch_Array (PHP 3>= 3.0.1, PHP 4)

obtiene una fila en la forma de un array

array **pg_fetch_array** (int result, int row [, int result_type]) \linebreak

Devuelve: Un array que se corresponde con la fila obtenida, o `FALSE` si no hay más filas.

pg_fetch_array() es una versión extendida de pg_fetch_row(). Además de almacenar los datos en los índices numericos del array resultante, también almacena los datos usando índices asociativos, empleando para ello el nombre del campo como la llave o índice.

El tercer parámetro opcional *`result_type`* en **pg_fetch_array()** es una constante y puede tomar cualquiera de los siguientes valores: PGSQL_ASSOC, PGSQL_NUM, y PGSQL_BOTH.

Nota: *`Result_type`* se añadio en PHP 4.0.

Una cosa importante a tener en cuenta es que usar **pg_fetch_array()** NO es significativamente más lento que usar pg_fetch_row(), y sin embargo el valor añadido que aporta sí lo es.

Para más detalles, ver pg_fetch_row()

Ejemplo 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$arr = pg_fetch_array ($result, 0);
echo $arr[0] . " <- array\n";

$arr = pg_fetch_array ($result, 1);
echo $arr["author"] . " <- array\n";
?>
```

pg_Fetch_Object (PHP 3>= 3.0.1, PHP 4)

obtener una fila en forma de objeto

object **pg_fetch_object** (int result, int row [, int result_type]) \linebreak

Devuelve: Un objeto cuyas propiedades se corresponden con los campos de la fila obtenida, o `FALSE` si no hay más filas.

pg_fetch_object() es parecida a pg_fetch_array(), con una diferencia - se devuelve un objeto, en vez de un array. Indirectamente, eso significa que solo puedes acceder a los datos por medio de su nombre de campo, y no a través de sus posiciones (los números son nombres de propiedad invalidos).

El tercer parámetro opcional *`result_type`* en **pg_fetch_object()** es una constante y puede tomar cualquiera de los siguientes valores: PGSQL_ASSOC, PGSQL_NUM, y PGSQL_BOTH.

Nota: *`Result_type`* se añadio en PHP 4.0.

Referente a la velocidad, la función es identica a pg_fetch_array(), y practicamente tan rápida como pg_fetch_row() (la diferencia es insignificante).

Ver también: pg_fetch_array() y pg_fetch_row().

Ejemplo 1. Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect ("localhost", "5432", "", "", $database);
if (!$db_conn): ?>
    <H1>Failed connecting to postgres database <? echo $database ?></H1> <?
    exit;
endif;

$qu = pg_exec ($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // postgres needs a row counter other dbs might not

while ($data = pg_fetch_object ($qu, $row)):
    echo $data->autor." (";
    echo $data->jahr ."): ";
    echo $data->titel."<BR>";
    $row++;
endwhile; ?>

<PRE><?php
$fields[] = Array ("autor", "Author");
$fields[] = Array ("jahr",  "  Year");
$fields[] = Array ("titel", " Title");

$row= 0; // postgres needs a row counter other dbs might not
while ($data = pg_fetch_object ($qu, $row)):
    echo "----------\n";
    reset ($fields);
    while (list (,$item) = each ($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
endwhile;
echo "----------\n"; ?>
</PRE> <?php
pg_freeResult ($qu);
pg_close ($db_conn);
?>
```

pg_fetch_result (PHP 4 >= 4.2.0)

Returns values from a result resource

mixed **pg_fetch_result** (resource result, int row, mixed field) \linebreak

pg_fetch_result() returns values from a *result* resource returned by pg_query(). *row* is integer. *field* is field name(string) or field index (integer). The *row* and *field* specify what cell in the table

of results to return. Row numbering starts from 0. Instead of naming the field, you may use the field index as an unquoted number. Field indices start from 0.

PostgreSQL has many built in types and only the basic ones are directly supported here. All forms of integer, boolean and void types are returned as integer values. All forms of float, and real types are returned as float values. All other types, including arrays are returned as strings formatted in the same default PostgreSQL manner that you would see in the **psql** program.

pg_Fetch_Row (PHP 3>= 3.0.1, PHP 4)

obtiene la fila como un array enumerado

array **pg_fetch_row** (int result, int row) \linebreak

Devuelve: Un array que se corresponde con la fila obtenida, o `FALSE` en el caso de que no haya más filas.

pg_fetch_row() obtiene una fila de datos a partir del resultado asociado con el identificador de resultado especificado. La fila se devuelve en forma de array. Cada columna del resultado se almacena en una posición del array, empezando a partir de la posición 0.

Las siguientes llamadas a **pg_fetch_row()** devolverán la fila siguiente en el conjunto resultado, o falso en el caso de que no haya más filas que devolver.

Ver también: pg_fetch_array(), pg_fetch_object(), **pg_result()**.

Ejemplo 1. Postgres fetch row

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$row = pg_fetch_row ($result, 0);
echo $row[0] . " <- row\n";

$row = pg_fetch_row ($result, 1);
echo $row[0] . " <- row\n";

$row = pg_fetch_row ($result, 2);
echo $row[1] . " <- row\n";
?>
```

pg_field_is_null (PHP 4 >= 4.2.0)

Test if a field is NULL

int **pg_field_is_null** (resource result, int row, mixed field) \linebreak

pg_field_is_null() test if a field is NULL or not. It returns 1 if the field in the given row is NULL. It returns 0 if the field in the given row is NOT NULL. Field can be specified as column index (number) or fieldname (string). Row numbering starts at 0.

Nota: This function used to be called pg_fieldisnull().

pg_field_name (PHP 4 >= 4.2.0)

Returns the name of a field

string **pg_field_name** (resource result, int field_number) \linebreak

pg_field_name() returns the name of the field occupying the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts from 0.

Nota: This function used to be called pg_fieldname().

See also pg_field_num().

pg_field_num (PHP 4 >= 4.2.0)

Returns the field number of the named field

int **pg_field_num** (resource result, string field_name) \linebreak

pg_field_num() will return the number of the column (field) slot that corresponds to the *field_name* in the given PostgreSQL *result* resource. Field numbering starts at 0. This function will return -1 on error.

Nota: This function used to be called pg_fieldnum().

See also pg_field_name().

pg_field_prtlen (PHP 4 >= 4.2.0)

Returns the printed length

int **pg_field_prtlen** (resource result, int row_number, string field_name) \linebreak

pg_field_prtlen() returns the actual printed length (number of characters) of a specific value in a PostgreSQL *result*. Row numbering starts at 0. This function will return -1 on an error.

Nota: This function used to be called pg_field_prtlen().

See also pg_field_size().

pg_field_size (PHP 4 >= 4.2.0)

Returns the internal storage size of the named field

int **pg_field_size** (resource result, int field_number) \linebreak

pg_field_size() returns the internal storage size (in bytes) of the field number in the given PostgreSQL *result*. Field numbering starts at 0. A field size of -1 indicates a variable length field. This function will return FALSE on error.

Nota: This function used to be called pg_fieldsize().

See also **pg_field_len()** and pg_field_type().

pg_field_type (PHP 4 >= 4.2.0)

Returns the type name for the corresponding field number

string **pg_field_type** (resource result, int field_number) \linebreak

pg_field_type() returns a string containing the type name of the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts at 0.

Nota: This function used to be called pg_fieldtype().

See also **pg_field_len()** and pg_field_name().

pg_free_result (PHP 4 >= 4.2.0)

Free result memory

bool **pg_free_result** (resource result) \linebreak

pg_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **pg_free_result()** with the *result* resource as an argument and the associated result memory will be freed. It returns true on success and false if an error occurs.

Nota: This function used to be called `pg_freeresult()`.

See also pg_query().

pg_get_result (PHP 4 >= 4.2.0)

Get asynchronous query result

resource **pg_get_result** ([resource connection]) \linebreak

pg_get_result() get result resource from async query executed by pg_send_query(). pg_send_query() can send multiple queries to PostgreSQL server and **pg_get_result()** is used to get query result one by one. It returns result resource. If there is no more results, it returns `FALSE`.

pg_Host (PHP 3, PHP 4)

Devuelve el nombre del host

string **pg_host** (int connection_id) \linebreak

pg_Host() devuelve el nombre del host al que identificador conexión PostgreSQL pasado está conectado.

pg_insert (PHP 4 CVS only)

Insert array into table.

bool **pg_insert** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_insert() inserts `assoc_array` which has `field=>value` into table specified as `table_name`. If `options` is specified, pg_convert() is applied to `assoc_array` with specified option.

Ejemplo 1. pg_insert

```
<?php
    $db = pg_connect ('dbname=foo');
    // This is safe, since $_POST is converted autotmatically
    $res = pg_insert($db, 'post_log', $_POST);
    if ($res) {
        echo "POST data is succesfully logged\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Nota: This function is experimental.

See also pg_convert()

pg_last_error (PHP 4 >= 4.2.0)

Get the last error message string of a connection

string **pg_last_error** (resource connection) \linebreak

pg_last_error() returns the last error message for given *`connection`*.

Error messages may be overwritten by internal PostgreSQL(libpq) function calls. It may not return appropriate error message, if multiple errors are occured inside a PostgreSQL module function.

Use pg_result_error(), pg_result_status() and pg_connection_status() for better error handling.

Nota: This function used to be called `pg_errormessage()`.

See also pg_result_error().

pg_last_notice (PHP 4 >= 4.0.6)

Returns the last notice message from PostgreSQL server

string **pg_last_notice** (resource connection) \linebreak

pg_last_notice() returns the last notice message from the PostgreSQL server specified by *connection*. The PostgreSQL server sends notice messages in several cases, e.g. if the transactions can't be continued. With **pg_last_notice()**, you can avoid issuing useless queries, by checking whether the notice is related to the transaction or not.

Aviso

This function is EXPERIMENTAL and it is not fully implemented yet. **pg_last_notice()** was added in PHP 4.0.6. However, PHP 4.0.6 has problem with notice message handling. Use of the PostgreSQL module with PHP 4.0.6 is not recommended even if you are not using **pg_last_notice()**.

This function is fully implemented in PHP 4.3.0. PHP earlier than PHP 4.3.0 ignores database connection parameter.

Notice message tracking can be set to optional by setting 1 for `pgsql.ignore_notice` ini from PHP 4.3.0.

Notice message logging can be set to optional by setting 0 for `pgsql.log_notice` ini from PHP 4.3.0. Unless `pgsql.ignore_notice` is set to 0, notice message cannot be logged.

See also pg_query() and pg_last_error().

pg_last_oid (PHP 4 >= 4.2.0)

Returns the last object's oid

int **pg_last_oid** (resource result) \linebreak

pg_last_oid() is used to retrieve the `oid` assigned to an inserted tuple (record) if the result resource is used from the last command sent via pg_query() and was an SQL INSERT. Returns a positive integer if there was a valid `oid`. It returns `FALSE` if an error occurs or the last command sent via pg_query() was not an INSERT or INSERT is failed.

OID field became an optional field from PostgreSQL 7.2. When OID field is not defined in a table, programmer must use pg_result_status() to check if record is is inserted successfully or not.

Nota: This function used to be called `pg_getlastoid()`.

See also pg_query() and pg_result_status()

pg_lo_close (PHP 4 >= 4.2.0)

Close a large object

bool **pg_lo_close** (resource large_object) \linebreak

pg_lo_close() closes a Large Object. *large_object* is a resource for the large object from pg_lo_open().

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called pg_loclose().

See also pg_lo_open(), pg_lo_create() and pg_lo_import().

pg_lo_create (PHP 4 >= 4.2.0)

Create a large object

int **pg_lo_create** (resource connection) \linebreak

pg_lo_create() creates a Large Object and returns the oid of the large object. *connection* specifies a valid database connection opened by pg_connect() or pg_pconnect(). PostgreSQL access modes INV_READ, INV_WRITE, and INV_ARCHIVE are not supported, the object is created always with both read and write access. INV_ARCHIVE has been removed from PostgreSQL itself (version 6.3 and above). It returns large object oid otherwise. It returns FALSE, if an error occurred,

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called pg_locreate().

pg_lo_export (PHP 4 >= 4.2.0)

Export a large object to file

bool **pg_lo_export** (int oid, string pathname [, resource connection]) \linebreak

The *oid* argument specifies oid of the large object to export and the *pathname* argument specifies the pathname of the file. It returns FALSE if an error occurred, TRUE otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called pg_loexport().

See also pg_lo_import().

pg_lo_import (PHP 4 >= 4.2.0)

Import a large object from file

int **pg_lo_import** ([resource connection, string pathname]) \linebreak

In versions before PHP 4.2.0 the syntax of this function was different, see the following definition:

int **pg_lo_import** ([resource connection, string pathname]) \linebreak

The *pathname* argument specifies the pathname of the file to be imported as a large object. It returns FALSE if an error occurred, oid of the just created large object otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: Cuando safe-mode (modo-seguro) está activado, PHP comprueba si el fichero(s)/directorios que vas a utilizar, tienen la misma UID que el script que está siendo ejecutado.

Nota: This function used to be called pg_loimport().

See also pg_lo_export() and pg_lo_open().

pg_lo_open (PHP 4 >= 4.2.0)

Open a large object

resource **pg_lo_open** (resource connection, int oid, string mode) \linebreak

pg_lo_open() open a Large Object and returns large object resource. The resource encapsulates information about the connection. *oid* specifies a valid large object oid and *mode* can be either "r", "w", or "rw". It returns FALSE if there is an error.

Aviso

Do not close the database connection before closing the large object resource.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called pg_loopen().

See also pg_lo_close() and pg_lo_create().

pg_lo_read_all (PHP 4 >= 4.2.0)

Read a entire large object and send straight to browser

int **pg_lo_read_all** (resource large_object) \linebreak

pg_lo_read_all() reads a large object and passes it straight through to the browser after sending all pending headers. Mainly intended for sending binary data like images or sound. It returns number of bytes read. It returns `FALSE`, if an error occurred.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called `pg_loreadall()`.

See also pg_lo_read().

pg_lo_read (PHP 4 >= 4.2.0)

Read a large object

string **pg_lo_read** (resource large_object, int len) \linebreak

pg_lo_read() reads at most *`len`* bytes from a large object and returns it as a string. *`large_object`* specifies a valid large object resource and *`len`* specifies the maximum allowable size of the large object segment. It returns `FALSE` if there is an error.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called `pg_loread()`.

See also pg_lo_read_all().

pg_lo_seek (PHP 4 >= 4.2.0)

Seeks position of large object

bool **pg_lo_seek** (resource large_object, int offset [, int whence]) \linebreak

pg_lo_seek() seeks position of large object resource. *`whence`* is PGSQL_SEEK_SET, PGSQL_SEEK_CUR or PGSQL_SEEK_END.

See also pg_lo_tell().

pg_lo_tell (PHP 4 >= 4.2.0)

Returns current position of large object

int **pg_lo_tell** (resource large_object) \linebreak

pg_lo_tell() returns current position (offset from the beginning of large object).

See also pg_lo_seek().

pg_lo_unlink (PHP 4 >= 4.2.0)

Delete a large object

bool **pg_lo_unlink** (resource connection, int oid) \linebreak

pg_lo_unlink() deletes a large object with the *oid*. It returns `TRUE` on success, otherwise returns `FALSE`.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called `pg_lo_unlink()`.

See also pg_lo_create() and pg_lo_import().

pg_lo_write (PHP 4 >= 4.2.0)

Write a large object

int **pg_lo_write** (resource large_object, string data) \linebreak

pg_lo_write() writes at most to a large object from a variable *data* and returns the number of bytes actually written, or `FALSE` in the case of an error. *large_object* is a large object resource from pg_lo_open().

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called `pg_lo_write()`.

See also pg_lo_create() and pg_lo_open().

pg_metadata (PHP 4 CVS only)

Get metadata for table.

array **pg_metadata** (resource connection, string table_name) \linebreak

pg_metadata() returns table definition for `table_name` as array. If there is error, it returns `FALSE`

Nota: This function is experimental.

See also pg_convert()

pg_num_fields (PHP 4 >= 4.2.0)

Returns the number of fields

int **pg_num_fields** (resource result) \linebreak

pg_num_fields() returns the number of fields (columns) in a PostgreSQL *`result`*. The argument is a result resource returned by pg_query(). This function will return -1 on error.

Nota: This function used to be called `pg_numfields()`.

See also pg_num_rows() and pg_affected_rows().

pg_num_rows (PHP 4 >= 4.2.0)

Returns the number of rows

int **pg_num_rows** (resource result) \linebreak

pg_num_rows() will return the number of rows in a PostgreSQL *`result`* resource. *`result`* is a query result resource returned by pg_query(). This function will return -1 on error.

Nota: Use pg_affected_rows() to get number of rows affected by INSERT, UPDATE and DELETE query.

Nota: This function used to be called `pg_numrows()`.

See also pg_num_fields() and pg_affected_rows().

pg_Options (PHP 3, PHP 4)

Devuelve opciones

string **pg_options** (int connection_id) \linebreak

pg_Options() devuelve una cadena que contiene las opciones especificadas en el identificador de conexión con PostgreSQL dado.

pg_pConnect (PHP 3, PHP 4)

Crea una conexión persistente con una base de datos

int **pg_pconnect** (string host, string port, string options, string tty, string dbname) \linebreak

Devuelve un índice de conexión en caso de éxito, o `FALSE` si no es posible realizar la conexión. Abre una conexión persistente hacia una base de datos de PostgreSQL. Cada uno de los parámetros puede ser una cadena entrecomillada (quoted), incluyendo el número de puerto. Los parámetros options y tty son opcionales y pueden omitirse. Esta función devuelve un índice de conexión que luego será empleado al llamar a otras funciones PostgreSQL. Puedes tener multiples conexiones persistentes abiertas al mismo tiempo. Ver también **pg_Connect()**.

Una conexión también se puede establecer con el comando siguiente: **$conn = pg_pconnect("dbname=marliese port=5432");** Otros parámetros además de *`dbname`* y *`port`* son *`host`*, *`tty`*, *`options`*, *`user`* y *`password`*.

pg_Port (PHP 3, PHP 4)

Devuelve el número de puerto

int **pg_port** (int connection_id) \linebreak

pg_Port() devuelve el número del puerto al que el identificador de conexión con PostgreSQL está conectado.

pg_put_line (PHP 4 >= 4.0.3)

Send a NULL-terminated string to PostgreSQL backend

bool **pg_put_line** ([resource connection, string data]) \linebreak

pg_put_line() sends a NULL-terminated string to the PostgreSQL backend server. This is useful for example for very high-speed inserting of data into a table, initiated by starting a PostgreSQL

copy-operation. That final NULL-character is added automatically. Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

> **Nota:** The application must explicitly send the two characters "\." on the last line to indicate to the backend that it has finished sending its data.

See also pg_end_copy().

Ejemplo 1. High-speed insertion of data into a table

```
<?php
    $conn = pg_pconnect ("dbname=foo");
    pg_query($conn, "create table bar (a int4, b char(16), d float8)");
    pg_query($conn, "copy bar from stdin");
    pg_put_line($conn, "3\thello world\t4.5\n");
    pg_put_line($conn, "4\tgoodbye world\t7.11\n");
    pg_put_line($conn, "\\.\n");
    pg_end_copy($conn);
?>
```

pg_query (PHP 4 >= 4.2.0)

Execute a query

resource **pg_query** (resource connection, string query) \linebreak

pg_query() returns a query result resource if query could be executed. It returns `FALSE` on failure or if connection is not a valid connection. Details about the error can be retrieved using the pg_last_error() function if connection is valid. pg_last_error() sends an SQL statement to the PostgreSQL database specified by the *`connection`* resource. The *`connection`* must be a valid connection that was returned by pg_connect() or pg_pconnect(). The return value of this function is an query result resource to be used to access the results from other PostgreSQL functions such as pg_fetch_array().

> **Nota:** *`connection`* is a optional parameter for **pg_query()**. If *`connection`* is not set, default connection is used. Default connection is the last connection made by pg_connect() or pg_pconnect().
>
> Although *`connection`* can be omitted, it is not recommended, since it could be a cause of hard to find bug in script.

Nota: This function used to be called `pg_exec()`. `pg_exec()` is still available for compatibility reasons but users are encouraged to use the newer name.

See also pg_connect(), pg_pconnect(), pg_fetch_array(), pg_fetch_object(), pg_num_rows(), and pg_affected_rows().

pg_result_error (PHP 4 >= 4.2.0)

Get error message associated with result

string **pg_result_error** (resource result) \linebreak

pg_result_error() returns error message associated with *`result`* resource. Therefore, user has better chance to get better error message than pg_last_error().

See also pg_query(), pg_send_query(), pg_get_result(), pg_last_error() and pg_last_notice()

pg_result_status (PHP 4 >= 4.2.0)

Get status of query result

int **pg_result_status** (resource result) \linebreak

pg_result_status() returns status of result resource. Possible retun values are PGSQL_EMPTY_QUERY, PGSQL_COMMAND_OK, PGSQL_TUPLES_OK, PGSQL_COPY_TO, PGSQL_COPY_FROM, PGSQL_BAD_RESPONSE, PGSQL_NONFATAL_ERROR and PGSQL_FATAL_ERROR.

See also pg_connection_status().

pg_select (PHP 4 CVS only)

Select records.

array **pg_select** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_select() selects records specified by `assoc_array` which has `field=>value`. For successful query, it returns array contains all records and fields that match the condition specified by `assoc_array`. If `options` is specified, pg_convert() s applied to `assoc_array` with specified option.

Ejemplo 1. pg_select

```
<?php
    $db = pg_connect ('dbname=foo');
```

```
    // This is safe, since $_POST is converted autotmatically
    $rec = pg_select($db, 'post_log', $_POST);
    if ($rec) {
        echo "Records selected\n";
        var_dump($rec);
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Nota: This function is experimental.

See also pg_convert()

pg_send_query (PHP 4 >= 4.2.0)

Send asynchronous query

bool **pg_send_query** (resource connection, string query) \linebreak bool **pg_send_query** (string query) \linebreak

pg_send_query() send asynchronous query to the *connection*. Unlike pg_query(), it can send multiple query to PostgreSQL and get the result one by one using pg_get_result(). Script execution is not block while query is executing. Use pg_connection_busy() to check connection is busy (i.e. query is executing) Query may be canceled by calling pg_cancel_query().

Although, user can send multiple query at once. User cannot send multiple query over busy connection. If query is sent while connection is busy, it waits until last query is finished and discards all result.

See also pg_query(), pg_cancel_query(), pg_get_result() and pg_connection_busy()

pg_set_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Set the client encoding

int **pg_set_client_encoding** ([resource connection, string encoding]) \linebreak

pg_set_client_encoding() sets the client encoding and return 0 if success or -1 if error.

encoding is the client encoding and can be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250. Available encoding depends on your PostgreSQL and libpq version. Refer to PostgreSQL manual for supported encodings for your PostgreSQL.

Nota: This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details.

The function used to be called **pg_setclientencoding()**.

See also pg_client_encoding().

pg_trace (PHP 4 >= 4.0.1)

Enable tracing a PostgreSQL connection

bool **pg_trace** (string pathname [, string mode [, resource connection]]) \linebreak

pg_trace() enables tracing of the PostgreSQL frontend/backend communication to a debugging file specified as *pathname*. To fully understand the results, one needs to be familiar with the internals of PostgreSQL communication protocol. For those who are not, it can still be useful for tracing errors in queries sent to the server, you could do for example **grep '^To backend' trace.log** and see what query actually were sent to the PostgreSQL server. For more information, refer to PostgreSQL manual.

Filename and *mode* are the same as in fopen() (*mode* defaults to 'w'), *connection* specifies the connection to trace and defaults to the last one opened.

It returns TRUE if *pathname* could be opened for logging, FALSE otherwise.

See also fopen() and pg_untrace().

pg_tty (PHP 3, PHP 4)

Devuelve el nombre del tty

string **pg_tty** (int connection_id) \linebreak

pg_tty() devuelve el nombre del tty hacia el que se dirije la salida de depuración del lado del servidor en el identificador de conexión de PostgreSQL dado.

pg_untrace (PHP 4 >= 4.0.1)

Disable tracing of a PostgreSQL connection

bool **pg_untrace** ([resource connection]) \linebreak

Stop tracing started by pg_trace(). *connection* specifies the connection that was traced and defaults to the last one opened.

Returns always TRUE.

See also pg_trace().

pg_update (PHP 4 CVS only)

Update table.

long **pg_update** (resource connection, string table_name, array condition, array data [, int options]) \linebreak

pg_update() updates records that matches `condition` with `data` If `options` is specified, pg_convert() is applied to `assoc_array` with specified options.

Ejemplo 1. pg_update

```
<?php
    $db = pg_connect ('dbname=foo');
    $data = array('field1'=>'AA', 'field2'=>'BB');
    // This is safe, since $_POST is converted automatically
    $res = pg_update($db, 'post_log', $_POST, $data);
    if ($res) {
        echo "Data is updated: $res\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Nota: This function is experimental.

See also pg_convert()

LXXXI. Process Control Functions

Process Control support in PHP implements the Unix style of process creation, program execution, signal handling and process termination. Process Control should not be enabled within a webserver environment and unexpected results may happen if any Process Control functions are used within a webserver environment.

This documentation is intended to explain the general usage of each of the Process Control functions. For detailed information about Unix process control you are encouraged to consult your systems documentation including fork(2), waitpid(2) and signal(2) or a comprehensive reference such as Advanced Programming in the UNIX Environment by W. Richard Stevens (Addison-Wesley).

Process Control support in PHP is not enabled by default. You will need to use the --enable-pcntl configuration option when compiling PHP to enable Process Control support.

> **Nota:** Currently, this module will not function on non-Unix platforms (Windows).

The following list of signals are supported by the Process Control functions. Please see your systems signal(7) man page for details of the default behavior of these signals.

Tabla 1. Supported Signals

SIGFPE	SIGCONT	SIGKILL
SIGSTOP	SIGUSR1	SIGTSTP
SIGHUP	SIGUSR2	SIGTTIN
SIGINT	SIGSEGV	SIGTTOU
SIGQUIT	SIGPIPE	SIGURG
SIGILL	SIGALRM	SIGXCPU
SIGTRAP	SIGTERM	SIGXFSZ
SIGABRT	SIGSTKFLT	SIGVTALRM
SIGIOT	SIGCHLD	SIGPROF
SIGBUS	SIGCLD	SIGWINCH
SIGPOLL	SIGIO	SIGPWR
SIGSYS		

Process Control Example

This example forks off a daemon process with a signal handler.

Ejemplo 1. Process Control Example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
     die("could not fork");
} else if ($pid) {
     exit(); // we are the parent
} else {
     // we are the child
}

// detatch from the controlling terminal
if (!posix_setsid()) {
    die("could not detach from terminal");
}

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");

// loop forever performing tasks
while(1) {

    // do something interesting here

}

function sig_handler($signo) {

     switch($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
             break;
         default:
             // handle all other signals
     }

}

?>
```

pcntl_exec (PHP 4 >= 4.2.0)

Executes specified program in current process space

bool **pcntl_exec** (string path [, array args [, array envs]]) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

pcntl_fork (PHP 4 >= 4.1.0)

Forks the currently running process

int **pcntl_fork** (void) \linebreak

The **pcntl_fork()** function creates a child process that differs from the parent process only in it's PID and PPID. Please see your system's fork(2) man page for specific details as to how fork works on your system.

On success, the PID of the child process is returned in the parent's thread of execution, and a 0 is returned in the child's thread of execution. On failure, a -1 will be returned in the parent's context, no child process will be created, and a PHP error is raised.

Ejemplo 1. pcntl_fork() Example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
     die("could not fork");
} else if ($pid) {
     // we are the parent
} else {
     // we are the child
}

?>
```

See also pcntl_waitpid() and pcntl_signal().

pcntl_signal (PHP 4 >= 4.1.0)

Installs a signal handler

bool **pcntl_signal** (int signo, mixed handle) \linebreak

The **pcntl_signal()** function installs a new signal handler for the signal indicated by *signo*. The signal handler is set to *handler* which may be the name of a user created function, or either of the two global constants SIG_IGN or SIG_DFL.

pcntl_signal() returns TRUE on success or FALSE on failure.

Ejemplo 1. pcntl_signal() Example

```
<?php

// signal handler function
function sig_handler($signo) {

     switch($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
             break;
         case SIGUSR1:
             print "Caught SIGUSR1...\n";
             break;
         default:
             // handle all other signals
     }

}

print "Installing signal handler...\n";

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");
pcntl_signal(SIGUSR1, "sig_handler");

print "Generating signal SIGTERM to self...\n";

// send SIGUSR1 to current process id
posix_kill(posix_getpid(), SIGUSR1);

print "Done\n"

?>
```

See also pcntl_fork() and pcntl_waitpid().

pcntl_waitpid (PHP 4 >= 4.1.0)

Waits on or returns the status of a forked child

int **pcntl_waitpid** (int pid, int status, int options) \linebreak

The **pcntl_waitpid()** function suspends execution of the current process until a child as specified by the *pid* argument has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal handling function. If a child as requested by *pid* has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed. Please see your system's waitpid(2) man page for specific details as to how waitpid works on your system.

pcntl_waitpid() returns the process ID of the child which exited, -1 on error or zero if WNOHANG was used and no child was available

The value of *pid* can be one of the following:

Tabla 1. possible values for *pid*

< -1	wait for any child process whose process group ID is equal to the absolute value of *pid*.
-1	wait for any child process; this is the same behaviour that the wait function exhibits.
0	wait for any child process whose process group ID is equal to that of the calling process.
> 0	wait for the child whose process ID is equal to the value of *pid*.

pcntl_waitpid() will store status information in the *status* parameter which can be evaluated using the following functions: pcntl_wifexited(), pcntl_wifstopped(), pcntl_wifsignaled(), pcntl_wexitstatus(), pcntl_wtermsig() and pcntl_wstopsig().

The value of *options* is the value of zero or more of the following two global constants OR'ed together:

Tabla 2. possible values for *options*

WNOHANG	return immediately if no child has exited.
WUNTRACED	return for children which are stopped, and whose status has not been reported.

See also pcntl_fork(), pcntl_signal(), pcntl_wifexited(), pcntl_wifstopped(), pcntl_wifsignaled(), pcntl_wexitstatus(), pcntl_wtermsig() and pcntl_wstopsig().

pcntl_wexitstatus (PHP 4 >= 4.1.0)

Returns the return code of a terminated child

int **pcntl_wexitstatus** (int status) \linebreak

Returns the return code of a terminated child. This function is only useful if pcntl_wifexited() returned `TRUE`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wifexited().

pcntl_wifexited (PHP 4 >= 4.1.0)

Returns `TRUE` if status code represents a successful exit

int **pcntl_wifexited** (int status) \linebreak

Returns `TRUE` if the child status code represents a successful exit.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wexitstatus().

pcntl_wifsignaled (PHP 4 >= 4.1.0)

Returns `TRUE` if status code represents a termination due to a signal

int **pcntl_wifsignaled** (int status) \linebreak

Returns `TRUE` if the child process exited because of a signal which was not caught.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_signal().

pcntl_wifstopped (PHP 4 >= 4.1.0)

Returns `TRUE` if child process is currently stopped

int **pcntl_wifstopped** (int status) \linebreak

Returns `TRUE` if the child process which caused the return is currently stopped; this is only possible if the call to pcntl_waitpid() was done using the option `WUNTRACED`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid().

pcntl_wstopsig (PHP 4 >= 4.1.0)

Returns the signal which caused the child to stop

int **pcntl_wstopsig** (int status) \linebreak

Returns the number of the signal which caused the child to stop. This function is only useful if pcntl_wifstopped() returned `TRUE`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wifstopped().

pcntl_wtermsig (PHP 4 >= 4.1.0)

Returns the signal which caused the child to terminate

int **pcntl_wtermsig** (int status) \linebreak

Returns the number of the signal that caused the child process to terminate. This function is only useful if pcntl_wifsignaled() returned `TRUE`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid(), pcntl_signal() and pcntl_wifsignaled().

LXXXII. Funciones de ejecución de programas

escapeshellarg (PHP 4 >= 4.0.3)

escape a string to be used as a shell argument

string **escapeshellarg** (string arg) \linebreak

escapeshellarg() adds single quotes around a string and quotes/escapes any existing single quotes allowing you to pass a string directly to a shell function and having it be treated as a single safe argument. This function should be used to escape individual arguments to shell functions coming from user input. The shell functions include exec(), system() and the backtick operator. A standard use would be:

```
system("ls ".escapeshellarg($dir));
```

See also exec(), popen(), system(), and the backtick operator.

escapeshellcmd (PHP 3, PHP 4)

enmascara los metacaracteres del intérprete de ordenes

string **escapeshellcmd** (string command) \linebreak

EscapeShellCmd() enmascara cualquier carácter en una cadena de caracteres que pueda usarse para introducir fraudulentamente una orden al intérprete de órdenes para que éste ejecute instrucciones arbitrarias. Esta función se debería usar para asegurarse que cualquier dato que venga del usuario se enmascare antes de que éste se le pase a las funciones exec() o system(), o al operador ‘ (apóstrofe invertido) . Un uso habitual podría ser:

```
system(EscapeShellCmd($cmd))
```

Véase también exec(), popen(), system(), y el operador ‘ (apóstrofe invertido).

exec (PHP 3, PHP 4)

Ejecuta un programa externo

string **exec** (string command [, string array [, int return_var]]) \linebreak

exec() ejecuta la orden indicada en *command*, sin embargo no produce ninguna salida. Simplemente devuelve la última línea de la salida resultado de la orden. Si necesita ejecutar una orden y obtener directamente todos los datos devueltos por la orden sin ninguna interferencia, use la función **PassThru()**.

Si el parámetro *array* existe, entonces el array especificado se rellenará con cada una de las líneas de la salida producida por la orden. Notar que si el array ya contiene algunos elementos, **exec()** los añadirá al final del array. Si no quiere que la función añada dichos elementos, haga un unset() sobre el array antes de pasárselo a **exec()**.

Si el parámetro *return_var* existe a la vez que el parámetro *array*, entonces el valor de retorno de la orden ejecutada se guardará en dicha variable.

Destacar que si usted va a permitir que se pasen datos provenientes de usuarios a esta función, entonces debería usar **EscapeShellCmd()** para asegurarse de que los usuarios no pueden engañar al sistema para ejecutar instrucciones arbitrarias.

Véase también system(), **PassThru()**, popen(), **EscapeShellCmd()**, y el operador ‘ (apóstrofe invertido).

passthru (PHP 3, PHP 4)

Ejecuta un programa externo y muestra su salida literal

string **passthru** (string command [, int return_var]) \linebreak

La función **passthru()** es similar a la función exec() en que ejecuta una orden (*command*). Si existe el parámetro *return_var*, el valor de estado devuelto por la orden Unix se guardará ahí. Esta función debería usarse en lugar de exec() o system() cuando la salida de la orden Unix sean datos binarios que deban ser pasados directamente al navegador. Un uso típico de ello es ejecutar algo como las utilidades pbmplus las cuales pueden dar como resultado directamente el flujo de datos de una imagen. Poniendo el content-type a *image/gif* y llamando al programa pbmplus para mostrar un gif, usted puede crear archivos de órdenes PHP que generen directamente imágenes.

Véase también exec(), system(), popen(), **EscapeShellCmd()**, y el operador ‘ (apóstrofe invertido).

proc_close (PHP 4 CVS only)

Close a process opened by proc_open and return the exit code of that process.

int **proc_close** (resource process) \linebreak

proc_close() is similar to popen() except that it only works on processes opened by proc_open(). **proc_close()** waits for the process to terminate, and returns it’s exit code. If you have open pipes to that process, you should fclose() them prior to calling this function in order to avoid a deadlock - the child process may not be able to exit while the pipes are open.

proc_open (PHP 4 CVS only)

Execute a command and open file pointers for input/output

resource **proc_open** (string cmd, array descriptorspec, array pipes) \linebreak

proc_open() is similar to popen() but provides a much greater degree of control over the program execution. *cmd* is the command to be executed by the shell. *descriptorspec* is an indexed array where the key represents the descriptor number and the value represents how PHP will pass that descriptor to the child process. *pipes* will be set to an indexed array of file pointers that correspond to PHP's end of any pipes that are created. The return value is a resource representing the process; you should free it using proc_close() when you are finished with it.

```
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
   2 => array("file", "/tmp/error-output.txt", "a"), // stderr is a file to write to
);
$process = proc_open("php", $descriptorspec, $pipes);
if (is_resource($process)) {
    // $pipes now looks like this:
    // 0 => writeable handle connected to child stdin
    // 1 => readable handle connected to child stdout
    // Any error output will be appended to /tmp/error-output.txt

    fwrite($pipes[0], "<?php echo \"Hello World!\"; ?>");
    fclose($pipes[0]);

    while(!feof($pipes[1])) {
        echo fgets($pipes[1], 1024);
    }
    fclose($pipes[1]);
    // It is important that you close any pipes before calling
    // proc_close in order to avoid a deadlock
    $return_value = proc_close($process);

    echo "command returned $return_value\n";
}
```

The file descriptor numbers in *descriptorspec* are not limited to 0, 1 and 2 - you may specify any valid file descriptor number and it will be passed to the child process. This allows your script to interoperate with other scripts that run as "co-processes". In particular, this is useful for passing passphrases to programs like PGP, GPG and openssl in a more secure manner. It is also useful for reading status information provided by those programs on auxillary file descriptors.

Nota: Windows compatibility: Descriptors beyond 2 (stderr) are made available to the child process as inheritable handles, but since the Windows architecture does not associate file descriptor numbers with low-level handles, the child process does not (yet) have a means of accessing those handles. Stdin, stdout and stderr work as expected.

Nota: This function was introduced in PHP 4.3.0.

Nota: If you only need a uni-directional (one-way) process pipe, use popen() instead, as it is much easier to use.

See also exec(), system(), passthru(), popen(), escapeshellcmd(), and the backtick operator.

shell_exec (PHP 4)

Execute command via shell and return complete output as string

string **shell_exec** (string cmd) \linebreak

This function is identical to the backtick operator.

system (PHP 3, PHP 4)

Ejecuta un programa externo y muestra su salida

string **system** (string command [, int return_var]) \linebreak

system() se parece a la versión C de la función de mismo nombre en que ejecuta la orden indicada en *command* y muestra el resultado. Si se indica una variable como segundo parámetro, el código de estado devuelto por la orden ejecutada se guardará en esta variable.

Destacar que si usted va a permitir que se pasen datos provenientes de usuarios a esta función, entonces debería usar **EscapeShellCmd()** para asegurarse de que los usuarios no pueden engañar al sistema para ejecutar instrucciones arbitrarias.

La llamada a **system()** también intenta vaciar automáticamente el buffer de salida del servidor web después de cada línea de salida si PHP está funcionando como un módulo del servidor.

Devuelve la última línea de la orden en caso de éxito, y falso en caso de fallo.

Si necesita ejecutar una orden y obtener de vuelta todo los datos del mismo sin interferencias, use la función **PassThru()**.

Véase también exec(), **PassThru()**, popen(), **EscapeShellCmd()**, y el operador ‘ (apóstrofe invertido).

LXXXIII. Printer functions

These functions are only available under Windows 9.x, ME, NT4 and 2000. They have been added in PHP 4 (4.0.4).

printer_abort (unknown)

Deletes the printer's spool file

void **printer_abort** (resource handle) \linebreak

This function deletes the printers spool file.

handle must be a valid handle to a printer.

Ejemplo 1. printer_abort() example

```
$handle = printer_open();
printer_abort($handle);
printer_close($handle);
```

printer_close (unknown)

Close an open printer connection

void **printer_close** (resource handle) \linebreak

This function closes the printer connection. **printer_close()** also closes the active device context.

handle must be a valid handle to a printer.

Ejemplo 1. printer_close() example

```
$handle = printer_open();
printer_close($handle);
```

printer_create_brush (unknown)

Create a new brush

mixed **printer_create_brush** (int style, string color) \linebreak

The function creates a new brush and returns a handle to it. A brush is used to fill shapes. For an example see printer_select_brush(). *color* must be a color in RGB hex format, i.e. "000000" for black, *style* must be one of the following constants:

- *PRINTER_BRUSH_SOLID*: creates a brush with a solid color.
- *PRINTER_BRUSH_DIAGONAL*: creates a brush with a 45-degree upward left-to-right hatch (/).
- *PRINTER_BRUSH_CROSS*: creates a brush with a cross hatch (+).
- *PRINTER_BRUSH_DIAGCROSS*: creates a brush with a 45 cross hatch (x).
- *PRINTER_BRUSH_FDIAGONAL*: creates a brush with a 45-degree downward left-to-right hatch (\ ).
- *PRINTER_BRUSH_HORIZONTAL*: creates a brush with a horizontal hatch (-).
- *PRINTER_BRUSH_VERTICAL*: creates a brush with a vertical hatch (|).
- *PRINTER_BRUSH_CUSTOM*: creates a custom brush from an BMP file. The second parameter is used to specify the BMP instead of the RGB color code.

printer_create_dc (unknown)

Create a new device context

void **printer_create_dc** (resource handle) \linebreak

The function creates a new device context. A device context is used to customize the graphic objects of the document. *handle* must be a valid handle to a printer.

Ejemplo 1. printer_create_dc() example

```
$handle = printer_open();
printer_start_doc($handle);
printer_start_page($handle);

printer_create_dc($handle);
/* do some stuff with the dc */
printer_set_option($handle, PRINTER_TEXT_COLOR, "333333");
printer_draw_text($handle, 1, 1, "text");
printer_delete_dc($handle);

/* create another dc */
printer_create_dc($handle);
printer_set_option($handle, PRINTER_TEXT_COLOR, "000000");
printer_draw_text($handle, 1, 1, "text");
/* do some stuff with the dc */

printer_delete_dc($handle);

printer_endpage($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_create_font (unknown)

Create a new font

mixed **printer_create_font** (string face, int height, int width, int font_weight, bool italic, bool underline, bool strikeout, int orientaton) \linebreak

The function creates a new font and returns a handle to it. A font is used to draw text. For an example see printer_select_font(). *face* must be a string specifying the font face. *height* specifies the font height, and *width* the font width. The *font_weight* specifies the font weight (400 is normal), and can be one of the following predefined constants.

- *PRINTER_FW_THIN*: sets the font weight to thin (100).
- *PRINTER_FW_ULTRALIGHT*: sets the font weight to ultra light (200).
- *PRINTER_FW_LIGHT*: sets the font weight to light (300).
- *PRINTER_FW_NORMAL*: sets the font weight to normal (400).
- *PRINTER_FW_MEDIUM*: sets the font weight to medium (500).
- *PRINTER_FW_BOLD*: sets the font weight to bold (700).
- *PRINTER_FW_ULTRABOLD*: sets the font weight to ultra bold (800).
- *PRINTER_FW_HEAVY*: sets the font weight to heavy (900).

italic can be TRUE or FALSE, and sets whether the font should be italic.

underline can be TRUE or FALSE, and sets whether the font should be underlined.

strikeout can be TRUE or FALSE, and sets whether the font should be striked out.

orientation specifies a rotation. For an example see printer_select_font().

printer_create_pen (unknown)

Create a new pen

mixed **printer_create_pen** (int style, int width, string color) \linebreak

The function creates a new pen and returns a handle to it. A pen is used to draw lines and curves. For an example see printer_select_pen(). *color* must be a color in RGB hex format, i.e. "000000" for black, *width* specifies the width of the pen whereas *style* must be one of the following constants:

- *PRINTER_PEN_SOLID*: creates a solid pen.
- *PRINTER_PEN_DASH*: creates a dashed pen.
- *PRINTER_PEN_DOT*: creates a dotted pen.
- *PRINTER_PEN_DASHDOT*: creates a pen with dashes and dots.
- *PRINTER_PEN_DASHDOTDOT*: creates a pen with dashes and double dots.

- *PRINTER_PEN_INVISIBLE*: creates an invisible pen.

printer_delete_brush (unknown)

Delete a brush

bool **printer_delete_brush** (resource handle) \linebreak

The function deletes the selected brush. For an example see printer_select_brush(). It returns TRUE on success, or FALSE otherwise. *handle* must be a valid handle to a brush.

printer_delete_dc (unknown)

Delete a device context

bool **printer_delete_dc** (resource handle) \linebreak

The function deletes the device context and returns TRUE on success, or FALSE if an error occurred. For an example see printer_create_dc(). *handle* must be a valid handle to a printer.

printer_delete_font (unknown)

Delete a font

bool **printer_delete_font** (resource handle) \linebreak

The function deletes the selected font. For an example see printer_select_font(). It returns TRUE on success, or FALSE otherwise. *handle* must be a valid handle to a font.

printer_delete_pen (unknown)

Delete a pen

bool **printer_delete_pen** (resource handle) \linebreak

The function deletes the selected pen. For an example see printer_select_pen(). It returns TRUE on success, or FALSE otherwise. *handle* must be a valid handle to a pen.

printer_draw_bmp (unknown)

Draw a bmp

void **printer_draw_bmp** (resource handle, string filename, int x, int y) \linebreak

The function simply draws an bmp the bitmap *`filename`* at position *`x`*, *`y`*. *`handle`* must be a valid handle to a printer.

The function returns `TRUE` on success, or otherwise `FALSE`.

Ejemplo 1. printer_draw_bmp() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_draw_bmp($handle, "c:\\image.bmp", 1, 1);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_chord (unknown)

Draw a chord

void **printer_draw_chord** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad_x, int rad_y, int rad_x1, int rad_y1) \linebreak

The function simply draws an chord. *`handle`* must be a valid handle to a printer.

`rec_x` is the upper left x coordinate of the bounding rectangle.

`rec_y` is the upper left y coordinate of the bounding rectangle.

`rec_x1` is the lower right x coordinate of the bounding rectangle.

`rec_y1` is the lower right y coordinate of the bounding rectangle.

`rad_x` is x coordinate of the radial defining the beginning of the chord.

`rad_y` is y coordinate of the radial defining the beginning of the chord.

`rad_x1` is x coordinate of the radial defining the end of the chord.

`rad_y1` is y coordinate of the radial defining the end of the chord.

Ejemplo 1. printer_draw_chord() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_chord($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_elipse (unknown)

Draw an ellipse

void **printer_draw_elipse** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y) \linebreak

The function simply draws an ellipse. *handle* must be a valid handle to a printer.

ul_x is the upper left x coordinate of the ellipse.

ul_y is the upper left y coordinate of the ellipse.

lr_x is the lower right x coordinate of the ellipse.

lr_y is the lower right y coordinate of the ellipse.

Ejemplo 1. printer_draw_elipse() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);
```

```
printer_draw_elipse($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_line (unknown)

Draw a line

void **printer_draw_line** (resource printer_handle, int from_x, int from_y, int to_x, int to_y) \linebreak

The function simply draws a line from position *`from_x`*, *`from_y`* to position *`to_x`*, *`to_y`* using the selected pen. *`printer_handle`* must be a valid handle to a printer.

Ejemplo 1. printer_draw_line() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "000000");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 10, 1000, 10);
printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_pie (unknown)

Draw a pie

void **printer_draw_pie** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad1_x, int rad1_y, int rad2_x, int rad2_y) \linebreak

The function simply draws an pie. *handle* must be a valid handle to a printer.

rec_x is the upper left x coordinate of the bounding rectangle.

rec_y is the upper left y coordinate of the bounding rectangle.

rec_x1 is the lower right x coordinate of the bounding rectangle.

rec_y1 is the lower right y coordinate of the bounding rectangle.

rad1_x is x coordinate of the first radial's ending.

rad1_y is y coordinate of the first radial's ending.

rad2_x is x coordinate of the second radial's ending.

rad2_y is y coordinate of the second radial's ending.

Ejemplo 1. printer_draw_pie() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_pie($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_rectangle (unknown)

Draw a rectangle

void **printer_draw_rectangle** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y) \linebreak

The function simply draws a rectangle.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

Ejemplo 1. printer_draw_rectangle() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_roundrect (unknown)

Draw a rectangle with rounded corners

void **printer_draw_roundrect** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y, int width, int height) \line-break

The function simply draws a rectangle with rounded corners.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

width is the width of the ellipse.

height is the height of the ellipse.

Ejemplo 1. printer_draw_roundrect() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_roundrect($handle, 1, 1, 500, 500, 200, 200);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_text (unknown)

Draw text

void **printer_draw_text** (resource printer_handle, string text, int x, int y) \linebreak

The function simply draws *text* at position *x*, *y* using the selected font. *printer_handle* must be a valid handle to a printer.

Ejemplo 1. printer_draw_text() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial",72,48,400,false,false,false,0);
printer_select_font($handle, $font);
printer_draw_text($handle, "test", 10, 10);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_end_doc (unknown)

Close document

bool **printer_end_doc** (resource handle) \linebreak

Closes a new document in the printer spooler. The document is now ready for printing. For an example see printer_start_doc(). *`handle`* must be a valid handle to a printer.

printer_end_page (unknown)

Close active page

bool **printer_end_page** (resource handle) \linebreak

The function closes the active page in the active document. For an example see printer_start_doc(). *`handle`* must be a valid handle to a printer.

printer_get_option (unknown)

Retrieve printer configuration data

mixed **printer_get_option** (resource handle, string option) \linebreak

The function retrieves the configuration setting of *`option`*. *`handle`* must be a valid handle to a printer. Take a look at printer_set_option() for the settings that can be retrieved, additionally the following settings can be retrieved:

- *`PRINTER_DEVICENAME`* returns the devicename of the printer.
- *`PRINTER_DRIVERVERSION`* returns the printer driver version.

Ejemplo 1. printer_get_option() example

```
$handle = printer_open();
print printer_get_option($handle, PRINTER_DRIVERVERSION);
printer_close($handle);
```

printer_list (unknown)

Return an array of printers attached to the server

array **printer_list** (int enumtype [, string name [, int level]]) \linebreak

The function enumerates available printers and their capabilities. *level* sets the level of information request. Can be 1,2,4 or 5. *enumtype* must be one of the following predefined constants:

- *PRINTER_ENUM_LOCAL*: enumerates the locally installed printers.
- *PRINTER_ENUM_NAME*: enumerates the printer of *name*, can be a server, domain or print provider.
- *PRINTER_ENUM_SHARED*: this parameter can't be used alone, it has to be OR'ed with other parameters, i.e. PRINTER_ENUM_LOCAL to detect the locally shared printers.
- *PRINTER_ENUM_DEFAULT*: (Win9.x only) enumerates the default printer.
- *PRINTER_ENUM_CONNECTIONS*: (WinNT/2000 only) enumerates the printers to which the user has made connections.
- *PRINTER_ENUM_NETWORK*: (WinNT/2000 only) enumerates network printers in the computer's domain. Only valid if *level* is 1.
- *PRINTER_ENUM_REMOTE*: (WinNT/2000 only) enumerates network printers and print servers in the computer's domain. Only valid if *level* is 1.

Ejemplo 1. printer_list() example

```
/* detect locally shared printer */
var_dump( printer_list(PRINTER_ENUM_LOCAL | PRINTER_ENUM_SHARED) );
```

printer_logical_fontheight (unknown)

Get logical font height

int **printer_logical_fontheight** (resource handle, int height) \linebreak

The function calculates the logical font height of *height*. *handle* must be a valid handle to a printer.

Ejemplo 1. printer_logical_fontheight() example

```
$handle = printer_open();
print printer_logical_fontheight($handle, 72);
printer_close($handle);
```

printer_open (unknown)

Open connection to a printer

mixed **printer_open** ([string devicename]) \linebreak

This function tries to open a connection to the printer *devicename*, and returns a handle on success or FALSE on failure.

If no parameter was given it tries to open a connection to the default printer (if not specified in php.ini as printer.default_printer, php tries to detect it).

printer_open() also starts a device context.

Ejemplo 1. printer_open() example

```
$handle = printer_open("HP Deskjet 930c");
$handle = printer_open();
```

printer_select_brush (unknown)

Select a brush

void **printer_select_brush** (resource printer_handle, resource brush_handle) \linebreak

The function selects a brush as the active drawing object of the actual device context. A brush is used to fill shapes. If you draw an rectangle the brush is used to draw the shapes, while the pen is used to draw the border. If you haven't selected a brush before drawing shapes, the shape won't be filled. *printer_handle* must be a valid handle to a printer. *brush_handle* must be a valid handle to a brush.

Ejemplo 1. printer_select_brush() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);
$brush = printer_create_brush(PRINTER_BRUSH_CUSTOM, "c:\\brush.bmp");
printer_select_brush($handle, $brush);
```

```
printer_draw_rectangle($handle, 1,1,500,500);

printer_delete_brush($brush);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "000000");
printer_select_brush($handle, $brush);
printer_draw_rectangle($handle, 1,501,500,1001);
printer_delete_brush($brush);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_select_font (unknown)

Select a font

void **printer_select_font** (resource printer_handle, resource font_handle) \linebreak

The function selects a font to draw text. *printer_handle* must be a valid handle to a printer. *font_handle* must be a valid handle to a font.

Ejemplo 1. printer_select_font() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 148, 76, PRINTER_FW_MEDIUM, false, false, false, -
50);
printer_select_font($handle, $font);
printer_draw_text($handle, "PHP is simply cool", 40, 40);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_select_pen (unknown)

Select a pen

void **printer_select_pen** (resource printer_handle, resource pen_handle) \linebreak

The function selects a pen as the active drawing object of the actual device context. A pen is used to draw lines and curves. I.e. if you draw a single line the pen is used. If you draw an rectangle the pen is used to draw the borders, while the brush is used to fill the shape. If you haven't selected a pen before drawing shapes, the shape won't be outlined. *printer_handle* must be a valid handle to a printer. *pen_handle* must be a valid handle to a pen.

Ejemplo 1. printer_select_pen() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "2222FF");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_set_option (unknown)

Configure the printer connection

bool **printer_set_option** (resource handle, int option, mixed value) \linebreak

The function sets the following options for the current connection. *handle* must be a valid handle to a printer. For *option* can be one of the following constants:

- *PRINTER_COPIES*: sets how many copies should be printed, *value* must be an integer.
- *PRINTER_MODE*: specifies the type of data (text, raw or emf), *value* must be a string.
- *PRINTER_TITLE*: specifies the name of the document, *value* must be a string.
- *PRINTER_ORIENTATION*: specifies the orientation of the paper, *value* can be either PRINTER_ORIENTATION_PORTRAIT or PRINTER_ORIENTATION_LANDSCAPE
- *PRINTER_RESOLUTION_Y*: specifies the y-resolution in DPI, *value* must be an integer.

- *PRINTER_RESOLUTION_X*: specifies the x-resolution in DPI, *value* must be an integer.
- *PRINTER_PAPER_FORMAT*: specifies the a predefined paper format, set *value* to PRINTER_FORMAT_CUSTOM if you want to specify a custom format with PRINTER_PAPER_WIDTH and PRINTER_PAPER_LENGTH. *value* can be one of the following constants.
  - *PRINTER_FORMAT_CUSTOM*: let's you specify a custom paper format.
  - *PRINTER_FORMAT_LETTER*: specifies standard letter format (8 1/2- by 11-inches).
  - *PRINTER_FORMAT_LETTER*: specifies standard legal format (8 1/2- by 14-inches).
  - *PRINTER_FORMAT_A3*: specifies standard A3 format (297- by 420-millimeters).
  - *PRINTER_FORMAT_A4*: specifies standard A4 format (210- by 297-millimeters).
  - *PRINTER_FORMAT_A5*: specifies standard A5 format (148- by 210-millimeters).
  - *PRINTER_FORMAT_B4*: specifies standard B4 format (250- by 354-millimeters).
  - *PRINTER_FORMAT_B5*: specifies standard B5 format (182- by 257-millimeter).
  - *PRINTER_FORMAT_FOLIO*: specifies standard FOLIO format (8 1/2- by 13-inch).

- *PRINTER_PAPER_LENGTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_LENGTH specifies a custom paper length in mm, *value* must be an integer.
- *PRINTER_PAPER_WIDTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_WIDTH specifies a custom paper width in mm, *value* must be an integer.
- *PRINTER_SCALE*: specifies the factor by which the printed output is to be scaled. the page size is scaled from the physical page size by a factor of scale/100. for example if you set the scale to 50, the output would be half of it's original size. *value* must be an integer.
- *PRINTER_BACKGROUND_COLOR*: specifies the background color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_COLOR*: specifies the text color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_ALIGN*: specifies the text alignment for the actual device context, *value* can be combined through OR'ing the following constants:
  - *PRINTER_TA_BASELINE*: text will be aligned at the base line.
  - *PRINTER_TA_BOTTOM*: text will be aligned at the bottom.
  - *PRINTER_TA_TOP*: text will be aligned at the top.
  - *PRINTER_TA_CENTER*: text will be aligned at the center.
  - *PRINTER_TA_LEFT*: text will be aligned at the left.
  - *PRINTER_TA_RIGHT*: text will be aligned at the right.

Ejemplo 1. printer_set_option() example

```
$handle = printer_open();
printer_set_option($handle, PRINTER_SCALE, 75);
printer_set_option($handle, PRINTER_TEXT_ALIGN, PRINTER_TA_LEFT);
printer_close($handle);
```

printer_start_doc (unknown)

Start a new document

bool **printer_start_doc** (resource handle [, string document]) \linebreak

The function creates a new document in the printer spooler. A document can contain multiple pages, it's used to schedule the print job in the spooler. *`handle`* must be a valid handle to a printer. The optional parameter *`document`* can be used to set an alternative document name.

Ejemplo 1. printer_start_doc() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_start_page (unknown)

Start a new page

bool **printer_start_page** (resource handle) \linebreak

The function creates a new page in the active document. For an example see printer_start_doc(). *`handle`* must be a valid handle to a printer.

printer_write (unknown)

Write data to the printer

bool **printer_write** (resource handle, string content) \linebreak

Writes *`content`* directly to the printer, and returns `TRUE` on success or `FALSE` if it failed.

`handle` must be a valid handle to a printer.

Ejemplo 1. printer_write() example

```
$handle = printer_open();
printer_write($handle, "Text to print");
printer_close($handle);
```

LXXXIV. Pspell Functions

The **pspell()** functions allow you to check the spelling of a word and offer suggestions.

You need the aspell and pspell libraries, available from http://aspell.sourceforge.net/ and http://aspell.net/ respectively, and add the `--with-pspell[=dir]` option when compiling php.

pspell_add_to_personal (PHP 4 >= 4.0.2)

Agrega la palabra a la lista personal de palabras

int **pspell_add_to_personal** (int dictionary_link, cadena palabra) \linebreak

pspell_add_to_personal() agrega una palabra a la lista personal de palabras. Si usas pspell_new_config() con pspell_config_personal() para abrir el diccionario, puedes salvar la lista de palabras luego, con pspell_save_wordlist(). Nota: Esta función no funcionará hasta que tengas pspell .11.2 y aspell .32.5 o superior.

Ejemplo 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_add_to_session (PHP 4 >= 4.0.2)

Agrega la palabra a la lista de palabras en la sesión actual

int **pspell_add_to_session** (int dictionary_link, cadena palabra) \linebreak

pspell_add_to_session() agrega la palabra a la lista de palabras asociada con la sesión actual. Esto es muy similar a pspell_add_to_personal()

pspell_check (PHP 4 >= 4.0.2)

Check a word

bool **pspell_check** (int dictionary_link, string word) \linebreak

pspell_check() checks the spelling of a word and returns `TRUE` if the spelling is correct, `FALSE` if not.

Ejemplo 1. pspell_check()

```
$pspell_link = pspell_new ("en");

if (pspell_check ($pspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

pspell_clear_session (PHP 4 >= 4.0.2)

Limpia la sesión actual

int **pspell_clear_session** (int dictionary_link) \linebreak

pspell_clear_session() limpia la sesión actual. La lista de palabras actual queda vacía, y, por ejemplo, si intentas salvarla con pspell_save_wordlist(), nada ocurre.

Ejemplo 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_clear_session ($pspell_link);
pspell_save_wordlist ($pspell_link);    //"Vlad" will not be saved
```

pspell_config_create (PHP 4 >= 4.0.2)

Create a config used to open a dictionary

int **pspell_config_create** (string language [, string spelling [, string jargon [, string encoding]]]) \linebreak

pspell_config_create() has a very similar syntax to pspell_new(). In fact, using **pspell_config_create()** immediatelly followed by pspell_new_config() will produce the exact same result. However, after

creating a new config, you can also use **pspell_config_*()** functions before calling pspell_new_config() to take advantage of some advanced functionality.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Ejemplo 1. pspell_config_create()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal ($pspell_config, "en");
```

pspell_config_ignore (PHP 4 >= 4.0.2)

Ignore words less than N characters long

int **pspell_config_ignore** (int dictionary_link, int n) \linebreak

pspell_config_ignore() should be used on a config before calling pspell_new_config(). This function allows short words to be skipped by the spellchecker. Words less then n characters will be skipped.

Ejemplo 1. pspell_config_ignore()

```
$pspell_config = pspell_config_create ("en");
pspell_config_ignore($pspell_config, 5);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "abcd");    //will not result in an error
```

pspell_config_mode (PHP 4 >= 4.0.2)

Change the mode number of suggestions returned

int **pspell_config_mode** (int dictionary_link, int mode) \linebreak

pspell_config_mode() should be used on a config before calling pspell_new_config(). This function determines how many suggestions will be returned by pspell_suggest().

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)

Ejemplo 1. pspell_config_mode()

```
$pspell_config = pspell_config_create ("en");
pspell_config_mode($pspell_config, PSPELL_FAST);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
```

pspell_config_personal (PHP 4 >= 4.0.2)

Set a file that contains personal wordlist

int **pspell_config_personal** (int dictionary_link, string file) \linebreak

pspell_config_personal() should be used on a config before calling pspell_new_config(). The personal wordlist will be loaded and used in addition to the standard one after you call pspell_new_config(). If the file does not exist, it will be created. The file is also the file where pspell_save_wordlist() will save personal wordlist to. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Ejemplo 1. pspell_config_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_repl (PHP 4 >= 4.0.2)

Set a file that contains replacement pairs

int **pspell_config_repl** (int dictionary_link, string file) \linebreak

pspell_config_repl() should be used on a config before calling pspell_new_config(). The replacement pairs improve the quality of the spellchecker. When a word is misspelled, and a proper suggestion was not found in the list, pspell_store_replacement() can be used to store a replacement pair and then pspell_save_wordlist() to save the wordlist along with the replacement pairs. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Ejemplo 1. pspell_config_repl()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_runtogether (PHP 4 >= 4.0.2)

Consider run-together words as valid compounds

int **pspell_config_runtogether** (int dictionary_link, bool flag) \linebreak

pspell_config_runtogether() should be used on a config before calling pspell_new_config(). This function determines whether run-together words will be treated as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Ejemplo 1. pspell_config_runtogether()

```
$pspell_config = pspell_config_create ("en");
pspell_config_runtogether ($pspell_config, true);
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_save_repl (PHP 4 >= 4.0.2)

Determine whether to save a replacement pairs list along with the wordlist

int **pspell_config_save_repl** (int dictionary_link, bool flag) \linebreak

pspell_config_save_repl() should be used on a config before calling pspell_new_config(). It determines whether pspell_save_wordlist() will save the replacement pairs along with the wordlist. Usually there is no need to use this function because if pspell_config_repl() is used, the replacement pairs will be saved by pspell_save_wordlist() anyway, and if it is not, the replacement pairs will not be saved. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

pspell_new_config (PHP 4 >= 4.0.2)

Load a new dictionary with settings based on a given config

int **pspell_new_config** (int config) \linebreak

pspell_new_config() opens up a new dictionary with settings specified in a config, created with with pspell_config_create() and modified with **pspell_config_*()** functions. This method provides you with the most flexibility and has all the functionality provided by pspell_new() and pspell_new_personal().

The config parameter is the one returned by pspell_config_create() when the config was created.

Ejemplo 1. pspell_new_config()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
```

pspell_new_personal (PHP 4 >= 4.0.2)

Load a new dictionary with personal wordlist

int **pspell_new_personal** (string personal, string language [, string spelling [, string jargon [, string encoding [, int mode]]]]) \linebreak

pspell_new_personal() opens up a new dictionary with a personal wordlist and returns the dictionary link identifier for use in other pspell functions. The wordlist can be modified and saved with pspell_save_wordlist(), if desired. However, the replacement pairs are not saved. In order to save replacement pairs, you should create a config using pspell_config_create(), set the personal wordlist file with pspell_config_personal(), set the file for replacement pairs with pspell_config_repl(), and open a new dictionary with pspell_new_config().

The personal parameter specifies the file where words added to the personal list will be stored. It should be an absolute filename beginning with '/' because otherwise it will be relative to $HOME, which is "/root" for most systems, and is probably not what you want.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)

- `PSPELL_RUN_TOGETHER` - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, `PSPELL_FAST`, `PSPELL_NORMAL` and `PSPELL_BAD_SPELLERS` are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Ejemplo 1. pspell_new_personal()

```
$pspell_link = pspell_new_personal ("/var/dictionaries/custom.pws",
        "en", "", "", "", PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_new (PHP 4 >= 4.0.2)

Load a new dictionary

int **pspell_new** (string language [, string spelling [, string jargon [, string encoding [, int mode]]]]) \linebreak

pspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other pspell functions.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are ’american’, ’british’, and ’canadian’.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are ’utf-8’, ’iso8859-*’, ’koi8-r’, ’viscii’, ’cp1252’, ’machine unsigned 16’, ’machine unsigned 32’. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)
- `PSPELL_RUN_TOGETHER` - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, `PSPELL_FAST`, `PSPELL_NORMAL` and `PSPELL_BAD_SPELLERS` are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Ejemplo 1. pspell_new()

```
$pspell_link = pspell_new ("en", "", "", "",
                           (PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_save_wordlist (PHP 4 >= 4.0.2)

Save the personal wordlist to a file

int **pspell_save_wordlist** (int dictionary_link) \linebreak

pspell_save_wordlist() saves the personal wordlist from the current session. The dictionary has to be open with pspell_new_personal(), and the location of files to be saved specified with pspell_config_personal() and (optionally) pspell_config_repl(). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Ejemplo 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/tmp/dicts/newdict");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_store_replacement (PHP 4 >= 4.0.2)

Store a replacement pair for a word

int **pspell_store_replacement** (int dictionary_link, string misspelled, string correct) \linebreak

pspell_store_replacement() stores a replacement pair for a word, so that replacement can be returned by pspell_suggest() later. In order to be able to take advantage of this function, you have to use pspell_new_personal() to open the dictionary. In order to permanently save the replacement pair, you have to use pspell_config_personal() and pspell_config_repl() to set the path where to save your custom wordlists, and then use pspell_save_wordlist() for the changes to be written to disk. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Ejemplo 1. pspell_store_replacement()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);

pspell_store_replacement ($pspell_link, $misspelled, $correct);
pspell_save_wordlist ($pspell_link);
```

pspell_suggest (PHP 4 >= 4.0.2)

Suggest spellings of a word

array **pspell_suggest** (int dictionary_link, string word) \linebreak

pspell_suggest() returns an array of possible spellings for the given word.

Ejemplo 1. pspell_suggest()

```
$pspell_link = pspell_new ("en");

if (!pspell_check ($pspell_link, "testt")) {
    $suggestions = pspell_suggest ($pspell_link, "testt");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br>";
    }
}
```

LXXXV. GNU Readline

Las funciones readline() implementan una interfaz con la librería GNU Readline. Un ejemplo de la manera de funcionar podría ser la forma en que el Bash permite usar las flechas de dirección para insertar carácteres o desplazarse a través del historial de comandos. Debido a la naturaleza interactiva de esta libreria, tendrá un uso muy reducido en la escritura de aplicaciones Web, aunque puede ser útil para scripts que han de ser ejecutados desde la consola.

La página principal del proyecto GNU Readline es http://cnswww.cns.cwru.edu/~chet/readline/rltop.html. Está actualizada por Chet Ramey, quien además es el autor de Bash.

readline_add_history (PHP 4)

Añade una línea al historial

void **readline_add_history** (string line) \linebreak

Esta función añade una línea al historial de líneas de comandos.

readline_clear_history (PHP 4)

Borra el historial

boolean **readline_clear_history** (void) \linebreak

Esta función borra por completo el historial de la línea de comandos.

readline_completion_function (PHP 4)

Registra una función de completitud

boolean **readline_completion_function** (string line) \linebreak

Esta función registra una función de completitud. Debe proporcionar el nombre de una función existente que acepte una línea de comandos parcial y devuelva una array con posibles coincidencias. Es el mismo tipo de funcionalidad que se obtiene al pulsar la tecla de tabulación cuando se está usando el Bash.

readline_info (PHP 4)

Establece/Obtiene diversas variables internas de readline

mixed **readline_info** ([string varname [, string newvalue]]) \linebreak

Si es llamada sin parámetros, esta función devuelve un array con los valores de todas las opciones que readline usa. Los elementos vendrán indexados por los siguientes valores: done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, y terminal_name.

Si es llamada con un parámetros, devuelve el valor de esa opción. Si es llamada con dos parámetros, el valor de la opción será cambiado al parámetro dado.

readline_list_history (PHP 4)

Lista el historial

array **readline_list_history** (void) \linebreak

Esta función devuelve un array con el historial de líneas de comandos completo. Los elementos están indexados por enteros comenzando por el cero.

readline_read_history (PHP 4)

Lee un historial

boolean **readline_read_history** (string filename) \linebreak

Esta función lee un historial de comandos desde un fichero.

readline_write_history (PHP 4)

Escribe el historial

boolean **readline_write_history** (string filename) \linebreak

Esta función escribe el historial de comandos en un archivo.

readline (PHP 4)

Lee una línea

string **readline** ([string prompt]) \linebreak

Esta función devuelve una única cadena del usuario. Puede especificar una cadena que se mostrará al usuario. La línea devuelta tiene el indicador final de nueva línea eliminado. Necesita añadir esta línea al historial usando la función readline_add_history().

Ejemplo 1. readline()

```
//obtiene 3 comandos del usuario
for ($i=0; $i < 3; $i++) {
        $line = readline ("Comando: ");
        readline_add_history ($line);
}

//Vuelca el historial
print_r (readline_list_history());

//Vuelca las variables
print_r (readline_info());
```

LXXXVI. Funciones GNU Recode

Este modulo contiene un interfaz para la biblioteca GNU Recode version 3.5. Para poder usar las funciones definidas en este modulo, debereis de compilar el interprete PHP con la opcion --with-recode. Para poder hacer esto debereis tener instalado en vuestro sistema GNU Recode 3.5 o superior.

La biblioteca GNU Recode convierte entre ficheros con diferentes codigos de caracteres y codificacion. Cuando esto no puede realizarse exactamente, puede desahacerse de los caracteres problematicos o crear una aproximacion. La biblioteca reconoce o produce alrededor de 150 codigos de caracteres y puede convertir ficheros entre casi todos los pares posibles. La gran mayoria de de codigos de caracteres RFC 1345 estan soportados.

recode_file (PHP 3>= 3.0.13, PHP 4)

Recodifica de fichero a fichero segun una peticion de recodificacion.

bool **recode_file** (int input, int output) \linebreak

Recodifica el fichero definido por *input* a el fichero definido por *output*, segun la peticion de recodificacion *request*. Devuelve FALSE si no puede realizar la recodificacion, TRUE si todo va bien.

Esta funcion no procesa ficheros remotos (URLs). Los dos ficheros deben de ser locales en el sistema.

recode_string (PHP 3>= 3.0.13, PHP 4)

Recodifica una cadena literal segun una peticion de recodificacion.

string **recode_string** (string request, string string) \linebreak

Recodifica la cadena *string* segun una peticion de recodificacion *request*. Devuelve FALSE si no puede realizar la recodificacion, TRUE si todo va bien.

Una simple peticion "recode" podria ser "lat1..iso646-de". Ver tambien la documentacion de GNU Recode de tu instalacion para obtener instrucciones detalladas sobre peticiones de recodificacion.

recode (PHP 4)

Recode a string according to a recode request

string **recode** (string request, string string) \linebreak

Nota: This is an alias for recode_string(). It has been added in PHP 4.

LXXXVII. Funciones de expresiones regulares compatibles con Perl

La sintaxis, para los patrones usados en estas funciones, es muy semejante al Perl. Las expresiones estarán encerradas por delimitadores, por ejemplo una barra de dividir (/). Cualquier carácter puede ser usado para delimitar incluso los que no son caracteres alfanuméricos o la barra invertida (\). Si el carácter delimitador ha sido usado en la propia expresión, es necesario que sea precedido por una barra inversa.

El delimitador de fin puede ser seguido por varios modificadores que afectarán al resultado. Examina Modificadores de Patrones.

Ejemplo 1. Ejemplos de patrones válidos

- /<\/\w+>/
- |(\d{3})-\d+|Sm
- /^(?i)php[34]/

Ejemplo 2. Ejemplos de patrones no válidos

- /href='(.*)' - falta el delimitador de fin
- /\w+\s*\w+/J - el modificador 'J' es desconocido
- 1-\d3-\d3-\d4| - falta el delimitador de inicio

> **Nota:** Para las funciones de expresiones compatibles con Perl se necesita PHP 4 o PHP 3.0.9 o superior.

Modificadores de Patrones (unknown)

describe los modificadores posibles en los patrones de expresiones regulares (regex)

Los posibles modificadores PRCE (Funciones de Expresiones Compatibles con Perl), en este momento, son mostrados a continuación. Los nombres entre paréntesis se refieren a nombres internos PCRE para dichos modificadores.

i (PCRE_CASELESS)

Si es usado, no se distinguirá entre mayúsculas y minúsculas.

m (PCRE_MULTILINE)

Por defecto, PCRE trata la cadena de entrada como si fuera una sola línea de caracteres (aun cuando tenga varias). El carácter especial de "inicio de línea" (^) empareja sólo al principio de la cadena, mientras el carácter especial de "fin de línea" ($) casa sólo el fin de la entrada, o antes un carácter de nueva línea (a menos que el modificador *E* sea definido). Esto es lo mismo que en Perl.

Cuando este modificador es utilizado, los constructores de "inicio de línea" y "fin de línea" son emparejados con el carácter de nueva línea. Esto es equivalente al modificador /m del Perl. Si no hay caracteres "\n" en la cadena de entrada, o no existen ^ o $ en el patrón, entonces este modificador no alterará el resultado.

s (PCRE_DOTALL)

Si se usa, el carácter especial de un punto en el patrón emparejará todos los caracteres, incluyendo el de nueva línea. Sin él, el carácter de nueva línea es excluido. Este modificador equivale a /s en Perl. Una cláusula como [^a] siempre casa con un carácter de nueva línea, independientemente de la utilización de este modificador.

x (PCRE_EXTENDED)

Si es definido, los caracteres de información con espacios en blanco en el patrón son ignorados excepto cuando son precedidos por una barra invertida o dentro de una clase carácter, y los caracteres entre un # fuera de una clase carácter y los siguientes caracteres de nueva línea, incluidos, son ignorados también. Esto es equivalente al /x en Perl y hace posible incluir comentarios dentro de patrones complejos. Sin embargo, esto es sólo aplicable a caracteres de información. Los caracteres de espacio en blanco nunca pueden aparecer en la secuencia de caracteres especiales de un patrón, por ejemplo en la secuencia (?(la cual introduce un subpatrón condicional.

e

Si es usado, preg_replace() hace las sustituciones \\ de forma habitual, evalúa el código PHP y usa el resultado para realizar una sustitución en la cadena de búsqueda.

Sólo preg_replace() hace uso de este modificador y es ignorado por las otras funciones PCRE.

Nota: Este modificador fue añadido en PHP 4.0.

A (PCRE_ANCHORED)

Si es definido, el patrón es forzado a ser "anclado", esto es, es obligado a emparejar sólo desde el inicio de la cadena (el "subject string"). Esta característica también puede realizarse con el apropiado patrón, y esta es la única manera de hacerlo en Perl.

E (PCRE_DOLLAR_ENDONLY)

Si es usado, el carácter del dólar en el patrón casará sólo con fin de la cadena de entrada (subject).Sin este modificador, un dólar es también emparejado con el carácter inmediatamente antes del de una nueva línea (pero no antes de cualquier otra nueva línea). Este modificador es ignorado si *m* es definido. No hay equivalente en Perl para este modificador.

S

Cuando un patrón va a ser usado varias veces, es mejor dedicar más tiempo a analizarlo para acelerar el proceso de casamientos. Si es definido entonces se realizar un análisis adicional. Estudiar a un patrón es sólo útil para los no anclados, esto es, no tienen un carácter de inicio fijado.

U (PCRE_UNGREEDY)

Este modificador invierte la "codicia" de los cuantificadores aunque no son ansiosos por defecto, se vuelven codiciosos si son seguidos por un "?". No es compatible con Perl. también puede usarse dentro del patrón.

X (PCRE_EXTRA)

Este modificador activa características adicionales del PCRE que no son compatible con Perl. Cualquier barra invertida en el patrón que sea seguida por una letra que no tenga una interpretación especial provocará un error, estas combinaciones están reservado para futuras ampliaciones. Por defecto, como en Perl, una barra invertida seguida por una letra sin un significado especial es tratada literalmente. No hay otras características controladas por este modificador a la fecha de hoy.

Sintaxis de los Patrones (unknown)

describe la sintaxis de PCRE regex

La librería PCRE es un conjunto de funciones que implementan emparejamientos dados patrones de expresiones regulares usando la misma sintaxis y semántica que Perl 5, con unas pocas diferencias (ver más adelante). La actual versión corresponde a Perl 5.005.

Las diferencias descritas aquí son con respecto a Perl 5.005.

1. Por defecto, un carácter de espacio en blanco es cualquier carácter que la función isspace() de la librería C reconozca, así es posible compilar PCRE con tablas alternativas de tipos de caracteres. Normalmente

isspace() casa con el espacio, salto de pagina, nueva línea, retorno de carro, tabulador horizontal y vertical. Perl 5 ya no incluye el tabulador vertical en su conjunto de caracteres de espacio en blanco. La secuencia de escape \n que estuvo durante mucho tiempo en la documentación de Perl nunca fue reconocida. Sin embargo, el carácter fue tratado como espacio en blanco hasta la 5.002. En 5.004 y 5.005 no casa \s.

2. PCRE no permite repetir cuantificadores sobre sentencias hacia adelante. Perl las permite, pero no de la forma que puedas pensar.
Por ejemplo, (?!a){3} no dice que los próximos tres caracteres no son "a". En realidad significa que los siguientes caracteres no son "a" tres veces.

3. Los subpatrones encontrados dentro de sentencias de más adelante negativas son contados, pero sus entradas en el vector de desplazamientos no son definidas. Perl define sus variables numéricas desde cualquiera de tales patrones que son casados antes de que la sentencia falle emparejar algo, pero solo si las sentencies de más adelante negativas contienen una opción sola.

4. Aunque los caracteres de cero binario son soportados en la cadena de entrada, no son permitidos en un patrón porque son pasados como un cadena típica de C, terminada por cero. La secuencia de escape "\0" puede ser usada en el patrón para representar el cero binario.

5. Las siguientes secuencias de Perl no son soportadas:
\l, \u, \L, \U, \E, \Q. En efecto, estas son implementadas por manipuladores de cadenas típicos de Perl y no son parte de los patrones del motor de búsqueda.

6. La secuencia \G de Perl no es soportada ya que no es relevante para emparejamientos de patrones sencillos.

7. Obviamente, PCRE no soporta el constructor (?{code})

8. Hay algunas diferencias en Perl 5.005_02 respecto a las definiciones de las cadenas de captura cuando parte de un patrón es repetido. Por ejemplo, casando "aba" con el patrón /^(a(b)?)+$/ define $2 al valor "b", pero emparejando "aabbaa" con /^(aa(bb)?)+$/ deja $2 sin definir. Sin embargo, si el patrón es cambiado a /^(aa(b(b))?)+$/ entones $2 (y $3) son definidos.

En Perl 5.004 $2 es definido en ambos casos, y también es cierto en PCRE. Si en el futuro Perl cambia a una regla diferente, PCRE puede cambiar para seguirla.

9. Otra discrepancia aún no resuelta es que en Perl 5.005_02 el patrón /^(a)?(?(1)a|b)+$/ casa la cadena "a", pero en PCRE eso no es así. Sin embargo, en ambos Perl y PCRE /^(a)?a/ empareja "a" dejando $1 sin definir.

10. PCRE da algunas extensiones para facilitar las expresiones de PERL:

(a) Aunque las sentencias de más adelante deben emparejar cadenas de longitud fija, cada opción de una sentencia de punto actual puede casar con una cadena de longitud diferente. Perl 5.005 requiere que todas ellas tengan la misma longitud.

(b) Si es definido PCRE_DOLLAR_ENDONLY y PCRE_MULTILINE no lo es, el carácter especial $ sólo casa con el final de la cadena.

(c) Si se define PCRE_EXTRA, una barra invertida seguida de una letra sin un significado especial provoca un error.

(d) Si defines PCRE_UNGREEDY, la voracidad de los cuantificadores de repetición es invertida, esto es, por defecto son no codiciosos, pero seguidos por una interrogación si lo son.

La sintaxis y la semántica de las expresiones soportadas por PCRE es descrita a continuación. Las expresiones son descritas en la documentación del Perl y en numerosos libros, algunos de los cuales tienen mucho ejemplares, Jeffrey Friedl's "Mastering Regular Expressions", publicado por O'Reilly (ISBN 1-56592-257-3), las cubre con gran detalle. La presente descripción es propuesta como documentación de referencia.

Una expresión es un patrón que es emparejada repetidamente, dada una cadena de entrada, de izquierda a derecha. Muchos caracteres se representan a ellos mismos en el patrón. Como un ejemplo trivial, el patrón

The quick brown fox

casa una parte de una cadena de entrada que es idéntica a ella. El poder de las expresiones proviene de la posibilidad de incluir alternativas y repeticiones en el patrón. Éstos son codificados en el patrón usando *meta-characters* (caracteres
especiales también llamados meta caracteres), los cuales no se representan a ellos mismos, en vez de eso, son interpretados de una manera especial.

Hay dos diferentes conjuntos de caracteres especiales: aquellos que son reconocidos en cualquier parte en el patrón excepto dentro corchetes ('[' y ']'), y aquellos que son reconocidos dentro. Fuera de los corchetes, los caracteres especiales son:

\ carácter de escape genérico con diferentes usos

^ secuencia de inicio de la cadena de entrada (o línea, en modo multilínea)
$ secuencia de fin de la cadena de entrada (o línea, en modo multilínea)
. empareja cualquier carácter excepto el de nueva línea (por defecto)
[inicia definición de clase de caracteres
| inicio de opción alternativa
(inicio de subpatrón
) fin de subpatrón
? amplia el significado de (
también es el cuantificador 0 ó 1
también es el cuantificador minimizado
* cero o más cuantificadores
+ uno o más cuantificadores
{ inicia el cuantificador min/max

Parte de un patrón dentro de corchetes ([]) es llamado un "character class" (clase de caracteres). En una clase de caracteres los únicos caracteres especiales son:

\ carácter de escape genérico
^ niega la clase, pero sólo si el primer carácter
- indica un rango de caracteres
] finaliza la clase de caracteres

Las secciones siguientes describen el uso de cada uno de los caracteres especiales (meta caracteres).

BARRA INVERTIDA

El carácter de barra invertida tiene varios usos. Primero, si es seguido por un carácter que no sea alfanumérico, toma el significado que el carácter pueda tener. Este uso de la barra invertida, como un carácter de escape, se aplica tanto dentro como fuera de las clases de caracteres.

Por ejemplo, si quieres casar un carácter "*", debes escribir "*" en el patrón. Esto es aplicable ya sea o no el carácter siguiente interpretado como un carácter especial, por eso siempre es aconsejable preceder un carácter no alfanumérico con "\" para especificar que se representa a él mismo. En particular, si quieres casar una barra invertida, escribe "\\".

Si el patrón es compilado con la opción PCRE_EXTENDED , los espacios en blanco en el patrón (fuera de una clase de caracteres) y los caracteres entre un "#" fuera de una clase de caracteres y el carácter de nueva línea son ignorado. Una barra invertida de escape puede usarse para incluir un espacio en blanco o el carácter "#" como parte del patrón.

Un segundo uso de la barra invertida sirve para codificar caracteres no imprimibles en los patrones de una manera visible. No hay restricciones sobre la apariencia de los caracteres no imprimibles, quitando el cero

binario de terminación de un patrón, pero cuando un patrón es preparado con un editor de texto, normalmente es fácil utilizar una de las siguientes secuencias de escape que representan sus caracteres binarios:

```
  \a     alarma, esto es, el carácter BEL (07 en hexadecimal)
  \cx    "control-x", donde x es cualquier carácter
  \e     escape (1B en hexadecimal)
  \f     nueva página (0C en hexadecimal)
  \n     nueva línea (0A en hexadecimal)
  \r     retorno de carro (0D en hexadecimal)
  \t     tabulador (09 en hexadecimal)
  \xhh   carácter con código hh en hexadecimal
  \ddd   carácter con código ddd en octal
```

El efecto de "\cx" es como sigue: si "x" es una letra minúscula, es convertida a mayúscula. Entonces el sexto bit del carácter (40 en hexadecimal) es invertido. Esto es, "\cz" es 1A en hexadecimal, pero "\c{" es 3B en hexadecimal, mientras "\c;" es 7B en hexadecimal.

Después de "\x", hasta dos dígitos hexadecimales son leídos (las letras pueden ser mayúsculas o minúsculas).

Después de "\0" son leídos dos dígitos octales más. En ambos casos, si hay menos de dos dígitos, se usrará lo que haya. Esto es, la secuencia "\0\x\07" indica dos ceros binarios seguidos por un carácter BEL. Asegúrate dar dos dígitos después del inicial cero si el carácter que sigue es un dígito octal.

El uso de una barra invertida seguido por otro dígito que no sea el cero es complejo. Fuera de una clase carácter, PCRE interpreta cualquier dígito como un número decimal. Si el número es menor que diez, o si ha habido al menos tantos paréntesis capturados a la izquierda en la expresión, entonces la secuencia entera es tomada como una *back reference* (referencia atrás). Una descripción de como trabaja esto es dada después, siguiendo la discusión de subpatrones con paréntesis.

Dentro de una clase carácter, o si el número decimal es mayor que nueve y no ha habido tantos subpatrones capturados PCRE relee los tres dígitos octales siguientes a la barra invertida y genera un byte desde los ocho bits menos significativos del valor. Cualquier dígito a continuación se representa a él mismo. Por ejemplo:

```
  \040   es otro modo de escribir un espacio
  \40    es lo mismo, siempre que haya menos de cuarenta subpatrones abiertos
  \7     siempre es una referencia atrás
  \11    puede ser una referencia atrás o un tabulador
  \011   siempre es un tabulador
  \0113  es el carácter con código octal 113 (ya que no puede haber más de
```

noventa y nueve referencias atrás)
\377 es un byte con todos sus bits a uno
\81 puede ser una referencia atrás o un cero binario seguido por
dos caracteres "8" y "1"

Ten en cuenta que el valor octal de un número mayor o igual a cien no debe ser precedido por un cero ya que no son leídos más de tres dígitos octales.

Todas las secuencias que definen el valor de un byte pueden ser usadas tanto dentro como fuera de la clase carácter. Además, la secuencia "\b" es interpretada como el carácter backspace (hex 08) dentro. Fuera es definido de otra manera (ver más adelante).

El tercer uso de la barra invertida es para especificar los tipos de caracteres genéricos:

\d cualquier un dígito decimal
\D cualquier carácter que no sea un dígito decimal
\s cualquier carácter de espacio en blanco (whitespace)
\S cualquier carácter que no sea un espacio en blanco
\w cualquier carácter de "palabra"
\W cualquier carácter que no se de "palabra"

Cada pareja de secuencia de escape divide el conjunto global de caracteres en dos. Cualquier carácter dado empareja en uno y sólo uno de cada pareja.

Un carácter de "palabra" es cualquier letra o dígito o el carácter subrayado, esto es, cualquier carácter puede ser parte de una "palabra" en Perl. La definición de letras y dígitos es controlada por la tabla de caracteres de PERL, y puede ser variada si las especificaciones regionales son tomadas en cuenta (ver "Soporte regional más adelante"). Por ejemplo, en Francia algunos caracteres tienen un código superior a 128, para representar las letras acentuadas, y son emparejados por \w.

Estas secuencias de tipos de caracteres pueden aparecer tanto dentro como fuera de las clases carácter. Cada una casa un carácter del tipo apropiado. Si el punto de casamiento actual es el final de la cadena, todo ello falla, ya que no hay más caracteres que casar.

El cuarto uso de la barra invertida es para ciertas sentencias (assertions). Una sentencia especifica una condición que tiene que ser encontrada en un punto particular de un emparejamiento, sin utilizar ningún carácter de la cadena de entrada. El uso de subpatrones para sentencias más complicadas es descrito después. Las sentencias de barra invertida son

\b limites de palabra
\B no sean limites de palabra
\A inicio de la cadena de entrada (independiente del modo multilínea)

\Z fin de la cadena de entrada o de una nueva línea delante del final
(independiente del modo multilínea)
\z fin de la cadena de entrada (independiente de modo multilínea)

Estas sentencias no pueden aparecer dentro de una clase carácter (pero ten en cuenta que "\b" tiene un significado diferente, quiere decir el carácter backspace dentro de una clase carácter)

Un límite de palabra es una posición en la cadena de entrada donde un carácter y el anterior no emparejan con \w o \W (por ejemplo, uno casa con \w y el otro con \W), o el principio o el final de la cadena si el primero o el último carácter emparejan con \w, respectivamente.

Las sentencias \A, \Z y \z se diferencian de los tradicionales circunflejo y dólar (ver más adelante) en que sólo emparejan el inicio y fin de la cadena de entrada sin tener en cuenta las opciones definidas. No les afectan las opciones PCRE_NOTBOL o PCRE_NOTEOL. La diferencia entre \Z y \z es que \Z casa antes una nueva línea que es el último carácter de la cadena como también el final de la cadena, sin embargo \z sólo casa el final.

CIRCUNFLEJO Y DOLAR

Fuera de una clase carácter, en el modo de emparejamiento por defecto, el carácter circunflejo es una sentencia la cual es verdadera sólo si el punto de casamiento actual es el inicio de la cadena de entrada. Dentro de una clase carácter, el circunflejo tiene significado completamente distinto (ver más adelante).

El circunflejo no necesita ser el primer carácter del patrón si son posibles un número de alternativas, pero será la primera cosa en cada alternativa en la cual aparezca si el patrón casa esa opción.
Si todas las alternativas posibles empiezan con un circunflejo, esto es, si el patrón es obligado a casar sólo con en el inicio de la cadena de entrada, se dice que es un patrón "anclado". También hay otros constructores que pueden hacer que un patrón sea anclado.

Un carácter de dólar es una sentencia que es verdadera sólo si el punto de emparejamiento actual es el final de la cadena de entrada, o inmediatamente antes de un carácter de nueva línea, el cual es el último carácter en la cadena, por defecto. El dólar no necesita ser el último carácter del patrón si hay varias alternativas, pero será el último elemento en cualquier alternativa en el que aparezca. El dólar no tiene un significado especial en una clase carácter.

El significado del dólar puede ser cambiado para que sólo empareje el final de la cadena de entrada definiendo la opción PCRE_DOLLAR_ENDONLY a la hora de compilar o tiempo de ejecución. Esto no afecta a la sentencia \Z.

El significado de los caracteres circunflejo y dólar cambia si la opción PCRE_MULTILINE es definida. Cuando éste es el caso, casan, respectivamente, inmediatamente antes y después de un carácter "\n" interno, además de emparejar con el inicio y el final de la cadena. Por ejemplo, el patrón /^abc$/ casa con la cadena de entrada "def\nabc" en modo multilínea, pero en otro modo no. Consecuentemente, los patrones anclados son en modo línea ya que todas las opciones que empiezan con "^" no son ancladas en modo multilínea. La opción PCRE_DOLLAR_ENDONLY es ignorada si PCRE_MULTILINE es definido.

Ten en cuenta que las secuencias \A, \Z y \z pueden ser usadas para casar el inicio y el final de la cadena en ambos modos, y si todas las opciones de un patrón empiezan con \A siempre es anclado, independientemente de si PCRE_MULTILINE es definido o no.

FINAL (PUNTO)

Fuera de una clase carácter, un punto en el patrón casa con un carácter cualquiera en la cadena de entrada, incluyendo un carácter no imprimible, exceptuando el de nueva línea (por defecto). Si la opción PCRE_DOTALL es definida, entonces los puntos casan con los de nueva línea también. El manejo de puntos es completamente independiente del uso del circunflejo y el dólar, la única relación entre ellos son los caracteres de nueva línea. Los puntos no tienen un significado especial dentro de una clase carácter.

CORCHETES

Un corchete de apertura crea una clase carácter, terminada por uno de cierre. Un corchete de cierre no tiene un significado especial. Si un corchete de cierre es necesitado como un miembro de la clase, será el primer carácter de datos en la clase (después de un circunflejo inicial, si está presente) o con una barra invertida antes.

Si una clase carácter casa con un carácter único en la cadena; el carácter debe estar en el conjunto de los caracteres definidos por la clase, a menos que el primero sea un circunflejo, en cuyo caso el carácter de la cadena de entrada no debe estar en el conjunto definido por la clase. Si un circunflejo es necesitado como un miembro de la clase, asegúrate que no es el primero o es precedido por una barra invertida.

Por ejemplo, la clase carácter [aeiou] empareja cualquier vocal minúscula, mientras [^aeiou] casa cualquier carácter que no sea una vocal minúscula. Ten en cuenta que un circunflejo es una notación convenida para especificar los caracteres que están en la clase enumerando los que no lo están. No es una sentencia: consume un carácter de la cadena de entrada y falla si el punto actual es final.

Cuando se define el emparejamiento sin tener en cuenta mayúsculas y minúsculas (caseless), cualquier letra en una clase representa ambas, por ejemplo,

un patrón caseless [aeiou] empareja tanto "A" como "a" y un caseless [^aeiou] no casa con "A"

El carácter de nueva línea nunca es tratado de un modo especial en una clase carácter, aunque se hallan definido cualquiera de las opciones PCRE_DOTALL o PCRE_MULTILINE. Una clase como [^a] siempre casa con una nueva línea.

El carácter de menos puede ser usado para especificar un rango de caracteres en una clase miembro. Por ejemplo, [d-m] casa con cualquier letra entre d y m ambas incluidas. Si un carácter de menos es necesitado en una clase, debe ser precedido por una barra invertida o aparecer en una posición donde no pueda ser interpretado como indicador de una rango, normalmente al inicio o al final de la clase.

No es posible tener el carácter literal "]" como el de final de un rango. Un patrón como [W-]46] es interpretado como una clase de dos caracteres ("W" y "-") seguido por la cadena literal "46]", por lo que emparejaría con "W46]" o "-46]". Sin embargo, si el carácter "]" es precedido con una barra invertida es tomado por el final del rango, así [W-\]46] es interpretado como una clase conteniendo un rango seguido por dos caracteres. La representación octal o hexadecimal de "]" puede ser usada para finalizar un rango.

Los rangos trabajan en la secuencia ASCII. Se pueden especificar mediante la representación numérica de los mismos, por ejemplo [\000-\037]. Si un rango que incluye letras es usado cuando es definida la opción de no tener en cuenta mayúsculas y minúsculas casan ambas. Por ejemplo, [W-c] es equivalente a [][\^_‘wxyzabc], teniendo en cuenta mayúsculas y minúsculas, y si la tabla de caracteres para la región "fr" es usada, entonces [\xc8-\xcb] empareja los caracteres E acentuados en ambos casos.

Los tipos de caracteres \d, \D, \s, \S, \w, y \W también pueden aparecer en una clase carácter y añaden los caracteres que ellos casen para la clase. Por ejemplo, [\dABCDEF] casa cualquier dígito hexadecimal. Un circunflejo puede ser usado convenientemente con el tipo de carácter mayúsculo para especificar un conjunto más restrictivo de caracteres que el de un casamiento con tipo de carácter minúsculo. Por ejemplo, la clase [^\W_] empareja cualquier letra o digito pero no el subrayado.

Todos los caracteres no alfanuméricos y los diferentes a \, -, ^ (al principio) y] no tienen un significado especial en una clase, y éstos tampoco si son definidos convenientemente.

BARRA VERTICAL

Los caracteres de barra vertical son usados para separar patrones alternativos. Por ejemplo, el patrón

gilbert|sullivan

casa con "gilbert" o "sullivan". Cualquier cantidad de opciones pueden ser implementadas, y una alternativa vacía se permite (emparejando la cadena vacía). El proceso de casamiento intenta cada una de izquierda a derecha, y la primera que valga es usada. Si las alternativas están dentro de un subpatrón, "valga" significa que casa el resto del patrón principal como también la alternativa en el subpatrón.

DEFINIENDO LAS OPCIONES INTERNAS

Las definiciones de PCRE_CASELESS, PCRE_MULTILINE, PCRE_DOTALL, y PCRE_EXTENDED pueden ser cambiadas desde dentro del patrón mediante una secuencia de letras de opciones de Perl encerradas entre "(?" y ")". Las letras de opciones son

i para PCRE_CASELESS
m para PCRE_MULTILINE
s para PCRE_DOTALL
x para PCRE_EXTENDED

Por ejemplo, (?im) define sin tener en cuenta mayúsculas y minúsculas y modo multilínea. También es posible eliminar estas opciones precediendo las letras con un menos y una combinación de definiciones y eliminaciones tal como (?im-sx), la cual define PCRE_CASELESS y PCRE_MULTILINE mientras elimina PCRE_DOTALL y PCRE_EXTENDED, también se permite. Si una letra aparece antes y después del menos, la opción es eliminada.

El ámbito de estas opciones cambia dependiendo dónde ocurra la definición. Las definiciones que son hechas fuera de subpatrones (como antes), el efecto es el mismo que si la opción se define o elimina al inicio del casamiento. Los siguientes patrones se comportan todos de la misma manera:

(?i)abc
a(?i)bc
ab(?i)c
abc(?i)

el cual tiene el mismo efecto que compilar el patrón abc con la opción PCRE_CASELESS. En otras palabras, tales definiciones de "nivel superior" se aplican a todo el patrón (a menos que haya otro cambio dentro del subpatrón). Si hay más de una definición de la misma opción en el mismo nivel superior, la definición más a la derecha se usa.

Si un cambio de opción sucede dentro de un subpatrón, el efecto es diferente. Esto es un cambio respecto de la conducta de Perl 5.005. Un cambio de opción dentro de un subpatrón afecta sólo a la parte del subpatrón que lo sigue,

por eso

(a(?i)b)c

empareja abc y aBc y ninguna otra cadena (asumiendo que no es usado PCRE_CASELESS). De este modo, las opciones pueden ser hechas para tener diferente significado en diferente partes del patrón. Cualquier cambio realizado en una alternativa provoca que todo el subpatrón la use. Por ejemplo,

(a(?i)b|c)

empareja "ab", "aB", "c", y "C", siempre y cuando case "C" la primera opción es abandonada antes de definir la opción. Esto es porque los efectos de definiciones de opción ocurren en tiempo de compilación. De otro modo, éstos sería una conducta muy rara.

Las opciones específicas PCRE PCRE_UNGREEDY y PCRE_EXTRA pueden ser cambiadas del mismo modo que las opciones compatibles con Perl usando los caracteres U y X respectivamente. La bandera (?X) es especial ya que siempre debe aparecer antes que cualquier otra en el patrón, incluso cuando es definida a nivel superior. Es mejor ponerla en el inicio.

SUBPATRONES

Los subpatrones son delimitados por paréntesis y pueden estar anidados. Marcando parte de un patrón como un subpatrón permite dos cosas:

1. Define un conjunto de opciones. Por ejemplo, el patrón

cat(aract|erpillar|)

empareja con "cat", "cataract", or "caterpillar". Sin los paréntesis, casaría "cataract", "erpillar" o la cadena vacía.

2. Define el subpatrón como un subpatrón capturado. Cuando el patrón sea emparejado por completo, esa porción de la cadena de entrada que casa con el subpatrón es devuelta mediante el argumento *ovector* de **pcre_exec()**. Los paréntesis abiertos son contados de izquierda a derecha (empezando por uno) para definir los números de subpatrones capturados.

Por ejemplo, si la cadena "the red king" es casada con el patrón

the ((red|white) (king|queen))

las subcadenas capturadas son "red king", "red", y "king" y los números son 1,2 y 3

El hecho de que los paréntesis realicen dos funciones no siempre es útil. A menudo, hay veces que un subpatrón agrupado es necesitado sin una querer una captura. Si un paréntesis abierto le sigue "?:", el subpatrón no hace ninguna captura, y no es contado cuando compute el número de subpatrones capturados. Por ejemplo, si la cadena "the white queen" es casada con el patrón

```
the ((?:red|white) (king|queen))
```

las subcadenas capturadas son "white queen" y "queen" y son numeradas como 1 y 2. El número máximo de subcadenas es de 99 y el número máximo de subpatrones, capturados o no, es de 200.

Como un atajo, si cualquier definición de opción es necesitada al inicio de un subpatrón no capturado, las letras de opciones pueden aparecer entre "?" y ":". Así los dos patrones

```
(?i:saturday|sunday)
(?:(?i)saturday|sunday)
```

emparejan como el mismo conjunto de cadena de entrada exactamente. Ya que las alternativas son intentadas de izquierda a derecha, y las opciones no son dejadas de tener en cuenta hasta que el final de subpatrón se alcanza, una definición de opción en una alternativa afecta al resto, por eso el patrón anterior empareja tanto con "SUNDAY" como con "Saturday".

REPETICION

La repetición es especificada por cuantificadores, la cual puede utilizarla cualquiera de los siguientes elementos:

```
un carácter, posiblemente precedido por el meta carácter .
una clase carácter
una referencia atrás (ver la próxima sección)
un subpatrón con paréntesis (a menos que sea una sentencia, ver más adelante)
```

El cuantificador de repetición general indica un número mínimo y un máximo de casamientos permitidos, dando los dos números entre llaves, separados por coma. El número debe ser menor que 65536, y el primero debe ser menor o igual que el segundo. Por ejemplo:

```
z{2,4}
```

casa con "zz", "zzz", o "zzzz". Una llave de cierre por si misma no es un carácter especial. Si el segundo número es omitido, pero aparece la coma, entonces no hay límite superior; si el segundo número y la coma son omitidos, el cuantificador indica el número exacto de repeticiones. Así

```
[aeiou]{3,}
```

empareja al menos tres vocales seguidas, pero pueden ser muchas más, mientras

```
\d{8}
```

casa exactamente ocho dígitos. Una llave abierta en una posición donde un cuantificador no es permitido o una que no empareje con la sintaxis de un cuantificador es tomada como un carácter literal. Por ejemplo, {,6} no es un cuantificador, pero sí una cadena literal de cuatro caracteres.

Se permite el cuantificado {0}, provocando que la expresión se comporte como si el elemento anterior y el cuantificador no estuvieran presentes.

Por conveniencia (y compatibilidad histórica) los cuantificadores más comunes tienen abreviaciones de un solo carácter.

```
*    es equivalente a {0,}
+    es equivalente a {1,}
?    es equivalente a {0,1}
```

Es posible construir bucles infinitos mediante un subpatrón que pueda casar ningún carácter con un cuantificador que no tenga límite superior, por ejemplo:

```
(a?)*
```

Las primeras versiones de Perl y PCRE dan un error en tiempo de compilación para tales patrones. Sin embargo, ya que existen casos donde esto puede ser útil, esto patrones son aceptados ahora, pero si cualquier repetición del subpatrón no casa ningún carácter, el bucle es roto.

Por defecto, los cuantificadores son "codiciosos", esto es, casan tantas veces como les es posible (hasta el número máximo de veces permitido), sin provocar que el resto del patrón falle. El ejemplo clásico de donde viene este problema es en intentar casar comentarios en los programas en C. Estos aparecen entre las secuencias /* y */ y dentro de la secuencia los caracteres * y / pueden aparecer individualmente. Un modo de casar comentarios en C es aplicando el patrón

```
/\*.*\*/
```

para la cadena

```
/* first command */  not comment  /* second comment */
```

falla, porque casa la cadena entera debido a la voracidad del elemento .*

Sin embargo, si un cuantificador le siguw un signo de interrogación entonces cesa la voracidad y empareja el mínimo número de veces posibles, así el patrón

```
/\*.*?\*/
```

hace las cosas correctamente con los comentarios en C. El significado de los cuantificadores variables no es cambiado en otro modo, justo el número preferido de casamientos. No confundas el uso de las interrogaciones con su uso como un cuantificador mas. Ya que tiene dos usos, a veces puede parecer doble, como en

```
\d??\d
```

el cual empareja un dígito normalmente, pero puede casar dos si ese el único modo de casar el resto del patrón.

Si se define la opción PCRE_UNGREEDY (la cual no es posible en Perl) entonces los cuantificadores no son voraces por defecto, pero uno puede serlo seguido por una interrogación. En otras palabras, invierte la conducta por defecto.

Cuando un subpatrón entre paréntesis es cuantificado con un número mínimo de repeticiones superior a uno o con un límite máximo, se necesita más almacenamiento para compilar el patrón, en proporción al tamaño del mínimo o del máximo.

Si un patrón empieza con .* o .{0,} y la opción PCRE_DOTALL (equivalente a /s del Perl) es definida, esta permitiendo el . para casar nuevas líneas, entonces el patrón es anclado implícitamente. PCRE trata tales patrones como si estuvieran precedidos por \A. En los casos donde se conoce que la cadena de entrada no contiene nuevas líneas, es conveniente definir PCRE_DOTALL cuando el patrón empieza con .* para obtener esta optimización o usar ^ para indicar explícitamente anclamiento.

Cuando un subpatrón capturado es repetido, el valor capturado es la subcadena que empareja la iteración final. Por ejemplo, el patrón

```
(tweedle[dume]{3}\s*)+
```

con la cadena de entrada "tweedledum tweedledee" el valor de la subcadena capturada es "tweedledee". Sin embargo, si hay subpatrones capturados anidadamente, los valores capturados correspondientes pueden haber sido definidos en las iteraciones anteriores. Por ejemplo, después de casar "aba" con

```
/(a|(b))+/
```

el valor de la segunda subcadena capturada es "b".

REFERENCIAS ATRAS

Fuera de una clase carácter, una barra invertida seguida por un digito mayor que cero (y posiblemente más dígitos) es una referencia atrás a un subpatrón capturado antes (a su izquierda) en el patrón, siempre que haya habido tantos paréntesis a la izquierda capturados.

Sin embargo, si el número decimal seguido por la barra invertida es menor que diez, siempre es tomado como una referencia atrás, y da error sólo si no hay los suficientes subpatrones capturados en todo el patrón. En otras palabras, los paréntesis que son referidos no necesitan estar a la izquierda de la referencia para un número menor de diez. Examina la sección anterior titulada "Barra invertida" para más detalles del manejo de los dígitos con la barra invertida.

Una referencia atrás empareja si casa el subpatrón capturado en el actual punto de la cadena de entrada, mejor que casar cualquier subpatrón de la misma. Así el patrón

```
(sens|respons)e and \1ibility
```

casa con "sense and sensibility" y "response and responsi bility", pero no "sense and responsibility". Si el casamiento con la distinción entre minúsculas y mayúsculas está activado en el momento de la referencia atrás, entonces la distinción de las letras es relevante. Por ejemplo,

```
((?i)rah)\s+\1
```

casa con "rah rah" y "RAH RAH", pero no "RAH rah", pero el subpatrón capturado originalmente es emparejado sin la distinción.

Puede haber más de una referencia atrás en el mismo subpatrón. Si un subpatrón no ha sido usado en un emparejamiento particular, entonces cualquier referencia atrás siempre fallara. Por ejemplo, el patrón

```
(a|(bc))\2
```

fallará siempre si inicia a casar con "a" mejor que con "bc". Ya que puede haber hasta 99 referencias atrás, todos los dígitos seguidos por una barra invertida son tomados como parte de número potencial de referencias atrás. Si el patrón continua con un carácter de dígito, entonces algún delimitador debe ser usado para terminar la referencia atrás. Si la opción PCRE_EXTENDED es definida, este puede ser el espacio en blanco. De otro modo un comentario vació puede ser usado.

Una referencia atrás ocurre dentro del paréntesis al cual refiere, falla cuando el subpatrón es usado por primera vez, así por ejemplo, (a\1) nunca emparejará. Sin embargo, tal referencia puede ser útil dentro de los subpatrones repetidos. Por ejemplo, el patrón

```
(a|b\1)+
```

casa con cualquier número de "a"s y también con "aba", "ababaa" etc. Para cada iteración del subpatrón, la referencia atrás casa la cadena de caracteres correspondiente a la iteración anterior. Para que esto trabaje, el patrón debe ser tal que la primera iteración no necesite casar la referencia atrás. Esto puede hacerse usando alternativas, como en el ejemplo anterior, o por medio de cuantificadores con un número mínimo de cero.

SENTENCIAS

Una sentencia es un test sobre los caracteres siguiendo o precediendo el punto actual de emparejamiento que no consume caracteres. Las sentencias codificadas como \b, \B, \A, \Z, \z, ^ y $ son descritas después. Las sentencias más complejas son codificadas como subpatrones. Hay dos clases: aquellas que condicionan más adelante de la posición actual en la cadena de entrada (lookahead)y las que lo hacen en este punto (lookbehind).

Un subpatrón de sentencia es emparejado del modo típico, excepto que no hace que el punto actual de emparejamiento cambie. Sentencias que condicionan más adelante empiezan con (?= para sentencias afirmativas y (?! para las negativas

```
\w+(?=;)
```

empareja una palabra seguida por un punto y coma. pero no incluye el punto y coma en el casamiento, y

```
foo(?!bar)
```

casa cualquier ocurrencia de "foo" que no es seguida por "bar". Ten en cuenta que el patrón similar

```
(?!foo)bar
```

no encuentra una ocurrencia de "bar" que es precedida por algo que no sea "foo"; encuentra cualquier ocurrencia de "bar", ya que la sentencia (?!foo) es siempre verdadera cuando los tres primeros caracteres son "bar". Una sentencia en el punto actual es necesaria para realizar este efecto. Las sentencias de punto actual empiezan con (?<= para sentencias afirmativas y (?<! para las negativas. Por ejemplo,

```
(?<!foo)bar
```

encuentra una ocurrencia de "bar" que no es precedida por "foo".
Los contenidos de un sentencia de punto actual están limitados para que todas las cadenas que emparejen deban tener una longitud fijada. Sin embargo, si hay varias alternativas, no todas tienen que tener la misma longitud. Así

```
(?<=bullock|donkey)
```

es permitido, pero

```
(?<!dogs?|cats?)
```

da error en tiempo de compilación. Opciones que emparejen diferentes longitudes de cadenas son permitidas sólo a nivel superior de la sentencia de punto actual. Ésta es una extensión comparada con Perl 5.005, la cual requiere que todas las opciones a casar tengan la misma longitud. Una sentencia como

```
(?<=ab(c|de))
```

no es permitida, ya que sus opciones a nivel superior pueden casar dos longitudes diferentes, pero es aceptable si se rescribe para usar dos opciones a nivel superior:

```
(?<=abc|abde)
```

La implementación de sentencias de punto actual es, para cada alternativa, mover temporalmente la posición actual hacia atrás por la longitud fijada e intentar casar. Si no hay suficientes caracteres antes de la posición actual, fallará. Las sentencias de punto actual en unión con subpatrones de sólo una vez pueden ser particularmente útiles para emparejamientos de finales de cadenas; un ejemplo es dado al final de la sección sobre subpatrones de una sola vez.

Varias sentencias (de cualquier tipo) pueden suceder consecutivamente.
Por ejemplo,

```
(?<=\d{3})(?<!999)foo
```

empareja "foo" precedido por tres dígitos que no sean "999".
Además, las sentencias puede ser anidadas en cualquier combinación. Por ejemplo,

```
(?<=(?<!foo)bar)baz
```

empareja una ocurrencia de "baz" que es precedida por "bar" la cual no sea precedida por "foo".

Los subpatrones de sentencias no son subpatrones capturados, y no pueden ser repetidos, ya que no tiene sentido la misma cosa varias veces. Si una sentencia contiene subpatrones capturados dentro de ella, éstos son siempre contados para el propósito de la numeración de los subpatrones capturados en todo el patrón. Las subcadenas capturadas son tenidas en cuenta para las sentencias afirmativas, pero no para las negativas (no tiene sentido).

El contador de sentencias llega hasta un máximo de doscientos subpatrones con paréntesis.

SUBPATRONES DE UNA SOLA VEZ

Maximizando y minimizando las repeticiones para ver si un número diferente de éstas permite al resto del patrón emparejar, causa múltiples evaluaciones de la cadena de entrada. A veces es útil prevenir esto, cambiando el patrón o provocando que la repetición falle pronto, cuando el creador del patrón conoce que no hay puntos en común.

Considera, por ejemplo, el patrón \d+foo cuando se aplica a esta cadena de entrada

```
123456bar
```

Después de emparejar los seis dígitos falla al emparejar "foo", la acción normal del casamiento es intentar otra vez con sólo cinco dígitos que emparejen con el elemento \d+, y entonces con cuatro, y así, antes de fallar. Subpatrones de una sola vez dan el modo de especificar que una parte del patrón tiene que emparejar, no es re-evaluado de esta manera, así el casamiento fallará al emparejar "foo" la primera vez. La notación es otra clase de paréntesis especial, iniciado con (?gt; como en este ejemplo:

```
(?>\d+)bar
```

Esta clase de paréntesis "bloquean" la parte del patrón que tiene que ser emparejada una vez y un fallo impide que la re-evalue.

Una descripción alternativa es que un subpatrón de este tipo case los caracteres de la cadena que un patrón fijo emparejaría, si estuviera anclado en el punto actual de la cadena de entrada.

Subpatrones de una sola vez no son subpatrones capturados. Estos casos tal como el ejemplo anterior pueden ser interpretado como de una repetición maximizada que debe tragar todo lo que pueda.
Por esto, mientras ambos \d+ y \d? están preparados para ajustar el número de dígitos que emparejan para hacer que el resto del patrón case, (?>\d+) sólo puede emparejar un secuencia de dígitos entera.

Esta construcción, por supuesto, puede contener subpatrones arbitriamente

complicados y pueden estar anidados.

Subpatrones de una sola vez pueden usarse con sentencias de punto actual para especificar eficientes emparejamientos al final de la cadena de entrada. Consideremos un patrón sencillo como este

```
abcd$
```

cuando se aplica a una cadena larga con la cual no empareja. Ya que el casamiento va de izquierda a derecha, PCRE buscará cada "a" en la cadena y entonces verá si lo que sigue casa con el resto del patrón. Si el patrón se escribe así

```
^.*abcd$
```

entonces el .* inicial casará primero la cadena entera, pero cuando esto falle, volverá atrás para emparejar todo menos el último carácter, entonces los dos últimos y así sucesivamente. Otra vez la búsqueda de "a" cubre la cadena completa, de derecha a izquierda, de esta manera no se mejora. Sin embargo, si el patrón fuese este

```
^(?>.*)(?<=abcd)
```

entonces no hay vuelta atrás para el elemento .*; sólo puede casar la cadena entera. La sentencia de punto actual subsiguiente hace un test sencillo sobre los últimos cuatro caracteres. Si falla, el casamiento inmediatamente da un resultado negativo. Para cadena largas, este acercamiento da una diferencia significativa en tiempo de ejecución.

SUBPATRONES CONDICIONALES

Es posible hacer que el casamiento procese un subpatrón condicionalmente o elegir entre dos subpatrones alternativos, dependiendo del resultado de una sentencia o si un subpatrón capturado previamente casó o no.

Las dos formas posibles de subpatrones condicionales son

```
(?(condition)yes-pattern)
(?(condition)yes-pattern|no-pattern)
```

Si la condición es satisfecha, el yes-pattern es usado; sino el no-pattern es utilizado si existe. Si hay más de dos alternativas en el subpatrón, se produce un error en tiempo de compilación.

Hay dos clases de condiciones. Si el texto entre los paréntesis consiste de una secuencia de dígitos, entonces la condición es verdadera si el subpatrón capturado de ese número ha sido casado previamente. Consideremos el siguiente patrón, contiene espacios en blanco para hacerlo más leíble

(asumimos la opción PCRE_EXTENDED) y lo dividimos en tres partes para facilitar la discusión:

```
( \( )?    [^()]+    (?(1) \) )
```

La primera parte empareja un paréntesis opcional abierto, y si el carácter esta presente, lo define como la primera subcadena capturada. La segunda parte casa uno o más caracteres que no están entre paréntesis. La tercera parte es un subpatrón condicional que examina si el primer conjunto de paréntesis casa o no. Si fuera así, esto es, si la cadena de entrada empieza por un paréntesis abierto, la condición es cierta, y el yes-pattern es ejecutado y un paréntesis de cierre es requerido. De otro modo, ya que no-pattern no esta presente, el subpatrón no casa con nada. En otras palabras, este patrón casa una secuencia de datos sin paréntesis opcionalmente limitada por ellos.

Si la condición no es una secuencia de dígitos, debe ser una sentencia. Esto puede ser una sentencia de más adelante positiva o negativa o una de punto actual. Consideremos este patrón, otra vez conteniendo espacios en blanco sin significado y con la segunda alternativa en la siguiente línea:

```
(?(?=[^a-z]*[a-z])
\d{2}[a-z]{3}-\d{2} | \d{2}-\d{2}-\d{2} )
```

La condición es una sentencia de más adelante positiva que empareja una secuencia opcional de cualquier cosas menos letras seguido por una letra. En otras palabras, examina la presencia de al menos una letra en la cadena de entrada. Si una letra es encontrada, la cadena es casada con la primera alternativa; sino lo es con la segunda. Este patrón casa cadenas de una de estas dos formas dd-aaa-dd o dd-dd-dd, donde aaa son letra y dd son dígitos.

COMENTARIOS

La secuencia (?# marca el inicio de un comentario el cual continua hasta el primer paréntesis. Los paréntesis anidados no son permitidos. Los caracteres que forman un comentario no forman parte del patrón de emparejamiento.

Si la opción PCRE_EXTENDED es definida, un carácter # fuera de una clase carácter crea un comentario que continua hasta la próxima línea del patrón.

RENDIMIENTO

Ciertos elementos que pueden aparecer en los patrones son más eficientes que otros. Es más eficiente usar una clase carácter como [aeiou] que un conjunto de alternativas tal como (a|e|i|o|u). En general, los constructores más sencillos que dan la conducta requerida son, normalmente, más eficientes. El libro de Jeffrey Friedl contiene un montón de discusiones sobre la

optimización de expresiones regulares para un rendimiento eficiente.

Cuando un patrón empieza con .* y la opción PCRE_DOTALL está definida, el patrón es anclado implícitamente por PCRE, ya que sólo puede casar el inicio de la cadena de entrada. Sin embargo, si PCRE_DOTALL no es definido, PCRE no puede hacer esta optimización, ya que el meta carácter . no tiene porque casar con una nueva línea y si la cadena de entrada contiene varias nuevas líneas, el patrón puede emparejar desde el carácter inmediatamente siguiente a uno de ellos en vez del inicio. Por ejemplo, el patrón

(.*) second

casa la cadena de entrada "first\nand second" (donde \n representa un carácter de nueva línea) con la primera subcadena capturada empezando con "and". En otras palabras, PCRE tiene que intentar los casamientos iniciándolos después de cada nueva línea en la cadena de entrada.

Si estas usando un patrón con cadenas de entrada que no contienen nuevas líneas, el mejor rendimiento se obtiene definiendo PCRE_DOTALL o iniciando el patrón con ^.* para indicar anclamiento explícito. Esto previene a PCRE tener que examinar toda la cadena de entrada buscando nuevas líneas para empezar de nuevo.

preg_grep (PHP 4)

Devuelve un array con los elementos que casen con el patrón

array **preg_grep** (string pattern, array input) \linebreak

preg_grep() devuelve un array conteniendo los elementos del array *input* que emparejen con el patrón (*pattern*) dado.

Ejemplo 1. Ejemplo de la función preg_grep()

```
preg_grep("/^(\d+)?\.\d+$/", $array); // encuentra todos los números reales en el array
```

Nota: Esta función fue añadida en PHP 4.0.

preg_match_all (PHP 3>= 3.0.9, PHP 4)

Realiza un completo emparejamiento de expresiones

int **preg_match_all** (string pattern, string subject, array matches [, int order]) \linebreak

Busca en *subject* todos los emparejamientos de la expresión *pattern* y los pone en *matches* de la forma indicada por *order*.

Después de encontrar el primer emparejamiento, las subsiguientes búsquedas empiezan desde el punto del último casamiento.

order puede tener los siguientes valores:

PREG_PATTERN_ORDER

Los resultados serán devueltos de manera que $matches[0] es un array con el patrón de búsqueda completo, $matches[1] es una array de las cadenas casadas por el primer subpatrón que esté entre paréntesis y así sucesivamente.

```
preg_match_all("|<[^>]+>(.*)</[^>]+>|U", "<b>example: </b><div align=left>this is a test
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n"
```

Esta ejemplo dará como resultado:

```
<b>example: </b>, <div align=left>this is a test</div>
example: , this is a test
```

Así, $out[0] contiene el array con las cadena que casan completamente con el patrón y $out[1] con las cadenas que se encuentran entre los tags.

PREG_SET_ORDER

Los resultados son dados de manera que $matches[0] es una array del primer conjunto de emparejamientos, $matches[1] es un array de los segundos conjuntos de casamientos y así sucesivamente.

```
preg_match_all("|<[^>]+>(.*)</[^>]+>|U", "<b>example: </b><div align=left>this is a test
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n"
```

Este ejemplo dará como resultado:

```
<b>example: </b>, example:
<div align=left>this is a test</div>, this is a test
```

En este caso, $matches[0] es el primer conjunto de emparejamientos y $matches[0][0] tiene el casamiento completo, $matches[0][1] el del primer subpatrón y así sucesivamente. Similarmente, $matches[1] es el segundo conjunto de emparejamientos, etc.

Si *order* no es dado, se asume PREG_PATTERN_ORDER.

Devuelve el número de casamientos completos, FALSE si no hubo o se produjo error.

Ejemplo 1. Obtener los número de teléfonos de un texto.

```
preg_match_all("/\(?  (\d{3})?  \)?  (?(1)  [\-\s] ) \d{3}-\d{4}/x",
               "Call 555-1212 or 1-800-555-1212", $phones);
```

Examina también preg_match(), preg_replace() y preg_split().

preg_match (PHP 3>= 3.0.9, PHP 4)

Realiza un emparejamiento dada una expresión

int **preg_match** (string pattern, string subject [, array matches]) \linebreak

Busca en *subject* para un emparejamiento, dada la expresión *pattern*.

Si *matches* es dado, entonces será definido con el resultado de la búsqueda. $matches[0] contendrá el texto que empareja con el patrón en su totalidad. $matches[1] tendrá la cadena que empareje con el primer subpatrón que esté entre paréntesis y así sucesivamente.

Devuelve TRUE si se encontró en la cadena un emparejamiento dado el patrón *pattern*, FALSE si no se produjo o hubo un error.

Ejemplo 1. Obtener el número de la siguiente página dada una cadena

```
if (preg_match("/page\s+#(\d+)/i", "Go to page #9.", $parts))
    print "Next page is $parts[1]";             // La siguiente página es $parts[1]
else
    print "Page not found.";                    // Página no encontrada
```

Examinar también preg_match_all(), preg_replace(), y preg_split().

preg_quote (PHP 3>= 3.0.9, PHP 4)

Prepara los caracteres de expresiones

string **preg_quote** (string str) \linebreak

preg_quote() toma *`str`* y pone una barra invertida (\) delante de todo carácter que sea parte de la sintaxis de las expresiones. Es útil si tienes una cadena en tiempo de ejecución y puede contener caracteres especiales.

Los caracteres especiales de las expresiones son:

```
. \\ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Nota: Esta función fue añadida en PHP 3.0.9.

preg_replace_callback (PHP 4 >= 4.0.5)

Perform a regular expression search and replace using a callback

mixed **preg_replace_callback** (mixed pattern, mixed callback, mixed subject [, int limit]) \linebreak

The behavior of this function is almost identical to preg_replace(), except for the fact that instead of *`replacement`* parameter, one should specify a *`callback`* that will be called and passed an array of matched elements in the subject string. The callback should return the replacement string. This function was added in PHP 4.0.5.

See also preg_replace().

preg_replace (PHP 3>= 3.0.9, PHP 4)

Lleva a cabo la búsqueda de una expresión y su sustitución

mixed **preg_replace** (mixed pattern, mixed replacement, mixed subject) \linebreak

Busca en *`subject`* los emparejamientos con *`pattern`* y los sustituye por *`replacement`*.

`replacement` puede contener referencias de la forma `\\`*`n`*. Éstas serán sustituidas por el texto obtenido por el patrón del paréntesis *`n`*ésimo. *`n`* puede tener un valor de cero a noventa y nueve, y `\\0` se refiere al texto casado por el patrón completo. Para obtener el número del subpatrón de búsqueda, los paréntesis abiertos son contados de izquierda derecha tomando el primero como uno.

Si el patrón no es encontrado en *`subject`*, entonces no se realizarán cambios.

Todos los parámetros de la función **preg_replace()** pueden ser un array.

Si *`subject`* es un array, entonces la búsqueda y sustitución es realizada para todos los elementos de *`subject`*, y el valor devuelto es también un array.

Si *pattern* y *replacement* son arrays, entonces **preg_replace()** toma un valor desde cada array y los usas para buscar y sustituir sobre *subject*. Si *replacement* tiene menos valores que *pattern*, entonces la cadena vacía es usada como valor para el resto de sustituciones. Si *pattern* es una array y *replacement* es una cadena, entonces esta cadena de sustitución es usada para todos los valores de *pattern*. Sin embargo, lo contrario no tiene sentido.

El modificador /e hace que la función **preg_replace()** trate el parámetro *replacement* como código PHP después de que la apropiada sustitución sea hecha. Atención, asegúrate que *replacement* es un código PHP correcto, de otro modo PHP dará un error de parse en la línea que contenga **preg_replace()**.

Nota: Este modificador fue añadido en PHP 4.0.

Ejemplo 1. Sustituir varios valores

```
$patterns = array("/(19|20\d{2})-(\d{1,2})-(\d{1,2})/", "/^\s*{(\w+)}\s*=/");
$replace = array("\\3/\\4/\\1", "$\\1 =");
print preg_replace($patterns, $replace, "{startDate} = 1999-5-27");
```

Esta ejemplo dará como resultado:

```
$startDate = 5/27/1999
```

Ejemplo 2. Usar el modificador /e

```
preg_replace("/(<\/?)(\w+)([^>]*>)/e", "'\\1'.strtoupper('\\2').'\\3'", $html_body);
```

Pondrá en mayúscula todos los tags HTML del texto de entrada.

Examina también preg_match(), preg_match_all(), y preg_split().

preg_split (PHP 3>= 3.0.9, PHP 4)

Divide una cadena dada una expresión

array **preg_split** (string pattern, string subject [, int limit [, int flags]]) \linebreak

Nota: El parámetro *flags* fue añadido en la Beta 3 de PHP

Devuelve un array conteniendo las subcadenas de *`subject`* divididas mediante los emparejamientos limitados por *`pattern`*.

Si *`limit`* es proporcionado, entonces sólo *`limit`* subcadenas son devueltas.

Si el flags es PREG_SPLIT_NO_EMPTY entonces las cadenas vacías no serán devueltas por **preg_split()**.

Ejemplo 1. Obtener las partes de una cadena de búsqueda

```
$keywords = preg_split("/[\s,]+/", "hypertext language, programming");
```

Examinar también preg_match(), preg_match_all(), y preg_replace().

LXXXVIII. qtdom functions

qdom_error (PHP 4 >= 4.0.5)

Returns the error string from the last QDOM operation or FALSE if no errors occured

string **qdom_error** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

qdom_tree (PHP 4 >= 4.0.4)

creates a tree of an xml string

object **qdom_tree** (string) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

LXXXIX. Funciones para expresiones regulares

Las expresiones regulares se usan en PHP para manipular cadenas complejas. Las funciones que soportan expresiones regulares son:

- ereg()
- ereg_replace()
- eregi()
- eregi_replace()
- split()

En todas estas funciones, el primer argumento es una expresión regular. PHP utiliza las expresiones regulares extendidas de POSIX, definidas en POSIX 1003.2. Para una descripción completa de las expresiones regulares POSIX, ver las páginas de manual de regex incluidas en el directorio regex de la distribución de PHP. Están en formato de página de manual, por lo que se deben leer con una orden como **man /usr/local/src/regex/regex.7**.

Ejemplo 1. Ejemplos de expresiones regulares

```
ereg("abc",$string);
/* Devuelve true si "abc"
   se encuentra en $string. */

ereg("^abc",$string);
/* Devuelve true si "abc"
   se encuentra al comienzo de $string. */

ereg("abc$",$string);
/* Devuelve true si "abc"
   se encuentra al final de $string. */

eregi("(ozilla.[23]|MSIE.3)",$HTTP_USER_AGENT);
/* Devuelve true si el navegador cliente
   es Netscape 2, 3 o MSIE 3. */

ereg("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)",
     $string,$regs);
/* Pone tres palabras separadas por espacios
   en $regs[1], $regs[2] y $regs[3]. */

$string = ereg_replace("^","<BR>",$string);
/* Coloca la etiqueta <BR> al comienzo de $string. */

$string = ereg_replace("$","<BR>",$string);
/* Coloca la etiqueta <BR> al final de $string. */

$string = ereg_replace("\n","",$string);
/* Elimina los caracteres fin-de-línea de $string. */
```

ereg_replace (PHP 3, PHP 4)

reemplaza expresiones regulares

string **ereg_replace** (string pattern, string replacement, string string) \linebreak

Esta función examina *string* buscando coincidencias de *pattern*, y reemplaza el texto encontrado con *replacement*.

Devuelve la cadena modificada. Si no hay coincidencias que reemplazar, devuelve la cadena original.

Si *pattern* contiene subcadenas entre paréntesis, *replacement* puede contener subcadenas de la forma *cifra*, que serán reemplazadas por el texto que coincide con la subcadena entre paréntesis que ocupa el lugar indicado por la cifra; \\0 produce el contenido total de la cadena. Se pueden usar hasta nueve subcadenas. Los paréntesis pueden anidarse; en este caso se cuentan los paréntesis de apertura.

Si no se encuentran coincidencias en *string*, se devuelve *string* sin cambios.

Por ejemplo, el siguiente fragmento de código imprime "This was a test" tres veces:

Ejemplo 1. ereg_replace() example

```
$string = "This is a test";
echo ereg_replace( " is", " was", $string );
echo ereg_replace( "( )is", "\\1was", $string );
echo ereg_replace( "(( )is)", "\\2was", $string );
```

Ver también ereg(), eregi(), y eregi_replace().

ereg (PHP 3, PHP 4)

Coincidencia de expresiones regulares

int **ereg** (string pattern, string string [, array regs]) \linebreak

Busca en *string* las coincidencias con la expresión regular *pattern*.

Si se encuentran coincidencias con subcadenas entre paréntesis de *pattern* y la función se ha llamado con el tercer argumento *regs*, las coincidencias se almacenarán en los elementos de *regs*. $regs[1] contendrá la subcadena que empieza en el primer paréntesis izquierdo; $regs[2] la que comienza en el segundo, etc. $regs[0] contendrá una copia de *string*.

La búsqueda diferencia mayúsculas y minúsculas.

Devuelve un valor verdadero si se encontró alguna coincidencia, o falso in no se encontraron coincidencias u ocurrió algún error.

El siguiente fragmento de código toma una fecha en formato ISO (AAAA-MM-DD) y la imprime en formato DD.MM.AAAA:

Ejemplo 1. ereg() example

```
if ( ereg( "([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs ) ) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Invalid date format: $date";
}
```

Ver también eregi(), ereg_replace(), y eregi_replace().

eregi_replace (PHP 3, PHP 4)

reemplaza expresiones regularse sin diferenciar mayúsculas y minúsculas

string **eregi_replace** (string pattern, string replacement, string string) \linebreak

Esta función es idéntica a ereg_replace(), excepto en que ignora la distinción entre mayúsculas y minúsculas.

Ver también ereg(), eregi(), y ereg_replace().

eregi (PHP 3, PHP 4)

coincidencia de expresiones regulares sin diferenciar mayúsculas y minúsculas

int **eregi** (string pattern, string string [, array regs]) \linebreak

Esta función es idéntica a ereg(), excepto en que ignora la distinción entre mayúsculas y minúsculas.

Ver también ereg(), ereg_replace(), y eregi_replace().

split (PHP 3, PHP 4)

divide la cadena en elementos de un array según una expresión regular

array **split** (string pattern, string string [, int limit]) \linebreak

Devuelve un array de cadenas, cada una de las cuales es una subcadena de *`string`* formada al dividir esta en los límites formados por la expresión regular *`pattern`*. Si ocurre un error, devuelve un valor falso.

Para obtener los cinco primeros campos de una línea de `/etc/passwd`:

Ejemplo 1. split() example

```
$passwd_list = split( ":", $passwd_line, 5 );
```

Para examinar una fecha que puede estar delimitada por barras, puntos o guiones:

Ejemplo 2. split() example

```
$date = "04/30/1973";  // Los delimitadores pueden ser barras, puntos o guiones
list( $month, $day, $year ) = split( '[/.-]', $date );
echo "Month: $month; Day: $day; Year: $year<br>\n";
```

Observar que *`pattern`* distingue entre mayúsculas y minúsculas.

Observar que si no se necesita la potencia de las expresiones regulares, es más rápido utilizar explode(), que no carga el motor de expresiones regulares.

Por favor, observar que *`pattern`* es una expresión regular. Si se quiere dividir con alguno de los caracteres especiales de las expresiones regulares, se necesita protegerlo antes. Si pareciera que **split()** (o cualquier otra función de regex) está haciendo algo irregular, léase el archivo `regex.7`, incluido en el subdirectorio `regex` de la distribución de PHP. Está en formato de página de manual, por lo que para leerlo es necesaria una orden como **man /usr/local/src/regex/regex.7**.

Ver también: explode() e implode().

spliti (PHP 4 >= 4.0.1)

Split string into array by regular expression case insensitive

array **spliti** (string pattern, string string [, int limit]) \linebreak

This function is identical to split() except that this ignores case distinction when matching alphabetic characters.

See also **preg_spliti()**, split(), explode(), and implode().

sql_regcase (PHP 3, PHP 4)

construye una expresión regular para buscar coincidencias sin diferenciar mayúsculas y minúsculas

string **sql_regcase** (string string) \linebreak

Devuelve una expresión regular válida que coincide con *`string`* sin distinguir mayúsculas y minúsculas. Esta expresión es *`string`* con cada carácter convertido a una expresión entre corchetes que

contiene el carácter en mayúscula y minúscula, si es posible; en caso contrario, contiene el carácter original dos veces.

Ejemplo 1. sql_regcase() example

```
echo sql_regcase( "Foo bar" );
```

imprime

```
[Ff][Oo][Oo][ ][Bb][Aa][Rr]
```

.

Se puede utilizar para lograr coincidencias que no diferencien mayúsculas de minúsculas en productos que sólo soportan expresiones regulares que sí distinguen.

XC. Funciones Semáforo y de memoria compartida

Este módulo provee funciones semáforo utilizando los semaforos de System V. Los semáforos pueden usarse para obtener acceso exclusivo a algun recurso del ordenador en cuestión, o para limitar el número de procesos que pueden usar un recurso simultaneamente.

Este módulo provee tambien funciones de memoria compartida, usando el compartimiento de memoria de System V. La memoria compartida puede usarse para proveer acceso a variables globales. Los diferentes demonios http e incluso otros programas, (como Perl, C, ...) son capaces de utilizar estos datos, para intercambiarlos de modo global. Recuerde que, la memoria compartida NO es segura para los accesos simultáneos. Use los semáforos para obtener sincronismo.

Tabla 1. Limites de la memoria compartida del SO Unix

SHMMAX	máximo tamaño de memoria compartida, normalmente 131072 bytes
SHMMIN	minimo tamaño de memoria compartida, por lo general 1 byte
SHMMNI	máxima cantidad de segmentos de memoria compartida, normalmente 100
SHMSEG	máximo de memoria compartida por proceso, normalmente 6

ftok (PHP 4 >= 4.2.0)

Convert a pathname and a project identifier to a System V IPC key

int **ftok** (string pathname, string proj) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

msg_get_queue (PHP 4 CVS only)

Create or attach to a message queue

int **msg_get_queue** (int key [, int perms]) \linebreak

msg_get_queue() returns an id that can be used to access the System V message queue with the given *key*. The first call creates the message queue with the optional *perms* (default: 0666). A second call to **msg_get_queue()** for the same *key* will return a different message queue identifier, but both identifiers access the same underlying message queue. If the message queue already exists, the *perms* will be ignored.

See also: msg_remove_queue(), msg_receive(), msg_send(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_receive (PHP 4 CVS only)

Receive a message from a message queue

bool **msg_receive** (int queue, int desiredmsgtype, int msgtype, int maxsize, mixed message [, bool unserialize [, int flags [, int errorcode]]]) \linebreak

msg_receive() will receive the first message from the specified *queue* of the type specified by *desiredmsgtype*. The type of the message that was received will be stored in *msgtype*. The maximum size of message to be accepted is specified by the *maxsize*; if the message in the queue is larger than this size the function will fail (unless you set *flags* as described below). The received message will be stored in *message*, unless there were errors receiving the message, in which case the optional *errorcode* will be set to the value of the system errno variable to help you identify the cause.

If *desiredmsgtype* is 0, the message from the front of the queue is returned. If *desiredmsgtype* is greater than 0, then the first message of that type is returned. If *desiredmsgtype* is less than 0, the first message on the queue with the lowest type less than or equal to the absolute value of

desiredmsgtype will be read. If no messages match the criteria, your script will wait until a suitable message arrives on the queue. You can prevent the script from blocking by specifying MSG_IPC_NOWAIT in the *flags* parameter.

unserialize defaults to TRUE; if it is set to TRUE, the message is treated as though it was serialized using the same mechanism as the session module. The message will be unserialized and then returned to your script. This allows you to easily receive arrays or complex object structures from other PHP scripts, or if you are using the WDDX serializer, from any WDDX compatible source. If *unserialize* is FALSE, the message will be returned as a binary-safe string.

The optional *flags* allows you to pass flags to the low-level msgrcv system call. It defaults to 0, but you may specify one or more of the following values (by adding or ORing them together).

Tabla 1. Flag values for msg_receive

MSG_IPC_NOWAIT	If there are no messages of the *desiredmsgtype*, return immediately and do not wait. The function will fail and return an integer value corresponding to ENOMSG.
MSG_EXCEPT	Using this flag in combination with a *desiredmsgtype* greater than 0 will cause the function to receive the first message that is not equal to *desiredmsgtype*.
MSG_NOERROR	If the message is longer than *maxsize*, setting this flag will truncate the message to *maxsize* and will not signal an error.

Upon successful completion the message queue data structure is updated as follows: *msg_lrpid* is set to the process-ID of the calling process, *msg_qnum* is decremented by 1 and *msg_rtime* is set to the current time.

msg_receive() returns TRUE on success or FALSE on failure. If the function fails, the optional *errorcode* will be set to the value of the system errno variable.

See also: msg_remove_queue(), msg_send(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_remove_queue (PHP 4 CVS only)

Destroy a message queue

bool **msg_remove_queue** (int queue) \linebreak

msg_remove_queue() destroys the message queue specified by the *queue*. Only use this function when all processes have finished working with the message queue and you need to release the system resources held by it.

See also: **msg_remove_queue()**, msg_receive(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_send (PHP 4 CVS only)

Send a message to a message queue

bool **msg_send** (int queue, int msgtype, mixed message [, bool serialize [, bool blocking [, int errorcode]]]) \linebreak

msg_send() sends a *message* of type *msgtype* (which MUST be greater than 0) to a the message queue specified by *queue*.

If the message is too large to fit in the queue, your script will wait until another process reads messages from the queue and frees enough space for your message to be sent. This is called blocking; you can prevent blocking by setting the optional *blocking* parameter to FALSE, in which case **msg_send()** will immediately return FALSE if the message is too big for the queue, and set the optional *errorcode* to EAGAIN, indicating that you should try to send your message again a little later on.

The optional *serialize* controls how the *message* is sent. *serialize* defaults to TRUE which means that the *message* is serialized using the same mechanism as the session module before being sent to the queue. This allows complex arrays and objects to be sent to other PHP scripts, or if you are using the WDDX serializer, to any WDDX compatible client.

Upon successful completion the message queue data structure is updated as follows: *msg_lspid* is set to the process-ID of the calling process, *msg_qnum* is incremented by 1 and *msg_stime* is set to the current time.

See also: msg_remove_queue(), msg_receive(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_set_queue (PHP 4 CVS only)

Set information in the message queue data structure

bool **msg_set_queue** (int queue, array data) \linebreak

msg_set_queue() allows you to change the values of the msg_perm.uid, msg_perm.gid, msg_perm.mode and msg_qbytes fields of the underlying message queue data structure. You specify the values you require by setting the value of the keys that you require in the *data* array.

Changing the data structure will require that PHP be running as the same user that created the the queue, owns the queue (as determined by the existing msg_perm.xxx fields), or be running with root privileges. root privileges are required to raise the msg_qbytes values above the system defined limit.

See also: msg_remove_queue(), msg_receive(), msg_stat_queue() and **msg_set_queue()**.

This function was introduced in PHP 4.3.0 (not yet released).

msg_stat_queue (PHP 4 CVS only)

Returns information from the message queue data structure

array **msg_stat_queue** (int queue) \linebreak

msg_stat_queue() returns the message queue meta data for the message queue specified by the `queue`. This is useful, for example, to determine which process sent the message that was just received.

The return value is an array whose keys and values have the following meanings:

Tabla 1. Array structure for msg_stat_queue

msg_perm.uid	The uid of the owner of the queue.
msg_perm.gid	The gid of the owner of the queue.
msg_perm.mode	The file access mode of the queue.
msg_stime	The time that the last message was sent to the queue.
msg_rtime	The time that the last message was received from the queue.
msg_ctime	The time that the queue was last changed.
msg_qnum	The number of messages waiting to be read from the queue.
msg_qbytes	The number of bytes of space currently available in the queue to hold sent messages until they are received.
msg_lspid	The pid of the process that sent the last message to the queue.
msg_lrpid	The pid of the process that received the last message from the queue.

See also: msg_remove_queue(), msg_receive(), **msg_stat_queue()** and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

sem_acquire (PHP 3>= 3.0.6, PHP 4)

adquiere un semáforo, lo toma para sí

int **sem_acquire** (int sem_identifier) \linebreak

Devuelve: Verdadero si hay éxito, falso si hay errores

sem_acquire() Produce un bloqueo (de ser necesario) hasta que el semáforo puede adquirirse. Un proceso intentando adquirir un semáforo ya adquirido, se bloqueará permanentemente si adquiriendo el

semáforo, excede su valor de max_acquire.

Despues de procesar una petición, cualquier semáforo adquirido por un proceso, que no sea explícitamente liberado, será liberado automñaticamente, generando un mensaje de alerta.

Véase tambien: sem_get() and sem_release().

sem_get (PHP 3>= 3.0.6, PHP 4)

obtiene la identificacion de un semáforo (semaphore id)

int **sem_get** (int key [, int max_acquire [, int perm]]) \linebreak

Devuelve: Un identificador positivo de semáforo en caso de éxito, o falso en caso de error.

sem_get() Devuelve un id que puede usarse para acceder al semáforo de System V con la llave dada. El semáforo se usa si es necesario usando los bits de permisos especificados en perm (por defecto 0666). El número de procesos que pueden adquirir el semáforo simultáneamente, se define en max_acquire (por defecto es 1). Actualmente este valor se fija sólo si el proceso determina que es el único relacionado actualmente al semáforo.

Una segunda llamada a **sem_get()** para la misma llave, devolverá un id de semáforo diferente, pero con ambos identificadoes, se accederá al mismo semáforo.

Véase también: sem_acquire() y sem_release().

sem_release (PHP 3>= 3.0.6, PHP 4)

release a semaphore

int **sem_release** (int sem_identifier) \linebreak

Returns: `TRUE` on success, `FALSE` on error

sem_release() releases the semaphore if it is currently acquired by the calling process, otherwise a warning is generated.

After releasing the semaphore, sem_acquire() may be called to re-acquire it.

Véase tambien: sem_get() y sem_acquire().

sem_remove (PHP 4 >= 4.1.0)

Remove a semaphore

bool **sem_remove** (int sem_identifier) \linebreak

Returns: `TRUE` on success, `FALSE` on error.

sem_remove() removes the semaphore *sem_identifier* if it has been created by sem_get(), otherwise generates a warning.

After removing the semaphore, it is no more accessible.

See also: sem_get(), sem_release() and sem_acquire().

Nota: This function does not work on Windows systems. It was added on PHP 4.1.0.

shm_attach (PHP 3>= 3.0.6, PHP 4)

Crea o abre un segmento de memoria compartida

int **shm_attach** (int key [, int memsize [, int perm]]) \linebreak

shm_attach() devuelve un identificador que puede usarse para acceder a la memoria compartida de SystemV, con la llave dada, la primera llamada creará el segmento de memoria compartida con mem_size de tamaño (por defecto: sysvshm.init_mem en el archivo de configuración, o bien de 10000 bytes) y los bits de permiso mas apropiados (por defecto: 0666).

Una segunda llamada a **shm_attach()** con la misma *llave* devolvera un id diferente de memoria compartida, pero ambos identificadores acceden a la misma porción de memoria compartida. *memsize* y *perm* serán ignorados.

shm_detach (PHP 3>= 3.0.6, PHP 4)

Finaliza conexión con un segmento de memoria compartida

int **shm_detach** (int shm_identifier) \linebreak

shm_detach() finaliza la conexión con la memoria compartida, especificada por el identificador *shm_identifier* creado con shm_attach(). Hay que recordar que la memoria compartida aún existe en el sistema Unix, y los datos aún están presentes.

shm_get_var (PHP 3>= 3.0.6, PHP 4)

Devuelve una variable de la memoria compartida

mixed **shm_get_var** (int id, int variable_key) \linebreak

shm_get_var() devuelve la variable con la llave *variable_key* dada. La variable, queda presente en la memoria compartida.

shm_put_var (PHP 3>= 3.0.6, PHP 4)

Inserta o actualiza una variable en la memoria compartida

int **shm_put_var** (int shm_identifier, int variable_key, mixed variable) \linebreak

Inserta o actualiza una *`variable`* con una llave *`variable_key`*. Son válidos todos los tipos de variable (doble, entera, cadena, arreglo).

shm_remove_var (PHP 3>= 3.0.6, PHP 4)

Elimina una variable de la memoria compartida

int **shm_remove_var** (int id, int variable_key) \linebreak

Elimina la varible que se corresponde con la llave *`variable_key`* dada, liberando la memoria que ocupaba aquella.

shm_remove (PHP 3>= 3.0.6, PHP 4)

Elimina memoria compartida del sistma Unix

int **shm_remove** (int shm_identifier) \linebreak

Elimina la memoria compartida de un sistema Unix, Todos los datos serán destruídos.

XCI. SESAM database functions

SESAM/SQL-Server is a mainframe database system, developed by Fujitsu Siemens Computers, Germany. It runs on high-end mainframe servers using the operating system BS2000/OSD.

In numerous productive BS2000 installations, SESAM/SQL-Server has proven ...

- the ease of use of Java-, Web- and client/server connectivity,
- the capability to work with an availability of more than 99.99%,
- the ability to manage tens and even hundreds of thousands of users.

Now there is a PHP3 SESAM interface available which allows database operations via PHP-scripts.

Configuration notes: There is no standalone support for the PHP SESAM interface, it works only as an integrated Apache module. In the Apache PHP module, this SESAM interface is configured using Apache directives.

Tabla 1. SESAM Configuration directives

Directive	Meaning
`php3_sesam_oml`	Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. Example: `php3_sesam_oml $.SYSLNK.SESAM-SQL.030`
`php3_sesam_configfile`	Name of SESAM application configuration file. Required for using SESAM functions. Example: `php3_sesam_configfile $SESAM.SESAM.CONF.AW` It will usually contain a configuration like (see SESAM reference manual): `CNF=B` `NAM=K` `NOTYPE`
`php3_sesam_messagecatalog`	Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive. Example: `php3_sesam_messagecatalog $.SYSMES.SESAM-SQL.030`

In addition to the configuration of the PHP/SESAM interface, you have to configure the

SESAM-Database server itself on your mainframe as usual. That means:

- starting the SESAM database handler (DBH), and
- connecting the databases with the SESAM database handler

To get a connection between a PHP script and the database handler, the `CNF` and `NAM` parameters of the selected SESAM configuration file must match the id of the started database handler.

In case of distributed databases you have to start a SESAM/SQL-DCN agent with the distribution table including the host and database names.

The communication between PHP (running in the POSIX subsystem) and the database handler (running outside the POSIX subsystem) is realized by a special driver module called SQLSCI and SESAM connection modules using common memory. Because of the common memory access, and because PHP is a static part of the web server, database accesses are very fast, as they do not require remote accesses via ODBC, JDBC or UTM.

Only a small stub loader (SESMOD) is linked with PHP, and the SESAM connection modules are pulled in from SESAM's OML PLAM library. In the configuration, you must tell PHP the name of this PLAM library, and the file link to use for the SESAM configuration file (As of SESAM V3.0, SQLSCI is available in the SESAM Tool Library, which is part of the standard distribution).

Because the SQL command quoting for single quotes uses duplicated single quotes (as opposed to a single quote preceded by a backslash, used in some other databases), it is advisable to set the PHP configuration directives `php3_magic_quotes_gpc` and `php3_magic_quotes_sybase` to `On` for all PHP scripts using the SESAM interface.

Runtime considerations: Because of limitations of the BS2000 process model, the driver can be loaded only after the Apache server has forked off its server child processes. This will slightly slow down the initial SESAM request of each child, but subsequent accesses will respond at full speed.

When explicitly defining a Message Catalog for SESAM, that catalog will be loaded each time the driver is loaded (i.e., at the initial SESAM request). The BS2000 operating system prints a message after successful load of the message catalog, which will be sent to Apache's error_log file. BS2000 currently does not allow suppression of this message, it will slowly fill up the log.

Make sure that the SESAM OML PLAM library and SESAM configuration file are readable by the user id running the web server. Otherwise, the server will be unable to load the driver, and will not allow to call any SESAM functions. Also, access to the database must be granted to the user id under which the Apache server is running. Otherwise, connections to the SESAM database handler will fail.

Cursor Types: The result cursors which are allocated for SQL "select type" queries can be either "sequential" or "scrollable". Because of the larger memory overhead needed by "scrollable" cursors, the default is "sequential".

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: `SESAM_SEEK_NEXT`) and the scrolling offset which can either be set once by sesam_seek_row() or each time when fetching a row using sesam_fetch_row(). When fetching a row using a "scrollable"

cursor, the following post-processing is done for the global default values for the scrolling type and scrolling offset:

Tabla 2. Scrolled Cursor Post-Processing

Scroll Type	Action
SESAM_SEEK_NEXT	none
SESAM_SEEK_PRIOR	none
SESAM_SEEK_FIRST	set scroll type to SESAM_SEEK_NEXT
SESAM_SEEK_LAST	set scroll type to SESAM_SEEK_PRIOR
SESAM_SEEK_ABSOLUTE	Auto-Increment internal offset value
SESAM_SEEK_RELATIVE	none. (maintain global default *offset* value, which allows for, e.g., fetching each 10th row backwards)

Porting note: Because in the PHP world it is natural to start indexes at zero (rather than 1), some adaptions have been made to the SESAM interface: whenever an indexed array is starting with index 1 in the native SESAM interface, the PHP interface uses index 0 as a starting point. E.g., when retrieving columns with sesam_fetch_row(), the first column has the index 0, and the subsequent columns have indexes up to (but not including) the column count ($array["count"]). When porting SESAM applications from other high level languages to PHP, be aware of this changed interface. Where appropriate, the description of the respective php sesam functions include a note that the index is zero based.

Security concerns: When allowing access to the SESAM databases, the web server user should only have as little privileges as possible. For most databases, only read access privilege should be granted. Depending on your usage scenario, add more access rights as you see fit. Never allow full control to any database for any user from the 'net! Restrict access to php scripts which must administer the database by using password control and/or SSL security.

Migration from other SQL databases: No two SQL dialects are ever 100% compatible. When porting SQL applications from other database interfaces to SESAM, some adaption may be required. The following typical differences should be noted:

- Vendor specific data types

  Some vendor specific data types may have to be replaced by standard SQL data types (e.g., TEXT

could be replaced by `VARCHAR(max. size)`).

- Keywords as SQL identifiers

  In SESAM (as in standard SQL), such identifiers must be enclosed in double quotes (or renamed).

- Display length in data types

  SESAM data types have a precision, not a display length. Instead of `int(4)` (intended use: integers up to '9999'), SESAM requires simply `int` for an implied size of 31 bits. Also, the only datetime data types available in SESAM are: `DATE`, `TIME(3)` and `TIMESTAMP(3)`.

- SQL types with vendor-specific `unsigned`, `zerofill`, or `auto_increment` attributes

  `Unsigned` and `zerofill` are not supported. `Auto_increment` is automatic (use `"INSERT ... VALUES(*, ...)"` instead of `"... VALUES(0, ...)"` to take advantage of SESAM-implied auto-increment.

- **int ... DEFAULT '0000'**

  Numeric variables must not be initialized with string constants. Use **DEFAULT 0** instead. To initialize variables of the datetime SQL data types, the initialization string must be prefixed with the respective type keyword, as in: `CREATE TABLE exmpl ( xtime timestamp(3) DEFAULT TIMESTAMP '1970-01-01 00:00:00.000' NOT NULL );`

- **$count = xxxx_num_rows();**

  Some databases promise to guess/estimate the number of the rows in a query result, even though the returned value is grossly incorrect. SESAM does not know the number of rows in a query result before actually fetching them. If you REALLY need the count, try **SELECT COUNT(...) WHERE ...**, it will tell you the number of hits. A second query will (hopefully) return the results.

- **DROP TABLE thename;**

  In SESAM, in the **DROP TABLE** command, the table name must be either followed by the keyword `RESTRICT` or `CASCADE`. When specifying `RESTRICT`, an error is returned if there are dependent objects (e.g., VIEWs), while with `CASCADE`, dependent objects will be deleted along with the specified table.

Notes on the use of various SQL types: SESAM does not currently support the BLOB type. A future version of SESAM will have support for BLOB.

At the PHP interface, the following type conversions are automatically applied when retrieving SQL fields:

Tabla 3. SQL to PHP Type Conversions

SQL Type	PHP Type
SMALLINT, INTEGER	"integer"
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	"double"
DATE, TIME, TIMESTAMP	"string"
VARCHAR, CHARACTER	"string"

When retrieving a complete row, the result is returned as an array. Empty fields are not filled in, so you will have to check for the existence of the individual fields yourself (use isset() or empty() to test for empty fields). That allows more user control over the appearance of empty fields (than in the case of an empty string as the representation of an empty field).

Support of SESAM's "multiple fields" feature: The special "multiple fields" feature of SESAM allows a column to consist of an array of fields. Such a "multiple field" column can be created like this:

Ejemplo 1. Creating a "multiple field" column

```
CREATE TABLE multi_field_test (
    pkey CHAR(20) PRIMARY KEY,
    multi(3) CHAR(12)
)
```

and can be filled in using:

Ejemplo 2. Filling a "multiple field" column

```
INSERT INTO multi_field_test (pkey, multi(2..3) )
    VALUES ('Second', <'first_val', 'second_val'>)
```

Note that (like in this case) leading empty sub-fields are ignored, and the filled-in values are collapsed, so that in the above example the result will appear as multi(1..2) instead of multi(2..3).

When retrieving a result row, "multiple columns" are accessed like "inlined" additional columns. In the example above, "pkey" will have the index 0, and the three "multi(1..3)" columns will be accessible as indices 1 through 3.

For specific SESAM details, please refer to the SESAM/SQL-Server documentation (english) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_en.htm) or the SESAM/SQL-Server documentation (german) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_gr.htm), both available online, or use the respective manuals.

sesam_affected_rows (PHP 3 CVS only)

Get number of rows affected by an immediate query

int **sesam_affected_rows** (string result_id) \linebreak

result_id is a valid result id returned by sesam_query().

Returns the number of rows affected by a query associated with *result_id*.

The **sesam_affected_rows()** function can only return useful values when used in combination with "immediate" SQL statements (updating operations like INSERT, UPDATE and DELETE) because SESAM does not deliver any "affected rows" information for "select type" queries. The number returned is the number of affected rows.

See also: sesam_query() and sesam_execimm()

```
$result = sesam_execimm ("DELETE FROM PHONE WHERE LASTNAME = '".strtoupper ($name)."'");
if (!$result) {
    ... error ...
}
print sesam_affected_rows ($result).
    " entries with last name ".$name." deleted.\n"
```

sesam_commit (PHP 3 CVS only)

Commit pending updates to the SESAM database

bool **sesam_commit** (void) \linebreak

Returns: TRUE on success, FALSE on errors

sesam_commit() commits any pending updates to the database.

Note that there is no "auto-commit" feature as in other databases, as it could lead to accidental data loss. Uncommitted data at the end of the current script (or when calling sesam_disconnect()) will be discarded by an implied sesam_rollback() call.

See also: sesam_rollback().

Ejemplo 1. Committing an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (!sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)"))
        die("insert failed");
    if (!sesam_commit())
        die("commit failed");
}
?>
```

sesam_connect (PHP 3 CVS only)

Open SESAM database connection

bool **sesam_connect** (string catalog, string schema, string user) \linebreak

Returns `TRUE` if a connection to the SESAM database was made, or `FALSE` on error.

sesam_connect() establishes a connection to an SESAM database handler task. The connection is always "persistant" in the sense that only the very first invocation will actually load the driver from the configured SESAM OML PLAM library. Subsequent calls will reuse the driver and will immediately use the given catalog, schema, and user.

When creating a database, the *"catalog"* name is specified in the SESAM configuration directive **//ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)**

The *"schema"* references the desired database schema (see SESAM handbook).

The *"user"* argument references one of the users which are allowed to access this *"catalog"* / *"schema"* combination. Note that *"user"* is completely independent from both the system's user id's and from HTTP user/password protection. It appears in the SESAM configuration only.

See also sesam_disconnect().

Ejemplo 1. Connect to a SESAM database

```
<?php
if (!sesam_connect ("mycatalog", "myschema", "otto")
    die("Unable to connect to SESAM";
?>
```

sesam_diagnostic (PHP 3 CVS only)

Return status information for last SESAM call

array **sesam_diagnostic** (void) \linebreak

Returns an associative array of status and return codes for the last SQL query/statement/command. Elements of the array are:

Tabla 1. Status information returned by sesam_diagnostic()

Element	Contents
$array["sqlstate"]	5 digit SQL return code (see the SESAM manual for the description of the possible values of SQLSTATE)
$array["rowcount"]	number of affected rows in last update/insert/delete (set after "immediate" statements only)
$array["errmsg"]	"human readable" error message string (set after errors only)
$array["errcol"]	error column number of previous error (0-based; or -1 if undefined. Set after errors only)
$array["errlin"]	error line number of previous error (0-based; or -1 if undefined. Set after errors only)

In the following example, a syntax error (E SEW42AE ILLEGAL CHARACTER) is displayed by including the offending SQL statement and pointing to the error location:

Ejemplo 1. Displaying SESAM error messages with error position

```
<?php
// Function which prints a formatted error message,
// displaying a pointer to the syntax error in the
// SQL statement
function PrintReturncode ($exec_str) {
    $err = Sesam_Diagnostic();
    $colspan=4; // 4 cols for: sqlstate, errlin, errcol, rowcount
    if ($err["errlin"] == -1)
        --$colspan;
    if ($err["errcol"] == -1)
        --$colspan;
    if ($err["rowcount"] == 0)
        --$colspan;
    echo "<TABLE BORDER>\n";
    echo "<TR><TH COLSPAN=".$colspan."><FONT COLOR=red>ERROR:</FONT> ".
   htmlspecialchars($err["errmsg"])."</TH></TR>\n";
    if ($err["errcol"] >= 0) {
        echo "<TR><TD COLSPAN=".$colspan."><PRE>\n";
        $errstmt = $exec_str."\n";
        for ($lin=0; $errstmt != ""; ++$lin) {
            if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
                if (!($i = strchr ($errstmt, "\n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                if ($line != "\n")
                    print htmlspecialchars ($line);
            } else {
                if (! ($i = strchr ($errstmt, "\n")))
```

```
                    $i = "";
                $line = substr ($errstmt, 0, strlen ($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                for ($col=0; $col < $err["errcol"]; ++$col)
                    echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>\\</BLINK></FONT>\n";
                print "<FONT COLOR=\"#880000\">".htmlspecialchars($line)."</FONT>";
                for ($col=0; $col < $err["errcol"]; ++$col)
                    echo (substr ($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>/</BLINK></FONT>\n";
            }
        }
        echo "</PRE></TD></TR>\n";
    }
    echo "<TR>\n";
    echo " <TD>sqlstate=" . $err["sqlstate"] . "</TD>\n";
    if ($err["errlin"] != -1)
        echo " <TD>errlin=" . $err["errlin"] . "</TD>\n";
    if ($err["errcol"] != -1)
        echo " <TD>errcol=" . $err["errcol"] . "</TD>\n";
    if ($err["rowcount"] != 0)
         echo " <TD>rowcount=" . $err["rowcount"] . "</TD>\n";
    echo "</TR>\n";
    echo "</TABLE>\n";
}

if (!sesam_connect ("mycatalog", "phoneno", "otto"))
  die ("cannot connect");

$stmt = "SELECT * FROM phone\n".
        " WHERE@ LASTNAME='KRAEMER'\n".
        " ORDER BY FIRSTNAME";
if (!($result = sesam_query ($stmt)))
    PrintReturncode ($stmt);
?>
```

See also: sesam_errormsg() for simple access to the error string only

sesam_disconnect (PHP 3 CVS only)

Detach from SESAM connection

bool **sesam_disconnect** (void) \linebreak

Returns: always `TRUE`.

sesam_disconnect() closes the logical link to a SESAM database (without actually disconnecting and unloading the driver).

Note that this isn't usually necessary, as the open connection is automatically closed at the end of the script's execution. Uncommitted data will be discarded, because an implicit sesam_rollback() is executed.

sesam_disconnect() will not close the persistent link, it will only invalidate the currently defined *"catalog"*, *"schema"* and *"user"* triple, so that any sesam function called after **sesam_disconnect()** will fail.

See also: sesam_connect().

Ejemplo 1. Closing a SESAM connection

```
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    ... some queries and stuff ...
    sesam_disconnect();
}
```

sesam_errormsg (PHP 3 CVS only)

Returns error message of last SESAM call

string **sesam_errormsg** (void) \linebreak

Returns the SESAM error message associated with the most recent SESAM error.

```
if (!sesam_execimm ($stmt))
  printf ("%s<br>\n", sesam_errormsg());
```

See also: sesam_diagnostic() for the full set of SESAM SQL status information

sesam_execimm (PHP 3 CVS only)

Execute an "immediate" SQL-statement

string **sesam_execimm** (string query) \linebreak

Returns: A SESAM "result identifier" on success, or `FALSE` on error.

sesam_execimm() executes an "immediate" statement (i.e., a statement like UPDATE, INSERT or DELETE which returns no result, and has no INPUT or OUTPUT variables). "select type" queries can not be used with **sesam_execimm()**. Sets the *affected_rows* value for retrieval by the sesam_affected_rows() function.

Note that sesam_query() can handle both "immediate" and "select-type" queries. Use **sesam_execimm()** only if you know beforehand what type of statement will be executed. An attempt to use SELECT type queries with **sesam_execimm()** will return `$err["sqlstate"] == "42SBW"`.

The returned "result identifier" can not be used for retrieving anything but the sesam_affected_rows(); it is only returned for symmetry with the sesam_query() function.

```
$stmt = "INSERT INTO mytable VALUES ('one', 'two')";
$result = sesam_execimm ($stmt);
$err = sesam_diagnostic();
print ("sqlstate = ".$err["sqlstate"]."\n".
       "Affected rows = ".$err["rowcount"]." == ".
       sesam_affected_rows($result)."\n");
```

See also: sesam_query() and sesam_affected_rows().

sesam_fetch_array (PHP 3 CVS only)

Fetch one row as an associative array

array **sesam_fetch_array** (string result_id [, int whence [, int offset]]) \linebreak

Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

sesam_fetch_array() is an alternative version of sesam_fetch_row(). Instead of storing the data in the numeric indices of the result array, it stores the data in associative indices, using the field names as keys.

`result_id` is a valid result id returned by sesam_query() (select type queries only!).

For the valid values of the optional *`whence`*and *`offset`* parameters, see the sesam_fetch_row() function for details.

sesam_fetch_array() fetches one row of data from the result associated with the specified result identifier. The row is returned as an associative array. Each result column is stored with an associative index equal to its column (aka. field) name. The column names are converted to lower case.

Columns without a field name (e.g., results of arithmetic operations) and empty fields are not stored in the array. Also, if two or more columns of the result have the same column names, the later column will take precedence. In this situation, either call sesam_fetch_row() or make an alias for the column.

```
SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2
```

A special handling allows fetching "multiple field" columns (which would otherwise all have the same column names). For each column of a "multiple field", the index name is constructed by appending the string "(n)" where n is the sub-index of the multiple field column, ranging from 1 to its declared repetition factor. The indices are NOT zero based, in order to match the nomenclature used in the respective query syntax. For a column declared as:

```
CREATE TABLE ... ( ... MULTI(3) INT )
```

the associative indices used for the individual "multiple field" columns would be `"multi(1)"`, `"multi(2)"`, and `"multi(3)"` respectively.

Subsequent calls to **sesam_fetch_array()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or `FALSE` if there are no more rows.

Ejemplo 1. SESAM fetch array

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table:
print "<TABLE BORDER>\n";
while (($row = sesam_fetch_array ($result)) && count ($row) > 0) {
    print " <TR>\n";
    print " <TD>".htmlspecialchars ($row["firstname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["lastname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["phoneno"])."</TD>\n";
    print " </TR>\n";
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: sesam_fetch_row() which returns an indexed array.

sesam_fetch_result (PHP 3 CVS only)

Return all or part of a query result

mixed **sesam_fetch_result** (string result_id [, int max_rows]) \linebreak

Returns a mixed array with the query result entries, optionally limited to a maximum of *max_rows* rows. Note that both row and column indexes are zero-based.

Tabla 1. Mixed result set returned by sesam_fetch_result()

Array Element	Contents
int $arr["count"]	number of columns in result set (or zero if this was an "immediate" query)

Array Element	Contents
int $arr["rows"]	number of rows in result set (between zero and *max_rows*)
bool $arr["truncated"]	TRUE if the number of rows was at least *max_rows*, FALSE otherwise. Note that even when this is TRUE, the next **sesam_fetch_result()** call may return zero rows because there are no more result entries.
mixed $arr[col][row]	result data for all the fields at row(row) and column(col), (where the integer index row is between 0 and $arr["rows"]-1, and col is between 0 and $arr["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php isset() function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Note that the amount of memory used up by a large query may be gigantic. Use the *max_rows* parameter to limit the maximum number of rows returned, unless you are absolutely sure that your result will not use up all available memory.

See also: sesam_fetch_row(), and sesam_field_array() to check for "multiple fields". See the description of the sesam_query() function for a complete example using **sesam_fetch_result()**.

sesam_fetch_row (PHP 3 CVS only)

Fetch one row as an array

array **sesam_fetch_row** (string result_id [, int whence [, int offset]]) \linebreak

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

The number of columns in the result set is returned in an associative array element $array["count"]. Because some of the result columns may be empty, the count() function can not be used on the result row returned by **sesam_fetch_row()**.

result_id is a valid result id returned by sesam_query() (select type queries only!).

whence is an optional parameter for a fetch operation on "scrollable" cursors, which can be set to the following predefined constants:

Tabla 1. Valid values for *"whence"* parameter

Value	Constant	Meaning

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially (after fetch, the internal default is set to SESAM_SEEK_NEXT)
1	SESAM_SEEK_PRIOR	read sequentially backwards (after fetch, the internal default is set to SESAM_SEEK_PRIOR)
2	SESAM_SEEK_FIRST	rewind to first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	seek to last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	seek to absolute row number given as *offset* (Zero-based. After fetch, the internal default is set to SESAM_SEEK_ABSOLUTE, and the internal offset value is auto-incremented)
5	SESAM_SEEK_RELATIVE	seek relative to current scroll position, where *offset* can be a positive or negative offset value.

This parameter is only valid for "scrollable" cursors.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. If the *whence* parameter is omitted, the global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT, and settable by sesam_seek_row()) are used. If *whence* is supplied, its value replaces the global default.

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE. This parameter is only valid for "scrollable" cursors.

sesam_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array (indexed by values between 0 and $array["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php isset() function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Subsequent calls to **sesam_fetch_row()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or FALSE if there are no more rows.

Ejemplo 1. SESAM fetch rows

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
```

```
}
// print the table in backward order
print "<TABLE BORDER>\n";
$row = sesam_fetch_row ($result, SESAM_SEEK_LAST);
while (is_array ($row)) {
    print " <TR>\n";
    for ($col = 0; $col < $row["count"]; ++$col) {
        print "  <TD>".htmlspecialchars ($row[$col])."</TD>\n";
    }
    print " </TR>\n";
    // use implied SESAM_SEEK_PRIOR
    $row = sesam_fetch_row ($result);
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: sesam_fetch_array() which returns an associative array, and sesam_fetch_result() which returns many rows per invocation.

sesam_field_array (PHP 3 CVS only)

Return meta information about individual columns in a result

array **sesam_field_array** (string result_id) \linebreak

result_id is a valid result id returned by sesam_query().

Returns a mixed associative/indexed array with meta information (column name, type, precision, ...) about individual columns of the result after the query associated with *result_id*.

Tabla 1. Mixed result set returned by sesam_field_array()

Array Element	Contents
int $arr["count"]	Total number of columns in result set (or zero if this was an "immediate" query). SESAM "multiple fields" are "inlined" and treated like the respective number of columns.
string $arr[col]["name"]	column name for column(col), where col is between 0 and $arr["count"]-1. The returned value can be the empty string (for dynamically computed columns). SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same column name.

Array Element	Contents
string $arr[col]["count"]	The "count" attribute describes the repetition factor when the column has been declared as a "multiple field". Usually, the "count" attribute is 1. The first column of a "multiple field" column however contains the number of repetitions (the second and following column of the "multiple field" contain a "count" attribute of 1). This can be used to detect "multiple fields" in the result set. See the example shown in the sesam_query() description for a sample use of the "count" attribute.
string $arr[col]["type"]	php variable type of the data for column(`col`), where `col` is between 0 and `$arr["count"]-1`. The returned value can be one of • "integer" • "double" • "string" depending on the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same php type.

Array Element	Contents
string $arr[col]["sqltype"]	SQL variable type of the column data for column(`col`), where `col` is between 0 and `$arr["count"]-1`. The returned value can be one of • "CHARACTER" • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" describing the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same SQL type.
string $arr[col]["length"]	The SQL "length" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "length" attribute is used with "CHARACTER" and "VARCHAR" SQL types to specify the (maximum) length of the string variable. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same length attribute.
string $arr[col]["precision"]	The "precision" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "precision" attribute is used with numeric and time data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same precision attribute.

Array Element	Contents
string $arr[col]["scale"]	The "scale" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "scale" attribute is used with numeric data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same scale attribute.

See the sesam_query() function for an example of the **sesam_field_array()** use.

sesam_field_name (PHP 3 CVS only)

Return one column name of the result set

int **sesam_field_name** (string result_id, int index) \linebreak

Returns the name of a field (i.e., the column name) in the result set, or `FALSE` on error.

For "immediate" queries, or for dynamic columns, an empty string is returned.

Nota: The column index is zero-based, not one-based as in SESAM.

See also: sesam_field_array(). It provides an easier interface to access the column names and types, and allows for detection of "multiple fields".

sesam_free_result (PHP 3 CVS only)

Releases resources for the query

int **sesam_free_result** (string result_id) \linebreak

Releases resources for the query associated with *`result_id`*. Returns `FALSE` on error.

sesam_num_fields (PHP 3 CVS only)

Return the number of fields/columns in a result set

int **sesam_num_fields** (string result_id) \linebreak

After calling sesam_query() with a "select type" query, this function gives you the number of columns in the result. Returns an integer describing the total number of columns (aka. fields) in the current *result_id* result set or FALSE on error.

For "immediate" statements, the value zero is returned. The SESAM "multiple field" columns count as their respective dimension, i.e., a three-column "multiple field" counts as three columns.

See also: sesam_query() and sesam_field_array() for a way to distinguish between "multiple field" columns and regular columns.

sesam_query (PHP 3 CVS only)

Perform a SESAM SQL query and prepare the result

string **sesam_query** (string query [, bool scrollable]) \linebreak

Returns: A SESAM "result identifier" on success, or FALSE on error.

A "result_id" resource is used by other functions to retrieve the query results.

sesam_query() sends a query to the currently active database on the server. It can execute both "immediate" SQL statements and "select type" queries. If an "immediate" statement is executed, then no cursor is allocated, and any subsequent sesam_fetch_row() or sesam_fetch_result() call will return an empty result (zero columns, indicating end-of-result). For "select type" statements, a result descriptor and a (scrollable or sequential, depending on the optional boolean *scrollable* parameter) cursor will be allocated. If *scrollable* is omitted, the cursor will be sequential.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT) and the scrolling offset which can either be set once by sesam_seek_row() or each time when fetching a row using sesam_fetch_row().

For "immediate" statements, the number of affected rows is saved for retrieval by the sesam_affected_rows() function.

See also: sesam_fetch_row() and sesam_fetch_result().

Ejemplo 1. Show all rows of the "phone" table as a html table

```
<?php
if (!sesam_connect ("phonedb", "demo", "otto"))
    die ("cannot connect");
$result = sesam_query ("select * from phone");
if (!$result) {
    $err = sesam_diagnostic();
    die ($err["errmsg"]);
}
echo "<TABLE BORDER>\n";
// Add title header with column names above the result:
if ($cols = sesam_field_array ($result)) {
    echo " <TR><TH COLSPAN=".$cols["count"].">Result:</TH></TR>\n";
    echo " <TR>\n";
```

```
    for ($col = 0; $col < $cols["count"]; ++$col) {
        $colattr = $cols[$col];
        /* Span the table head over SESAM's "Multiple Fields": */
        if ($colattr["count"] > 1) {
            echo "  <TH COLSPAN=".$colattr["count"].">".$colattr["name"].
                "(1..".$colattr["count"].")</TH>\n";
            $col += $colattr["count"] - 1;
        } else
            echo "  <TH>" . $colattr["name"] . "</TH>\n";
    }
    echo " </TR>\n";
}

do {
    // Fetch the result in chunks of 100 rows max.
    $ok = sesam_fetch_result ($result, 100);
    for ($row=0; $row < $ok["rows"]; ++$row) {
        echo " <TR>\n";
        for ($col = 0; $col < $ok["cols"]; ++$col) {
            if (isset($ok[$col][$row]))
                echo "  <TD>" . $ok[$col][$row] . "</TD>\n";
            } else {
                echo "  <TD>-empty-</TD>\n";
            }
        }
        echo " </TR>\n";
    }
}
while ($ok["truncated"]) { // while there may be more data
    echo "</TABLE>\n";
}
// free result id
sesam_free_result($result);
?>
```

sesam_rollback (PHP 3 CVS only)

Discard any pending updates to the SESAM database

bool **sesam_rollback** (void) \linebreak

Returns: `TRUE` on success, `FALSE` on errors

sesam_rollback() discards any pending updates to the database. Also affected are result cursors and result descriptors.

At the end of each script, and as part of the sesam_disconnect() function, an implied **sesam_rollback()** is executed, discarding any pending changes to the database.

See also: sesam_commit().

Ejemplo 1. Discarding an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)")
        && sesam_execimm ("INSERT INTO othertable VALUES (*, 'Another Test', 1)"))
        sesam_commit();
    else
        sesam_rollback();
}
?>
```

sesam_seek_row (PHP 3 CVS only)

Set scrollable cursor mode for subsequent fetches

bool **sesam_seek_row** (string result_id, int whence [, int offset]) \linebreak

result_id is a valid result id (select type queries only, and only if a "scrollable" cursor was requested when calling sesam_query()).

whence sets the global default value for the scrolling type, it specifies the scroll type to use in subsequent fetch operations on "scrollable" cursors, which can be set to the following predefined constants:

Tabla 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially
1	SESAM_SEEK_PRIOR	read sequentially backwards
2	SESAM_SEEK_FIRST	fetch first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	fetch last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	fetch absolute row number given as *offset* (Zero-based. After fetch, the default is set to SESAM_SEEK_ABSOLUTE, and the offset value is auto-incremented)

Value	Constant	Meaning
5	SESAM_SEEK_RELATIVE	fetch relative to current scroll position, where *offset* can be a positive or negative offset value (this also sets the default "offset" value for subsequent fetches).

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE.

sesam_settransaction (PHP 3 CVS only)

Set SESAM transaction parameters

bool **sesam_settransaction** (int isolation_level, int read_only) \linebreak

Returns: TRUE if the values are valid, and the **settransaction()** operation was successful, FALSE otherwise.

sesam_settransaction() overrides the default values for the "isolation level" and "read-only" transaction parameters (which are set in the SESAM configuration file), in order to optimize subsequent queries and guarantee database consistency. The overridden values are used for the next transaction only.

sesam_settransaction() can only be called before starting a transaction, not after the transaction has been started already.

To simplify the use in php scripts, the following constants have been predefined in php (see SESAM handbook for detailed explanation of the semantics):

Tabla 1. Valid values for *"Isolation_Level"* parameter

Value	Constant	Meaning
1	SESAM_TXISOL_READ_UNCOMMITTED	Read Uncommitted
2	SESAM_TXISOL_READ_COMMITTED	Read Committed
3	SESAM_TXISOL_REPEATABLE_READ	Repeatable Read
4	SESAM_TXISOL_SERIALIZABLE	Serializable

Tabla 2. Valid values for *"Read_Only"* parameter

Value	Constant	Meaning
0	SESAM_TXREAD_READWRITE	Read/Write
1	SESAM_TXREAD_READONLY	Read-Only

The values set by **sesam_settransaction()** will override the default setting specified in the SESAM configuration file.

Ejemplo 1. Setting SESAM transaction parameters

```
<?php
sesam_settransaction (SESAM_TXISOL_REPEATABLE_READ,
                      SESAM_TXREAD_READONLY);
?>
```

XCII. Funciones para el manejo de sesiones

El apoyo que PHP proporciona para las sesiones consiste en una forma de conservar ciertos datos a lo largo de los subsiguientes accesos, lo cual le permite construir aplicaciones más personalizadas e incrementar el atractivo de su sitio web.

Si ya está familiarizado con el tratamiento de sesiones de PHPLIB, notará que algunos conceptos son similares al soporte de las sesiones de PHP.

A cada visitante que accede a su web se le asigna un identificador único, llamado "session id" (identificador de sesión). Éste se almacena en una cookie por parte del usuario o se propaga en la URL.

El soporte de las sesiones le permite registrar un número arbitrario de variables que se conservarán en las siguientes peticiones. Cuando un visitante acceda a su web, PHP comprobará automáticamente (si session.auto_start está puesto a 1) o cuando usted lo especifique (de forma explícita mendiante session_start() o implícita a través de session_register()) si se le ha enviado un "session id" específico con su petición, en cuyo caso se recrean las variables que se habían guardado anteriormente.

Todas las variables registradas son almacenadas tras finalizar la petición. Las variables que están indefinidas se marcan como no definidas. En los subsiguientes accesos, no estarán definidas por el módulo de sesiones a menos que el usuario las defina más tarde.

Las opciones de configuración `track_vars` y `register_globals` influyen notablemente en la forma en que las variables de la sesión se almacenan y restauran.

> **Nota:** A partir de PHP 4.0.3, `track_vars` siempre está activado.

> **Nota:** A partir de PHP 4.1.0, `$_SESSION` está disponible como variable global, al igual que `$_POST`, `$_GET`, `$_REQUEST` y demás. Al contrario que `$HTTP_SESSION_VARS`, `$_SESSION` siempre es global. Por tanto, no se debe usar global para `$_SESSION`.

Si `track_vars` está activado y `register_globals` está desactivado, sólo los miembros del vector asociativo global `$HTTP_SESSION_VARS` pueden ser registrados como variables de la sesión. Las variables restauradas de la sesión sólo estarán disponibles en el vector `$HTTP_SESSION_VARS`.

Ejemplo 1. Registrar una variable con `track_vars` activado

```
<?php
session_start();
if (isset($HTTP_SESSION_VARS['count'])) {
   $HTTP_SESSION_VARS['count']++;
}
else {
   $HTTP_SESSION_VARS['count'] = 0;
}
?>
```

Se recomienda usar `$_SESSION` (o `$HTTP_SESSION_VARS` con PHP 4.0.6 o inferior) por seguridad y para hacer el código más legible. Con `$_SESSION` o `$HTTP_SESSION_VARS`, no es necesario usar las funciones session_register() / session_unregister() / session_is_registered(). Los usuarios pueden acceder a una variable de la sesión como si se tratase de una variable normal.

Ejemplo 2. Registrar una variable con $_SESSION.

```
<?php
session_start();
// Use $HTTP_SESSION_VARS con PHP 4.0.6 o inferior
if (!isset($_SESSION['count'])) {
    $_SESSION['count'] = 0;
} else {
    $_SESSION['count']++;
}
?>
```

Ejemplo 3. Borrar una variable con $_SESSION.

```
<?php
session_start();
// Use $HTTP_SESSION_VARS con PHP 4.0.6 o inferior
unset($_SESSION['count']);
?>
```

Si `register_globals` está activado, todas las variables globales pueden ser registradas como variables de la sesión, y las variables de la sesión serán restauradas a sus correspondientes variables globales. Como PHP debe saber qué variables globles están registradas como variables de la sesión, los usuarios deben registrar las variables con la función session_register(), mientras que con

`$HTTP_SESSION_VARS`/`$_SESSION` no es necesario usar session_register().

> **Atención**
>
> Si está usando `$HTTP_SESSION_VARS`/`$_SESSION` y desactiva `register_globals`, no use session_register(), session_is_registered() ni session_unregister().
>
> Si activa `register_globals`, session_unregister() debería ser usado a partir de que las variables de la sesión sean registradas como variables globales cuando los datos de la sesión se guardan. Se recomienda desactivar `register_globals` por motivos de seguridad y rendimiendo.

Ejemplo 4. Registrar una variable con `register_globals` activado

```
<?php
if (!session_is_registered('count')) {
    session_register("count");
    $count = 0;
}
else {
    $count++;
}
?>
```

Si `track_vars` y `register_globals` están activados, las variables globales y las entradas de `$HTTP_SESSION_VARS`/`$_SESSION` harán referencia al mismo valor para variables ya registradas.

Si el usuario utiliza session_register() para registrar una variable, el vector `$HTTP_SESSION_VARS`/`$_SESSION` no contendrá esa variable hasta que se cargue de los datos de la sesión. (p.ej. hasta la próxima petición).

Hay dos formas de propagar un "session id":

- Cookies
- Parámetro en la URL

El módulo de sesiones admite ambas formas. Las Cookies son la mejor opción, pero como no son fiables (los clientes no están obligados a aceptarlas), no podemos confiar en ellas. El segundo método incrusta el "session id" directamente en las URLs.

PHP es capaz de hacerlo de forma transparente al usuario cuando se compila con `--enable-trans-sid`. Si activa esta opción, las URIs relativas serán modificadas de forma que contengan el session id automáticamente. Alternativamente, puede usar la constante `SID` que está

definida, si el cliente no envía la cookie adecuada. El `SID` puede tener la forma de `nombre_de_sesion=session_id` o ser una cadena vacía.

El ejemplo siguiente demuestra cómo registrar una variable, y cómo colocar correctamente un enlace a otra página usando la constante SID.

Ejemplo 5. Contar el número de impresiones de un usuario

```
<?php
if (!session_is_registered('count')) {
    session_register('count');
    $count = 1;
}
else {
    $count++;
}
?>

Hola, visitante. Has visto esta p&aacute;gina <?php echo $count; ?> veces.

<?php
# el <?php echo SID?> (Se puede usar <?=SID?> si short tag est&aacute; activado)
# es necesario para conservar el session id
# en caso de que el usuario haya desactivado las cookies
?>

Para continuar, haga click <A HREF="nextpage.php?<?php echo SID?>">aqu&iacute;</A>.
```

El `<?=SID?>` no es necesario si se ha usado `--enable-trans-sid` al compilar PHP.

> **Nota:** Se asume que las URLs no relativas apuntan a sitios web externos, y por tanto no se añade el SID, ya que pasar el SID a un servidor diferente podría ocasionar un agujero de seguridad.

Para implementar el almacenamiento en bases de datos o en otro tipo de almacenamiento, necesitará usar **session_set_save_handler()()** para crear una colección de funciones de almacenamiento a nivel de usuario.

El sistema de control de sesiones soporta varias opciones de configuración que puede colocar en su archivo `php.ini`. Les daremos un pequeño repaso.

- `session.save_handler` define el nombre del controlador que se usa para almacenar y recuperar los datos asociados a la sesión. Su valor por defecto es `files`.
- `session.save_path` define el argumento que se pasa al controlador de almacenamiento. Si elige el controlador de archivos por defecto, esta es la ruta donde los archivos se crean. Por defecto es `/tmp`.

Si la profundidad de la ruta de `session.save_path` es mayor que 2, no se llevará a cabo la recolección de basura.

> **Aviso**
>
> Si lo deja apuntando a un directorio con permiso de lectura por el resto de usuarios, como `/tmp` (la opción por defecto), los demás usuarios del servidor pueden conseguir robar las sesiones obteniéndolas de la lista de archivos de ese directorio.

- `session.name` especifica el nombre de la sesión que se usa como nombre de la cookie. Sólo debería contener caracteres alfanuméricos. Por defecto vale `PHPSESSID`.
- `session.auto_start` especifica si el módulo de las sesión inicia una sesión automáticamente al comenzar la petición. Por defecto está `0` (desactivado).
- `session.cookie_lifetime` especifica la duración de la cookie en segundos que se manda al navegador. El valor `0` significa "hasta que se cierra el navegador", y es el que se encuentra por defecto.
- `session.serialize_handler` define el nombre del controlador que se utiliza para guardar y restaurar los datos. Actualmente se soportan un formato interno de PHP (cuyo nombre es `php`) y WDDX (cuyo nombre es `wddx`). WDDX sólo está disponible si PHP está compilado con Soporte para WDDX. Por defecto es `php`.
- `session.gc_probability` especifica la probabilidad de que se inicie la rutina gc (garbage collection - recoleción de basura) en cada petición en porcentaje. Por defecto es `1`.
- `session.gc_maxlifetime` especifica el número de segundos tras los cuales los datos se considerarán como ’basura’ y serán eliminados.
- `session.referer_check` contiene la subcadena que usted quiera que se compruebe en cada "HTTP Referer" (N.T.: Página desde donde proviene el enlace a la página actual). Si el "Referer" fue enviado por el cliente y la subcadena no se ha encontrado, el session id incrustado será marcado como inválido. Por defecto es una cadena vacía.
- `session.entropy_file` indica la ruta a un recurso externo (un archivo) que se usará como fuente adicional de entropía en el proceso de creación de session id’s. Por ejemplo `/dev/random` o `/dev/urandom`, que están disponibles en muchos sistemas Unix.
- `session.entropy_length` especifica el número de bytes que serán leidos del archivo indicado más arriba. Por defecto es `0` (desactivado).
- `session.use_cookies` indica si el módulo puede usar cookies para guardar el session id en el lado del cliente. Por defecto está a `1` (activado).
- `session.cookie_path` especifica la ruta a colocar en session_cookie. Por defecto es `/`.
- `session.cookie_domain` especifica el dominio a establecer en session_cookie. Por defecto no se especifica ninguno en ningún caso.
- `session.cache_limiter` especifica el método de control del caché a usar en las páginas de la sesión (none/nocache/private/private_no_expire/public). Por defecto es `nocache` (no guardar las

páginas en el caché).

- `session.cache_expire` especifica el tiempo-de-vida de las páginas de la sesión que se encuentran en el caché en minutos. No tiene efecto para el limitador nocache. Por defecto vale `180`.
- `session.use_trans_sid` indica si la inclusión del sid transparente está activada o no, si fue activada compilando con `--enable-trans-sid`. Por defecto está a `1` (activado).
- `url_rewriter.tags` especifica qué etiquetas html serán reescritas para incluir el session id si la inclusión del sid transparente está activada. Las etiquetas por defecto son `a=href,area=href,frame=src,input=src,form=fakeentry`

Nota: El manejo de sesiones fue añadido en PHP 4.0.

session_cache_expire (PHP 4 >= 4.2.0)

Devuelve la caducidad actual del caché

int **session_cache_expire** ([int nueva_caducidad_cache]) \linebreak

session_cache_expire() devuelve la caducidad actual del caché. Si se proporciona *nueva_caducidad_cache*, se reemplazará la caducidad actual con *nueva_caducidad_cache*.

session_cache_limiter (PHP 4 >= 4.0.3)

Lee y/o cambia el limitador del caché actual

string **session_cache_limiter** ([string limitador_del_cache]) \linebreak

session_cache_limiter() devuelve el nombre del limitador de caché actual. Si se especifica *limitador_del_cache*, el nombre del limitador de caché actual se cambia al nuevo valor.

El limitador de caché controla las cabeceras HTTP de control del caché enviadas al cliente. Estas cabeceras determinan las reglas por las que el contenido de la página puede ser guardado en el caché local del cliente. Cambiando el limitador de caché a `nocache`, por ejemplo, impedirá cualquier tipo de almacenamiento en el caché por parte del cliente. Un valor de `public`, en cambio, permitiría el almacenamiento en el caché. Tambié se puede cambiar a `private`, que es un poco más restrictivo que el `public`.

En el modo `private`, la cabecera Expire (caducidad) enviada al cliente puede confundir a algunos navegadores incluyendo Mozilla. Puede evitar este problema con el modo `private_no_expire`. La cabecera Expire nunca se envía al cliente en este modo.

Nota: `private_no_expire` fue añadida en PHP 4.2.0dev.

Al comenzar la ejecución del script, el limitador de caché se reestablece al valor por defecto guardado en `session.cache_limiter`. De este modo, es necesario llamar a **session_cache_limiter()** en cada petición (y antes de llamar a session_start()).

Ejemplo 1. Ejemplos con session_cache_limiter()

```
<?php

# cambia el limitador del cach&eacute; a 'private'

session_cache_limiter('private');
$cache_limiter = session_cache_limiter();

echo "El limitador de cach&eacute; est&aacute; puesto ahora en $cache_limiter<p>";
?>
```

session_decode (PHP 4)

Decodifica los datos de una sesión a partir de una cadena

bool **session_decode** (string datos) \linebreak

session_decode() decodifica los datos de una sesión que se encuentran en *datos*, creando las variables guardadas en la sesión.

session_destroy (PHP 4)

Destruye todos los datos guardados en una sesión

bool **session_destroy** (void) \linebreak

session_destroy() destruye todos los datos asociados con la sesión actual. No destruye ninguna de las variables globales asociadas a la sesión ni la cookie.

Esta función devuelve TRUE si se ha destruido la sesión correctamente y FALSE si ha habido algún problema al intentarlo.

Ejemplo 1. Destrucción de una sesión

```
<?php

// Inicializa de la sesi&oacute;n.
// Si est&aacute; usando session_name("algo"), &iexcl;no lo olvide ahora!
session_start();
// Destruye todas las variables de la sesi&oacute;n
session_unset();
// Finalmente, destruye la sesi&oacute;n
session_destroy();

?>
```

Ejemplo 2. Destrucción de una sesión con $_SESSION

```
<?php

// Inicializa la sesi&oacute;n.
// Si est&aacute; usando session_name("algo"), &iexcl;no lo olvide ahora!
session_start();
// Destruye todas las variables de la sesi&oacute;n
$_SESSION = array();
// Finalmente, destruye la sesi&oacute;n
session_destroy();

?>
```

session_encode (PHP 4)

Codifica los datos de la sesión actual en una cadena

string **session_encode** (void) \linebreak

session_encode() devuelve una cadena con el contenido de la sesión actual en su interior.

session_get_cookie_params (PHP 4)

Obtiene los parámetros de la cookie de la sesión

array **session_get_cookie_params** (void) \linebreak

La función **session_get_cookie_params()** devuelve un vector con información sobre la cookie de la sesión actual, conteniendo los siguientes elementos:

- "lifetime" - La duración de la cookie.
- "path" - La ruta donde se guarda la información.
- "domain" - El dominio de la cookie.
- "secure" - La cookie debe ser enviada sólo bajo conexiones seguras. (Este elemento fue añadido en PHP 4.0.4.)

session_id (PHP 4)

Lee y/o cambia el session id actual

string **session_id** ([string id]) \linebreak

session_id() devuelve el session id de la sesión actual. Si se especifica un *id*, reemplazará el session id actual.

También se puede utilizar la constante SID para recuperar el nombre y el session id de la sesión actual como una cadena adecuada para añadir a las URLs.

session_is_registered (PHP 4)

Comprueba si una variable está registrada en la sesión

bool **session_is_registered** (string nombre) \linebreak

session_is_registered() devuelve `TRUE` si hay una variable registrada en la sesión actual cuyo nombre es *nombre*.

Nota: Si utiliza `$_SESSION` (o `$HTTP_SESSION_VARS` con PHP 4.0.6 o inferior), use isset() para comprobar si una variable está registrada en `$_SESSION`.

Atención

Si utiliza `$HTTP_SESSION_VARS`/`$_SESSION`, no use session_register(), **session_is_registered()** ni session_unregister().

session_module_name (PHP 4)

Lee y/o cambia el módulo de la sesión actual

string **session_module_name** ([string módulo]) \linebreak

session_module_name() devuelve el nombre del módulo de la sesión actual. Si se especifica *módulo*, se usará ese módulo en su lugar.

session_name (PHP 4)

Lee y/o cambia el nombre de la sesión actual

string **session_name** ([string nombre]) \linebreak

session_name() devuelve el nombre de la sesión actual. Si se especifica un *nombre*, el nombre de de la sesión actual se cambia a este valor.

El nombre de la sesión hace referencia al session id utilizado en las cookies y en las URLs. Debería contener únicamente caracteres alfanuméricos; también debería ser corto y descriptivo (p.ej. para usuarios con los avisos de las cookies activados). El nombre de la sesión se restaura al valor guardado por defecto en `session.name` al comenzar la petición. Así, pues, es necesario llamar a **session_name()** en cada petición (y antes de llamar a session_start() o a session_register()).

Ejemplo 1. Ejemplos de session_name()

```
<?php

// Cambiar el nombre de la sesi&oacute;n a WebsiteID

$nombre_anterior = session_name("WebsiteID");

echo "El anterior nombre de la sesi&oacute;n era $nombre_anterior<p>";
?>
```

session_readonly (unknown)

Inicia una sesión - reinicializa las variables, pero sin guardar los cambios al terminar

void **session_readonly** (void) \linebreak

Lee los datos de una sesión sin bloquearlos. No es posible cambiar los datos de la sesión, pero el rendimiento de los framesets mejorará.

session_register (PHP 4)

Register one or more variables with the current session

bool **session_register** (mixed name [, mixed ...]) \linebreak

session_register() accepts a variable number of arguments, any of which can be either a string holding the name of a variable or an array consisting of variable names or other arrays. For each name, **session_register()** registers the global variable with that name in the current session.

> **Atención**
>
> This registers a *global* variable. If you want to register a session variable inside a function, you need to make sure to make it global using **global()** or use the session arrays as noted below.

> **Atención**
>
> If you are using `$_SESSION` (or `$HTTP_SESSION_VARS`), do not use **session_register()**, session_is_registered() and session_unregister().

This function returns `TRUE` when all of the variables are successfully registered with the session.

If session_start() was not called before this function is called, an implicit call to session_start() with no parameters will be made.

You can also create a session variable by simply setting the appropriate member of the `$_SESSION` or `$HTTP_SESSION_VARS` (PHP < 4.1.0) array.

```
$barney = "A big purple dinosaur.";
session_register("barney");

$_SESSION["zim"] = "An invader from another planet.";

# The old way was to use $HTTP_SESSION_VARS
$HTTP_SESSION_VARS["spongebob"] = "He's got square pants.";
```

Nota: It is not currently possible to register resource variables in a session. For example, you can not create a connection to a database and store the connection id as a session variable and expect the connection to still be valid the next time the session is restored. PHP functions that return a resource are identified by having a return type of `resource` in their function definitions. A list of functions that return resources are available in the resource types appendix.

If `$_SESSION` (or `$HTTP_SESSION_VARS` for PHP 4.0.6 or less) is used, assign variable to `$_SESSION`. i.e. $_SESSION['var'] = 'ABC';

See also session_is_registered() and session_unregister().

session_save_path (PHP 4)

Lee y/o cambia la ruta donde se guardan los datos de la sesión actual

string **session_save_path** ([string path]) \linebreak

session_save_path() devuelve la ruta del directorio usado actualmente para guardar los datos de la sesión. Si se especifica *path*, se cambiará la ruta donde se guardan los datos.

Nota: En algunos sistemas operativos, puede que quiera especificar una ruta en un sistema de ficheros que maneja muchos ficheros pequeños de forma eficiente. Por ejemplo, en Linux, reiserfs puede dar un rendimiento mejor que ext2fs.

session_set_cookie_params (PHP 4)

Cambia los parámetros de la cookie de la sesión

void **session_set_cookie_params** (int duración [, string path [, string dominio [, bool segura]]]) \linebreak

Cambia los parámetros de la cookie definidos en el archivo `php.ini`. El efecto de esta función sólo dura hasta que termina el script.

Nota: El parámetro *segura* fue añadido en PHP 4.0.4.

session_set_save_handler (PHP 4)

Establece unas funciones para el almacenamiento de los datos de la sesión a nivel de usuario

bool **session_set_save_handler** (string abrir, string cerrar, string leer, string escribir, string destruir, string gc) \linebreak

session_set_save_handler() establece las funciones que se utilizan a nivel de usuario para el almacenamiento y recuperación de los datos asociados a una sesión. Es lo más útil cuando se prefiere utilizar otro método de almacenamiento distinto del proporcionaddo por las sesiones de PHP. p.ej. Almacenar los datos de la sesión en una base de datos local. Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

Nota: Debe cambiar la opción *session.save_handler* en la configuración a *user* en su archivo `php.ini` para que **session_set_save_handler()** tenga efecto.

Nota: El manejador "escribir" no se ejecuta hasta que se cierra la salida. Por ello, la salida de las sentencias que coloquemos en el manejador "escribir" para el depurado nunca será enviadas al

navegador. Si se necesita producir una salida para el depurado, se sugiere que la salida se produzca en un archivo.

El siguiente ejemplo proporciona almacenamiento de las sesiones basado en archivos de forma similar al manejador de sesiones por defecto de PHP *files*. Este ejemplo puede ser extendido fácilmente para cubrir el almacenamiento en bases de datos usando su motor de soporte de bases de datos de PHP favorito.

La función de lectura debe devolver siempre una cadena para que el manejador de escritura funcione como se espera. Devuelva una cadena vacía si no hay ningún dato a leer. Los valores devueltos de otros manejadores son convertidos a una expresión booleana. TRUE si todo ha ido correctamente, FALSE si ha habido algún problema.

Ejemplo 1. Ejemplo de session_set_save_handler()

```
<?php
function abrir ($save_path, $session_name) {
  global $sess_save_path, $sess_session_name;

  $sess_save_path = $save_path;
  $sess_session_name = $session_name;
  return(true);
}

function cerrar() {
  return(true);
}

function leer ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "r")) {
    $sess_data = fread($fp, filesize($sess_file));
    return($sess_data);
  } else {
    return(""); // Debe devolver "" aqu&iacute;.
  }

}

function escribir ($id, $sess_data) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w")) {
    return(fwrite($fp, $sess_data));
  } else {
    return(false);
  }
```

```
}

function destruir ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  return(@unlink($sess_file));
}

/*****************************************
 * ATENCI&Oacute;N - Necesitar&aacute; implementar alg&uacute;n     *
 * tipo de rutinas recolectoras de basura aqu&iacute; *
 *****************************************/
function rb ($maxlifetime) {
  return true;
}

session_set_save_handler ("abrir", "cerrar", "leer", "escribir", "destruir", "rb");

session_start();

// proceed to use sessions normally

?>
```

session_start (PHP 4)

Inicializar los datos de una sesión

bool **session_start** (void) \linebreak

session_start() crea una sesió (o la continúa basandose en el session id pasado por GET o mediante una cookie).

Si desea usar una sesión con un nombre en concreto, debe llamar a session_name() antes de llamar a **session_start()**.

Esta función siempre devuelve `TRUE`.

> **Nota:** Si está usando sesiones basadas en las cookies, debe llamar a **session_start()** antes de que haya ninguna salida al navegador.

session_start() registrará un manejador de salida interno para la reescritura de las URL's si `trans-sid` está activado. Si un usuario utiliza `ob_gzhandler` o algo como ob_start(), el orden del manejador de

salida es importante para que la salida sea la adecuada. Por ejemplo, el usuario debe registrar `ob_gzhandler` antes de iniciar la sesión.

> **Nota:** Se recomienda utilizar `zlib.output_compression` en lugar de `ob_gzhandler`

session_unregister (PHP 4)

Desregistrar una variable de la sesión actual

bool **session_unregister** (string nombre) \linebreak

session_unregister() desregistra (olvida) la variable global llamada *`nombre`* de la sesión actual.

Esta función devuelve `TRUE` cuando la variable es eliminada de la sesión correctamente.

> **Nota:** Si utiliza `$_SESSION` (o `$HTTP_SESSION_VARS` con PHP 4.0.6 o inferior), use unset() para eliminar una variable de la sesión actual.

> **Atención**
>
> Esta función no borra la variable global correspondiente a *`nombre`*, sólo evita que la variable sea guardada como parte de la sesión. Debe llamar a unset() para eliminar la variable global correspondiente.

> **Atención**
>
> Si está trabajando con `$HTTP_SESSION_VARS`/`$_SESSION`, no utilice session_register(), session_is_registered() ni **session_unregister()**.

session_unset (PHP 4)

Elimina todas las variables de la sesión

void **session_unset** (void) \linebreak

La función **session_unset()** elimina y libera el espacio ocupado por todas las variables de la sesión actual registradas.

> **Nota:** Si utiliza `$_SESSION` (o `$HTTP_SESSION_VARS` con PHP 4.0.6 o inferior), use unset() en su lugar para desregistrar una variable de la sesión. p.ej. $_SESSION = array();

session_write_close (PHP 4 >= 4.0.4)

Escribe los datos de la sesión y la finaliza

void **session_write_close** (void) \linebreak

Finaliza la sesión actual y guarda sus datos

Los datos de la sesió se suelen guardar tras finalizar la ejecución de su script sin necesidad de llamar a **session_write_close()**, pero como los datos de la sesió están bloqueados para evitar escrituras concurrentes, sólo un script puede trabajar con una sesión a la vez. Cuando se usan framesets junto con sesiones, podrá comprobar que los frames se cargan uno a uno debido a este bloqueo. Puede reducir el tiempo necesario para cargar los frames finalizando la sesión tan pronto como haya terminado los cambios a las variables de la sesión.

XCIII. Shared Memory Functions

Shmop is an easy to use set of functions that allows php to read, write, create and delete UNIX shared memory segments. The functions will not work on windows, as it does not support shared memory. To use shmop you will need to compile php with the --enable-shmop parameter in your configure line.

> **Nota:** The functions explained in the chapter begin all with **shm_()** in PHP 4.0.3, but in PHP 4.0.4 and later versions these names are changed to begin with **shmop_()**.

Ejemplo 1. Shared Memory Operations Overview

```
<?php

// Create 100 byte shared memory block with system id if 0xff3
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if(!$shm_id) {
echo "Couldn't create shared memory segment\n";
}

// Get shared memory block's size
$shm_size = shmop_size($shm_id);
echo "SHM Block Size: ".$shm_size. " has been created.\n";

// Lets write a test string into shared memory
$shm_bytes_written = shmop_write($shm_id, "my shared memory block", 0);
if($shm_bytes_written != strlen("my shared memory block")) {
echo "Couldn't write the entire length of data\n";
}

// Now lets read the string back
$my_string = shmop_read($shm_id, 0, $shm_size);
if(!$my_string) {
echo "Couldn't read from shared memory block\n";
}
echo "The data inside shared memory was: ".$my_string."\n";

//Now lets delete the block and close the shared memory segment
if(!shmop_delete($shm_id)) {
echo "Couldn't mark shared memory block for deletion.
}
shmop_close($shm_id);

?>
```

shmop_close (PHP 4 >= 4.0.4)

Close shared memory block

int **shmop_close** (int shmid) \linebreak

shmop_close() is used to close a shared memory block.

shmop_close() takes the shmid, which is the shared memory block identifier created by **shmop_open()**.

Ejemplo 1. Closing shared memory block

```
<?php
shmop_close($shm_id);
?>
```

This example will close shared memory block identified by `$shm_id`.

shmop_delete (PHP 4 >= 4.0.4)

Delete shared memory block

int **shmop_delete** (int shmid) \linebreak

shmop_delete() is used to delete a shared memory block.

shmop_delete() takes the shmid, which is the shared memory block identifier created by **shmop_open()**. On success 1 is returned, on failure 0 is returned.

Ejemplo 1. Deleting shared memory block

```
<?php
shmop_delete($shm_id);
?>
```

This example will delete shared memory block identified by `$shm_id`.

shmop_open (PHP 4 >= 4.0.4)

Create or open shared memory block

int **shmop_open** (int key, string flags, int mode, int size) \linebreak

shmop_open() can create or open a shared memory block.

shmop_open() takes 4 parameters: key, which is the system's id for the shared memory block, this parameter can be passed as a decimal or hex. The second parameter are the flags that you can use:

- "a" for access (sets IPC_EXCL) use this flag when you need to open an existing shared memory segment
- "c" for create (sets IPC_CREATE) use this flag when you need to create a new shared memory segment.

The third parameter is the mode, which are the permissions that you wish to assign to your memory segment, those are the same as permission for a file. Permissions need to be passed in octal form ex. 0644. The last parameter is size of the shared memory block you wish to create in bytes.

> **Nota:** Note: the 3rd and 4th should be entered as 0 if you are opening an existing memory segment. On success **shmop_open()** will return an id that you can use to access the shared memory segment you've created.

Ejemplo 1. Create a new shared memory block

```
<?php
$shm_id = shmop_open(0x0fff, "c", 0644, 100);
?>
```

This example opened a shared memory block with a system id of 0x0fff.

shmop_read (PHP 4 >= 4.0.4)

Read data from shared memory block

string **shmop_read** (int shmid, int start, int count) \linebreak

shmop_read() will read a string from shared memory block.

shmop_read() takes 3 parameters: shmid, which is the shared memory block identifier created by **shmop_open()**, offset from which to start reading and count on the number of bytes to read.

Ejemplo 1. Reading shared memory block

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

This example will read 50 bytes from shared memory block and place the data inside `$shm_data`.

shmop_size (PHP 4 >= 4.0.4)

Get size of shared memory block

int **shmop_size** (int shmid) \linebreak

shmop_size() is used to get the size, in bytes of the shared memory block.

shmop_size() takes the shmid, which is the shared memory block identifier created by **shmop_open()**, the function will return and int, which represents the number of bytes the shared memory block occupies.

Ejemplo 1. Getting the size of the shared memory block

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

This example will put the size of shared memory block identified by `$shm_id` into `$shm_size`.

shmop_write (PHP 4 >= 4.0.4)

Write data into shared memory block

int **shmop_write** (int shmid, string data, int offset) \linebreak

shmop_write() will write a string into shared memory block.

shmop_write() takes 3 parameters: shmid, which is the shared memory block identifier created by **shmop_open()**, data, a string that you want to write into shared memory block and offset, which specifies where to start writing data inside the shared memory segment.

Ejemplo 1. Writing to shared memory block

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

This example will write data inside `$my_string` into shared memory block, `$shm_bytes_written` will contain the number of bytes written.

XCIV. Shockwave Flash functions

PHP offers the ability to create Shockwave Flash files via Paul Haeberli's libswf module. You can download libswf at ftp://ftp.sgi.com/sgi/graphics/grafica/flash. Once you have libswf all you need to do is to configure `--with-swf[=DIR]` where DIR is a location containing the directories include and lib. The include directory has to contain the swf.h file and the lib directory has to contain the libswf.a file. If you unpack the libswf distribution the two files will be in one directory. Consequently you will have to copy the files to the proper location manually.

Once you've successfully installed PHP with Shockwave Flash support you can then go about creating Shockwave files from PHP. You would be surprised at what you can do, take the following code:

Ejemplo 1. SWF example

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);

swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);

swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p,  'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p,  'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
```

```
}

swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();

swf_closefile ();
?>
```

It will produce the animation found at the following url (http://www.designmultimedia.com/swfphp/test.swf).

Nota: SWF support was added in PHP4 RC2.

swf_actiongeturl (PHP 4)

Get a URL from a Shockwave Flash movie

void **swf_actiongeturl** (string url, string target) \linebreak

The **swf_actionGetUrl()** function gets the URL specified by the parameter *`url`* with the target *`target`*.

swf_actiongotoframe (PHP 4)

Play a frame and then stop

void **swf_actiongotoframe** (int framenumber) \linebreak

The **swf_actionGotoFrame()** function will go to the frame specified by *`framenumber`*, play it, and then stop.

swf_actiongotolabel (PHP 4)

Display a frame with the specified label

void **swf_actiongotolabel** (string label) \linebreak

The **swf_actionGotoLabel()** function displays the frame with the label given by the *`label`* parameter and then stops.

swf_actionnextframe (PHP 4)

Go foward one frame

void **swf_actionnextframe** (void) \linebreak

Go foward one frame.

swf_actionplay (PHP 4)

Start playing the flash movie from the current frame

void **swf_actionplay** (void) \linebreak

Start playing the flash movie from the current frame.

swf_actionprevframe (PHP 4)

Go backwards one frame

void **swf_actionprevframe** (void) \linebreak

swf_actionsettarget (PHP 4)

Set the context for actions

void **swf_actionsettarget** (string target) \linebreak

The **swf_actionSetTarget()** function sets the context for all actions. You can use this to control other flash movies that are currently playing.

swf_actionstop (PHP 4)

Stop playing the flash movie at the current frame

void **swf_actionstop** (void) \linebreak

Stop playing the flash movie at the current frame.

swf_actiontogglequality (PHP 4)

Toggle between low and high quality

void **swf_actiontogglequality** (void) \linebreak

Toggle the flash movie between high and low quality.

swf_actionwaitforframe (PHP 4)

Skip actions if a frame has not been loaded

void **swf_actionwaitforframe** (int framenumber, int skipcount) \linebreak

The **swf_actionWaitForFrame()** function will check to see if the frame, specified by the *`framenumber`* parameter has been loaded, if not it will skip the number of actions specified by the *`skipcount`* parameter. This can be useful for "Loading..." type animations.

swf_addbuttonrecord (PHP 4)

Controls location, appearance and active area of the current button

void **swf_addbuttonrecord** (int states, int shapeid, int depth) \linebreak

The **swf_addbuttonrecord()** function allows you to define the specifics of using a button. The first parameter, *states*, defines what states the button can have, these can be any or all of the following constants: BSHitTest, BSDown, BSOver or BSUp. The second parameter, the *shapeid* is the look of the button, this is usually the object id of the shape of the button. The *depth* parameter is the placement of the button in the current frame.

Ejemplo 1. swf_addbuttonrecord() function example

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_addcolor (PHP 4)

Set the global add color to the rgba value specified

void **swf_addcolor** (float r, float g, float b, float a) \linebreak

The **swf_addcolor()** function sets the global add color to the *rgba* color specified. This color is then used (implicitly) by the swf_placeobject(), swf_modifyobject() and the swf_addbuttonrecord() functions. The color of the object will be add by the *rgba* values when the object is written to the screen.

Nota: The *rgba* values can be either positive or negative.

swf_closefile (PHP 4)

Close the current Shockwave Flash file

void **swf_closefile** (void) \linebreak

Close a file that was opened by the swf_openfile() function.

swf_definebitmap (PHP 4)

Define a bitmap

void **swf_definebitmap** (int objid, string image_name) \linebreak

The **swf_definebitmap()** function defines a bitmap given a GIF, JPEG, RGB or FI image. The image will be converted into a Flash JPEG or Flash color map format.

swf_definefont (PHP 4)

Defines a font

void **swf_definefont** (int fontid, string fontname) \linebreak

The **swf_definefont()** function defines a font given by the *fontname* parameter and gives it the id specified by the *fontid* parameter. It then sets the font given by *fontname* to the current font.

swf_defineline (PHP 4)

Define a line

void **swf_defineline** (int objid, float x1, float y1, float x2, float y2, float width) \linebreak

The **swf_defineline()** defines a line starting from the x coordinate given by *x1* and the y coordinate given by *y1* parameter. Up to the x coordinate given by the *x2* parameter and the y coordinate given by the *y2* parameter. It will have a width defined by the *width* parameter.

swf_definepoly (PHP 4)

Define a polygon

void **swf_definepoly** (int objid, array coords, int npoints, float width) \linebreak

The **swf_definepoly()** function defines a polygon given an array of x, y coordinates (the coordinates are defined in the parameter *coords*). The parameter *npoints* is the number of overall points that are contained in the array given by *coords*. The *width* is the width of the polygon's border, if set to 0.0 the polygon is filled.

swf_definerect (PHP 4)

Define a rectangle

void **swf_definerect** (int objid, float x1, float y1, float x2, float y2, float width) \linebreak

The **swf_definerect()** defines a rectangle with an upper left hand coordinate given by the x, *x1*, and the y, *y1*. And a lower right hand coordinate given by the x coordinate, *x2*, and the y coordinate, *y2* . Width of the rectangles border is given by the *width* parameter, if the width is 0.0 then the rectangle is filled.

swf_definetext (PHP 4)

Define a text string

void **swf_definetext** (int objid, string str, int docenter) \linebreak

Define a text string (the *str* parameter) using the current font and font size. The *docenter* is where the word is centered, if *docenter* is 1, then the word is centered in x.

swf_endbutton (PHP 4)

End the definition of the current button

void **swf_endbutton** (void) \linebreak

The **swf_endButton()** function ends the definition of the current button.

swf_enddoaction (PHP 4)

End the current action

void **swf_enddoaction** (void) \linebreak

Ends the current action started by the swf_startdoaction() function.

swf_endshape (PHP 4)

Completes the definition of the current shape

void **swf_endshape** (void) \linebreak

The **swf_endshape()** completes the definition of the current shape.

swf_endsymbol (PHP 4)

End the definition of a symbol

void **swf_endsymbol** (void) \linebreak

The **swf_endsymbol()** function ends the definition of a symbol that was started by the swf_startsymbol() function.

swf_fontsize (PHP 4)

Change the font size

void **swf_fontsize** (float size) \linebreak

The **swf_fontsize()** function changes the font size to the value given by the *size* parameter.

swf_fontslant (PHP 4)

Set the font slant

void **swf_fontslant** (float slant) \linebreak

Set the current font slant to the angle indicated by the *slant* parameter. Positive values create a foward slant, negative values create a negative slant.

swf_fonttracking (PHP 4)

Set the current font tracking

void **swf_fonttracking** (float tracking) \linebreak

Set the font tracking to the value specified by the *tracking* parameter. This function is used to increase the spacing between letters and text, positive values increase the space and negative values decrease the space between letters.

swf_getbitmapinfo (PHP 4)

Get information about a bitmap

array **swf_getbitmapinfo** (int bitmapid) \linebreak

The **swf_getbitmapinfo()** function returns an array of information about a bitmap given by the *bitmapid* parameter. The returned array has the following elements:

- "size" - The size in bytes of the bitmap.
- "width" - The width in pixels of the bitmap.
- "height" - The height in pixels of the bitmap.

swf_getfontinfo (PHP 4)

The height in pixels of a capital A and a lowercase x

array **swf_getfontinfo** (void) \linebreak

The **swf_getfontinfo()** function returns an associative array with the following parameters:

- Aheight - The height in pixels of a capital A.
- xheight - The height in pixels of a lowercase x.

swf_getframe (PHP 4)

Get the frame number of the current frame

int **swf_getframe** (void) \linebreak

The **swf_getframe()** function gets the number of the current frame.

swf_labelframe (PHP 4)

Label the current frame

void **swf_labelframe** (string name) \linebreak

Label the current frame with the name given by the *name* parameter.

swf_lookat (PHP 4)

Define a viewing transformation

void **swf_lookat** (double view_x, double view_y, double view_z, double reference_x, double reference_y, double reference_z, double twist) \linebreak

The **swf_lookat()** function defines a viewing transformation by giving the viewing position (the parameters *view_x*, *view_y*, and *view_z*) and the coordinates of a reference point in the scene, the reference point is defined by the *reference_x*, *reference_y* , and *reference_z* parameters. The *twist* controls the rotation along with viewer's z axis.

swf_modifyobject (PHP 4)

Modify an object

void **swf_modifyobject** (int depth, int how) \linebreak

Updates the position and/or color of the object at the specified depth, *depth*. The parameter *how* determines what is updated. *how* can either be the constant MOD_MATRIX or MOD_COLOR or it can be a combination of both (MOD_MATRIX|MOD_COLOR).

MOD_COLOR uses the current mulcolor (specified by the function swf_mulcolor()) and addcolor (specified by the function swf_addcolor()) to color the object. MOD_MATRIX uses the current matrix to position the object.

swf_mulcolor (PHP 4)

Sets the global multiply color to the rgba value specified

void **swf_mulcolor** (float r, float g, float b, float a) \linebreak

The **swf_mulcolor()** function sets the global multiply color to the *rgba* color specified. This color is then used (implicitly) by the swf_placeobject(), swf_modifyobject() and the swf_addbuttonrecord() functions. The color of the object will be multiplied by the *rgba* values when the object is written to the screen.

Nota: The *rgba* values can be either positive or negative.

swf_nextid (PHP 4)

Returns the next free object id

int **swf_nextid** (void) \linebreak

The **swf_nextid()** function returns the next available object id.

swf_oncondition (PHP 4)

Describe a transition used to trigger an action list

void **swf_oncondition** (int transition) \linebreak

The **swf_onCondition()** function describes a transition that will trigger an action list. There are several types of possible transitions, the following are for buttons defined as TYPE_MENUBUTTON:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- IdletoOverDown
- OutDowntoIdle
- MenuEnter (IdletoOverUp|IdletoOverDown)
- MenuExit (OverUptoIdle|OverDowntoIdle)

For TYPE_PUSHBUTTON there are the following options:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- OverDowntoOutDown
- OutDowntoOverDown
- OutDowntoIdle
- ButtonEnter (IdletoOverUp|OutDowntoOverDown)
- ButtonExit (OverUptoIdle|OverDowntoOutDown)

swf_openfile (PHP 4)

Open a new Shockwave Flash file

void **swf_openfile** (string filename, float width, float height, float framerate, float r, float g, float b) \linebreak

The **swf_openfile()** function opens a new file named *`filename`* with a width of *`width`* and a height of *`height`* a frame rate of *`framerate`* and background with a red color of *`r`* a green color of *`g`* and a blue color of *`b`*.

The **swf_openfile()** must be the first function you call, otherwise your script will cause a segfault. If you want to send your output to the screen make the filename: "php://stdout" (support for this is in 4.0.1 and up).

swf_ortho2 (PHP 4)

Defines 2D orthographic mapping of user coordinates onto the current viewport

void **swf_ortho2** (double xmin, double xmax, double ymin, double ymax) \linebreak

The **swf_ortho2()** function defines a two dimensional orthographic mapping of user coordinates onto the current viewport, this defaults to one to one mapping of the area of the Flash movie. If a perspective transformation is desired, the **swf_perspective ()** function can be used.

swf_ortho (PHP 4 >= 4.0.1)

Defines an orthographic mapping of user coordinates onto the current viewport

void **swf_ortho** (double xmin, double xmax, double ymin, double ymax, double zmin, double zmax) \linebreak

The **swf_ortho()** funcion defines a orthographic mapping of user coordinates onto the current viewport.

swf_perspective (PHP 4)

Define a perspective projection transformation

void **swf_perspective** (double fovy, double aspect, double near, double far) \linebreak

The **swf_perspective()** function defines a perspective projection transformation. The *fovy* parameter is field-of-view angle in the y direction. The *aspect* parameter should be set to the aspect ratio of the viewport that is being drawn onto. The *near* parameter is the near clipping plane and the *far* parameter is the far clipping plane.

> **Nota:** Various distortion artifacts may appear when performing a perspective projection, this is because Flash players only have a two dimensional matrix. Some are not to pretty.

swf_placeobject (PHP 4)

Place an object onto the screen

void **swf_placeobject** (int objid, int depth) \linebreak

Places the object specified by *objid* in the current frame at a depth of *depth*. The *objid* parameter and the *depth* must be between 1 and 65535.

This uses the current mulcolor (specified by swf_mulcolor()) and the current addcolor (specified by swf_addcolor()) to color the object and it uses the current matrix to position the object.

Nota: Full RGBA colors are supported.

swf_polarview (PHP 4)

Define the viewer's position with polar coordinates

void **swf_polarview** (double dist, double azimuth, double incidence, double twist) \linebreak

The **swf_polarview()** function defines the viewer's position in polar coordinates. The *dist* parameter gives the distance between the viewpoint to the world space origin. The *azimuth* parameter defines the azimuthal angle in the x,y coordinate plane, measured in distance from the y axis. The *incidence* parameter defines the angle of incidence in the y,z plane, measured in distance from the z axis. The incidence angle is defined as the angle of the viewport relative to the z axis. Finally the *twist* specifies the amount that the viewpoint is to be rotated about the line of sight using the right hand rule.

swf_popmatrix (PHP 4)

Restore a previous transformation matrix

void **swf_popmatrix** (void) \linebreak

The **swf_popmatrix()** function pushes the current transformation matrix back onto the stack.

swf_posround (PHP 4)

Enables or Disables the rounding of the translation when objects are placed or moved

void **swf_posround** (int round) \linebreak

The **swf_posround()** function enables or disables the rounding of the translation when objects are placed or moved, there are times when text becomes more readable because rounding has been enabled. The *round* is whether to enable rounding or not, if set to the value of 1, then rounding is enabled, if set to 0 then rounding is disabled.

swf_pushmatrix (PHP 4)

Push the current transformation matrix back unto the stack

void **swf_pushmatrix** (void) \linebreak

The **swf_pushmatrix()** function pushes the current transformation matrix back onto the stack.

swf_removeobject (PHP 4)

Remove an object

void **swf_removeobject** (int depth) \linebreak

Removes the object at the depth specified by *depth*.

swf_rotate (PHP 4)

Rotate the current transformation

void **swf_rotate** (double angle, string axis) \linebreak

The **swf_rotate()** rotates the current transformation by the angle given by the *angle* parameter around the axis given by the *axis* parameter. Valid values for the axis are 'x' (the x axis), 'y' (the y axis) or 'z' (the z axis).

swf_scale (PHP 4)

Scale the current transformation

void **swf_scale** (double x, double y, double z) \linebreak

The **swf_scale()** scales the x coordinate of the curve by the value of the *x* parameter, the y coordinate of the curve by the value of the *y* parameter, and the z coordinate of the curve by the value of the *z* parameter.

swf_setfont (PHP 4)

Change the current font

void **swf_setfont** (int fontid) \linebreak

The **swf_setfont()** sets the current font to the value given by the *fontid* parameter.

swf_setframe (PHP 4)

Switch to a specified frame

void **swf_setframe** (int framenumber) \linebreak

The **swf_setframe()** changes the active frame to the frame specified by *framenumber*.

swf_shapearc (PHP 4)

Draw a circular arc

void **swf_shapearc** (float x, float y, float r, float ang1, float ang2) \linebreak

The **swf_shapeArc()** function draws a circular arc from angle A given by the *ang1* parameter to angle B given by the *ang2* parameter. The center of the circle has an x coordinate given by the *x* parameter and a y coordinate given by the *y*, the radius of the circle is given by the *r* parameter.

swf_shapecurveto3 (PHP 4)

Draw a cubic bezier curve

void **swf_shapecurveto3** (float x1, float y1, float x2, float y2, float x3, float y3) \linebreak

Draw a cubic bezier curve using the x,y coordinate pairs *x1*, *y1* and *x2*,*y2* as off curve control points and the x,y coordinate *x3*, *y3* as an endpoint. The current position is then set to the x,y coordinate pair given by *x3*,*y3*.

swf_shapecurveto (PHP 4)

Draw a quadratic bezier curve between two points

void **swf_shapecurveto** (float x1, float y1, float x2, float y2) \linebreak

The **swf_shapecurveto()** function draws a quadratic bezier curve from the x coordinate given by *x1* and the y coordinate given by *y1* to the x coordinate given by *x2* and the y coordinate given by *y2*. The current position is then set to the x,y coordinates given by the *x2* and *y2* parameters

swf_shapefillbitmapclip (PHP 4)

Set current fill mode to clipped bitmap

void **swf_shapefillbitmapclip** (int bitmapid) \linebreak

Sets the fill to bitmap clipped, empty spaces will be filled by the bitmap given by the *bitmapid* parameter.

swf_shapefillbitmaptile (PHP 4)

Set current fill mode to tiled bitmap

void **swf_shapefillbitmaptile** (int bitmapid) \linebreak

Sets the fill to bitmap tile, empty spaces will be filled by the bitmap given by the *bitmapid* parameter (tiled).

swf_shapefilloff (PHP 4)

Turns off filling

void **swf_shapefilloff** (void) \linebreak

The **swf_shapeFillOff()** function turns off filling for the current shape.

swf_shapefillsolid (PHP 4)

Set the current fill style to the specified color

void **swf_shapefillsolid** (float r, float g, float b, float a) \linebreak

The **swf_shapeFillSolid()** function sets the current fill style to solid, and then sets the fill color to the values of the *rgba* parameters.

swf_shapelinesolid (PHP 4)

Set the current line style

void **swf_shapelinesolid** (float r, float g, float b, float a, float width) \linebreak

The **swf_shapeLineSolid()** function sets the current line style to the color of the *rgba* parameters and width to the *width* parameter. If 0.0 is given as a width then no lines are drawn.

swf_shapelineto (PHP 4)

Draw a line

void **swf_shapelineto** (float x, float y) \linebreak

The **swf_shapeLineTo()** draws a line to the x,y coordinates given by the *x* parameter & the *y* parameter. The current position is then set to the x,y parameters.

swf_shapemoveto (PHP 4)

Move the current position

void **swf_shapemoveto** (float x, float y) \linebreak

The **swf_shapeMoveTo()** function moves the current position to the x coordinate given by the *x* parameter and the y position given by the *y* parameter.

swf_showframe (PHP 4)

Display the current frame

void **swf_showframe** (void) \linebreak

The swf_showframe function will output the current frame.

swf_startbutton (PHP 4)

Start the definition of a button

void **swf_startbutton** (int objid, int type) \linebreak

The **swf_startbutton()** function starts off the definition of a button. The *type* parameter can either be TYPE_MENUBUTTON or TYPE_PUSHBUTTON. The TYPE_MENUBUTTON constant allows the focus to travel from the button when the mouse is down, TYPE_PUSHBUTTON does not allow the focus to travel when the mouse is down.

swf_startdoaction (PHP 4)

Start a description of an action list for the current frame

void **swf_startdoaction** (void) \linebreak

The **swf_startdoaction()** function starts the description of an action list for the current frame. This must be called before actions are defined for the current frame.

swf_startshape (PHP 4)

Start a complex shape

void **swf_startshape** (int objid) \linebreak

The **swf_startshape()** function starts a complex shape, with an object id given by the *objid* parameter.

swf_startsymbol (PHP 4)

Define a symbol

void **swf_startsymbol** (int objid) \linebreak

Define an object id as a symbol. Symbols are tiny flash movies that can be played simultaneously. The *objid* parameter is the object id you want to define as a symbol.

swf_textwidth (PHP 4)

Get the width of a string

float **swf_textwidth** (string str) \linebreak

The **swf_textwidth()** function gives the width of the string, *str*, in pixels, using the current font and font size.

swf_translate (PHP 4)

Translate the current transformations

void **swf_translate** (double x, double y, double z) \linebreak

The **swf_translate()** function translates the current transformation by the *x*, *y*, and *z* values given.

swf_viewport (PHP 4)

Select an area for future drawing

void **swf_viewport** (double xmin, double xmax, double ymin, double ymax) \linebreak

The **swf_viewport()** function selects an area for future drawing for *xmin* to *xmax* and *ymin* to *ymax*, if this function is not called the area defaults to the size of the screen.

XCV. Funciones SNMP

Para usar las funciones SNMP en Unix necesita instalar el paquete UCD SNMP (http://net-snmp.sourceforge.net/). En Windows estas funciones están solamente disponibles en NT y no en Win95/98.

Importante: Para usar el paquete UCD SNMP, necesita definir NO_ZEROLENGTH_COMMUNITY a 1 antes de compilarlo. Despues de configurar UCD SNMP, edite config.h y busque NO_ZEROLENGTH_COMMUNITY. Descomente la línea #define. Debería de verse como sigue:

```
#define NO_ZEROLENGTH_COMMUNITY 1
```

Si ve faltas de segmentación desconocidas en combinación con los comandos SNMP, no siga las siguientes instrucciones. Si no desea recompilar UCD SNMP, puede compilar PHP con la opción --enable-ucd-snmp-hack la cual trabajará entorno a las mismas características.

snmp_get_quick_print (PHP 3>= 3.0.8, PHP 4)

Va a buscar el valor actual de la biblioteca UCD estableciendo quick_print

boolean **snmp_get_quick_print** (void) \linebreak

Delvuele el valor actual almacenado en la biblioteca UCD para quick_print. quick_print está desactivado por defecto.

```
$quickprint = snmp_get_quick_print();
```

La llamada a la función superior podría devolver `FALSE` si quick_print está activo, y `TRUE` si quick_print está activo.

snmp_get_quick_print() está solamente disponible cuando estemos usando la biblioteca UCD SNMP. Esta función no está disponible cuando estemos usando la biblioteca Windows SNMP.

Ver: **snmp_get_quick_print()** para una descripción completa de lo que hace quick_print.

snmp_set_quick_print (PHP 3>= 3.0.8, PHP 4)

Establece el valor de quick_print con el de la biblioteca UCD SNMP.

void **snmp_set_quick_print** (boolean quick_print) \linebreak

Establece el valor de quick_print con la biblioteca UCD SNMP. Cuando esto está establecido (1), la biblioteca SNMP devolverá valores 'quick printed'. De esta manera sólo el valor será impreso. Cuando quick_print no está activada (por defecto) la biblioteca UCD SNMP imprime información extra incluyendo el tipo del valor (p. Ej. IPAddress o OID). Adicionalmente, si quick_print no está activado, la biblioteca imprime valores hexadecimales adicionales para todas las cadenas de 3 o menos caracteres.

El ajuste de quick_print es generalmente usado cuando usando la información devuelta con anterioridad se muestra.

```
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
```

El primer valor impreso debe de ser: 'Timeticks: (0) 0:00:00.00', donde qick_print se activa, solo se imprimira '0:00:00.00'.

Por defecto la biblioteca UCD SNMP devuelve valores detallados, quick_print es usado para devolver solamente el valor.

Las cadenas son mantenidas normalmente con comillas extra, esto será corregido en versiones posteriores.

snmp_get_quick_print() está sólo disponible cuando estemos usando la biblioteca UCD SNMP. Esta función no está disponible cuando estemos usando la biblioteca Windows SNMP.

snmpget (PHP 3, PHP 4)

Va a buscar un objeto SNMP

string **snmpget** (string hostname, string community, string object_id [, int timeout [, int retries]]) \linebreak

Devuelve el valor de un objeto SNMP en caso de exito y `FALSE` en caso de error.

La función **snmpget()** es usada para leer el valor de un objeto SNMP especificado por el *`object_id`*. El agente SNMP es especificado por el *`hostname`* y la comunidad lectora es especificada por el parametro *`community`*.

$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0")

snmprealwalk (PHP 3>= 3.0.8, PHP 4)

Return all objects including their respective object ID within the specified one

array **snmprealwalk** (string host, string community, string object_id [, int timeout [, int retries]]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

snmpset (PHP 3>= 3.0.12, PHP 4)

Va a buscar un objeto SNMP

string **snmpget** (string hostname, string community, string object_id, string type, mixed value [, int timeout [, int retries]]) \linebreak

Establece el valor especificado para el objeto SNMP, devolviendo `TRUE` en caso de exito o `FALSE` en caso de error.

La función **snmpset()** es usada para establecer el valor de un objeto SNMP especificado por el *`object_id`*. El agente SNMP es espcificado por el *`hostname`* y la comunidad lectora por el parametro *`community`*.

snmpwalk (PHP 3, PHP 4)

Busqueda por un arbol de información acerca de un entidad de red

array **snmpwalk** (string hostname, string community, string object_id [, int timeout [, int retries]]) \linebreak

Devuelve una matriz de valores de objetos SMNP comenzando por el **object_id()** como raíz y `FALSE` en caso de error.

La función **snmpwalk()** es usada para leer todos los valores de un agente SNMP especificado por el *`hostname`*. *`Community`* especifica la comunidad lectora para el agente. Un *`object_id`* nulo se toma como la raíz del arbol de los objetos SNMP y todos los objetos por debajo de ese arbol son devueltos como una matriz. Si *`object_id`* es especificado, todos los objetos SNMP por debajo de *`object_id`* son devueltos.

```
$a = snmpwalk("127.0.0.1", "public", "");
```

Encima de una función de llamada podrían devolverse todos los objetos SNMP del agente SNMP en ejecución en el servidor local. Uno puede pasar por todos los valores con un bucle.

```
for ($i=0; $i<count($a); $i++) {
    echo $a[$i];
}
```

snmpwalkoid (PHP 3>= 3.0.8, PHP 4)

Busqueda por un arbol de información acerca de un entidad de red

array **snmpwalkoid** (string hostname, string community, string object_id [, int timeout [, int retries]]) \linebreak

Devuelve una matriz asociativa con los identificadores de los objetos y sus respectivos valores comenzando por el *`object_id`* como raíz y `FALSE` en caso de error.

La función **snmpwalkoid()** es usada para leer todos los identificadores de objetos y sus respectivos valores de un agente SNMP especificado por el nombre del servidor. La lectura de *`community`*

especifica la comunidad para el agente. Un *object_id* nulo es tomado como la raíz del arbol de objetos SNMP y todos los objetos por debajo de este arbol son devueltos como una matriz. Si *object_id* es especificado, todos los objetos SNMP inferiores al *object_id* son devueltos.

La existencia de **snmpwalkoid()** y snmpwalk() tiene razones historicas. Ambas funciones son proporcionadas para compatibilidad hacia atrás.

```
$a = snmpwalkoid("127.0.0.1", "public", "");
```

La llamada a las funciones superiores devuelve todos los objetos SNMP del agente SNMP en ejecución en el servidor local. Uno puede pasar por todos los valores con un bucle.

```
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br>\n";
}
```

XCVI. Socket functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The socket extension implements a low-level interface to the socket communication functions, providing the possibility to act as a socket server as well as a client.

For a more generic client-side socket interface, see fsockopen() and pfsockopen().

When using the socket functions described here, it is important to remember that while many of them have identical names to their C counterparts, they often have different declarations. Please be sure to read the descriptions to avoid confusion.

That said, those unfamiliar with socket programming can still find a lot of useful material in the appropriate Unix man pages, and there is a great deal of tutorial information on socket programming in C on the web, much of which can be applied, with slight modifications, to socket programming in PHP.

Ejemplo 1. Socket example: Simple TCP/IP server

This example shows a simple talkback server. Change the `address` and `port` variables to suit your setup and execute. You may then connect to the server with a command similar to: **telnet 192.168.1.53 10000** (where the address and port match your setup). Anything you type will then be output on the server side, and echoed back to you. To disconnect, enter 'quit'.

```
<?php
error_reporting(E_ALL);

/* Allow the script to hang around waiting for connections. */
set_time_limit(0);

$address = '192.168.1.53';
$port = 10000;

if (($sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    echo "socket() failed: reason: " . strerror($sock) . "\n";
}

if (($ret = bind($sock, $address, $port)) < 0) {
    echo "bind() failed: reason: " . strerror($ret) . "\n";
}

if (($ret = listen($sock, 5)) < 0) {
    echo "listen() failed: reason: " . strerror($ret) . "\n";
}

do {
    if (($msgsock = accept_connect($sock)) < 0) {
```

```
        echo "accept_connect() failed: reason: " . strerror($msgsock) . "\n";
        break;
    }
    do {
        $buf = '';
        $ret = read($msgsock, $buf, 2048);
        if ($ret < 0) {
            echo "read() failed: reason: " . strerror($ret) . "\n";
            break 2;
        }
        if ($ret == 0) {
            break 2;
        }
        $buf = trim($buf);
        if ($buf == 'quit') {
            close($msgsock);
            break 2;
        }
        $talkback = "PHP: You said '$buf'.\n";
        write($msgsock, $talkback, strlen($talkback));
        echo "$buf\n";
    } while (true);
    close($msgsock);
} while (true);

close($sock);
?>
```

Ejemplo 2. Socket example: Simple TCP/IP client

This example shows a simple, one-shot HTTP client. It simply connects to a page, submits a HEAD request, echoes the reply, and exits.

```
<?php
error_reporting(E_ALL);

echo "<h2>TCP/IP Connection</h2>\n";

/* Get the port for the WWW service. */
$service_port = getservbyname('www', 'tcp');

/* Get the IP address for the target host. */
$address = gethostbyname('www.php.net');

/* Create a TCP/IP socket. */
$socket = socket(AF_INET, SOCK_STREAM, 0);
if ($socket < 0) {
    echo "socket() failed: reason: " . strerror($socket) . "\n";
} else {
    "socket() successful: " . strerror($socket) . "\n";
```

```
}

echo "Attempting to connect to '$address' on port '$service_port'...";
$result = connect($socket, $address, $service_port);
if ($result < 0) {
    echo "connect() failed.\nReason: ($result) " . strerror($result) . "\n";
} else {
    echo "OK.\n";
}

$in = "HEAD / HTTP/1.0\r\n\r\n";
$out = '';

echo "Sending HTTP HEAD request...";
write($socket, $in, strlen($in));
echo "OK.\n";

echo "Reading response:\n\n";
while (read($socket, $out, 2048)) {
    echo $out;
}

echo "Closing socket...";
close($socket);
echo "OK.\n\n";
?>
```

socket_accept (PHP 4 >= 4.1.0)

Accepts a connection on a socket

resource **socket_accept** (resource socket) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

After the socket `socket` has been created using socket_create(), bound to a name with socket_bind(), and told to listen for connections with socket_listen(), this function will accept incoming connections on that socket. Once a successful connection is made, a new socket resource is returned, which may be used for communication. If there are multiple connections queued on the socket, the first will be used. If there are no pending connections, **socket_accept()** will block until a connection becomes present. If `socket` has been made non-blocking using socket_set_blocking() or socket_set_nonblock(), `FALSE` will be returned.

The socket resource returned by **socket_accept()** may not be used to accept new connections. The original listening socket `socket`, however, remains open and may be reused.

Returns a new socket resource on success, or `FALSE` on error. The actual error code can be retrieved by calling socket_last_error(). This error code may be passed to socket_strerror() to get a textual explanation of the error.

See also socket_bind(), socket_connect(), socket_listen(), socket_create(), and socket_strerror().

socket_bind (PHP 4 >= 4.1.0)

Binds a name to a socket

bool **socket_bind** (resource socket, string address [, int port]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

socket_bind() binds the name given in `address` to the socket described by `socket`, which must be a valid socket resource created with socket_create().

The `address` parameter is either an IP address in dotted-quad notation (e.g. `127.0.0.1`), if the socket is of the `AF_INET` family; or the pathname of a Unix-domain socket, if the socket family is `AF_UNIX`.

The *port* parameter is only used when connecting to an AF_INET socket, and designates the port on the remote host to which a connection should be made.

Devuelve TRUE si todo fue bien, FALSE en caso de fallo. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

See also socket_connect(), socket_listen(), socket_create(), socket_last_error() and socket_strerror().

socket_clear_error (PHP 4 >= 4.2.0)

Clears the error on the socket or the last error code

void **socket_clear_error** ([resource socket]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

This function clears the error code on the given socket or the global last socket error.

This function allows explicitely resetting the error code value either of a socket or of the extension global last error code. This may be useful to detect within a part of the application if an error occured or not.

See also socket_last_error() and socket_strerror().

socket_close (PHP 4 >= 4.1.0)

Closes a socket resource

void **socket_close** (resource socket) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

socket_close() closes the socket resource given by *socket*.

> **Nota: socket_close()** can't be used on PHP file resources created with fopen(), popen(), fsockopen(), or **psockopen()**; it is meant for sockets created with socket_create() or socket_accept().

See also socket_bind(), socket_listen(), socket_create() and socket_strerror().

socket_connect (PHP 4 >= 4.1.0)

Initiates a connection on a socket

bool **socket_connect** (resource socket, string address [, int port]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Initiates a connection using the socket resource *`socket`*, which must be a valid socket resource created with socket_create().

The *`address`* parameter is either an IP address in dotted-quad notation (e.g. `127.0.0.1`), if the socket is of the `AF_INET` family; or the pathname of a Unix-domain socket, if the socket family is `AF_UNIX`.

The *`port`* parameter is only used when connecting to an `AF_INET` socket, and designates the port on the remote host to which a connection should be made.

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

See also socket_bind(), socket_listen(), socket_create(), socket_last_error() and socket_strerror().

socket_create_listen (PHP 4 >= 4.1.0)

Opens a socket on port to accept connections

resource **socket_create_listen** (int port [, int backlog]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

This function is meant to ease the task of creating a new socket which only listens to accept new connections.

socket_create_listen() create a new socket resource of type `AF_INET` listening on *all* local interfaces on the given port waiting for new connections.

The *backlog* parameter defines the maximum length the queue of pending connections may grow to. SOMAXCONN may be passed as *backlog* parameter, see socket_listen() for more information.

socket_create_listen() returns a new socket resource on success or FALSE on error. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

> **Nota:** If you want to create a socket which only listens on a certain interfaces you need to use socket_create(), socket_bind() and socket_listen().

See also socket_create(), socket_bind(), socket_listen(), socket_last_error() and socket_strerror().

socket_create_pair (PHP 4 >= 4.1.0)

Creates a pair of indistinguishable sockets and stores them in fds.

bool **socket_create_pair** (int domain, int type, int protocol, array &fd) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_create (PHP 4 >= 4.1.0)

Create a socket (endpoint for communication)

resource **socket_create** (int domain, int type, int protocol) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Creates a communication endpoint (a socket), and returns a socket resource.

The *domain* parameter sets the domain (protocol family) to be used for communication. Currently, AF_INET and AF_UNIX are understood. AF_INET is typical used for internet based communication. AF_UNIX uses pathnames to identify sockets and can therefore only be used for local communication (which is faster, on the other hand).

The *type* parameter selects the socket type. This is one of SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET, SOCK_RAW, SOCK_RDM, or SOCK_PACKET. The two most common types are SOCK_DGRAM for UDP (connectionless) communication and SOCK_STREAM for TCP communication.

protocol sets the protocol which is either SOL_UDP or SOL_TCP.

Returns a socket resource on success, or FALSE on error. The actual error code can be retrieved by calling socket_last_error(). This error code may be passed to socket_strerror() to get a textual explanation of the error.

For more information on the usage of **socket_create()**, as well as on the meanings of the various parameters, see the Unix man page socket (2).

Nota: If an invalid *domain* or *type* is given, **socket_create()** defaults to AF_INET and SOCK_STREAM respectively and additionally emits an E_WARNING message.

See also socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_last_error(), and socket_strerror().

socket_get_option (PHP 4 CVS only)

Gets socket options for the socket

mixed **socket_get_option** (resource socket, int level, int optname) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

Nota: This function used to be called `socket_getopt()` prior to PHP 4.3.0

socket_getpeername (PHP 4 >= 4.1.0)

Queries the remote side of the given socket which may either result in host/port or in a UNIX filesystem path, dependent on its type.

bool **socket_getpeername** (resource socket, string &addr [, int &port]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

If the given socket is of type `AF_INET`, **socket_getpeername()** will return the peers (remote) *IP address* in dotted-quad notation (e.g. `127.0.0.1`) in the *`address`* parameter and, if the optional *`port`* parameter is present, also the associated port.

If the given socket is of type `AF_UNIX`, **socket_getpeername()** will return the UNIX filesystem path (e.g. `/var/run/daemon.sock`) in the *`address`* parameter.

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo. **socket_getpeername()** may also return `FALSE` if the socket type is not any of `AF_INET` or `AF_UNIX`, in which case the last socket error code is *not* updated.

See also **socket_getpeername()**, socket_last_error() and socket_strerror().

socket_getsockname (PHP 4 >= 4.1.0)

Queries the local side of the given socket which may either result in host/port or in a UNIX filesystem path, dependent on its type.

bool **socket_getsockname** (resource socket, string &addr [, int &port]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

If the given socket is of type `AF_INET`, **socket_getsockname()** will return the local *IP address* in dotted-quad notation (e.g. `127.0.0.1`) in the *`address`* parameter and, if the optional *`port`* parameter is present, also the associated port.

If the given socket is of type `AF_UNIX`, **socket_getsockname()** will return the UNIX filesystem path (e.g. `/var/run/daemon.sock`) in the *`address`* parameter.

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo. **socket_getsockname()** may also return `FALSE` if the socket type is not any of `AF_INET` or `AF_UNIX`, in which case the last socket error code is *not* updated.

See also socket_getpeername(), socket_last_error() and socket_strerror().

socket_iovec_add (PHP 4 >= 4.1.0)

Adds a new vector to the scatter/gather array

bool **socket_iovec_add** (resource iovec, int iov_len) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_iovec_alloc (PHP 4 >= 4.1.0)

...]) Builds a 'struct iovec' for use with sendmsg, recvmsg, writev, and readv

resource **socket_iovec_alloc** (int num_vectors [, int]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_iovec_delete (PHP 4 >= 4.1.0)

Deletes a vector from an array of vectors

bool **socket_iovec_delete** (resource iovec, int iov_pos) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_iovec_fetch (PHP 4 >= 4.1.0)

Returns the data held in the iovec specified by iovec_id[iovec_position]

string **socket_iovec_fetch** (resource iovec, int iovec_position) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_iovec_free (PHP 4 >= 4.1.0)

Frees the iovec specified by iovec_id

bool **socket_iovec_free** (resource iovec) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_iovec_set (PHP 4 >= 4.1.0)

Sets the data held in iovec_id[iovec_position] to new_val

bool **socket_iovec_set** (resource iovec, int iovec_position, string new_val) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_last_error (PHP 4 >= 4.1.0)

Returns the last error on the socket

int **socket_last_error** ([resource socket]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

This function returns a socket error code.

If a socket resource is passed to this function, the last error which occured on this particular socket is returned. If the socket resource is ommited, the error code of the last failed socket function is returned. The latter is in particular helpful for functions like socket_create() which don't return a socket on failure and socket_select() which can fail for reasons not directly tied to a particular socket. The error code is suitable to be fed to socket_strerror() which returns a string describing the given error code.

```
if (false == ($socket = @socket_create(AF_INET, SOCK_STREAM, SOL_TCP))) {
    die("Couldn't create socket, error code is: " . socket_last_error() .
        ",error message is: " . socket_strerror(socket_last_error()));
}
```

Nota: socket_last_error() does not clear the error code, use socket_clear_error() for this purpose.

socket_listen (PHP 4 >= 4.1.0)

Listens for a connection on a socket

bool **socket_listen** (resource socket [, int backlog]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

After the socket *socket* has been created using socket_create() and bound to a name with socket_bind(), it may be told to listen for incoming connections on *socket*.

A maximum of *backlog* incoming connections will be queued for processing. If a connection request arrives with the queue full the client may receive an error with an indication of ECONNREFUSED, or, if the underlying protocol supports retransmission, the request may be ignored so that retries may succeed.

> **Nota:** The maximum number passed to the *backlog* parameter highly depends on the underlying platform. On linux, it is silently truncated to SOMAXCONN. On win32, if passed SOMAXCONN, the underlying service provider responsible for the socket will set the backlog to a maximum *reasonable* value. There is no standard provision to find out the actual backlog value on this platform.

socket_listen() is applicable only to sockets of type SOCK_STREAM or SOCK_SEQPACKET.

Devuelve TRUE si todo fue bien, FALSE en caso de fallo. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

See also socket_accept(), socket_bind(), socket_connect(), socket_create() and socket_strerror().

socket_read (PHP 4 >= 4.1.0)

Reads a maximum of length bytes from a socket

string **socket_read** (resource socket, int length [, int type]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The function **socket_read()** reads from the socket resource *socket* created by the socket_create() or socket_accept() functions. The maximum number of bytes read is specified by the *length* parameter. Otherwise you can use \r, \n, or \0 to end reading (depending on the *type* parameter, see below).

socket_read() returns the data as a string on success, or FALSE on error. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual representation of the error.

Nota: socket_read() may return a zero length string ("") indicating the end of communication (i.e. the remote end point has closed the connection).

Optional *type* parameter is a named constant:

- PHP_BINARY_READ - use the system read() function. Safe for reading binary data. (Default in PHP >= 4.1.0)
- PHP_NORMAL_READ - reading stops at \n or \r. (Default in PHP <= 4.0.6)

See also socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_last_error(), socket_strerror() and socket_write().

socket_readv (PHP 4 >= 4.1.0)

Reads from an fd, using the scatter-gather array defined by iovec_id

bool **socket_readv** (resource socket, resource iovec_id) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_recv (PHP 4 >= 4.1.0)

Receives data from a connected socket

string **socket_recv** (resource socket, int len, int flags) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_recvfrom (PHP 4 >= 4.1.0)

Receives data from a socket, connected or not

int **socket_recvfrom** (resource socket, string &buf, int len, int flags, string &name [, int &port]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_recvmsg (PHP 4 >= 4.1.0)

Used to receive messages on a socket, whether connection-oriented or not

bool **socket_recvmsg** (resource socket, resource iovec, array &control, int &controllen, int &flags, string &addr [, int &port]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_select (PHP 4 >= 4.1.0)

Runs the select() system call on the given arrays of sockets with a timeout specified by tv_sec and tv_usec

int **socket_select** (resource &read, resource &write, resource &except, int tv_sec [, int tv_usec]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The **socket_select()** accepts arrays of sockets and waits for them to change status. Those coming with BSD sockets background will recognize that those socket resource arrays are in fact the so-called file descriptor sets. Three independent arrays of socket resources are watched.

The sockets listed in the *read* array will be watched to see if characters become available for reading (more precisely, to see if a read will not block - in particular, a socket resource is also ready on end-of-file, in which case a socket_read() will return a zero length string).

The sockets listed in the *write* array will be watched to see if a write will not block.

The sockets listed in the *except* array will be watched for exceptions.

> **Aviso**
>
> On exit, the arrays are modified to indicate which socket resource actually changed status.

You do not need to pass every array to **socket_select()**. You can leave it out and use an empty array or NULL instead. Also do not forget that those arrays are passed *by reference* and will be modified after **socket_select()** returns.

Example:

```
/* Prepare the read array */
$read = array($socket1, $socket2);

if (false === ($num_changed_sockets = socket_select($read, $write = NULL, $ex-
cept = NULL, 0))) {
    /* Error handling */
else if ($num_changed_sockets > 0) {
    /* At least at one of the sockets something interesting happened */
}
```

> **Nota:** Due a limitation in the current Zend Engine it is not possible to pass a constant modifier like NULL directly as a parameter to a function which expects this parameter to be passed by reference. Instead use a temporary variable or an expression with the leftmost member being a temporary variable:
>
> ```
> socket_select($r, $w, $e = NULL, 0);
> ```

The *tv_sec* and *tv_usec* together form the *timeout* parameter. The *timeout* is an upper bound on the amount of time elapsed before **socket_select()** return. *tv_sec* may be zero , causing **socket_select()** to

return immediately. This is useful for polling. If *tv_sec* is NULL (no timeout), **socket_select()** can block indefinitely.

On success **socket_select()** returns the number of socket resorces contained in the modified arrays, which may be zero if the timeout expires before anything interesting happens. On error FALSE is returned. The error code can be retrieved with socket_last_error().

Nota: Be sure to use the === operator when checking for an error. Since the **socket_select()** may return 0 the comparison with == would evaluate to TRUE:

```
if (false === socket_select($r, $w, $e = NULL, 0)) {
    echo "socket_select() failed, reason: " . socket_strerror(socket_last_error()) . "\n";
}
```

Nota: Be aware that some socket implementations need to be handled very carefully. A few basic rules:

- You should always try to use **socket_select()** without timeout. Your program should have nothing to do if there is no data available. Code that depends on timeouts is not usually portable and difficult to debug.
- No socket resource must be added to any set if you do not intend to check its result after the **socket_select()** call, and respond appropriately. After **socket_select()** returns, all socket resources in all arrays must be checked. Any socket resource that is available for writing must be written to, and any socket resource available for reading must be read from.
- If you read/write to a socket returns in the arrays be aware that they do not necessarily read/write the full amount of data you have requested. Be prepared to even only be able to read/write a single byte.
- It's common to most socket implementations that the only exception caught with the *except* array is out-of-bound data received on a socket.

See also socket_read(), socket_write(), socket_last_error() and socket_strerror().

socket_send (PHP 4 >= 4.1.0)

Sends data to a connected socket

int **socket_send** (resource socket, string buf, int len, int flags) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_sendmsg (PHP 4 >= 4.1.0)

Sends a message to a socket, regardless of whether it is connection-oriented or not

bool **socket_sendmsg** (resource socket, resource iovec, int flags, string addr [, int port]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_sendto (PHP 4 >= 4.1.0)

Sends a message to a socket, whether it is connected or not

int **socket_sendto** (resource socket, string buf, int len, int flags, string addr [, int port]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_set_nonblock (PHP 4 >= 4.1.0)

Sets nonblocking mode for file descriptor fd

bool **socket_set_nonblock** (resource socket) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_set_option (PHP 4 CVS only)

Sets socket options for the socket

bool **socket_set_option** (resource socket, int level, int optname, int) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

Nota: This function used to be called `socket_setopt()` prior to PHP 4.3.0

socket_shutdown (PHP 4 >= 4.1.0)

Shuts down a socket for receiving, sending, or both.

bool **socket_shutdown** (resource socket [, int how]) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

socket_strerror (PHP 4 >= 4.1.0)

Return a string describing a socket error

string **socket_strerror** (int errno) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

socket_strerror() takes as its *errno* parameter a socket error code as returned by socket_last_error() and returns the corresponding explanatory text. This makes it a bit more pleasant to figure out why something didn't work; for instance, instead of having to track down a system include file to find out what '-111' means, you just pass it to **socket_strerror()**, and it tells you what happened.

Ejemplo 1. socket_strerror() example

```
<?php
if (false == ($socket = @socket_create(AF_INET, SOCK_STREAM, 0))) {
   echo "socket_create() failed: reason: " . socket_strerror(socket_last_error()) . "\n";
}

if (false == (@socket_bind($socket, '127.0.0.1', 80))) {
   echo "socket_bind() failed: reason: " . socket_strerror(socket_last_error($socket)) . "\r
}
?>
```

The expected output from the above example (assuming the script is not run with root privileges):

```
socket_bind() failed: reason: Permission denied
```

See also socket_accept(), socket_bind(), socket_connect(), socket_listen(), and socket_create().

socket_write (PHP 4 >= 4.1.0)

Write to a socket

int **socket_write** (resource socket, string buffer [, int length]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

The function **socket_write()** writes to the socket *`socket`* from *`buffer`*.

The optional parameter *`length`* can specify an alternate length of bytes written to the socket. If this length is greater then the buffer length, it is silently truncated to the length of the buffer.

Returns the number of bytes successfully written to the socket or `FALSE` one error. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

> **Nota: socket_write()** does not necessarily write all bytes from the given buffer. It's valid that, depending on the network buffers etc., only a certain amount of data, even one byte, is written though your buffer is greater. You have to watch out so you don't unintentionally forget to transmit the rest of your data.

> **Nota:** It is perfectly valid for **socket_write()** to return zero which means no bytes have been written. Be sure to use the `===` operator to check for `FALSE` in case of an error.

See also socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_read() and socket_strerror().

socket_writev (PHP 4 >= 4.1.0)

Writes to a file descriptor, fd, using the scatter-gather array defined by iovec_id

bool **socket_writev** (resource socket, resource iovec_id) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

XCVII. Funciones de cadenas

Todas estas funciones manipulan cadenas de varias maneras. En las secciones sobre expresiones regulares y manejo de URL se pueden encontrar secciones más especializadas.

AddCSlashes (PHP 4)

Marca una cadena con barras al estilo del C

string **addcslashes** (string cad, string listcar) \linebreak

Devuelve una cadena con barras invertidas antes de los caracteres listados en el parámetro *listcar*. También marca \n, \r etc. Al estilo del C, los caracteres con códico ASCII inferior a 32 y superior a 126 son convertidos a representación octal. Tenga cuidado cuando marque caracteres alfanuméricos. Puede especificar un rango en *listcar* como el "\0..\37", que marcaría todos los caracteres con código ASCII entre 0 y 31.

Ejemplo 1. Ejemplo de addcslashes()

```
$tradformado = addcslashes ($no_transf, "\0..\37!@\177..\377");
```

Nota: Añadida en PHP4b3-dev.

Vea también stripcslashes(), stripslashes(), htmlspecialchars(), htmlspecialchars(), y quotemeta().

AddSlashes (PHP 3, PHP 4)

Marca una cadena con barras

string **addslashes** (string cad) \linebreak

Devuelve una cadena con barras invertidas frente a los caracteres que necesitan marcarse en consultas de bases de datos, etc. Estos son la comilla simple (’), comilla doble ("), barra invertida (\) y NUL (el byte nulo).

Vea también stripslashes(), htmlspecialchars(), y quotemeta().

bin2hex (PHP 3>= 3.0.9, PHP 4)

Convierte datos binarios en su representación hexadecimal

string **bin2hex** (string cad) \linebreak

Devuelve una cadena ASCII que contiene la representación hexadecimal de *cad*. La conversión se realiza byte a byte, con los 4 bits superiores primero.

chop (PHP 3, PHP 4)

Elimina espacios sobrantes al final

string **chop** (string cad) \linebreak

Devuelve la cadena argumento sin los espacios sobrantes, incluyendo los saltos de línea.

Ejemplo 1. Ejemplo de chop()

```
$recortada = chop ($linea);
```

Vea también trim().

chr (PHP 3, PHP 4)

Devuelve un caracter específico

string **chr** (int ascii) \linebreak

Devuelve una cadena de un caracter que congiene el caracter especificado por *ascii*.

Ejemplo 1. Ejemplo de chr()

```
$cad .= chr (27); /* añade un caracter de escape al final de $cad */

/* A veces esto es más útil */

$cad = sprintf ("La cadena termina en escape: %c", 27);
```

Esta función complementa a ord(). Vea también sprintf() con una cadena de formato `%c`.

chunk_split (PHP 3>= 3.0.6, PHP 4)

Divide una cadena en trozos más pequeños

string **chunk_split** (string cadena [, int tamatrozo [, string final]]) \linebreak

Se puede utilizar para trocear una cadena en pedazos más pequeños, lo que es útil, p.ej., para convertir la salida de la función base64_encode a la semántica del RFC 2045. Inserta la cadena *final* cada *tamatrozo* (por defecto vale 76) caracteres. Devuelve la nueva cadena y deja intacta la original.

Ejemplo 1. Ejemplo de chunk_split()

```
# formatear $datos usando la semántica del RFC 2045

$nueva_cad = chunk_split (base64_encode($datos));
```

Esta función es notablemente más rápida que ereg_replace().

Nota: Esta función se añadió en la 3.0.6.

convert_cyr_string (PHP 3>= 3.0.6, PHP 4)

Convierte de un juego de caracteres Cirílico a otro

string **convert_cyr_string** (string cad, string desde, string hasta) \linebreak

Esta función convierte la cadena dada de un juego de caracteres Cirílico a otro. Los argumentos *desde* y *hasta* son caracteres sencillos que representan los juegos de caracteres Cirílicos fuente y destino. Los tipos soportados son:

- k - koi8-r
- w - windows-1251
- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

count_chars (PHP 4)

Devuelve información sobre los caracteres usados en una cadena

mixed **count_chars** (string cadena [, modo]) \linebreak

Cuenta el número de apariciones de cada valor de byte (0..255) en *cadena* y lo devuelve de varias maneras. El parámetro opcional *modo* vale por defecto 0. Dependiendo de *modo*, **count_chars()** puede devolver:

- 0 - una matriz con el valor del byte como clave y la frecuencia de cada uno como valor.
- 1 - como el 0, pero listando únicamente los valores de byte con frecuencia superior a cero.

- 2 - como el 0, pero listando únicamente los valores de byte con frecuencia igual a 0.
- 3 - se devuelve una cadena que contiene todos los valores de byte utilizados.
- 4 - se devuelve una cadena que contiene todos los valores de byte no utilizados.

Nota: Esta función se añadió en el PHP 4.0.

crc32 (PHP 4 >= 4.0.1)

Calcula el polinomio crc32 de una cadena

int **crc32** (string cad) \linebreak

Genera el polinomio de comprobación de reduncancia cíclica de 32 bits de *cad*. Se suele utilizar para validad la integridad de los datos transmitidos.

Vea también: md5()

crypt (PHP 3, PHP 4)

Encripta una cadena mediante DES

string **crypt** (string cad [, string semilla]) \linebreak

crypt() encriptará una cadena utilizando el método estándar de encriptación del Unix DES. Los argumentos son una cadena a encriptar y una cadena semilla de 2 caracteres en la que basar la encriptación. Vea la página de manual de Unix sobre crypt para más información.

Si el argumento de semilla no se proporciona, será generado aleatoriamente por el PHP.

Algunos sistemas operativos soportan más de un tipo de encriptación. De hecho, algunas veces la encriptación estándar DES es sustituída por un algoritmo de encriptación basado en MD5. El tipo de encriptación es disparado por el argumento semilla. En tiempo de instalación, el PHP determina la capacidad de la función de encriptación y aceptará semillas para otros tipos de encriptación. Si no se proporciona la semilla, el PHP intentará generar una semilla estándar DES de 2 caraceres por defecto, excepto si el tipo de encriptación estándar del sistema es el MD5, en cuyo caso se generará una semilla aleatoria compatible con MD5. El PHP fija una constante llamada CRYPT_SALT_LENGTH que le especifica si su sistema soporta una semilla de 2 caracteres o si se debe usar la semilla de 12 caracteres del NDS.

La función estándar de encriptación **crypt()** contiene la semilla como los dos primeros caracteres de la salida.

En los sistemas en los que la función crypt() soporta múltiples tipos de encriptación, las siguienes constantes son fijadas a 0 ó 1 dependiendo de si está disponible el tipo dado:

- CRYPT_STD_DES - Encriptación DES estándar con semilla de 2 caracteres
- CRYPT_EXT_DES - Encriptación DES extendida con semilla de 9 caracteres
- CRYPT_MD5 - Encriptación MD5 con semilla de 12 caracteres y comenzando por 1
- CRYPT_BLOWFISH - Encriptación DES extendida con semilla de 16 caracteres y comenzando por 2

No hay función de desencriptado porque **crypt()** utiliza un algoritmo de una sola vía.

Vea también: md5().

echo (unknown)

Da salida a una o más cadenas

echo (string arg1 [, string argn...]) \linebreak

Da salida a todos sus parámetros.

echo() no es realmente una función (es una sentencia del lenguaje) de modo que no se requiere el uso de los paréntesis.

Ejemplo 1. Ejemplo de echo()

```
echo "Hola Mundo";

echo "Esto se extiende
por varias líneas. Los saltos de línea
también se envían";

echo "Esto se extiende\npor varias líneas. Los saltos de línea\ntambién se envían";
```

Nota: De hecho, si desea pasar más de un parámetro a echo no debe encerrarlos entre paréntesis.

Vea también: print(), printf(), y flush().

explode (PHP 3, PHP 4)

Divide una cadena por otra

array **explode** (string separador, string cadena [, int limite]) \linebreak

Devuelve una matriz de cadenas, cada una de las cuales es una subcadena de *cadena* formada mediante su división en las fronteras marcadas por la cadena *separador*. Si se especifica *limite*, la matriz devuelta contendrá un máximo de *limite* elementos con el último conteniendo el resto de la *cadena*.

Ejemplo 1. Ejemplo de explode()

```
$pizza = "trozo1 trozo2 trozo3 trozo4 trozo5 trozo6";
$trozos = explode (" ", $pizza);
```

Vea también split() e implode().

get_html_translation_table (PHP 4)

Devuelve la tabla de traducción utilizada por htmlspecialchars() y htmlentities()

string **get_html_translation_table** (int tabla) \linebreak

get_html_translation_table() devolverá la tabla de traducción que se usa internamente para htmlspecialchars() y htmlentities(). Hay dos nuevas definiciones (*HTML_ENTITIES*, *HTML_SPECIALCHARS*) que le permiten especificar la tabla deseada.

Ejemplo 1. Ejemplo de Tabla de Traducción

```
$trad = get_html_translation_table (HTML_ENTITIES);
$cad = "Hallo & <Frau> & Krämer";
$codif = strtr ($cad, $trad);
```

La variable `$codif` contendrá ahora: "Hallo & <Frau> & Krämer".

Lo interesante es usar la función array_flip() para cambiar la dirección de la traducción.

```
$trad = array_flip ($trad);
$original = strtr ($cad, $trad);
```

El contenido de `$original` sería: "Hallo & <Frau> & Krämer".

Nota: Esta función fue añadida en PHP 4.0.

Vea también: htmlspecialchars(), htmlentities(), strtr(), y array_flip().

get_meta_tags (PHP 3>= 3.0.4, PHP 4)

Extrae todas las etiquetas meta de un archivo y retorna una matriz

array **get_meta_tags** (string nombrefich [, int use_ruta_include]) \linebreak

Abre el *nombrefich* y lo trocea línea a línea buscando etiquetas <meta> de la forma

Ejemplo 1. Ejemplo de Etiquetas Meta

```
<meta name="autor" content="nombre">
<meta name="etiquetas" content="documentación de php3">
</head> <!-- el proceso se detiene aquí -->
```

(preste atención a los finales de línea - el PHP utiliza una función nativa para trocear la entrada, de modo que un archivo de Mac no funcionará en Unix).

El valor de la propiedad name queda como clave y el valor de la propiedad content queda como el valor de la matriz devuelta, de modo que pueda usar fácilmente funciones estándar de matrices para recorrerla o para acceder a valores individuales. Los caracteres especiales en el valor de name son sustituídos por '_' y el resto es convertido a minúsculas.

Fijando *use_ruta_include* a 1 hará que el PHP intente abrir el archivo a través de la ruta de inclusión.

hebrev (PHP 3, PHP 4)

Convierte Hebreo lógico a texto visual

string **hebrev** (string texto_hebreo [, int max_cars_por_linea]) \linebreak

El parámetro opcional *max_cars_por_linea* indica el máximo número de caracteres que se emitirán por línea. La función intenta evitar cortar palabras.

Vea también hebrevc()

hebrevc (PHP 3, PHP 4)

Convierte Hebreo lógico a texto visual con conversión de saltos de línea

string **hebrevc** (string texto_hebreo [, int max_cars_por_linea]) \linebreak

Esta funcion es similar a hebrev() con la diferencia que convierte las nuevas líneas (\n) a "
\n". El parámetro opcional *max_cars_por_linea* indica el máximo número de caracteres que se emitirán por línea. La función intenta evitar cortar palabras.

Vea también hebrev()

htmlentities (PHP 3, PHP 4)

Convierte todos los caracteres aplicables a entidades HTML

string **htmlentities** (string cadena) \linebreak

Esta función es del todo idéntica a htmlspecialchars(), excepto que traduce todos los caracteres que tienen equivalente como entidad HTML.

Actualmente se utiliza el juego de caracteres ISO-8859-1.

Vea también htmlspecialchars() y nl2br().

htmlspecialchars (PHP 3, PHP 4)

Convierte caracteres especiales a entidades HTML

string **htmlspecialchars** (string cadena) \linebreak

Ciertos caracteres tienen significados especiales en HTML, y deben ser representados por entidades HTML si se desea preservar su significado. Esta función devuelve una cadena con dichas conversiones realizadas.

Esta función es útil para evitar que el texo entrado por el usuario contenga marcas HTML, como ocurre en aplicaciones de foros o libros de visita.

Actualmente, las traducciones hechas son:

- '&' (ampersand) se convierte en '&'
- '"' (doble comilla) se convierte en '"'
- '<' (menor que) se convierte en '<'
- '>' (mayor que) se convierte en '>'

Nótese que esta función no traduce nada más que lo mostrado más arriba. Para una traducción de entidades completa, vea htmlentities().

Vea también htmlentities() y nl2br().

implode (PHP 3, PHP 4)

Unir elementos de una matriz mediante una cadena

string **implode** (string cola, array piezas) \linebreak

Devuelve una cadena que contiene una representación de todos los elementos de la matriz en el mismo orden, pero con la cadena `cola` en medio de los mismos.

Ejemplo 1. Ejemplo de implode()

```
$separada_dospuntos = implode (":", $matrizay);
```

Vea también explode(), join(), y split().

join (PHP 3, PHP 4)

Une elementos de una tabla mediante una cadena

string **join** (string cola, array piezas) \linebreak

join() es un alias para implode(), y es idéntica en todo.

Vea también explode(), implode(), y split().

levenshtein (PHP 3>= 3.0.17, PHP 4 >= 4.0.1)

Calcula la distancia Levenshtein entre dos cadenas

int **levenshtein** (string cad1, string cad2) \linebreak

Esta función devuelve la distancia Levenshtein entre las dos cadenas argumento, ó -1 si alguna de las cadenas tiene más de 255 caracteres.

La distancia Levenshtein se define como el mínimo número de caracteres que se tienen que sustituir, insertar o borrar para transformar *`cad1`* en *`cad2`*. La complejidad del algoritmo es `O(m*n)`, donde `n` y `m` son las longitudes de *`cad1`* y *`cad2`* (bastante bueno si se la compara con similar_text(), que es O(max(n,m)**3), pero aún es cara).

Vea también soundex(), similar_text() y metaphone().

localeconv (PHP 4 >= 4.0.5)

Get numeric formatting information

array **localeconv** (void) \linebreak

Returns an associative array containing localized numeric and monetary formatting information.

localeconv() returns data based upon the current locale as set by setlocale(). The associative array that is returned contains the following fields:

Array element	Description

Array element	Description
decimal_point	Decimal point character
thousands_sep	Thousands separator
grouping	Array containing numeric groupings
int_curr_symbol	International currency symbol (i.e. USD)
currency_symbol	Local currency symbol (i.e. $)
mon_decimal_point	Monetary decimal point character
mon_thousands_sep	Monetary thousands separator
mon_grouping	Array containing monetary groupings
positive_sign	Sign for positive values
negative_sign	Sign for negative values
int_frac_digits	International fractional digits
frac_digits	Local fractional digits
p_cs_precedes	`TRUE` if currency_symbol precedes a positive value, `FALSE` if it succeeds one
p_sep_by_space	`TRUE` if a space separates currency_symbol from a positive value, `FALSE` otherwise
n_cs_precedes	`TRUE` if currency_symbol precedes a negative value, `FALSE` if it succeeds one
n_sep_by_space	`TRUE` if a space separates currency_symbol from a negative value, `FALSE` otherwise
p_sign_posn	0 Parentheses surround the quantity and currency_symbol 1 The sign string precedes the quantity and currency_symbol 2 The sign string succeeds the quantity and currency_symbol 3 The sign string immediately precedes the currency_symbol 4 The sign string immediately succeeds the currency_symbol

Array element	Description
n_sign_posn	0 Parentheses surround the quantity and currency_symbol 1 The sign string precedes the quantity and currency_symbol 2 The sign string succeeds the quantity and currency_symbol 3 The sign string immediately precedes the currency_symbol 4 The sign string immediately succeeds the currency_symbol

The grouping fields contain arrays that define the way numbers should be grouped. For example, the grouping field for the en_US locale, would contain a 2 item array with the values 3 and 3. The higher the index in the array, the farther left the grouping is. If an array element is equal to CHAR_MAX, no further grouping is done. If an array element is equal to 0, the previous element should be used.

Ejemplo 1. localeconv() example

```
setlocale(LC_ALL, "en_US");

$locale_info = localeconv();

echo "<PRE>\n";
echo "-------------------------------------------\n";
echo "  Monetary information for current locale:  \n";
echo "-------------------------------------------\n\n";

echo "int_curr_symbol:   {$locale_info["int_curr_symbol"]}\n";
echo "currency_symbol:   {$locale_info["currency_symbol"]}\n";
echo "mon_decimal_point: {$locale_info["mon_decimal_point"]}\n";
echo "mon_thousands_sep: {$locale_info["mon_thousands_sep"]}\n";
echo "positive_sign:     {$locale_info["positive_sign"]}\n";
echo "negative_sign:     {$locale_info["negative_sign"]}\n";
echo "int_frac_digits:   {$locale_info["int_frac_digits"]}\n";
echo "frac_digits:       {$locale_info["frac_digits"]}\n";
echo "p_cs_precedes:     {$locale_info["p_cs_precedes"]}\n";
echo "p_sep_by_space:    {$locale_info["p_sep_by_space"]}\n";
echo "n_cs_precedes:     {$locale_info["n_cs_precedes"]}\n";
echo "n_sep_by_space:    {$locale_info["n_sep_by_space"]}\n";
echo "p_sign_posn:       {$locale_info["p_sign_posn"]}\n";
echo "n_sign_posn:       {$locale_info["n_sign_posn"]}\n";
echo "</PRE>\n";
```

The constant CHAR_MAX is also defined for the use mentioned above.

See also: setlocale().

ltrim (PHP 3, PHP 4)

Elimina el espacio en blanco del principio de una cadena

string **ltrim** (string cad) \linebreak

Esta función elimina el espacio en blanco del principio de una cadena y devuelve la cadena resultante. Los caracteres de espacio que elimina realmente son: "\n", "\r", "\t", "\v", "\0", y el espacio en sí.

Vea también chop() y trim().

md5_file (PHP 4 >= 4.2.0)

Calculates the md5 hash of a given filename

string **md5_file** (string filename) \linebreak

Calculates the MD5 hash of the specified *`filename`* using the RSA Data Security, Inc. MD5 Message-Digest Algorithm (http://www.faqs.org/rfcs/rfc1321.html), and returns that hash.

This function has the same purpose of the command line utility md5sum.

See also: md5() and crc32()

md5 (PHP 3, PHP 4)

Calcula el hash md5 de una cadena

string **md5** (string cad) \linebreak

Calcula el hash (extracto) MD5 de *`cad`* usaneo el Algoritmo de Resumen de Mensajes MD5 de RSA Data Security, Inc. (http://www.faqs.org/rfcs/rfc1321.html).

Vea también: crc32()

metaphone (PHP 4)

Calcula la clave "metáfona" de una cadena

string **metaphone** (string cad) \linebreak

Calcula la clave "metáfona" de *cad*.

Similarmente a soundex(), metaphone crea la misma clave para palabras que suenan parecidas. Es más precisa que la función soundex(), pues conoce las reglas básicas de la pronunciación del Inglés. Las claves metafónicas generadas son de longitud variable.

Metaphone fue desarrollado por Lawrence Philips <lphilips@verity.com>. Se describe en ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

Nota: Esta función se añadió en PHP 4.0.

nl_langinfo (PHP 4 >= 4.1.0)

Query language and locale information

string **nl_langinfo** (int item) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

nl2br (PHP 3, PHP 4)

Convierte nuevas líneas a saltos de línea HTML

string **nl2br** (string cadena) \linebreak

Devuelve la *cadena* con '
' insertados antes de cada nueva línea.

Vea también htmlspecialchars(), htmlentities() y wordwrap().

ord (PHP 3, PHP 4)

Devuelve el valor ASCII de un caracter

int **ord** (string cadena) \linebreak

Devuelve el valor ASCII del primer caracter de *cadena*. Esta función complementa a chr().

Ejemplo 1. Ejemplo de ord()

```
if (ord ($cad) == 10) {
    echo "El primer caracter de \$cad es un salto de línea.\n";
}
```

Vea también chr().

parse_str (PHP 3, PHP 4)

Divide la cadena en variables

void **parse_str** (string cad) \linebreak

Divide *cad* como si fuera la cadena de consulta enviada por un URL y crea las variables en el ámbito actual.

Ejemplo 1. Usando parse_str()

```
$cad = "primero=valor&segundo[]=esto+funciona&segundo[]=otro";
parse_str($cad);
echo $primero;    /* escribe "valor" */
echo $segundo[0]; /* escribe "esto funciona" */
echo $segundo[1]; /* escribe "otro" */
```

print (unknown)

Emite una cadena

print (string arg) \linebreak

Emite *arg*.

Vea también: echo(), printf(), y flush().

printf (PHP 3, PHP 4)

Emite una cadena con formato

int **printf** (string formato [, mixed args]) \linebreak

Produce una salida según el *formato*, que es descrito en la documentación para sprintf().

Vea también: print(), sprintf(), sscanf(), fscanf(), y flush().

quoted_printable_decode (PHP 3>= 3.0.6, PHP 4)

Convierte una cadena con marcación imprimible a una cadena de 8 bits

string **quoted_printable_decode** (string cad) \linebreak

Esta función devuelve una cadena binaria de 8 bit que se corresponde con la cadena con marcación imprimible decodificada. Esta función es similar a imap_qprint(), pero sin requerir que el módulo IMAP funcione.

quotemeta (PHP 3, PHP 4)

Quote meta characters

string **quotemeta** (string cad) \linebreak

Devuelve una versión de la cadena con una barra invertida (\) antes de cada caracter de este conjunto:

```
. \\ + * ? [ ^ ] ( $ )
```

Vea también addslashes(), htmlentities(), htmlspecialchars(), nl2br(), y stripslashes().

rtrim (PHP 3, PHP 4)

Elimina espacios en blanco al final de la cadena.

string **rtrim** (string cad) \linebreak

Devuelve la cadena argumento sin espacios en blanco ni saltos de línea al final. Es un alias para chop().

Ejemplo 1. Ejemplo de rtrim()

```
$recortada = rtrim ($linea);
```

Vea también trim(), ltrim().

setlocale (PHP 3, PHP 4)

Fija la información de localidad

string **setlocale** (string categoria, string localidad) \linebreak

categoria es una cadena que especifica la categoría de las funciones afectadas por el ajuste de localidad:

- LC_ALL para todas las funciones
- LC_COLLATE para la comparación de cadenas - aún no incluída en el PHP
- LC_CTYPE para la conversión y clasificación de caracteres, como por ejemplo strtoupper()
- LC_MONETARY para localeconv() - aún no incluída en el PHP
- LC_NUMERIC para el separador decimal
- LC_TIME para el formato de fecha y hora con strftime()

Si *localidad* es la cadena vacía `""`, los nombres de localidad se fijarán a partir de las variables de entorno con los mismos nombres de las categorías anteriores, o desde "LANG".

Si la localidad es cero o `"0"`, el ajuste de localidad no se ve afectado y sólo se devuelve el ajuste actual.

setlocale devuelve la nueva localidad, o `FALSE` si la funcionalidad de localización no está disponible en la plataforma, la localidad especificada no existe o el nombre de categoría no es válido. Un nombre de categoría no válido también produce un mensaje de aviso.

similar_text (PHP 3>= 3.0.7, PHP 4)

Calcula la similitud entre dos cadenas

int **similar_text** (string primera, string segunda [, double porcentaje]) \linebreak

Esta función calcula la similitud entre dos cadenas según se describe en Oliver [1993]. Nótese que esta implementación no utiliza una pila como en el pseudo-código de Oliver, sino llamadas recursivas que pueden o no acelerar el proceso completo. Nótese también que la complejidad de este algoritmo es O(N**3), donde N es la longitud de la cadena más larga.

Pasando una referencia como tercer argumento, **similar_text()** calculará para usted la similitud como porcentaje. Devuelve el número de caracteres coincidentes en ambas cadenas.

soundex (PHP 3, PHP 4)

Calcula la clave soundex de una cadena

string **soundex** (string cad) \linebreak

Calcula la clave soundex de *cad*.

Las claves soundex tienen la propiedad de que las palabras que se pronuncian de forma parecida tienen la misma clave, de modo que se pueden usar para simplificar la búsqueda en las bases de datos cuando se conoce la pronunciación pero no la transcripción. Esta función soundex devuelve una cadena de 4 caracteres que comienza por una letra.

Esta función soundex en particular es la descrita por Donald Knuth en "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison-Wesley (1973), pp. 391-392.

Ejemplo 1. Ejemplos de Soundex

```
soundex ("Euler") == soundex ("Ellery") == 'E460';
soundex ("Gauss") == soundex ("Ghosh") == 'G200';
soundex ("Knuth") == soundex ("Kant") == 'H416';
soundex ("Lloyd") == soundex ("Ladd") == 'L300';
soundex ("Lukasiewicz") == soundex ("Lissajous") == 'L222';
```

sprintf (PHP 3, PHP 4)

Devuelve una cadena con formato

string **sprintf** (string formato [, mixed args]) \linebreak

Devuelve una cadena producida de acuerdo a la cadena de *formato*.

La cadena de formato está compuesta por cero o más directivas: caracteres ordinarios (excepto `%`) que son copiados directamente al resultado, y *especificaciones de conversión*, cada una de las cuales provoca la obtención de su propio parámetro. Esto se aplica tanto a **sprintf()** como a printf().

Cada especificación de conversión consiste en uno de estos elementos, por orden:

1. Un *especificador de relleno* opcional que indica qué caracter se utilizará para rellenar el resultado hasta el tamaño de cadena correcto. Este puede ser un espacio o un `0` (caracter cero). El valor por defecto es rellenar con espacios. Un caracter de relleno alternativo se puede especificar prefijándolo con una comilla simple (`'`). Vea los ejemplos más abajo.

2. Un *especificador de alineación* opcional que indica si el resultado debe ser alineado a la izquierda o a la derecha. Por defecto se alinea a la derecha; un caracter - aquí lo justificará a la izquierda.
3. Un número opcional, un *especificador de ancho* que dice el número de caracteres (mínimo) en que debería resultar esta conversión.
4. Un *especificador de precisión* opcional que indica cuántos dígitos decimales deben mostrarse para los números en coma flotante. Esta opción no tienen efecto para otros tipos que no sean double. (Otra función útil para formatear números es number_format()).
5. Un *especificador de tipo* que indica el tipo a usar para tratar los datos de los argumentos. Los tipos posibles son:
   `%` - un caracter literal de porcentaje. No se precisa argumento.
   `b` - el argumento es tratado como un entero y presentado como un número binario.
   `c` - el argumento es tratado como un entero, y presentado como el caracter con dicho valor ASCII.
   `d` - el argumento es tratado como un entero y presentado como un número decimal.
   `f` - el argumento es tratado como un doble y presentado como un número de coma flotante.
   `o` - el argumento es tratado como un entero, y presentado como un número octal.
   `s` - el argumento es tratado como una cadena y es presentado como tal.
   `x` - el argumento es tratado como un entero y presentado como un número hexadecimal (con minúsculas).
   `X` - el argumento es tratado como un entero y presentado como un número hexadecimal (con mayúsculas).

Vea también: printf(), sscanf(), fscanf(), y number_format().

Ejemplo 1. Ejemplo de sprintf(): enteros rellenos con ceros

```
$fechaiso = sprintf ("%04d-%02d-%02d", $anno, $mes, $dia);
```

Ejemplo 2. Ejemplo de sprintf(): formateando monedas

```
$pelas1 = 68.75;
$pelas2 = 54.35;
$pelas = $pelas1 + $pelas2;
// echo $pelas mostrará "123.1";
$formateado = sprintf ("%01.2f", $pelas);
// echo $formateado mostrará "123.10"
```

sscanf (PHP 4 >= 4.0.1)

Trocea la entrada desde una cadena según un formato dado

mixed **sscanf** (string cad, string formato [, string var1]) \linebreak

La función**sscanf()** es la función de entrada análoga de printf(). **sscanf()** lee del parámetro de cadena *cad* y lo interpreta según el *formato* especificado. Si sólo se pasan dos parámetros a esta función, los valores devueltos se harán en una matriz.

Ejemplo 1. Ejemplo de sscanf()

```
// obteniendo el número de serie
$numserie = sscanf("SN/2350001","SN/%d");
// y la fecha de fabricación
$fecha = "01 Enero 2000";
list($dia, $mes, $anno) = sscanf($fecha,"%d %s %d");
echo "El objeto $numserie fue fabricado el: $anno-".substr($mes,0,3)."-$dia\n";
```

Si se pasan los parámetros opcionales, la función devolverá el número de valores asignados. Los parámetros opcionales deben ser pasados por referencia.

Ejemplo 2. Ejemplo de sscanf() - usando parámetros opcionales

```
// obtener autor y generar la ficha DocBook
$autor = "24\tLewis Carroll";
$n = sscanf($autor,"%d\t%s %s", &$id, &$nombre, &$apell);
echo "<autor id='$id'>
        <firstname>$nombre</firstname>
        <surname>$apell</surname>
</author>\n";
```

Vea también: fscanf(), printf(), y sprintf().

str_pad (PHP 4 >= 4.0.1)

Rellena una cadena con otra hasta una longitud dada

string **str_pad** (string entrada, int tama_relleno [, string cad_relleno [, int tipo_relleno]]) \linebreak

Esta función rellena la cadena *entrada* por la derecha, la izquierda o por ambos lados hasta el largo indicado. Si no se especifica el argumento opcional *cad_relleno*, *entrada* es rellenada con espacios. En caso contrario, será rellenada con los caracteres de *cad_relleno* hasta el límite.

El argumento opcional *tipo_relleno* puede valer STR_PAD_RIGHT, STR_PAD_LEFT, o STR_PAD_BOTH. Si no se especifica, se asume que vale STR_PAD_RIGHT.

Si el valor de *tama_relleno* es negativo o menor que la longitud de la cadena de entrada, no se produce rellenado alguno.

Ejemplo 1. Ejemplo de str_pad()

```
$entrada = "Alien";
print str_pad($entrada, 10);                      // produce "Alien     "
print str_pad($entrada, 10, "-=", STR_PAD_LEFT);  // produce "-=-=-Alien"
print str_pad($entrada, 10, "_", STR_PAD_BOTH);   // produce "__Alien___"
```

str_repeat (PHP 4)

Repite una cadena

string **str_repeat** (string cad_entrada, int veces) \linebreak

Devuelve la *cad_entrada* repetida *veces*. *veces* debe ser mayor que 0.

Ejemplo 1. Ejemplo de str_repeat()

```
echo str_repeat ("-=", 10);
```

Esto mostrará "-=-=-=-=-=-=-=-=-=-=".

Nota: Esta función fue añadida en el PHP 4.0.

str_replace (PHP 3>= 3.0.6, PHP 4)

Sustitiye todas las apariciones de la aguja en el pajar por la cadena

string **str_replace** (string aguja, string cad, string pajar) \linebreak

Esta función sustituye todas las apariciones de la *aguja* en el *pajar* por la *cad* dada. Si no precisa reglas especiales de sustitución, deberá usar siempre esta función en lugar de ereg_replace().

Ejemplo 1. Ejemplo de str_replace()

```
$bodytag = str_replace ("%cuerpo%", "negro", "<body text=%cuerpo%>");
```

Esta función tiene seguridad binaria.

Nota: str_replace() fue añadida en PHP 3.0.6, pero tuvo errores hasta el PHP 3.0.8.

Vea también ereg_replace() y strtr().

str_rot13 (PHP 4 >= 4.2.0)

Perform the rot13 transform on a string

string **str_rot13** (string str) \linebreak

This function performs the ROT13 encoding on the *str* argument and returns the resulting string. The ROT13 encoding simply shifts every letter by 13 places in the alphabet while leaving non-alpha characters untouched. Encoding and decoding are done by the same function, passing an encoded string as argument will return the original version.

strcasecmp (PHP 3>= 3.0.2, PHP 4)

Comparación de cadenas insensible a mayúsculas y minúsculas y segura en modo binario

int **strcasecmp** (string cad1, string cad2) \linebreak

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Ejemplo 1. Ejemplo de strcasecmp()

```
$var1 = "Hello";
$var2 = "hello";
if (!strcasecmp ($var1, $var2)) {
    echo '$var1 es igual a $var2 en una comparación sin tener en cuenta '
     .'mayúsculas o minúsculas';
}
```

Vea también ereg(), strcmp(), substr(), stristr(), y strstr().

strchr (PHP 3, PHP 4)

Encuentra la primera aparición de un caracter

string **strchr** (string pajar, string aguja) \linebreak

Esta función es un alias para strstr(), y es idéntica en todo.

strcmp (PHP 3, PHP 4)

Comparación de cadenas con seguridad binaria

int **strcmp** (string cad1, string cad2) \linebreak

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Nótese que esta comparación es sensible a mayúsculas y minúsculas.

Vea también ereg(), strcasecmp(), substr(), stristr(), strncmp(), y strstr().

strcoll (PHP 4 >= 4.0.5)

Locale based string comparison

int **strcoll** (string str1, string str2) \linebreak

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal. **strcoll()** uses the current locale for doing the comparisons. If the current locale is C or POSIX, this function is equivalent to strcmp().

Note that this comparison is case sensitive, and unlike strcmp() this function is not binary safe.

See also ereg(), strcmp(), strcasecmp(), substr(), stristr(), strncasecmp(), strncmp(), strstr(), and setlocale().

strcspn (PHP 3>= 3.0.3, PHP 4)

Encuentra la longitud del elemento inicial que no coincide con la máscara

int **strcspn** (string cad1, string cad2) \linebreak

Devuelve la longitud del segmento inicial de *cad1* que *no* contiene ninguno de los caracteres de *cad2*.

Vea también strspn().

strip_tags (PHP 3>= 3.0.8, PHP 4)

Elimina marcas HTML y PHP de una cadena

string **strip_tags** (string cad [, string etiq_permitidas]) \linebreak

Esta función intenta eliminar todas las etiquetas HTML y PHP de la cadena dada. Causa error por precaución en caso de etiquetas incompletas o falsas. Utiliza la misma máquina de estados para eliminar las etiquetas que la función fgetss().

Puede usar el parámetro opcional para especificar las etiquetas que no deben eliminarse.

Nota: *etiq_permitidas* fue añadido en PHP 3.0.13, PHP4B3.

stripcslashes (PHP 4)

Desmarca la cadena marcada con addcslashes()

string **stripcslashes** (string cad) \linebreak

Devuelve una cadena con las barras invertidas eliminadas. Reconoce las marcas tipo C `\n`, `\r` ..., y la representación octal y hexadecimal.

Nota: Añadida en PHP4b3-dev.

Vea también addcslashes().

stripslashes (PHP 3, PHP 4)

Desmarca la cadena marcada con addslashes()

string **stripslashes** (string cad) \linebreak

Devuelve una cadena con las barras invertidas eliminadas (`\'` se convierte en `'`, etc.). Las barras invertidas dobles se convierten en sencillas.

Vea también addslashes().

stristr (PHP 3>= 3.0.6, PHP 4)

strstr() sin tener en cuenta mayúsculas o minúsculas

string **stristr** (string pajar, string aguja) \linebreak

Devuelve todo el *pajar* desde la primera aparición de la *aguja*, siendo el *pajar* examinado sin tener en cuenta mayúsculas o minúsculas.

Si la *aguja* no se encuentra, devuelve `FALSE`.

Si la *aguja* no es una cadena, es convertida a entero y usada como código de un carácter ASCII.

Vea también strchr(), strrchr(), substr(), y ereg().

strlen (PHP 3, PHP 4)

Obtiene la longitud de la cadena

int **strlen** (string cad) \linebreak

Devuelve la longitud de la *cadena*.

strnatcasecmp (PHP 4)

Comparación de cadenas insensible a mayúsculas y minúsculas usando un algoritmo de "orden natural"

int **strnatcasecmp** (string cad1, string cad2) \linebreak

Esta función implementa un algoritmo de comparación que ordena las cadenas alfanuméricas como lo haría un ser humano. El comportamiento de esta función es similar a strnatcmp(), pero la comparación no es sensible a mayúsculas y minúsculas. Para más información, vea la página de Martin Pool sobre Comparación de Cadenas en Orden Natural (http://naturalordersort.org/).

De forma similar a otras funciones de comparación de cadenas, esta devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Vea también ereg(), strcasecmp(), substr(), stristr(), strcmp(), strncmp(), strnatcmp(), y strstr().

strnatcmp (PHP 4)

Compara cadenas usando un algoritmo de "orden natural"

int **strnatcmp** (string cad1, string cad2) \linebreak

Esta función implementa un algoritmo de comparación que ordena las cadenas alfanuméricas como lo haría un ser humano, que es lo que se denomina "orden natural". A continuación se puede ver un ejemplo de la diferencia entre este algoritmo y los algoritmos de ordenación de cadenas habituales en los ordenadores (utilizados en strcmp()):

```
$matriz1 = $matriz2 = array ("img12.png","img10.png","img2.png","img1.png");
echo "Comparación de cadenas estándar\n";
usort($matriz1,"strcmp");
print_r($matriz1);
echo "\nComparación de cadenas en orden natural\n";
usort($matriz2,"strnatcmp");
print_r($matriz2);
```

El código anterior generará la siguiente salida:

```
Comparación de cadenas estándar
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Comparación de cadenas en orden natural
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

Para más información, vea la página de Martin Pool sobre Comparación de Cadenas en Orden Natural (http://naturalordersort.org/).

De forma similar a otras funciones de comparación de cadenas, esta devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Nótese que esta comparación es sensible a mayúsculas y minúsculas.

Vea también ereg(), strcasecmp(), substr(), stristr(), strcmp(), strncmp(), strnatcasecmp(), y strstr().

strncasecmp (PHP 4 >= 4.0.2)

Binary safe case-insensitive string comparison of the first n characters

int **strncasecmp** (string str1, string str2, int len) \linebreak

This function is similar to strcasecmp(), with the difference that you can specify the (upper limit of the) number of characters (*len*) from each string to be used in the comparison. If any of the strings is shorter than *len*, then the length of that string will be used for the comparison.

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

See also ereg(), strcasecmp(), strcmp(), substr(), stristr(), and strstr().

strncmp (PHP 4)

Comparación de los n primeros caracteres de cadenas, con seguridad binaria

int **strncmp** (string cad1, string cad2, int largo) \linebreak

Esta función es similar a strcmp(), con la diferencia que se puede especificar el (límite superior del) número de caractares (*largo*) de cada cadena que se usarán en la comparación. Si alguna de las cadenas es menor que el *largo*, se usará su longitud para la comparación.

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Nótese que esta comparación es sensible a mayúsculas y minúsculas.

Vea también ereg(), strcasecmp(), substr(), stristr(), strcmp(), y strstr().

strpos (PHP 3, PHP 4)

Encuentra la posición de la primera aparición de una cadena

int **strpos** (string pajar, string aguja [, int desplazamiento]) \linebreak

Devuelve la posición numérica de la primera aparición de la *aguja* en la cadena *pajar*. A diferencia de strrpos(), esta función puede tomar una cadena completa como *aguja* y se utilizará en su totalidad.

Si la *aguja* no es hayada, devuelve FALSE.

Nota: Es fácil confundir los valores de retorno para "caracter encontrado en la posición 0" y "caracter no encontrado". Aquí se indica cómo detectar la diferencia:

```
// en PHP 4.0b3 y posteriores:
$pos = strpos ($micadena, "b");
if ($pos === false) { // nota: tres signos igual
    // no encontrado ...
}

// en versiones anteriores a la 4.0b3:
$pos = strpos ($micadena, "b");
if (is_string ($pos) && !$pos) {
    // no encontrado ...
}
```

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un caracter.

El parámetro opcional *desplazamiento* le permite especificar a partir de qué caracter del *pajar* comenzar a buscar. La posición devuelta es aún relativa al comienzo de *pajar*.

Vea también strrpos(), strrchr(), substr(), stristr(), y strstr().

strrchr (PHP 3, PHP 4)

Encuentra la última aparición de un caracter en una cadena

string **strrchr** (string pajar, string aguja) \linebreak

Esta función devuelve la porción del *pajar* que comienza en la última aparición de la *aguja* y continúa hasta el final del *pajar*.

Devuelve FALSE si la *aguja* no es hallada.

Si la *aguja* contiene más de un caracter, sólo se usará el primero.

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un caracter.

Ejemplo 1. Ejemplo de strrchr()

```
// obtener el último directorio de $PATH
$dir = substr (strrchr ($PATH, ":"), 1);

// obtener todo tras el último salto de línea
$texto = "Line 1\nLine 2\nLine 3";
$apell = substr (strrchr ($texto, 10), 1 );
```

Vea también substr(), stristr(), y strstr().

strrev (PHP 3, PHP 4)

Invierte una cadena

string **strrev** (string cadena) \linebreak

Devuele la *cadena* invertida.

strrpos (PHP 3, PHP 4)

Encuentra la posición de la última aparición de un caracter en una cadena

int **strrpos** (string pajar, char aguja) \linebreak

Devuele la posición numérica de la última aparición de la *aguja* en el *pajar*. Nótese que la aguja en este caso sólo puede ser un caracter único. Si se pasa una cadena como aguja, sólo se utilizará el primer caracter de la misma.

Si la *aguja* no es hayada, devuelve FALSE.

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un caracter.

Vea también strpos(), strrchr(), substr(), stristr(), y strstr().

strspn (PHP 3>= 3.0.3, PHP 4)

Encuentra la longitud del segmento inicial que coincide con la máscara

int **strspn** (string cad1, string cad2) \linebreak

Devuelve la longitud del segmento inicial de *cad1* que consiste por entero en caracteres contenidos en *cad2*.

```
strspn ("42 es la respuesta. ¿Cuál es la pregunta ...?", "1234567890");
```

devolverá 2 como resultado.

Vea también strcspn().

strstr (PHP 3, PHP 4)

Encuentra la primera aparición de una cadena

string **strstr** (string pajar, string aguja) \linebreak

Devuelve todo el *pajar* desde la primera aparición de la *aguja* hasta el final.

Si la *aguja* no es hayada, devuelve FALSE.

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un caracter.

Nota: Nótese que esta función es sensible a mayúsculas y minúsculas. Para búsquedas no sensibles, utilice stristr().

Ejemplo 1. Ejemplo de strstr()

```
$email = 'sterling@designmultimedia.com';
$dominio = strstr ($email, '@');
print $dominio; // imprime @designmultimedia.com
```

Vea también stristr(), strrchr(), substr(), y ereg().

strtok (PHP 3, PHP 4)

Divide una cadena en elementos

string **strtok** (string arg1, string arg2) \linebreak

strtok() se usa para dividir en elementos una cadena. Es decir, que si tiene una cadena como "Esta es una cadena de ejemplo" podría dividirla en palabras individuales utilizando el espacio como divisor.

Ejemplo 1. Ejemplo de strtok()

```
$cadena = "Esta es una cadena de ejemplo";
$tok = strtok ($cadena," ");
while ($tok) {
    echo "Palabra=$tok<br>";
    $tok = strtok (" ");
}
```

Nótese que sólo la primera llamada a strtok utiliza el argumento cadena. Cada llamada subsiguiente necesita sólo el divisor a utilizar, puesto que ella guarda la posición actual en la cadena. Para comenzar de nuevo o para dividir otra cadena, simplemente llame a strtok con el argumento de cadena y se inicializará. Nótese que puede poner divisores múltiples como parámetro. La cadena será dividida cuando alguno de los caracteres del argumento sea hallado.

Además tenga cuidado si sus divisores valen "0", pues evalúa como `FALSE` en las expresiones condicionales.

Vea también split() y explode().

strtolower (PHP 3, PHP 4)

Pasa a minúsculas una cadena

string **strtolower** (string cad) \linebreak

Devuelve la *`cadena`* con todas sus letras en minúsculas.

Nótese que las letras son definidad por el locale actual. Esto quiere decir que, por ejemplo, en el locale por defecto ("C"), los caracteres como la Ñ no serán convertidos.

Ejemplo 1. Ejemplo de strtolower()

```
$cad = "María Tenía Un Corderito al que QUERÍA Mucho";
$cad = strtolower($cad);
print $cad; # Visualiza maría tenía un corderito al que quería mucho
```

Vea también strtoupper() y ucfirst().

strtoupper (PHP 3, PHP 4)

Pasa a mayúsculas una cadena

string **strtoupper** (string cadena) \linebreak

Devuelve la *cadena* con todas sus letras en mayúsculas.

Nótese que las letras son definidad por el locale actual. Esto quiere decir que, por ejemplo, en el locale por defecto ("C"), los caracteres como la ñ no serán convertidos.

Ejemplo 1. Ejemplo de strtoupper()

```
$cad = "María Tenía Un Corderito al que QUERÍA Mucho";
$cad = strtoupper ($cad);
print $cad; # Visuaiza MARíA TENíA UN CORDERITO AL QUE QUERÍA MUCHO
```

Vea también strtolower() and ucfirst().

strtr (PHP 3, PHP 4)

Traduce ciertos caracteres

string **strtr** (string cad, string desde, string hasta) \linebreak

Esta función trabaja sobre *cad*, traduciendo todas las apariciones de cada caracter en *desde* por el caracter correspondiente en *hasta* y devolviendo el resultado.

Si *desde* y *hasta* son de distinta longitud, los caracteres extra en la más larga son ignorados.

Ejemplo 1. Ejemplo de strtr()

```
$addr = strtr($addr, "äåö", "aao");
```

strtr() puede llamarso sólo con dos argumentos. Si se llama de esta manera, se comporta de otro modo: *desde* debe ser entonces una matriz que contenga pares cadena -> cadena que serán sustituídos en la

cadena fuente. **strtr()** siempre buscará la coincidencia más larga primero y *NO* intentará sustituir nada en lo que haya trabajado ya.

Ejemplos:

```
$trad = array ("hola" => "hey", "hey" => "hola");
echo strtr("hey a todos, dije hola", $trad) . "\n";
```

Mostrará: "hola a todos, dije hey",

Nota: Esta característica (2 argumentos) fue añadida en el PHP 4.0

Vea también ereg_replace().

substr_count (PHP 4)

Cuenta el número de apariciones de la subcadena

int **substr_count** (string pajar, string aguja) \linebreak

substr_count() devuelve el número de veces que la subcadena *aguja* se encuentra en la cadena *pajar*.

Ejemplo 1. Ejemplo de substr_count()

```
print substr_count("This is a test", "is"); // prints out 2
```

substr_replace (PHP 4)

Sustituye texto en una parte de una cadena

string **substr_replace** (string cadena, string sustituto, int comienzo [, int largo]) \linebreak

substr_replace() sustituye la parte de *cadena* delimitada por los parámetros *comienzo* y (opcionalmente) *largo* por la cadena dada en *sustituto*. Se devuelve el resultado.

Si *comienzo* es positivo, la sustitución comenzará en dicha posición dentro de la *cadena*.

Si *comienzo* es negativo, la sustitución comenzará en dicha posición pero contando desde el final de *cadena*.

Si se especifica el *largo* y es positivo, representa el largo de la porción de *cadena* a sustituir. Si es negativo, representa el número de caracteres desde el final de *cadena* en los que dejar de sustituir. Si no se especifica, valdrá por defecto strlen(*cadena*); es decir, que acabará la sustitución al final de *cadena*.

Ejemplo 1. Ejemplo de substr_replace()

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";

/* Estos dos ejemplos sustituyen toda $var por 'bob'. */
echo substr_replace ($var, 'bob', 0) . "<br>\n";
echo substr_replace ($var, 'bob', 0, strlen ($var)) . "<br>\n";

/* Inserta 'bob' justo al inicio de $var. */
echo substr_replace ($var, 'bob', 0, 0) . "<br>\n";

/* Los dos siguientes cambian 'MNRPQR' en $var por 'bob'. */
echo substr_replace ($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace ($var, 'bob', -7, -1) . "<br>\n";

/* Borrar 'MNRPQR' de $var. */
echo substr_replace ($var, '', 10, -1) . "<br>\n";
?>
```

Vea también str_replace() y substr().

Nota: substr_replace() fue añadida en el PHP 4.0.

substr (PHP 3, PHP 4)

Devuelve parte de una cadena

string **substr** (string cadena, int comienzo [, int largo]) \linebreak

substr devuelve la porción de *cadena* especificada por los parámetros *comienzo* y *largo*.

Si *comienzo* es positivo, la cadena devuelta comenzará en dicho caracter de *cadena*.

Ejemplos:

```
$resto = substr ("abcdef", 1);    // devuelve "bcdef"
$resto = substr ("abcdef", 1, 3); // devuelve "bcd"
```

Si *comienzo* es negativo, la cadena devuelta comenzará en dicha posición desde el final de *cadena*.

Ejemplos:

```
$resto = substr ("abcdef", -1);    // devuelve "f"
$resto = substr ("abcdef", -2);    // devuelve "ef"
$resto = substr ("abcdef", -3, 1); // devuelve "d"
```

Si se especifica *largo* y es positivo, la cadena devuelta terminará *largo* caracteres tras el *comienzo*. Si esto resulta en una cadena con longitud negativa (porque el comienzo está pasado el final de la cadena), la cadena devuelta contendrá únicamente el caracter que haya en *comienzo*.

Si se especifica *largo* y es negativo, la cadena devuelta terminará a *largo* caracteres desde el final de *cadena*. Si esto resulta en una cadena con longitud negativa, la cadena devuelta contendrá únicamente el caracter que haya en *comienzo*.

Examples:

```
$resto = substr ("abcdef", 1, -1); // devuelve "bcde"
```

Vea también strrchr() y ereg().

trim (PHP 3, PHP 4)

Elimina espacios del principio y final de una cadena

string **trim** (string cad) \linebreak

Esta función elimina los espacios en blanco del comienzo y del final de una cadena y devuelve el resultado. Los caracteres de espacio que elimina realmente son: "\n", "\r", "\t", "\v", "\0", y el espacio en sí.

Vea también chop() y ltrim().

ucfirst (PHP 3, PHP 4)

Pasar a mayúsculas el primer caracter de una cadena

string **ucfirst** (string cad) \linebreak

Pone en mayúsculas el primer carácter de *cad* si es alfabético.

Nótese que ’alfabético’ está determinado por la localidad actual. Por ejemplo, en la localidad por defecto "C", los caracteres como la a con diéresis (ä) no serán convertidos.

Ejemplo 1. Ejemplo de ucfirst()

```
$texto = 'susanita tiene un ratón, un ratón chiquitín.';
$texto = ucfirst ($texto); // $texto vale ahora: Susanita tiene un
// ratón, un ratón chiquitín.
```

Vea también strtoupper() y strtolower()

ucwords (PHP 3>= 3.0.3, PHP 4)

Pone en mayúsculas el primer caracter de cada palabra de una cadena

string **ucwords** (string cad) \linebreak

Pasa a mayúsculas la primera letra de cada palabra en *cad* si dicho caracter es alfabético.

Ejemplo 1. Ejemplo de ucwords()

```
$texto = "susanita tiene un ratón, un ratón chiquitín.";
$texto = ucwords($texto); // $texto vale ahora: Susanita Tiene Un
                          // Ratón, Un Ratón Chiquitín.
```

Vea también strtoupper(), strtolower() y ucfirst().

vprintf (PHP 4 >= 4.1.0)

Output a formatted string

void **vprintf** (string format, array args) \linebreak

Display array values as a formatted string according to *format* (which is described in the documentation for sprintf()).

Operates as printf() but accepts an array of arguments, rather than a variable number of arguments.

See also: printf(), sprintf(), vsprintf()

vsprintf (PHP 4 >= 4.1.0)

Return a formatted string

string **vsprintf** (string format, array args) \linebreak

Return array values as a formatted string according to *format* (which is described in the documentation for sprintf()).

Operates as sprintf() but accepts an array of arguments, rather than a variable number of arguments.

See also: sprintf(), **vsprintf()**, vprintf()

wordwrap (PHP 4 >= 4.0.2)

Corta una cadena en un número dado de caracteres usando un caracter de ruptura de cadenas.

string **wordwrap** (string cad [, int ancho [, string ruptura]]) \linebreak

Corta la cadena *cad* en la columna especificada por el parámetro (opcional) *ancho*. La línea se rompe utilizando el parámetro (opcional) *ruptura*.

wordwrap() automáticamente cortará en la columna 75 y usará '\n' (nueva línea) si no se especifican el *ancho* o la *ruptura*.

Ejemplo 1. Ejemplo de wordwrap()

```
$texto = "El veloz murciélago hindú comía feliz cardillo y kiwi.";
$textonuevo = wordwrap( $texto, 20 );

echo "$textonuevo\n";
```

Este ejemplo mostraría:

```
El veloz murciélago
hindú comía feliz cardillo y kiwi.
```

Vea también nl2br().

XCVIII. Funciones de Sybase

sybase_affected_rows (PHP 3>= 3.0.6, PHP 4)

obtiene el número de filas afectadas por la última consulta

int **sybase_affected_rows** ([int link_identifier]) \linebreak

Devuelve: El número de filas afectadas por la última consulta.

sybase_affected_rows() devuelve el número de filas afectadas por la última acción e tipo INSERT, UPDATE o DELETE en el servidor asociado al identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto.

Esta instrucción no es efectiva para sentencias de tipo SELECT, sólo en sentencias que modifican registros. Para obtener el número de filas afectadas por un SELECT, use sybase_num_rows().

Nota: Esta función sólo esta disponible usando el interface de la librería CT, y no con la librería DB.

sybase_close (PHP 3, PHP 4)

cierra una conexión Sybase

int **sybase_close** (int link_identifier) \linebreak

Devuelve: `TRUE` si lo consigue, `FALSE` ante un error

sybase_close() cierra el enlace a la base de datos Sybase asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto.

Note que esto no es necesario usualmente, ya que los enlaces no persistentes abiertos son cerrados automaticamente al final de la ejecución del script.

sybase_close() no cerrará enlaces persistentes generados por sybase_pconnect().

Vea también: sybase_connect(), sybase_pconnect().

sybase_connect (PHP 3, PHP 4)

abre una conexión con un servidor Sybase

int **sybase_connect** (string servername, string username, string password) \linebreak

Devuelve: Un identificador de enlace Sybase positivo, o `FALSE` ante un error.

sybase_connect() establece una conexión son un servidor Sybase. El parámetro servername tiene que ser un nombre de servidor válido que esta definido en el fichero 'interfaces'.

En el caso que se haga una segunda llamada a sybase_connect() con los mismos argumentos, no se establecerá un nuevo enlace, en vez de esto, se devolverá el identificador de enlace que ya está abierto.

El enlace al servidor será cerrado tan pronto como la ejecución del script finalice, a menos que sea cerrado antes llamando explicitamente a sybase_close().

Vea también sybase_pconnect(), sybase_close().

sybase_data_seek (PHP 3, PHP 4)

mueve el puntero interno de la fila

int **sybase_data_seek** (int result_identifier, int row_number) \linebreak

Devuelve: `TRUE` si lo hace, `FALSE` en caso de fallo

sybase_data_seek() mueve el puntero interno de la fila del resultado asociado con el identificador de resultado especificado hacia el número de fila introducido. La siguiente llamada a sybase_fetch_row() devolverá esa fila.

Vea también: **sybase_data_seek()**.

sybase_fetch_array (PHP 3, PHP 4)

carga una fila como un array

int **sybase_fetch_array** (int result) \linebreak

Returns: An array that corresponds to the fetched row, or `FALSE` if there are no more rows.

sybase_fetch_array() es la versión extendida de sybase_fetch_row(). Además de almacenar los datos en los indices numericos del array de resultados, también almacena los datos en indices asociativos, usando los nombres de campo como claves.

Una cosa importante a remarcar es que el uso de sybase_fetch_array() NO es significativamente más lento que el uso de sybase_fetch_row(), mientrás que proporciona un valor añadido significativo.

Para más detalles, vea también sybase_fetch_row()

sybase_fetch_field (PHP 3, PHP 4)

obtiene la información del campo

object **sybase_fetch_field** (int result, int field_offset) \linebreak

Devuelve un objeto conteniendo la información del campo

sybase_fetch_field() puede usarse para obtener información sobre los campos de una consulta determinada. Si no se especifica el offset del campo, el siguiente campo que aún no halla sido tomado por sybase_fetch_field() es el que se obtiene.

Las propiedades del objeto son:

- name - column name. si la columna es el resultado de una función, esta propiedad se establece a computed#N, donde #N es un número de serie.
- column_source - la tabla de la cual se ha cogido la columna
- max_length - máxima longitud de la columna
- numeric - 1 si la columna es numérica

Vea también sybase_field_seek()

sybase_fetch_object (PHP 3, PHP 4)

carga una fila como un objeto

int **sybase_fetch_object** (int result) \linebreak

Devuelve: Un objeto con las propiedades que corresponden a la fila tomada, o `FALSE` si no hay más filas.

sybase_fetch_object() es similar a sybase_fetch_array(), con una diferencia - se devuelve un objeto, en vez de un array. Indirectamente, esto significa que sólo se puede acceder a los datos por los nombres de campo, y no por sus offsets (los números son nombres de propiedades ilegales).

En el tema de velocidad, la función es idéntica a sybase_fetch_array(), y al menos tan rápida como sybase_fetch_row() (la diferencia es insignificante).

Vea también: sybase_fetch-array() y sybase_fetch-row().

sybase_fetch_row (PHP 3, PHP 4)

obtiene una fila como un array enumerado

array **sybase_fetch_row** (int result) \linebreak

Devuelve: Un array que corresponde a la fila obtenida, o `FALSE` si no hay más filas.

sybase_fetch_row() obtiene una fila de datos del resultado asociado con el identificador de resultado especificado. La fila se devuelve como un array. Cada columna del resultado es almacenada en un offset del arrat, comenzando en el offset 0.

Las siguientes llamadas a sybase_fetch_rows() devolverán la siguiente fila del conjunto de resultados, o `FALSE` si no hay más filas.

Vea también: sybase_fetch_array(), sybase_fetch_object(), sybase_data_seek(), **sybase_fetch_lengths()**, y sybase_result().

sybase_field_seek (PHP 3, PHP 4)

establece el offset de un campo

int **sybase_field_seek** (int result, int field_offset) \linebreak

Localiza el campo especificado por el offset. Si la siguiente llamada sybase_fetch_field() no incluye un offset se devuelve este campo.

Vea también: sybase_fetch_field().

sybase_free_result (PHP 3, PHP 4)

libera el resultado de la memoria

int **sybase_free_result** (int result) \linebreak

sybase_free_result() sólo se necesita usar en el caso de que este preocupado por el uso de demasiada memoria mientras se ejecuta su script. Todos los resultados en memoria son liberados cuando el script finaliza, puede llamar a **sybase_free_result()** con el identificador de resultado como argumento y la memoria asociada a ese resultado será liberada.

sybase_get_last_message (PHP 3, PHP 4)

Returns the last message from the server

string **sybase_get_last_message** (void) \linebreak

sybase_get_last_message() returns the last message reported by the server.

sybase_min_client_severity (PHP 3, PHP 4)

Sets minimum client severity

void **sybase_min_client_severity** (int severity) \linebreak

sybase_min_client_severity() sets the minimum client severity level.

Nota: This function is only available using the CT library interface to Sybase, and not the DB library.

See also: sybase_min_server_severity().

sybase_min_error_severity (PHP 3, PHP 4)

Sets minimum error severity

void **sybase_min_error_severity** (int severity) \linebreak

sybase_min_error_severity() sets the minimum error severity level.

See also: sybase_min_message_severity().

sybase_min_message_severity (PHP 3, PHP 4)

Sets minimum message severity

void **sybase_min_message_severity** (int severity) \linebreak

sybase_min_message_severity() sets the minimum message severity level.

See also: sybase_min_error_severity().

sybase_min_server_severity (PHP 3, PHP 4)

Sets minimum server severity

void **sybase_min_server_severity** (int severity) \linebreak

sybase_min_server_severity() sets the minimum server severity level.

Nota: This function is only available using the CT library interface to Sybase, and not the DB library.

See also: sybase_min_client_severity().

sybase_num_fields (PHP 3, PHP 4)

obtiene el número de campos de un resultado

int **sybase_num_fields** (int result) \linebreak

sybase_num_fields() devuelve el número de campos en un conjunto de resultados.

Vea también: **sybase_db_query()**, sybase_query(), sybase_fetch_field(), sybase_num_rows().

sybase_num_rows (PHP 3, PHP 4)

obtiene el número de filas de un resultado

int **sybase_num_rows** (string result) \linebreak

sybase_num_rows() devuelve el número de filas de un conjunto de resultados.

Vea también: **sybase_db_query()**, sybase_query() and, sybase_fetch_row().

sybase_pconnect (PHP 3, PHP 4)

abre una conexión con Sybase persistente

int **sybase_pconnect** (string servername, string username, string password) \linebreak

Devuelve: Un identificador de enlace persistente de Sybase positivo en caso de que pueda abrirlo, en caso de error devuelve `FALSE`

sybase_pconnect() actua de una forma muy parecida a sybase_connect() con dos grandes diferencias.

Primera, cuando se conecta, esta función primero tratará de encontrar un enlace (persistente) que ya este abierto con el mismo host, nombre de usuario y contraseña. Si encuentra uno, devolverá un identificador para él en vez de abrir una nueva conexión.

Segunda, la conexión al servidor SQL no se cerrará cuando finalice la ejecución del script. En vez de esto, el enlace permanecerá abierto para un futuro uso (sybase_close() no podrá cerrar enlaces establecidos con`sybase_pconnect()`).

Este tipo de enlaces son llamados 'persistentes'.

sybase_query (PHP 3, PHP 4)

envía una consulta a Sybase

int **sybase_query** (string query, int link_identifier) \linebreak

Devuelve: Un identificador de resultado Sybase positivo si va bien, o `FALSE` ante un error.

sybase_query() envía una consulta a la actual base de datos activa en el servidor que está asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto. Si no hay un enlace abierto, la función intentará establecer un enlace como si sybase_connect() fuese llamada, y lo usará.

Vea también: **sybase_db_query()**, sybase_select_db(), y sybase_connect().

sybase_result (PHP 3, PHP 4)

obtiene datos de un resultado

int **sybase_result** (int result, int i, mixed field) \linebreak

Devuelve: El contenido de la celda en la fila y el offset especificado de un conjunto de resultados de Sybase.

sybase_result() devuelve el contenido de una celda de un conjunto de resultados de Sybase. El parámetro field puede ser el offset del campo, o el nombre del campo, o el nombre de la tabla, un punto y el nombre del campo (nombre_tabla.nombre_campo). Si el nombre de la columna tiene un alias (’select foo as bar from...’), use el alias en vez del nombre de la columna.

Cuando trabaje con conjuntos de resultados grandes, debe considerar el uso de alguna de las funciones que cargan una fila entera (especificadas abajo). Ya que estas funciones devuelven el contenido de multiples celdas en una única llamada, son MUCHO más rápidas que sybase_result(). Además, note que especificar un offset númerico en el parámetro field es mucho más rápido que especificar un nombre de campo o nombre_tabla.nombre_campo.

Alternativas recomendadas para alto rendimiento: sybase_fetch_row(), sybase_fetch_array(), y sybase_fetch_object().

sybase_select_db (PHP 3, PHP 4)

seleccciona una base de datos Sybase

int **sybase_select_db** (string database_name, int link_identifier) \linebreak

Returns: `TRUE` on success, `FALSE` on error

sybase_select_db() establece como activa la base de datos en el servidor asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto. Si no hay un enlace abierto, la función intentará establecer un enlace como si sybase_connect() fuese llamada, y lo usará.

Cada llamada subsiguiente a sybase_query() será hecha en la base de datos activa.

Vea también: sybase_connect(), sybase_pconnect(), y sybase_query()

XCIX. Tokenizer functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

token_get_all (PHP 4 >= 4.2.0)

Split given source in tokens

array **token_get_all** (string source) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

token_name (PHP 4 >= 4.2.0)

Get the name of a given token

string **token_name** (int type) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

C. Funciones URL

base64_decode (PHP 3, PHP 4)

decodifica datos cifrados con MIME base64

string **base64_decode** (string datos_cifrados) \linebreak

base64_decode() decodifica *`datos_cifrados`* y devuelve los datos originales. Los datos devueltos pueden ser binarios.

Vea también: base64_encode(), RFC-2045 sección 6.8.

base64_encode (PHP 3, PHP 4)

Codifica datos en MIME base64

string **base64_encode** (string datos) \linebreak

base64_encode() devuelve *`datos`* cifrados en base64. Esta codificación está pensada para que los datos binarios sobrevivan al transporte a través de capas que no son de 8 bits, como por ejemplo los cuerpos de los mensajes de correo.

Los datos codificados con Base64 ocupan aproximadamente un 33% más de espacio que los datos originales.

Vea también: base64_decode(), chunk_split(), RFC-2045 sección 6.8.

parse_url (PHP 3, PHP 4)

Analiza una URL y devuelve sus componentes

array **parse_url** (string url) \linebreak

Esta función devuelve una matriz que apunta a alguno de los componentes de la URL que estén presentes. Esto incluye el "esquema", "host", "puerto", "usuario", "pass", "path", "consulta", y "fragmento".

rawurldecode (PHP 3, PHP 4)

Decode URL-encoded strings

string **rawurldecode** (string str) \linebreak

Returns a string in which the sequences with percent (`%`) signs followed by two hex digits have been replaced with literal characters. For example, the string

```
foo%20bar%40baz
```

decodes into

```
foo bar@baz
```

.

> **Nota: rawurldecode()** does not decode plus symbols ('+') into spaces. urldecode() does.

See also rawurlencode(), urldecode(), urlencode().

rawurlencode (PHP 3, PHP 4)

URL-encode according to RFC1738

string **rawurlencode** (string str) \linebreak

Returns a string in which all non-alphanumeric characters except

```
-_.
```

have been replaced with a percent (`%`) sign followed by two hex digits. This is the encoding described in RFC1738 for protecting literal characters from being interpreted as special URL delimiters, and for protecting URL's from being mangled by transmission media with character conversions (like some email systems). For example, if you want to include a password in an FTP URL:

Ejemplo 1. rawurlencode() example 1

```
echo '<a href="ftp://user:', rawurlencode('foo @+%/'),
     '@ftp.my.com/x.txt">';
```

Or, if you pass information in a PATH_INFO component of the URL:

Ejemplo 2. rawurlencode() example 2

```
echo '<a href="http://x.com/department_list_script/',
    rawurlencode('sales and marketing/Miami'), '">';
```

See also rawurldecode(), urldecode(), urlencode().

urldecode (PHP 3, PHP 4)

decodifica URL-cifradas en una cadena de texto

string **urldecode** (string cadena) \linebreak

Decodifica cualquier `%##` cifrado en la cadena dada. Se devuelve la cadena decodificada.

Ejemplo 1. Ejemplo urldecode()

```
$a = split ('&', $querystring);
$i = 0;
while ($i < count ($a)) {
  $b = split ('=', $a [$i]);
  echo 'El valor para el parámetro ', htmlspecialchars (urldecode ($b [0])),
       ' es ', htmlspecialchars (urldecode ($b [1])), "<BR>";
  $i++;
}
```

Vea también urlencode()

urlencode (PHP 3, PHP 4)

Codifica una URL en una cadena de texto

string **urlencode** (string cadena) \linebreak

Devuelve una cadena en la que todos los carácteres no alfanuméricos excepto `-_.` han sido reemplazados con un signo de porcentaje (`%`) seguido por dos dígitos hexadecimales y los espacios han sido codificados como signos positivos (`+`). Está codificado de la misma manera que los datos que se envian desde un formulario WWW, es decir de la misma forma que el tipo `application/x-www-form-urlencoded`. Esto difiere del cifrado RFC1738 (vea rawurlencode()) en el que por razones históricas, los espacios son codificados como signos positivos (+). Esta función es conveniente para codificar una cadena de texto que va a ser usada como parte de una consulta de una URL, y es una forma adecuada de pasar variables a la página siguiente:

Ejemplo 1. Ejemplo urlencode()

```
echo '<A HREF="mycgi?foo=', urlencode ($userinput), '">';
```

Vea también urldecode()

CI. Funciones sobre variables

doubleval (PHP 3, PHP 4)

Obtiene el valor double (decimal) de una variable.

double **doubleval** (mixed var) \linebreak

Devuelve el valor double (decimal en punto flotante) de *var*.

var puede ser cualquier tipo escalar. No se puede usar **doubleval()** sobre arrays u objetos.

Ver también intval(), strval(), settype() y Type juggling.

empty (unknown)

Determina si una variable está definida

int **empty** (mixed var) \linebreak

Devuelve FALSE si *var* está definida y tiene un valor no-vacío o distinto de cero; en otro caso devuelve TRUE.

Ver también isset() y unset().

floatval (PHP 4 >= 4.2.0)

Get float value of a variable

float **floatval** (mixed var) \linebreak

Returns the float value of *var*.

Var may be any scalar type. You cannot use **floatval()** on arrays or objects.

```
$var = '122.34343The';
$float_value_of_var = floatval ($var);
print $float_value_of_var; // prints 122.34343
```

See also intval(), strval(), settype() and Type juggling.

get_defined_vars (PHP 4 >= 4.0.4)

Returns an array of all defined variables

array **get_defined_vars** (void) \linebreak

This function returns an multidimensional array containing a list of all defined variables, be them environment, server or user-defined variables.

```
$b = array(1,1,2,3,5,8);

$arr = get_defined_vars();

// print $b
print_r($arr["b"]);

// print path to the PHP interpreter (if used as a CGI)
// e.g. /usr/local/bin/php
echo $arr["_"];

// print the command-line paramaters if any
print_r($arr["argv"]);

// print all the server vars
print_r($arr["_SERVER"]);

// print all the available keys for the arrays of variables
print_r(array_keys(get_defined_vars()));
```

See also get_defined_functions() and get_defined_constants().

get_resource_type (PHP 4 >= 4.0.2)

Returns the resource type

string **get_resource_type** (resource handle) \linebreak

This function returns a string representing the type of the resource passed to it. If the paramater is not a valid resource, it generates an error.

```
$c = mysql_connect();
echo get_resource_type($c)."\n";
// prints: mysql link

$fp = fopen("foo","w");
```

```
echo get_resource_type($fp)."\n";
// prints: file

$doc = new_xmldoc("1.0");
echo get_resource_type($doc->doc)."\n";
// prints: domxml document
```

gettype (PHP 3, PHP 4)

Obtiene el tipo de una variable.

string **gettype** (mixed var) \linebreak

Devuelve el tipo de la variable PHP *var*.

Los valores posibles de la cadena devuelta son:

- "integer"
- "double"
- "string"
- "array"
- "object"
- "unknown type"

Ver también settype().

import_request_variables (PHP 4 >= 4.1.0)

Import GET/POST/Cookie variables into the global scope

bool **import_request_variables** (string types [, string prefix]) \linebreak

Imports GET/POST/Cookie variables into the global scope. It is useful if you disabled register_globals, but would like to see some variables in the global scope.

Using the *types* parameter, you can specify which request variables to import. You can use 'G', 'P' and 'C' characters respectively for GET, POST and Cookie. These characters are not case sensitive, so you can also use any combination of 'g', 'p' and 'c'. POST includes the POST uploaded file information. Note that the order of the letters matters, as when using "gp", the POST variables will overwrite GET variables with the same name. Any other letters than GPC are discarded.

The *prefix* parameter is used as a variable name prefix, prepended before all variable's name imported into the global scope. So if you have a GET value named "userid", and provide a prefix "pref_", then you'll get a global variable named $pref_userid.

If you're interested in importing other variables into the global scope, such as SERVER, consider using extract().

> **Nota:** Although the *prefix* parameter is optional, you will get an E_NOTICE level error if you specify no prefix, or specify an empty string as a prefix. This is a possible security hazard. Notice level errors are not displayed using the default error reporting level.

```
// This will import GET and POST vars
// with an "rvar_" prefix
import_request_variables("gP", "rvar_");

print $rvar_foo;
```

See also $_REQUEST, register_globals, Predefined Variables, and extract().

intval (PHP 3, PHP 4)

Obtiene el valor entero de una variable.

int **intval** (mixed var [, int base]) \linebreak

Devuelve el valor entero de *var*, usando la base de conversión especificada (por defecto es base 10).

var puede ser cualquier tipo escalar. No se puede usar **intval()** sobre arrays u objetos.

Ver también doubleval(), strval(), settype() y Type juggling.

is_array (PHP 3, PHP 4)

Averigua si una variable es un array.

int **is_array** (mixed var) \linebreak

Devuelve TRUE si *var* es un array, y FALSE en otro caso.

Ver también is_double(), is_float(), is_int(), is_integer(), is_real(), is_string(), is_long(), y is_object().

is_bool (PHP 4)

Finds out whether a variable is a boolean

bool **is_bool** (mixed var) \linebreak

Returns `TRUE` if the *`var`* parameter is a boolean.

See also is_array(), is_float(), is_int(), is_integer(), is_string(), and is_object().

is_callable (PHP 4 >= 4.0.6)

Find out whether the argument is a valid callable construct

bool **is_callable** (mixed var [, bool syntax_only [, string callable_name]]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

is_double (PHP 3, PHP 4)

Averigua si una variable es un valor double (número decimal).

int **is_double** (mixed var) \linebreak

Devuelve `TRUE` si *`var`* es un double (número decimal), y `FALSE` en otro caso.

Ver también is_array(), is_float(), is_int(), is_integer(), is_real(), is_string(), is_long(), y is_object().

is_float (PHP 3, PHP 4)

Averigua si una variable es un flotante.

int **is_float** (mixed var) \linebreak

Esta función es un alias de is_double().

Ver también is_double(), is_real(), is_int(), is_integer(), is_string(), is_object(), is_array(), y is_long().

is_int (PHP 3, PHP 4)

Averigua si una variable es un valor entero.

int **is_int** (mixed var) \linebreak

Esta función es un alias de is_long().

Ver también is_double(), is_float(), is_integer(), is_string(), is_real(), is_object(), is_array(), y is_long().

is_integer (PHP 3, PHP 4)

Averigua si una variable es un valor entero.

int **is_integer** (mixed var) \linebreak

Esta función es un alias de is_long().

Ver también is_double(), is_float(), is_int(), is_string(), is_real(), is_object(), is_array(), y is_long().

is_long (PHP 3, PHP 4)

Averigua si una varible es un valor entero.

int **is_long** (mixed var) \linebreak

Devuelve `TRUE` si *`var`* es un entero (long), y `FALSE` en otro caso.

Ver también is_double(), is_float(), is_int(), is_real(), is_string(), is_object(), is_array(), y is_integer().

is_null (PHP 4 >= 4.0.4)

Finds whether a variable is `NULL`

bool **is_null** (mixed var) \linebreak

Returns `TRUE` if *`var`* is null, `FALSE` otherwise.

See the `NULL` type when a variable is considered to be `NULL` and when not.

See also `NULL`, is_bool(), is_numeric(), is_float(), is_int(), is_string(), is_object(), is_array(), is_integer(), and is_real()

is_numeric (PHP 4)

Finds whether a variable is a number or a numeric string

bool **is_numeric** (mixed var) \linebreak

Returns `TRUE` if *`var`* is a number or a numeric string, `FALSE` otherwise.

See also is_bool(), is_float(), is_int(), is_string(), is_object(), is_array(), and is_integer().

is_object (PHP 3, PHP 4)

Averigua si una varible es un objeto.

int **is_object** (mixed var) \linebreak

Devuelve `TRUE` si *`var`* es un objeto, y `FALSE` en otro caso.

Ver también is_long(), is_int(), is_integer(), is_float(), is_double(), is_real(), is_string(), y is_array().

is_real (PHP 3, PHP 4)

Averigua si una varible es un número real.

int **is_real** (mixed var) \linebreak

Esta función es un alias de is_double().

Ver también is_long(), is_int(), is_integer(), is_float(), is_double(), is_object(), is_string(), y is_array().

is_resource (PHP 4)

Finds whether a variable is a resource

bool **is_resource** (mixed var) \linebreak

is_resource() returns `TRUE` if the variable given by the *`var`* parameter is a resource, otherwise it returns `FALSE`.

See the documentation on the resource-type for more information.

is_scalar (PHP 4 >= 4.0.5)

Finds whether a variable is a scalar

bool **is_scalar** (mixed var) \linebreak

is_scalar() returns `TRUE` if the variable given by the *`var`* parameter is a scalar, otherwise it returns `FALSE`.

Scalar variables are those containing an integer, float, string or boolean. Types array, object and resource or not scalar.

```
function show_var($var) {
    if (is_scalar($var)) {
        echo $var;
    } else {
        var_dump($var);
    }
}
$pi = 3.1416;
$proteins = array("hemoglobin", "cytochrome c oxidase", "ferredoxin");

show_var($pi);
// prints: 3.1416

show_var($proteins)
// prints:
// array(3) {
//   [0]=>
//   string(10) "hemoglobin"
//   [1]=>
//   string(20) "cytochrome c oxidase"
//   [2]=>
//   string(10) "ferredoxin"
// }
```

Nota: is_scalar() does not consider resource type values to be scalar as resources are abstract datatypes which are currently based on integers. This implementation detail should not be relied upon, as it may change.

See also is_bool(), is_numeric(), is_float(), is_int(), is_real(), is_string(), is_object(), is_array(), and is_integer().

is_string (PHP 3, PHP 4)

Averigua si una varible es una cadena de caracteres (string).

int **is_string** (mixed var) \linebreak

Devuelve `TRUE` si *`var`* es una cadena, y `FALSE` en otro caso.

Ver también is_long(), is_int(), is_integer(), is_float(), is_double(), is_real(), is_object(), y is_array().

isset (unknown)

Determina si una varible está definida

int **isset** (mixed var) \linebreak

Devuelve `TRUE` si *`var`* existe; y `FALSE` en otro caso.

Si una variable ha sido destruida con unset(), ya no estará definida (no será **isset()**).

```
$a = "test";
echo isset($a); // true
unset($a);
echo isset($a); // false
```

Ver también empty() y unset().

print_r (PHP 4)

Prints human-readable information about a variable

void **print_r** (mixed expression) \linebreak

print_r() displays information about a variable in a way that's readable by humans. If given a string, integer or float, the value itself will be printed. If given an array, values will be presented in a format that shows keys and elements. Similar notation is used for objects.

Remember that **print_r()** will move the array pointer to the end. Use reset() to bring it back to beginning.

> **Sugerencia:** Como con todo lo que presenta un resultado directamente en el navegador, se pueden utilizar las funciones de control de salida para capturar el resultado de esta función y grabarlo - por ejemplo - in una cadena literal.

```
<pre>
<?php
    $a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x','y','z'));
    print_r ($a);
```

```
?>
</pre>
```

Which will output:

```
<pre>
Array
(
    [a] => apple
    [b] => banana
    [c] => Array
        (
            [0] => x
            [1] => y
            [2] => z
        )
)
</pre>
```

> **Nota:** Prior to PHP 4.0.4, **print_r()** will continue forever if given an array or object that contains a direct or indirect reference to itself. An example is `print_r($GLOBALS)` because `$GLOBALS` is itself a global variable that contains a reference to itself.

See also ob_start(), var_dump(), and var_export().

serialize (PHP 3>= 3.0.5, PHP 4)

Generates a storable representation of a value

string **serialize** (mixed value) \linebreak

serialize() returns a string containing a byte-stream representation of *value* that can be stored anywhere.

This is useful for storing or passing PHP values around without losing their type and structure.

To make the serialized string into a PHP value again, use unserialize(). **serialize()** handles all types, except the resource-type. You can even **serialize()** arrays that contain references to itself. References inside the array/object you are **serialize()**ing will also be stored.

Nota: In PHP 3, object properties will be serialized, but methods are lost. PHP 4 removes that limitation and restores both properties and methods. Please see the Serializing Objects section of Classes and Objects for more information.

Ejemplo 1. serialize() example

```
// $session_data contains a multi-dimensional array with session
// information for the current user.  We use serialize() to store
// it in a database at the end of the request.

$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn,
      "UPDATE sessions SET data = ? WHERE id = ?");
$sqldata = array (serialize($session_data), $PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata)) {
    $stmt = odbc_prepare($conn,
     "INSERT INTO sessions (id, data) VALUES(?, ?)");
    if (!odbc_execute($stmt, &$sqldata)) {
    /* Something went wrong.  Bitch, whine and moan. */
    }
}
```

See Also: unserialize().

settype (PHP 3, PHP 4)

Establece el tipo de una variable.

int **settype** (string var, string type) \linebreak

Establece el tipo de la varible *var* como *type*.

Los valores posibles para *type* son:

- "integer"
- "double"
- "string"
- "array"
- "object"

Devuelve `TRUE` si se lleva a cabo con éxito; en otro caso devuelve `FALSE`.

Ver también gettype().

strval (PHP 3, PHP 4)

Obtiene una cadena de caracteres a partir de una variable.

string **strval** (mixed var) \linebreak

Devuelve una cadena con el valor de *`var`*.

`var` puede ser cualquier tipo escalar. No se puede usar **strval()** sobre arrays u objetos.

Ver también doubleval(), intval(), settype() y Type juggling.

unserialize (PHP 3>= 3.0.5, PHP 4)

Creates a PHP value from a stored representation

mixed **unserialize** (string str) \linebreak

unserialize() takes a single serialized variable (see serialize()) and converts it back into a PHP value. The converted value is returned, and can be an integer, float, string, array or object.

> **Nota:** It's possible to set a callback-function which will be called, if an undefined class should be instanciated during unserializing. (to prevent getting an incomplete object "__PHP_Incomplete_Class".) Use your `php.ini`, ini_set() or .htaccess-file to define 'unserialize_callback_func'. Everytime an undefined class should be instanciated, it'll be called. To disable this feature just empty this setting.

Ejemplo 1. unserialize_callback_func example

```
$serialized_object='O:1:"a":1:{s:5:"value";s:3:"100";}';

ini_set('unserialize_callback_func','mycallback'); // set your callback_function

function mycallback($classname) {
    // just include a file containing your classdefinition
    // you get $classname to figure out which classdefinition is required
}
```

Nota: In PHP 3, methods are not preserved when unserializing a serialized object. PHP 4 removes that limitation and restores both properties and methods. Please see the Serializing Objects section of Classes and Objects or more information.

Ejemplo 2. unserialize() example

```
// Here, we use unserialize() to load session data to the
// $session_data array from the string selected from a database.
// This example complements the one described with serialize().

$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array ($PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata) || !odbc_fetch_into ($stmt, &$tmp)) {
    // if the execute or fetch fails, initialize to empty array
    $session_data = array();
} else {
    // we should now have the serialized data in $tmp[0].
    $session_data = unserialize ($tmp[0]);
    if (!is_array ($session_data)) {
        // something went wrong, initialize to empty array
        $session_data = array();
    }
}
```

See Also: serialize().

unset (unknown)

Destruye una variable dada

int **unset** (mixed var) \linebreak

unset() destruye la variable especificada y devuelve `TRUE`.

Ejemplo 1. Ejemplo de unset()

```
unset( $foo );
unset( $bar['quux'] );
```

Ver también isset() y empty().

var_dump (PHP 3>= 3.0.5, PHP 4)

Dumps information about a variable

void **var_dump** (mixed expression [, mixed expression [, ...]]) \linebreak

This function returns structured information about one or more expressions that includes its type and value. Arrays are explored recursively with values indented to show structure.

> **Sugerencia:** Como con todo lo que presenta un resultado directamente en el navegador, se pueden utilizar las funciones de control de salida para capturar el resultado de esta función y grabarlo - por ejemplo - in una cadena literal.

Compare **var_dump()** to print_r().

```
<pre>
<?php
$a = array (1, 2, array ("a", "b", "c"));
var_dump ($a);

/* output:
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}

*/

$b = 3.1;
$c = TRUE;
var_dump($b,$c);

/* output:
float(3.1)
```

```
bool(true)

*/
?>
</pre>
```

var_export (PHP 4 >= 4.2.0)

Outputs or returns a string representation of a variable

mixed **var_export** (mixed expression [, bool return]) \linebreak

This function returns structured information about the variable that is passed to this function. It is similar to var_dump() with the exception that the returned representation is valid PHP code.

You can also return the variable representation by using `TRUE` as second parameter to this function.

Compare **var_export()** to var_dump().

```
<pre>
<?php
$a = array (1, 2, array ("a", "b", "c"));
var_export ($a);

/* output:
array (
  0 => 1,
  1 => 2,
  2 =>
  array (
    0 => 'a',
    1 => 'b',
    2 => 'c',
  ),
)
*/

$b = 3.1;
$v = var_export($b, TRUE);
echo $v;

/* output:
3.1
*/
?>
</pre>
```

CII. vpopmail functions

vpopmail_add_alias_domain_ex (PHP 4 >= 4.0.5)

Add alias to an existing virtual domain

bool **vpopmail_add_alias_domain_ex** (string olddomain, string newdomain) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_add_alias_domain (PHP 4 >= 4.0.5)

Add an alias for a virtual domain

bool **vpopmail_add_alias_domain** (string domain, string aliasdomain) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_add_domain_ex (PHP 4 >= 4.0.5)

Add a new virtual domain

bool **vpopmail_add_domain_ex** (string domain, string passwd [, string quota [, string bounce [, bool apop]]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_add_domain (PHP 4 >= 4.0.5)

Add a new virtual domain

bool **vpopmail_add_domain** (string domain, string dir, int uid, int gid) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_add_user (PHP 4 >= 4.0.5)

Add a new user to the specified virtual domain

bool **vpopmail_add_user** (string user, string domain, string password [, string gecos [, bool apop]]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_alias_add (PHP 4 >= 4.1.0)

insert a virtual alias

bool **vpopmail_alias_add** (string user, string domain, string alias) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_alias_del_domain (PHP 4 >= 4.1.0)

deletes all virtual aliases of a domain

bool **vpopmail_alias_del_domain** (string domain) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_alias_del (PHP 4 >= 4.1.0)

deletes all virtual aliases of a user

bool **vpopmail_alias_del** (string user, string domain) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_alias_get_all (PHP 4 >= 4.1.0)

get all lines of an alias for a domain

array **vpopmail_alias_get_all** (string domain) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_alias_get (PHP 4 >= 4.1.0)

get all lines of an alias for a domain

array **vpopmail_alias_get** (string alias, string domain) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_auth_user (PHP 4 >= 4.0.5)

Attempt to validate a username/domain/password. Returns true/false

bool **vpopmail_auth_user** (string user, string domain, string password [, string apop]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_del_domain_ex (PHP 4 >= 4.0.5)

Delete a virtual domain

bool **vpopmail_del_domain_ex** (string domain) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_del_domain (PHP 4 >= 4.0.5)

Delete a virtual domain

bool **vpopmail_del_domain** (string domain) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_del_user (PHP 4 >= 4.0.5)

Delete a user from a virtual domain

bool **vpopmail_del_user** (string user, string domain) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_error (PHP 4 >= 4.0.5)

Get text message for last vpopmail error. Returns string

string **vpopmail_error** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_passwd (PHP 4 >= 4.0.5)

Change a virtual user’s password

bool **vpopmail_passwd** (string user, string domain, string password) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

vpopmail_set_user_quota (PHP 4 >= 4.0.5)

Sets a virtual user's quota

bool **vpopmail_set_user_quota** (string user, string domain, string quota) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

CIII. W32api functions

w32api_deftype (PHP 4 >= 4.2.0)

Defines a type for use with other w32api_functions

int **w32api_deftype** (string typename, string member1_type, string member1_name) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

w32api_init_dtype (PHP 4 >= 4.2.0)

Creates an instance to the data type typename and fills it with the values val1, val2, the function then returns a DYNAPARM which can be passed when invoking a function as a parameter

resource **w32api_init_dtype** (string typename, mixed val1, mixed val2) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

w32api_invoke_function (PHP 4 >= 4.2.0)

Invokes function funcname with the arguments passed after the function name

mixed **w32api_invoke_function** (string funcname) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

w32api_register_function (PHP 4 >= 4.2.0)

Registers function function_name from library with PHP

bool **w32api_register_function** (string libary, string function_name) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

w32api_set_call_method (PHP 4 >= 4.2.0)

Sets the calling method used

void **w32api_set_call_method** (int method) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

CIV. Funciones WDDX

Estas funciones permiten el uso de WDDX (http://www.openwddx.org/).

Debe saber que todas las funciones que serializan variables usan el primer elemento de un array para determinar si este ha de serializarse en forma de array o como estructura. Si el primer elemento esta indexado por una cadena, se serializa como estructura, y en caso contrario, como array.

Ejemplo 1. Serializacion de un valor simple

```
<?php
print wddx_serialize_value("Ejemplo de PHP a paquete WDDX", "Paquete PHP");
?>
```

Este ejemplo producira:

```
<wddxPacket version='0.9'><header comment='Paquete PHP'/><data>
<string>Ejemplo de PHP a paquete WDDX</string></data></wddxPacket>
```

Ejemplo 2. Uso de paquetes incrementales

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");

/* Suponga que $ciudades se ha obtenido de una base de datos */
$ciudades = array("Austin", "Novato", "Seattle");
wddx_add_vars($packet_id, "ciudades");

$packet = wddx_packet_end($packet_id);
print $packet;
?>
```

Este ejemplo producira:

```
<wddxPacket version='0.9'><header comment='PHP'/><data><struct>
<var name='pi'><number>3.1415926</number></var><var name='ciudades'>
<array length='3'><string>Austin</string><string>Novato</string>
<string>Seattle</string></array></var></struct></data></wddxPacket>
```

wddx_add_vars (PHP 3>= 3.0.7, PHP 4)

Finaliza un paquete WDDX con el identificador dado

wddx_add_vars (entero packet_id, varios-tipos name_var [, varios-tipos ...]) \linebreak

wddx_add_vars() se utiliza para serializar las variables dadas e incorporar el resultado al paquete especificado por *packet_id*. Las variables a serializar se especifican exactamente igual que en wddx_serialize_vars().

wddx_deserialize (PHP 3>= 3.0.7, PHP 4)

Des-serializa un paquete WDDX

varios-tipos **wddx_deserialize** (cadena packet) \linebreak

wddx_deserialized() toma una cadena *packet* y la desserializa. Devuelve el resultado que puede ser de tipo cadena, numerico o array. Las estructuras son desserializadas en forma de arrays asociativos.

wddx_packet_end (PHP 3>= 3.0.7, PHP 4)

Finaliza un paquete WDDX con el identificador dado

cadena **wddx_packet_end** (entero packet_id) \linebreak

wddx_packet_end() finaliza el paquete WDDX especificado por el *packet_id* y devuelve la cadena con el paquete.

wddx_packet_start (PHP 3>= 3.0.7, PHP 4)

Comienza un nuevo paquete WDDX con una estructura dentro

entero **wddx_packet_start** ([cadena comentario]) \linebreak

Utilice **wddx_packet_start()** para comenzar un nuevo paquete WDDX que permita la adicion sucesiva de variables. Recibe el parametro opcional *comentario* y devuelve un identificador de paquete para su uso en posteriores llamadas. Automaticamente define una estructura dentro del paquete para contener las variables.

wddx_serialize_value (PHP 3>= 3.0.7, PHP 4)

Serializa un valor simple en un paquete WDDX

cadena **wddx_serialize_value** (varios-tipos var [, cadena comentario]) \linebreak

wddx_serialize_value() se utiliza para crear un paquete WDDX desde un valor simple dado. Toma el valor contenido en *var*, y una cadena *comentario* opcional que aparecera en la cabecera del paquete, y devuelve el paquete WDDX.

wddx_serialize_vars (PHP 3>= 3.0.7, PHP 4)

Serializa variables en un paquete WDDX

cadena **wddx_serialize_vars** (varios-tipos nombre_var [, varios-tipos ...]) \linebreak

wddx_serialize_vars() se utiliza para crear un paquete WDDX con una estructura que contiene la representacion serializada de las variables pasadas como parametros.

wddx_serialize_vars() toma un numero variable de argumentos, cada uno de los cuales puede ser una cadena con el nombre de una variable o un array con nombres de variables, o de otro array, etc.

Ejemplo 1. wddx_serialize_vars example

```
<?php
$a = 1;
$b = 5.5;
$c = array("azul", "naranja", "violeta");
$d = "colores";

$clvars = array("c", "d");
print wddx_serialize_vars("a", "b", $clvars);
?>
```

El ejemplo anterior producira:

```
<wddxPacket version='0.9'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>azul</string><string>naranja</string><string>violeta</string></array></var>
<var name='d'><string>colores</string></var></struct></data></wddxPacket>
```

CV. Funciones de intérprete XML

Introducción

Acerca de XML

XML (eXtensible Markup Language) es un formato de información para el intercambio de documentos estructurado en la "Web" Es un estándar definido por el consorcio de la "World Wide Web" (W3C). Se puede encontrar información sobre XML y tecnologís relacionadas en http://www.w3.org/XML/.

Instalación

Esta extensión usa expat, que se puede encontrar en http://www.jclark.com/xml/. El Makefile que viene con expat no crea una biblioteca por defecto, se puede usar esta regla de make para eso:

```
libexpat.a: $(OBJS)
ar -rc $@ $(OBJS)
ranlib $@
```

Se puede conseguir un paquete RPM de expat en http://sourceforge.net/projects/expat/.

Nota que si se usa Apache-1.3.7 o posterior, ya tienes la biblioteca requerida expat. Simplemente, configura PHP usando `--with-xml` (sin ninguna ruta adicional) y usará automáticamente la biblioteca expat incluida en Apache.

En UNIX, ejecuta **configure** con la opción `--with-xml`. La biblioteca expat debería ser instalada en algún lugar donde el compilador pueda encontrarlo. Si se compila PHP como un módulo para Apache 1.3.9 o posterior, PHP automáticamente usará la biblioteca integrada expat de Apache. Puede necesitar establecer CPPFLAGS y LDFLAGS en su entorno antes de ejecutar "configure" si se ha instalado expat en algún lugar exótico.

Compila PHP. *¡Ta-tam!* Ya debería estar.

Sobre Esta Extensión

Esta extensión de PHP implementa soporte para expat de James Clarkin en PHP. Este conjunto de herramientas permite interpretar, pero no validar, documentos XML. Soporta tres codificaciones de caracteres fuente, también proporcionados por PHP: `US-ASCII`, `ISO-8859-1` y `UTF-8`. `UTF-16` no está soportado.

Esta extensión permite crear intérpretes de XML y definir entonces *gestores* para diferentes eventos

XML. Cada intérprete XML tiene también unos cuantos parámetros que se pueden ajustar.

Los gestores de eventos XML definidos son:

Tabla 1. Gestores de XML soportados

Función PHP para establecer gestor	Descripción del evento
xml_set_element_handler()	Los eventos de elemento ("element") se producen cuando el intérprete XML encuentra etiquetas de comienzo o fin. Hay gestores separados para etiquetas de comienzo y etiquetas de fin.
xml_set_character_data_handler()	La información de caracteres es, por definición, todo el contenido no "marcado" de los documentos XML, incluidos los espacios en blanco entre etiquetas. Nota que el intérprete XML no añade o elimina ningún espacio en blanco, depende de la aplicación (de ti) decidir si el espacio en blanco es significativo.
xml_set_processing_instruction_handler()	Los programadores de PHP deberían estar ya familiarizados con las instrucciones de procesado (PI). <?php ?> es una instrucción de procesado, donde *php* se denomina el "objetivo de procesado". El manejo de éstos es específico a cada aplicación, salvo que todos los objetivos PI que comienzan con "XML" están reservados.
xml_set_default_handler()	Todo lo que no va a otro gestor, va al gestor por defecto. Se tendrán en el gestor por defecto cosas como las declaraciones de tipos de documento y XML.
xml_set_unparsed_entity_decl_handler()	Este gestor se llamará para la declaración de una entidad no analizada (NDATA).
xml_set_notation_decl_handler()	Este gestor se llama para la declaración de una anotación.
xml_set_external_entity_ref_handler()	Este gestor se llama cuando el intérprete XML encuentra una referencia a una entidad general interpretada externa. Puede ser una referencia a un archivo o URL, por ejemplo. Ver el ejemplo de entidad externa para demostración.

Case Folding

Las funciones manejadoras de elementos pueden tomar sus nombres de elementos "*case-folded*". Case-folding se define en el estándar XML como "un proceso aplicado a una secuencia de caracteres, en el cual aquellos identificados como sin-mayúsculas son reemplazados por sus equivalentes en

mayúsculas". En otras palabras, cuando se trata de XML, case-folding simplemente significa poner en mayúsculas.

Por defecto, todos los nombres de elementos que se pasan a las funciones gestoras estan "pasados a mayúsculas". Esta conducta puede ser observada y controlada por el analizador XML con las funciones xml_parser_get_option() y xml_parser_set_option(), respectivamente.

Códigos de Error

Las siguientes constantes se definen para códigos de error XML (como los devuelve xml_parse()):

XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING

Codificación de caracteres

La extension XML de PHP soporta el conjunto de caracteres Unicode (http://www.unicode.org/) a través de diferentes *codificaciones de caracteres*. Hay dos tipos de codificaciones de caracteres, *coficación de fuente* y *codificación de destino*. La representación interna de PHP del documento está siempre codificada con `UTF-8`.

La codificación de fuente se hace cuando un documento XML es interpretado. Al crear un intérprete XML , se puede especificar una codificación de fuente (esta codificación no se puede cambiar má tarde durante el tiempo de vida del intérprete XML). Las codificaciones de fuente soportadas son `ISO-8859-1`, `US-ASCII` y `UTF-8`. Las dos primeras son codificaciones de byte-único, lo que significa que cada carácter se representa por un solo byte. `UTF-8` puede codificar caracteres compuestos por un número variable de bits (hasta 21) en de uno a cuatro bytes. La codificación fuente por defecto usada por PHP es `ISO-8859-1`.

La codificación de destino se hace cuando PHP pasa datos a las funciones gestoras XML. Cuando se crea un intérprete XML, la codificación de destino se crea igual a la codificación de fuente, pero se puede cambiar en cualquier momento. La codificación de destino afectará a la información de los caracteres así

como a los nombres de las etiquetas y a los objetivos de instrucciones de procesado.

Si el intérprete XML encuentra caracteres fuera del rango que su codificación de fuente es capaz de representar, devolverá un error.

Si PHP encuentra caracteres en el documento XML interpretado que no pueden ser representados en la codificación de destino elegida, los caracteres problema serán "degradados". Actualmente, esto significa que tales caracteres se reemplazan por un signo de interrogación.

Algunos Ejemplos

Aquí hay algunos ejemplos de archivos de comandos PHP que interpretan documentos XML.

Ejemplos de Estructuras de Elementos XML

Este primer ejemplo muestra la estructura del elemento inicio en un documento con indentación.

Ejemplo 1. Muestra la Estructura del Elemento XML

```
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs) {
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name\n";
    $depth[$parser]++;
}

function endElement($parser, $name) {
    global $depth;
    $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
```

```
xml_parser_free($xml_parser);
```

Ejemplo de Mapeo de Etiquetas XML

Ejemplo 2. Traduciendo XML a HTML

Este ejemplo transforma etiquetas de un documento XML directamente a etiquetas HTML. Los elementos no encontrados en el "array de traducción ("map array") son ignorados. Por supuesto, este ejemplo solamente funcionará con un tipo de documentos XML específico.

```
$file = "data.xml";
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);

function startElement($parser, $name, $attrs) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}

function endElement($parser, $name) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</$htmltag>";
    }
}

function characterData($parser, $data) {
    print $data;
}

$xml_parser = xml_parser_create();
// usa case-folding para que estemos seguros de encontrar la etiqueta
// en $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
```

```
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

Ejemplo de Entidad Externa XML

Este ejemplo resalta el código XML. Ilustra cómo usar un gestor de referencia de entidades extenas para incluir y analizar otros documentos, así como cuúntos PIs pueden ser procesados, y un modo de determinar "confianza" para PIs que contienen código.

Los documentos XML que se pueden usar en este ejemplo se encuentran bajo el ejemplo (`xmltest.xml` y `xmltest2.xml`.)

Ejemplo 3. Ejemplo de Entidades Externas

```
$file = "xmltest.xml";

function trustedFile($file) {
    // solamente confía en archivos locales que nos pertenezcan
    if (!eregi("^([a-z]+)://", $file)
        && fileowner($file) == getmyuid()) {
            return true;
    }
    return false;
}

function startElement($parser, $name, $attribs) {
    print "&lt;<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
            print " <font color=\"#009900\">$k</font>=\"<font
                   color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}

function endElement($parser, $name) {
    print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}

function characterData($parser, $data) {
    print "<b>$data</b>";
}
```

```
function PIHandler($parser, $target, $data) {
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
            // Si el documento analizado es "de confianza", diremos
            // que es seguro ejecutar código PHP en su interior.
            // Si no, en vez de ello mostrará el código.
            if (trustedFile($parser_file[$parser])) {
                eval($data);
            } else {
                printf("Untrusted PHP code: <i>%s</i>",
                        htmlspecialchars($data));
            }
            break;
    }
}

function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
                htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>',
                htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                   $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n", $openEntityNames,
                   $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                       xml_error_string(xml_get_error_code($parser)),
                       xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
    return false;
}

function new_xml_parser($file) {
    global $parser_file;
```

```
    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

    if (!($fp = @fopen($file, "r"))) {
        return false;
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("could not open XML input");
}

print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parse complete\n";
xml_parser_free($xml_parser);

?>
```

Ejemplo 4. xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
```

```
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <sect1 id="about">
  <title>About this Document</title>
  <para>
   <!-- this is a comment -->
   <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </sect1>
</chapter>
```

Este archivo se incluye desde `xmltest.xml`:

Ejemplo 5. xmltest2.xml

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
   <element attrib="value"/>
   &testEnt;
   <?php print "This is some more PHP code being executed."; ?>
</foo>
```

utf8_decode (PHP 3>= 3.0.6, PHP 4)

Convierte una cadena codificada UTF-8 a ISO-8859-1

string **utf8_decode** (string data) \linebreak

Esta función decodifica*data*, asume codificación `UTF-8` , a `ISO-8859-1`.

Mira utf8_encode() para una explicación de codificación UTF-8.

utf8_encode (PHP 3>= 3.0.6, PHP 4)

codifica una cadena ISO-8859-1 a UTF-8

string **utf8_encode** (string data) \linebreak

Esta función codifica la cadena *`data`* a `UTF-8`, y devuelve la versión codificada. `UTF-8` es un mecanismo estándar usado por Unicodepara codificar valores de *caracteres amplios* en un chorro de bytes. `UTF-8` es transparente a caracteres de ASCII plano, es auto-sincronizado (significa que es posible para un programa averiguar dónde comienzan los caracteres en el chorro de bytes) y se puede usar con funciones de comparación de cadenas normales para ordenar y otros fines. PHP codifica caracteres `UTF-8` en hasta cuatro bytes, como esto:

Tabla 1. Codificación UTF-8

bytes	bits	representación
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Cada *`b`* representa un bit que puede ser usado para almacenar datos de caracteres.

xml_error_string (PHP 3>= 3.0.6, PHP 4)

obtiene la cadena de error del analizador XML

string **xml_error_string** (int code) \linebreak

`code`

Un código de error de xml_get_error_code().

Devuelve una cadena con una descripción textual del código de error *code*, o FALSE si no se encontró descripción.

xml_get_current_byte_index (PHP 3>= 3.0.6, PHP 4)

obtiene el índice del byte actual para un analizador XML

int **xml_get_current_byte_index** (int parser) \linebreak

parser

Una referencia al analizador XML del que obtener el índice del byte.

Esta función devuelve FALSE si *parser* no referencia un analizador válido, o si no devuelve en qué índice de byte se encuentra el buffer de datos del analizador (empezando en 0).

xml_get_current_column_number (PHP 3>= 3.0.6, PHP 4)

Obtiene el número de columna actual para un analizador XML.

int **xml_get_current_column_number** (int parser) \linebreak

parser

Una referencia al analizador XML del que obtener el número de columna.

Esta función devuelve FALSE si *parser* no referencia un analizador válido, o si no devuelve en qué columna de la línea actual (como se obtiene de xml_get_current_line_number()) en la que se encuentra el analizador.

xml_get_current_line_number (PHP 3>= 3.0.6, PHP 4)

obtiene el número de línea actual de un analizador XML

int **xml_get_current_line_number** (int parser) \linebreak

parser

Una referencia al analizador XML del que obtener el número de línea.

Esta función devuelve `FALSE` si *`parser`* no referencia un analizador válido, o si no devuelve en qué línea se encuentra actualmente el buffer de datos del analizador.

xml_get_error_code (PHP 3>= 3.0.6, PHP 4)

obtiene el código de error del analizador XML

int **xml_get_error_code** (int parser) \linebreak

`parser`

Una referencia al analizador XML del que obtener el código de error.

Esta función devuelve `FALSE` si *`parser`* no referencia un analizador válido, o si no devuelve uno de los códigos de error listados en la sección de códigos de error.

xml_parse_into_struct (PHP 3>= 3.0.8, PHP 4)

Parse XML data into an array structure

int **xml_parse_into_struct** (resource parser, string data, array &values, array &index) \linebreak

This function parses an XML file into 2 parallel array structures, one (*`index`*) containing pointers to the location of the appropriate values in the *`values`* array. These last two parameters must be passed by reference.

Below is an example that illustrates the internal structure of the arrays being generated by the function. We use a simple `note` tag embeded inside a `para` tag, and then we parse this an print out the structures generated:

```
$simple = "<para><note>simple note</note></para>";
$p = xml_parser_create();
xml_parse_into_struct($p,$simple,$vals,$index);
xml_parser_free($p);
echo "Index array\n";
print_r($index);
echo "\nVals array\n";
print_r($vals);
```

When we run that code, the output will be:

```
Index array
Array
(
    [PARA] => Array
        (
            [0] => 0
            [1] => 2
        )

    [NOTE] => Array
        (
            [0] => 1
        )

)

Vals array
Array
(
    [0] => Array
        (
            [tag] => PARA
            [type] => open
            [level] => 1
        )

    [1] => Array
        (
            [tag] => NOTE
            [type] => complete
            [level] => 2
            [value] => simple note
        )

    [2] => Array
        (
            [tag] => PARA
            [type] => close
            [level] => 1
        )

)
```

Event-driven parsing (based on the expat library) can get complicated when you have an XML document that is complex. This function does not produce a DOM style object, but it generates structures amenable of being transversed in a tree fashion. Thus, we can create objects representing the data in the XML file easily. Let's consider the following XML file representing a small database of aminoacids information:

Ejemplo 1. moldb.xml - small database of molecular information

```
<?xml version="1.0"?>
<moldb>

    <molecule>
        <name>Alanine</name>
        <symbol>ala</symbol>
        <code>A</code>
        <type>hydrophobic</type>
    </molecule>

    <molecule>
        <name>Lysine</name>
        <symbol>lys</symbol>
        <code>K</code>
        <type>charged</type>
    </molecule>

</moldb>
```

And some code to parse the document and generate the appropriate objects:

Ejemplo 2. parsemoldb.php - parses moldb.xml into and array of molecular objects

```
<?php

class AminoAcid {
    var $name;  // aa name
    var $symbol;    // three letter symbol
    var $code;  // one letter code
    var $type;  // hydrophobic, charged or neutral

    function AminoAcid ($aa) {
        foreach ($aa as $k=>$v)
            $this->$k = $aa[$k];
    }
}

function readDatabase($filename) {
    // read the xml database of aminoacids
    $data = implode("",file($filename));
    $parser = xml_parser_create();
    xml_parser_set_option($parser,XML_OPTION_CASE_FOLDING,0);
    xml_parser_set_option($parser,XML_OPTION_SKIP_WHITE,1);
    xml_parse_into_struct($parser,$data,$values,$tags);
    xml_parser_free($parser);
```

```
    // loop through the structures
    foreach ($tags as $key=>$val) {
        if ($key == "molecule") {
            $molranges = $val;
            // each contiguous pair of array entries are the
            // lower and upper range for each molecule definition
            for ($i=0; $i < count($molranges); $i+=2) {
                    $offset = $molranges[$i] + 1;
                $len = $molranges[$i + 1] - $offset;
                $tdb[] = parseMol(array_slice($values, $offset, $len));
            }
        } else {
            continue;
        }
    }
    return $tdb;
}

function parseMol($mvalues) {
    for ($i=0; $i < count($mvalues); $i++)
        $mol[$mvalues[$i]["tag"]] = $mvalues[$i]["value"];
    return new AminoAcid($mol);
}

$db = readDatabase("moldb.xml");
echo "** Database of AminoAcid objects:\n";
print_r($db);

?>
```

After executing `parsemoldb.php`, the variable `$db` contains an array of `AminoAcid` objects, and the output of the script confirms that:

```
** Database of AminoAcid objects:
Array
(
    [0] => aminoacid Object
        (
            [name] => Alanine
            [symbol] => ala
            [code] => A
            [type] => hydrophobic
        )

    [1] => aminoacid Object
        (
            [name] => Lysine
            [symbol] => lys
            [code] => K
```

```
                    [type] => charged
                )

        )
```

xml_parse (PHP 3>= 3.0.6, PHP 4)

comienza a analizar un documento XML

int **xml_parse** (int parser, string data [, int isFinal]) \linebreak

parser

Una referencia al analizador XML que se va a utilizar.

data

Conjunto de información que se analizará. Un documento puede ser analizado por trozos llamando varias veces a **xml_parse()** con nueva información, siempre que se establezca el parámetro *isFinal* y sea TRUE cuando el último dato sea analizado.

isFinal (optional)

Si existe y es TRUE, *data* es el último pedazo de información envíado en este análisis.

Cuando el documento XML es analizado, se hacen llamadas a los gestores para los eventos configurados tantas veces como sea necesario, después de que esta función devuelva TRUE o FALSE.

Devuelve TRUE si el análisis se realiza con éxito, FALSE si no tiene éxito, o si *parser* no referencia a un analizador válido. Para análisis fallidos, se puede recuperar información de error con xml_get_error_code(), xml_error_string(), xml_get_current_line_number(), xml_get_current_column_number() y xml_get_current_byte_index().

xml_parser_create_ns (PHP 4 >= 4.0.5)

Create an XML parser

resource **xml_parser_create_ns** ([string encoding [, string sep]]) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xml_parser_create (PHP 3>= 3.0.6, PHP 4)

crea un analizador de XML

int **xml_parser_create** ([string encoding]) \linebreak

encoding (opcional)

Qué codificación de caracteres debería usar el analizador. Las siguientes codificación de caracteres están soportadas:
`ISO-8859-1` (por defecto)
`US-ASCII`
`UTF-8`

Esta función crea un analizador XML y devuelve un índice para usarlo con otras funciones XML. Devuelve `FALSE` en caso de fallo.

xml_parser_free (PHP 3>= 3.0.6, PHP 4)

Libera un analizador XML

string **xml_parser_free** (int parser) \linebreak

parser

Una referencia al analizador XML que se liberará.

Esta función devuelve `FALSE` si *parser* no referencia un analizador válido, o si no libera el analizador y devuelve `TRUE`.

xml_parser_get_option (PHP 3>= 3.0.6, PHP 4)

obtiene las opciones de un analizador XML

mixed **xml_parser_get_option** (int parser, int option) \linebreak

parser

Una referencia al analizador XML del que obtener opciones.

option

Qué opción recuperar. Ver xml_parser_set_option() para una lista de opciones.

Esta función devuelve FALSE si *parser* no referencia un analizador válido, o si la opción no pudo ser establecida. Si no, se devuelve la opción.

Mirar xml_parser_set_option() para la lista de opciones.

xml_parser_set_option (PHP 3>= 3.0.6, PHP 4)

establece las opciones de un analizador XML

int **xml_parser_set_option** (int parser, int option, mixed value) \linebreak

parser

Una referencia al analizador XML en el que establecer opciones.

option

Opción que se establecerá. Ver más abajo.

value

El nuevo valor de la opción.

Esta función devuelve FALSE si *parser* no referencia un analizador válido, o si la opción no pudo ser establecida. Si no, la opción se establece y devuelve TRUE.

Las opciones siguientes están disponibles:

Tabla 1. Opciones de analizador XML

Opción constante	Tipo de Datos	Descripción
XML_OPTION_CASE_FOLDING	integer	Controla si case-folding se habilita para este analizador XML. Habilitado por defecto.

Opción constante	Tipo de Datos	Descripción
XML_OPTION_TARGET_ENCODING	string	Establece qué codificación destino se usa en este analizador XML. Por defecto, esta puesta al mismo que la codificación fuente usada por xml_parser_create(). Las codificaciones de destino soportadas son `ISO-8859-1`, `US-ASCII` y `UTF-8`.

xml_set_character_data_handler (PHP 3>= 3.0.6, PHP 4)

Establece gestores de datos de caracteres

int **xml_set_character_data_handler** (int parser, string handler) \linebreak

Establece la función gestora de datos de caracteres para el analizador XML *`parser`*. *`handler`* es un string que contiene el nombre de la función que debe existir cuando xml_parse() es llamado por *`parser`*.

La función nombrada en *`handler`* debe aceptar dos parámetros: ***`handler`*** (int parser, string data) \linebreak

`parser`

El primer parámetro, *`parser`*, es una referencia al analizador XML que llama al gestor.

`data`

El segundo parámetro, *`data`*, contiene los datos caracteres como string.

Si una función gestora se establece como la cadena vacía, o `FALSE`, el gestor en cuestión se deshabilita.

Se devuelve `TRUE` si se estableció el gestor, `FALSE` si *`parser`* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_default_handler (PHP 3>= 3.0.6, PHP 4)

set up default handler

int **xml_set_default_handler** (int parser, string handler) \linebreak

Establece la función gestora por defecto para un analizador XML *`parser`*. *`handler`* es un string que contiene el nombre de la función que debe existir cuando xml_parse() es llamado por *`parser`*.

La función nombrada en *handler* debe aceptar dos parámetros: ***handler*** (int parser, string data) \linebreak

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

data

El segundo parámetro, *data*, contiene los caracteres de dato. Esto puede ser la declaración XML, la declaración de tipo de documento, entidades u otros datos para los cuales no existe otro gestor.

Si una función gestora se establece como la cadena vacía, o FALSE, el gestor en cuestión se deshabilita.

Se devuelve TRUE si se estableció el gestor, FALSE si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_element_handler (PHP 3>= 3.0.6, PHP 4)

establece gestores de los elementos principio y fin

int **xml_set_element_handler** (int parser, string startElementHandler, string endElementHandler) \linebreak

Establece las funciones de gestion de elementos para el analizador XML *parser*. *startElementHandler* y *endElementHandler* son strings que contienen los nombres de las funciones que deben existir cuando xml_parse() es llamado por *parser*.

La función denominada *startElementHandler* debe aceptar tres parámetros: ***startElementHandler*** (int parser, string name, string attribs) \linebreak

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

name

El segundo parámetro, *name*, contiene el nombre del elemento para el que se llama a este gestor. Si la propiedad de case-folding tiene efecto para este analizador, el nombre del elemento estará en mayúsculas.

attribs

El tercer parámetro, *attribs*, contiene un array asociativo con los atributos de los elementos (si hay). Las claves de este array son los nombres de los atributos, los valores son los valores de los atributos. Los nombres de los atributos están en mayúsculas (case-folded) con el mismo criterio que los nombres de los elementos. Los valores de los atributos *no* sufren las consecuencias de case-folding.

El orden original de los atributos se puede recuperar recorriendo *attribs* del modo usual, usando each(). La primera clave del array es el el primer atributo, y así sucesivamente.

La función llamada *endElementHandler* debe aceptar dos parámetros: ***endElementHandler*** (int parser, string name) \linebreak

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

name

El segundo parámetro, *name*, contiene el nombre del elemento para el que se llama a este gestor. Si la propiedad de case-folding tiene efecto para este analizador, el nombre del elemento estará en mayúsculas.

Si una función gestora se establece como la cadena vacía, o FALSE, el gestor en cuestión se deshabilita.

Se devuelve TRUE si se establecieron los gestores, FALSE si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_end_namespace_decl_handler (PHP 4 >= 4.0.5)

Set up character data handler

bool **xml_set_end_namespace_decl_handler** (resource pind, string hdl) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

Nota: En lugar de un nombre de función, se pueden proporcionar una matriz que contenga una referencia a un objeto o el nombre de un método.

xml_set_external_entity_ref_handler (PHP 3>= 3.0.6, PHP 4)

Establece gestores de referencia de entidades externas

int **xml_set_external_entity_ref_handler** (int parser, string handler) \linebreak

Establece la función gestora de declaraciones de notación para el analizador XML *parser*. *handler* es un string que contiene el nombre de una función que debe existir cuando xml_parse() es llamado por *parser*.

La función llamada por *handler* debe aceptar cinco parámetros, y debería devolver un valor entero. Si el valor devuelto desde el gestor (handler) es falso (lo cual ocurrirá si no se devuelve un valor), el analizador XML dejará de analizar y xml_get_error_code() devolverá XML_ERROR_EXTERNAL_ENTITY_HANDLING. int ***handler*** (int parser, string openEntityNames, string base, string systemId, string publicId) \linebreak

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

openEntityNames

El segundo parámetro, *openEntityNames*, es una lista, separada por espacios, de los nombres de las entidades que se abren para el análisis de esta entidad (incluido el nombre de la entidad referenciada).

base

Esta es la base para resolver el identificador de sistema (*systemid*) de la entidad externa. En la actualidad este parámetro es siempre la cadena vacía.

systemId

El cuarto parámetro, *systemId*, es el identificador del sistema tal como se especificó en la declaración de la entidad.

publicId

El quinto parámetro, *publicId*, es el identificador público como se especificó en la declaración de la entidad, o una cadena vacía si no se especificó ninguno; el espacio en blanco en el identificador público se habrá normalizado como se requiere en las especificaciones XML.

Si una función gestora se establece como la cadena vacía, o FALSE, el gestor en cuestión se deshabilita.

Se devuelve TRUE si se estableció el gestor, FALSE si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_notation_decl_handler (PHP 3>= 3.0.6, PHP 4)

Establece gestores de declaraciones de notación

int **xml_set_notation_decl_handler** (int parser, string handler) \linebreak

Establece las funciones gestoras de declaraciones de notación para el analizador XML *parser*. *handler* es un string que contiene el nombre de una función que debe existir cuando xml_parse() es llamado por *parser*.

Una declaración de notación es parte del DTD del documento y tiene el siguiente formato:

```
<!NOTATION name
{systemId | publicId}
>
```

Ver la sección 4.7 de las especificaciones XML 1.0 (http://www.w3.org/TR/1998/REC-xml-19980210#Notations) para la definición de declaraciones de notación.

La función llamada por *handler* debe aceptar cinco parámetros: ***handler*** (int parser, string notationName, string base, string systemId, string publicId) \linebreak

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

notationName

Este es el *nombre* de la notación, como se describió arriba en el formato de notación.

base

Esta es la base para resolver el identificador de sistema (*systemId*) de la declaración. En la actualidad este parámetro es siempre la cadena vacía.

systemId

Identificador de sistema de la declaración de notación externa.

publicId

Identificador público de la declaración de notación externa.

Si una función gestora se establece como la cadena vacía, o FALSE, el gestor en cuestión se deshabilita.

Se devuelve TRUE si se estableció el gestor, FALSE si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_object (PHP 4)

Usa un analizador XML dentro de un objecto

void **xml_set_object** (int parser, object &object) \linebreak

Esta función hace a *parser* utilizable dentro de *object*. Todas las funciones de callback establecidas por xml_set_element_handler() etc se asumen como métodos de *object*.

```
<?php
class xml  {
var $parser;
```

```
function xml() {
    $this->parser = xml_parser_create();
    xml_set_object($this->parser,&$this);
    xml_set_element_handler($this->parser,"tag_open","tag_close");
    xml_set_character_data_handler($this->parser,"cdata");
}

function parse($data) {
    xml_parse($this->parser,$data);
}

function tag_open($parser,$tag,$attributes) {
    var_dump($parser,$tag,$attributes);
}

function cdata($parser,$cdata) {
    var_dump($parser,$cdata);
}

function tag_close($parser,$tag) {
    var_dump($parser,$tag);
}

} // end of class xml

$xml_parser = new xml();
$xml_parser->parse("<A ID=\"hallo\">PHP</A>");
?>
```

Nota: xml_set_object() es gestionable a partir de PHP 4.0.

xml_set_processing_instruction_handler (PHP 3>= 3.0.6, PHP 4)

Establece el gestor de instrucciones de procesado (PI)

int **xml_set_processing_instruction_handler** (int parser, string handler) \linebreak

Establece ela función de gestión de instrucciones de procesado (PI) para el analizador XML *parser*. *handler* es un string que contiene el nombre de una función que debe existir cuando xml_parse() es llamada por *parser*.

Una instrucción de procedado tiene el siguiente formato:

```
<?
```

```
target
data?>
```

Puedes poner código PHP en esa etiqueta, pero ten en cuenta una limitación: en una PI XML, la etiqueta de fin de la PI (`?>`) no puede ser citada, por lo que esta secuencia de caracteres no debería aparecer en el código PHP que insertes con las PIs en documentos XML. Si lo hace, el resto del código PHP, así como la etiqueta de fin de PI "real", serán tratados como datos de caracteres.

La función nombrada en *`handler`* debe aceptar tres parámetros: ***`handler`*** (int parser, string target, string data) \linebreak

`parser`

El primer parámetro, *`parser`*, es una referencia al analizador XML que llama al gestor.

`target`

El segundo parámetro, *`target`*, contiene el objetivo PI.

`data`

El tercer parámetro, *`data`*, contiene los datos PI.

Si una función gestora se establece como la cadena vacía, o `FALSE`, el gestor en cuestión se deshabilita.

Se devuelve `TRUE` si se estableció el gestor, `FALSE` si *`parser`* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_start_namespace_decl_handler (PHP 4 >= 4.0.5)

Set up character data handler

bool **xml_set_start_namespace_decl_handler** (resource pind, string hdl) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

Nota: En lugar de un nombre de función, se pueden proporcionar una matriz que contenga una referencia a un objeto o el nombre de un método.

xml_set_unparsed_entity_decl_handler (PHP 3>= 3.0.6, PHP 4)

Establece un gestor de declaraciones de entidades no analizadas

int **xml_set_unparsed_entity_decl_handler** (int parser, string handler) \linebreak

Establece la función gestora de declaración de entidades no analizadas para el analizador XML *parser*. *handler* es una cadena que contiene el nombre de una función que debe existir cuando xml_parse() es llamada por *parser*.

Este gestor será llamado si el analizador XML encuentra una declaración de entidades externas con una declaración NDATA, como la siguiente:

```
<!ENTITY name {publicId | systemId}
        NDATA notationName>
```

Mira la sección 4.2.2 de las especificaciones XML 1.0 (http://www.w3.org/TR/1998/REC-xml-19980210#sec-external-ent) para la definición de entidades externas de notación declarada.

La función nombrada en *handler* debe aceptar seis parámetros: ***handler*** (int parser, string entityName, string base, string systemId, string publicId, string notationName) \linebreak

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

entityName

El nombre de la entidad que va a ser definida.

base

Esta es la base para resolver el identificador de sistema (*systemId*) de la entidad externa. Actualmente este parámetro siempre será una cadena vacía.

systemId

Identificador de Sistema para la entidad externa.

publicId

Identificador público para la entidad externa.

notationName

Nombre de la notación de esta entidad (ver xml_set_notation_decl_handler()).

Si una función gestora se establece como la cadena vacía, o FALSE, el gestor en cuestión se deshabilita.

Se devuelve TRUE si se estableció el gestor, FALSE si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

CVI. XMLRPC functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

xmlrpc_decode_request (PHP 4 >= 4.1.0)

Decodes XML into native PHP types

array **xmlrpc_decode_request** (string xml, string method [, string encoding]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_decode (PHP 4 >= 4.1.0)

Decodes XML into native PHP types

array **xmlrpc_decode** (string xml [, string encoding]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_encode_request (PHP 4 >= 4.1.0)

Generates XML for a method request

string **xmlrpc_encode_request** (string method, mixed params) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_encode (PHP 4 >= 4.1.0)

Generates XML for a PHP value

string **xmlrpc_encode** (mixed value) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_get_type (PHP 4 >= 4.1.0)

Gets xmlrpc type for a PHP value. Especially useful for base64 and datetime strings

string **xmlrpc_get_type** (mixed value) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_parse_method_descriptions (PHP 4 >= 4.1.0)

Decodes XML into a list of method descriptions

array **xmlrpc_parse_method_descriptions** (string xml) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_server_add_introspection_data (PHP 4 >= 4.1.0)

Adds introspection documentation

int **xmlrpc_server_add_introspection_data** (resource server, array desc) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_server_call_method (PHP 4 >= 4.1.0)

Parses XML requests and call methods

mixed **xmlrpc_server_call_method** (resource server, string xml, mixed user_data [, array output_options]) \linebreak

> **Aviso**
>
> Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_server_create (PHP 4 >= 4.1.0)

Creates an xmlrpc server

resource **xmlrpc_server_create** (void) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_server_destroy (PHP 4 >= 4.1.0)

Destroys server resources

void **xmlrpc_server_destroy** (resource server) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_server_register_introspection_callback (PHP 4 >= 4.1.0)

Register a PHP function to generate documentation

bool **xmlrpc_server_register_introspection_callback** (resource server, string function) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_server_register_method (PHP 4 >= 4.1.0)

Register a PHP function to resource method matching method_name

bool **xmlrpc_server_register_method** (resource server, string method_name, string function) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xmlrpc_set_type (PHP 4 >= 4.1.0)

Sets xmlrpc type, base64 or datetime, for a PHP string value

bool **xmlrpc_set_type** (string value, string type) \linebreak

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

CVII. XSLT functions

Aviso

Este módulo es *EXPERIMENTAL*. Esto significa que el comportamineto de estas funciones, nombre de funciones y en definitiva TODO lo documentado aqui, puede cambiar en una futura version de PHP SIN AVISO. Quedas avisado, y utilizar este módulo es tu responsabiliad.

Introduction

About XSLT and Sablotron

XSLT (Extensible Stylesheet Language (XSL) Transformations) is a language for transforming XML documents into other XML documents. It is a standard defined by The World Wide Web consortium (W3C). Information about XSLT and related technologies can be found at http://www.w3.org/TR/xslt.

Installation

This extension uses Sabloton and expat, which can both be found at http://www.gingerall.com/. Binaries are provided as well as source.

On UNIX, run **configure** with the `--with-sablot` and `--enable-sablot-errors-descriptive` options. The Sablotron library should be installed somewhere your compiler can find it.

About This Extension

This PHP extension implements support Sablotron from Ginger Alliance in PHP. This toolkit lets you transform XML documents into other documents, including new XML documents, but also into HTML or other target formats. It basically provides a standardized and portable template mechanism, separating content and design of a website.

xslt_create (PHP 4 >= 4.0.3)

Create a new XSL processor.

resource **xslt_create** (void) \linebreak

This function returns a handle for a new XSL processor. This handle is needed in all subsequent calls to XSL functions.

xslt_errno (PHP 4 >= 4.0.3)

Return the current error number

int **xslt_errno** ([int xh]) \linebreak

Return the current error number of the given XSL processor. If no handle is given, the last error number that occured anywhere is returned.

xslt_error (PHP 4 >= 4.0.3)

Return the current error string

mixed **xslt_error** ([int xh]) \linebreak

Return the current error string of the given XSL processor. If no handle is given, the last error string that occured anywhere is returned.

xslt_free (PHP 4 >= 4.0.3)

Free XSLT processor

void **xslt_free** (resource xh) \linebreak

Free the XSLT processor identified by the given handle.

xslt_output_process (unknown)

unknown

unknown **xslt_process** (unknown) \linebreak

This function lacks a prototype definition.

xslt_set_base (PHP 4 >= 4.0.5)

Set the base URI for all XSLT transformations

void **xslt_set_base** (resource xh, string uri) \linebreak

Sets the base URI for all XSLT transformations, the base URI is used with Xpath instructions to resolve document() and other commands which access external resources.

xslt_set_encoding (PHP 4 >= 4.0.5)

Set the encoding for the parsing of XML documents

void **xslt_set_encoding** (resource xh, string encoding) \linebreak

Set the output encoding for the XSLT transformations. When using the Sablotron backend this option is only available when you compile Sablotron with encoding support.

xslt_set_error_handler (PHP 4 >= 4.0.4)

Set an error handler for a XSLT processor

void **xslt_set_error_handler** (resource xh, mixed handler) \linebreak

Set an error handler function for the XSLT processor given by *xh*, this function will be called whenever an error occurs in the XSLT transformation (this function is also called for notices).

xslt_set_log (PHP 4 >= 4.0.6)

Set the log file to write log messages to

void **xslt_set_log** (resource xh, mixed log) \linebreak

xh

A reference to the XSLT parser.

log

This parameter is either a boolean value which toggles logging on and off, or a string containing the logfile in which log errors too.

This function allows you to set the file in which you want XSLT log messages to, XSLT log messages are different than error messages, in that log messages are not actually error messages but rather messages related to the state of the XSLT processor. They are useful for debugging XSLT, when something goes wrong.

By default logging is disabled, in order to enable logging you must first call **xslt_set_log()** with a boolean parameter which enables logging, then if you want to set the log file to debug to, you must then pass it a string containing the filename:

Ejemplo 1. Using the XSLT Logging features

```
<?php

$xh = xslt_create();
xslt_set_log($xh, true);
xslt_set_log($xh, getcwd() . 'myfile.log');

$result = xslt_process($xh, 'dog.xml', 'pets.xsl');
print($result);

xslt_free($xh);
?>
```

xslt_set_sax_handler (4.0.3 - 4.0.6 only)

Set SAX handlers for a XSLT processor

bool **xslt_set_sax_handler** (resource xh, handlers) \linebreak

Set SAX handlers on the ressource handle given by xh.

xslt_set_sax_handlers (PHP 4 >= 4.0.6)

Set the SAX handlers to be called when the XML document gets processed

void **xslt_set_sax_handlers** (resource processor, array handlers) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

xslt_set_scheme_handler (4.0.5 - 4.0.6 only)

Set Scheme handlers for a XSLT processor

void **xslt_set_scheme_handler** (resource xh, array handlers) \linebreak

Set Scheme handlers on the resource handle given by *xh*. Scheme handlers should be an array with the format (all elements are optional):

```
array(
[get_all] => get all handler,
[open] => open handler,
[get] => get handler,
[put] => put handler,
[close] => close handler
)
```

xslt_set_scheme_handlers (PHP 4 >= 4.0.6)

Set the scheme handlers for the XSLT processor

void **xslt_set_scheme_handlers** (resource processor, array handlers) \linebreak

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

CVIII. YAZ

The **yaz()** functions wrap the YAZ API. The home page of the project is http://www.indexdata.dk/yaz/. Information about the phpyaz module can be found at http://www.indexdata.dk/phpyaz/.

PHP/YAZ is much simpler to use than the C API for YAZ but less flexible. The intent is to make it easy to build basic client functions. It supports persistent stateless connections very similar to those offered by the various SQL APIs that are available for PHP. This means that sessions are stateless but shared amongst users, thus saving the connect and INIT steps in many cases.

Before compiling PHP with the PHP/YAZ module you'll need the YAZ toolkit. Build YAZ and install it. Build PHP with your favourite modules and add option --with-yaz. Your task is roughly the following:

```
gunzip -c yaz-1.6.tar.gz|tar xf -
gunzip -c php-4.0.X.tar.gz|tar xf -
cd yaz-1.6
./configure --prefix=/usr
make
make install
cd ../php-4.0.X
./configure --with-yaz=/usr/bin
make
make install
```

PHP/YAZ keeps track of connections with targets (Z-Associations). A positive integer represents the ID of a particular association.

The script below demonstrates the parallel searching feature of the API. When invoked it either prints a query form (if no arguments are supplied) or if there are arguments (term and one or more hosts) it searches the targets in array host.

Ejemplo 1. YAZ()

```
$num_hosts = count ($host);
if (empty($term) || count($host) == 0) {
    echo '<form method="get">
    <input type="checkbox"
    name="host[]" value="bagel.indexdata.dk/gils">
        GILS test
    <input type="checkbox"
    name="host[]" value="localhost:9999/Default">
        local test
    <input type="checkbox" checked="1"
    name="host[]" value="z3950.bell-labs.com/books">
        BELL Labs Library
    <br>
    RPN Query:
    <input type="text" size="30" name="term">
    <input type="submit" name="action" value="Search">
    ';
} else {
```

```
    echo 'You searced for ' . htmlspecialchars($term) . '<br>';
    for ($i = 0; $i > $num_hosts; $i++) {
        $id[] = yaz_connect($host[$i]);
        yaz_syntax($id[$i],"sutrs");
        yaz_search($id[$i],"rpn",$term);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr>' . $host[$i] . ":";
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
            echo "Error: $error";
        } else {
            $hits = yaz_hits($id[$i]);
            echo "Result Count $hits";
        }
        echo '<dl>';
        for ($p = 1; $p <= 10; $p++) {
            $rec = yaz_record($id[$i],$p,"string");
            if (empty($rec)) continue;
            echo "<dt><b>$p</b></dt><dd>";
            echo ereg_replace("\n", "<br>\n",$rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
```

yaz_addinfo (PHP 4 >= 4.0.1)

Returns additional error information

int **yaz_addinfo** (int id) \linebreak

Returns additional error message for target (last request). An empty string is returned if last operation was a success or if no additional information was provided by the target.

yaz_ccl_conf (PHP 4 >= 4.0.5)

Configure CCL parser

int **yaz_ccl_conf** (int id, array config) \linebreak

This function configures the CCL query parser for a target with definitions of access points (CCL qualifiers) and their mapping to RPN. To map a specific CCL query to RPN afterwards call the yaz_ccl_parse() function. Each index of the array *`config`* is the name of a CCL field and the corresponding value holds a string that specifies a mapping to RPN. The mapping is a sequence of attribute-type, attribute-value pairs. Attribute-type and attribute-value is separated by an equal sign (`=`). Each pair is separated by white space.

Ejemplo 1. CCL configuration

In the example below, the CCL parser is configured to support three CCL fields: `ti`, `au` and `isbn`. Each field is mapped to their BIB-1 equivalent. It is assumed that variable `$id` is a target ID.

```
$field["ti"] = "1=4";
$field["au"] = "1=1";
$field["isbn"] = "1=7";
yaz_ccl_conf($id,$field);
```

yaz_ccl_parse (PHP 4 >= 4.0.5)

Invoke CCL Parser

int **yaz_ccl_parse** (int id, string query, array & result) \linebreak

This function invokes the CCL parser. It converts a given CCL FIND query to an RPN query which may be passed to the yaz_search() function to perform a search. To define a set of valid CCL fields call yaz_ccl_conf() prior to this function. If the supplied *`query`* was successfully converted to RPN, this

function returns TRUE, and the index rpn of the supplied array *result* holds a valid RPN query. If the query could not be converted (because of invalid syntax, unknown field, etc.) this function returns FALSE and three indexes are set in the resulting array to indicate the cause of failure: errorcodeCCL error code (integer), errorstringCCL error string, and errorposapproximate position in query of failure (integer is character position).

yaz_close (PHP 4 >= 4.0.1)

Closes a YAZ connection

int **yaz_close** (int id) \linebreak

Closes a connection to a target. The application can no longer refer to the target with the given id.

yaz_connect (PHP 4 >= 4.0.1)

Returns a positive association ID on success; zero on failure

int **yaz_connect** (string zurl) \linebreak

yaz_connect() prepares for a connection to a Z39.50 target. The zurl argument takes the form host[:port][/database]. If port is omitted 210 is used. If database is omitted Default is used. This function is non-blocking and doesn't attempt to establish a socket - it merely prepares a connect to be performed later when yaz_wait() is called.

yaz_database (PHP 4 >= 4.0.6)

Specifies the databases within a session

int **yaz_database** (int id, string databases) \linebreak

This function specifies one or more databases to be used in search, retrieval, etc. - overriding databases specified in call to yaz_connect(). Multiple databases are separated by a plus sign +.

This function allows you to use different sets of databases within a session.

Returns TRUE on success; FALSE on error.

yaz_element (PHP 4 >= 4.0.1)

Specifies Element-Set Name for retrieval

int **yaz_element** (int id, string elementset) \linebreak

This function is used in conjunction with yaz_search() and yaz_present() to specify the element set name for records to be retrieved. Most servers support `F` (full) and `B` (brief).

Returns `TRUE` on success; `FALSE` on error.

yaz_errno (PHP 4 >= 4.0.1)

Returns error number

int **yaz_errno** (int id) \linebreak

Returns error for target (last request). A positive value is returned if the target returned a diagnostic code; a value of zero is returned if no errors occurred (success); negative value is returned for other errors targets didn't indicate the error in question.

yaz_errno() should be called after network activity for each target - (after yaz_wait() returns) to determine the success or failure of the last operation (e.g. search).

yaz_error (PHP 4 >= 4.0.1)

Returns error description

int **yaz_error** (int id) \linebreak

Returns error message for target (last request). An empty string is returned if last operation was a success.

yaz_error() returns a english message corresponding to the last error number as returned by yaz_errno().

yaz_hits (PHP 4 >= 4.0.1)

Returns number of hits for last search

int **yaz_hits** (int id) \linebreak

yaz_hits() returns number of hits for last search.

yaz_itemorder (PHP 4 >= 4.0.5)

Prepares for Z39.50 Item Order with an ILL-Request package

int **yaz_itemorder** (array args) \linebreak

This function prepares for an Extended Services request using the Profile for the Use of Z39.50 Item Order Extended Service to Transport ILL (Profile/1). See this (http://www.nlc-bnc.ca/iso/ill/stanprf.htm) and the specification (http://www.nlc-bnc.ca/iso/ill/document/standard/z-ill-1a.pdf). The args parameter must be a hash array with information about the Item Order request to be sent. The key of the hash is the name of the corresponding ASN.1 tag path. For example, the ISBN below the Item-ID has the key item-id,ISBN.

The ILL-Request parameters are:

protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-person
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-institution
transaction-id,transaction-group-qualifier
transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier
service-date-time,this,date
service-date-time,this,time
service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution
responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-institution
delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city
delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
billing-address,postal-address,name-of-person-or-institution,name-of-institution
billing-address,postal-address,extended-postal-delivery-address
billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city
billing-address,postal-address,region
billing-address,postal-address,country

billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess
ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED
requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service
search-type,need-before-date
search-type,expiry-date
search-type,expiry-flag
place-on-hold
client-id,client-name
client-id,client-status
client-id,client-identifier
item-id,item-type
item-id,call-number
item-id,author
item-id,title
item-id,sub-title
item-id,sponsoring-body
item-id,place-of-publication
item-id,publisher
item-id,series-title-number
item-id,volume-issue
item-id,edition
item-id,publication-date
item-id,publication-date-of-component
item-id,author-of-article
item-id,title-of-article
item-id,pagination
item-id,ISBN
item-id,ISSN
item-id,additional-no-letters
item-id,verification-reference-source
copyright-complicance
retry-flag
forward-flag
requester-note
forward-note

There are also a few parameters that are part of the Extended Services Request package and the ItemOrder package:

package-name
user-id

contact-name
contact-phone
contact-email
itemorder-item

yaz_present (PHP 4 >= 4.0.5)

Prepares for retrieval (Z39.50 present).

int **yaz_present** (void) \linebreak

This function prepares for retrieval of records after a successful search. The yaz_range() should be called prior to this function to specify the range of records to be retrieved.

yaz_range (PHP 4 >= 4.0.1)

Specifies the maximum number of records to retrieve

int **yaz_range** (int id, int start, int number) \linebreak

This function is used in conjunction with yaz_search() to specify the maximum number of records to retrieve (number) and the first record position (start). If this function is not invoked (only yaz_search()) start is set to 1 and number is set to 10.

Returns `TRUE` on success; `FALSE` on error.

yaz_record (PHP 4 >= 4.0.1)

Returns a record

int **yaz_record** (int id, int pos, string type) \linebreak

Returns record at position or empty string if no record exists at given position.

The **yaz_record()** function inspects a record in the current result set at the position specified. If no database record exists at the given position an empty string is returned. The argument, type, specifies the form of the returned record. If type is "string" the record is returned in a string representation suitable for printing (for XML and SUTRS). If type is "array" the record is returned as an array representation (for structured records).

yaz_scan_result (PHP 4 >= 4.0.5)

Returns Scan Response result

array **yaz_scan_result** (int id [, array & result]) \linebreak

Given a target ID this function returns and array with terms as received from the target in the last Scan Response. This function returns an array (0..n-1) where n is the number of terms returned. Each value is a pair where first item is term, second item is result-count. If the *result* is given it will be modified to hold additional information taken from the Scan Response: number (number of entries returned), stepsize (Step-size), position (position of term), status (Scan Status).

yaz_scan (PHP 4 >= 4.0.5)

Prepares for a scan

int **yaz_scan** (int id, string type, string startterm [, array flags]) \linebreak

This function prepares for a Z39.50 Scan Request. Argument *id* specifies target ID. Starting term point for the scan is given by *startterm*. The form in which is the starting term is specified is given by *type*. Currently type rpn is supported. The optional *flags* specifies additional information to control the behaviour of the scan request. Three indexes are currently read from the flags: number (number of terms requested), position (preferred position of term) and stepSize (preferred step size). To actually tranfer the Scan Request to the target and receive the Scan Response, yaz_wait() must be called. Upon completion of yaz_wait() call yaz_error() and yaz_scan_result() to handle the response.

The syntax of *startterm* is similar to the RPN query as described in yaz_search(). The startterm consists of zero or more @attr-operator specifications, then followed by exactly one token.

Ejemplo 1. PHP function that scans titles

```
function scan_titles($id, $starterm) {
  yaz_scan($id,"rpn", "@attr 1=4 " . $starterm);
  yaz_wait();
  $errno = yaz_errno($id);
  if ($errno == 0) {
    $ar = yaz_scan_result($id,&$options);
    echo 'Scan ok; ';
    $ar = yaz_scan_result($id, &$options);
    while(list($key,$val)=each($options)) {
       echo "$key = $val  ";
    }
    echo '<br><table><tr><td>';
    while(list($key,list($k, $term, $tcount))=each($ar)) {
      if (empty($k)) continue;
      echo "<tr><td>$term</td><td>";
      echo $tcount;
      echo "</td></tr>";
```

```
    }
    echo '</table>';
  } else {
    echo "Scan failed. Error: " . yaz_error($id) . "<br>";
  }
}
```

yaz_search (PHP 4 >= 4.0.1)

Prepares for a search

int **yaz_search** (int id, string type, string query) \linebreak

yaz_search() prepares for a search on the target with given id. The type represents the query type - only "rpn" is supported now in which case the third argument is a prefix notation query as used by YAZ. Like yaz_connect() this function is non-blocking and only prepares for a search to be executed later when yaz_wait() is called.

yaz_sort (PHP 4 >= 4.1.0)

Sets sorting criteria

int **yaz_sort** (int id, string criteria) \linebreak

This function sets sorting criteria and enables Z39.50 Sort. Use this function together with yaz_search() or yaz_present(). Using this function alone doesn't have any effect. If used in conjunction with yaz_search() a Z39.50 Sort will be sent after a search response has been received and before any records are retrieved with Z39.50 Present. The *criteria* takes the form

field1 flags1 field2 flags2 ...

where field1 specifies primary attributes for sort, field2 seconds, etc.. The field specifies either numerical attribute combinations consisting of type=value pairs separated by comma (e.g. `1=4,2=1`) ; or the field may specify a plain string criteria (e.g. `title`. The flags is a sequnce of the following characters which may not be separated by any white space.

Sort Flags

`a`

Sort ascending

`d`

Sort descending

`i`

Case insensitive sorting

`s`

Case sensitive sorting

Ejemplo 1. Sort Criterias

To sort on Bib1 attribute title, case insensitive, and ascending you'd use the following sort criteria:

```
1=4 ia
```

If the secondary sorting criteria should be author, case sensitive and ascending you'd use:

```
1=4 ia 1=1003 sa
```

yaz_syntax (PHP 4 >= 4.0.1)

Specifies the preferred record syntax for retrieval

int **yaz_syntax** (int id, string syntax) \linebreak

This function is used in conjunction with yaz_search() to specify the preferred record syntax for retrieval.

yaz_wait (PHP 4 >= 4.0.1)

Executes queries

int **yaz_wait** (int id, string syntax) \linebreak

This function carries out networked (blocked) activity for outstanding requests which have been prepared by the functions yaz_connect(), yaz_search(). **yaz_wait()** returns when all targets have either completed all requests or otherwise completed (in case of errors).

CIX. NIS funciona

NIS (anteriormente llamado Paginas Amarillas) permite la administracion de red de los archivos de administracion importantes (e.g.El archivo de contraseñas). Para mas informacion dirigirse a las paginas de ayuda de NIS y a la direccion. Introduccion a YP/NIS (http://www.desy.de/~sieversm/ypdoku/ypdoku/ypdoku.html) Hay tambien un libro llamado gestionando NFS Y NIS (http://www.oreilly.com/catalog/nfs/noframes.html) por Hal Stern.

Para obtener estas funciones de trabajo, usted tiene que configure PHP con `-- con- yp`.

yp_all (PHP 4 >= 4.0.6)

Traverse the map and call a function on each entry

void **yp_all** (string domain, string map, string callback) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

yp_cat (PHP 4 >= 4.0.6)

Return an array containing the entire map

array **yp_cat** (string domain, string map) \linebreak

> **Aviso**
>
> Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parametros.

yp_err_string (PHP 4 >= 4.0.6)

devuelve el mensaje de error asociado con la operacion previa.Util que indica el problema exacto.

cadena **yp_err_string** (void) \linebreak

yp_err_string() Retorna el mensaje de error asociado con la operacion previa.Util para indicar que salio mal exactamente.

Ejemplo 1. Ejemplo para errores de NIS

Vea tambien: yp errno

yp_errno (PHP 4 >= 4.0.6)

Retorna el codigo de error de la operacion previa.

int **yp_errno** (void) \linebreak

yp_errno() retorna el codigo de error de la operacion previa.

Los errores posibles son:

1 args para funcionar son malos
2 fallo de RPC- dominio ha sido unbound
3 no puede unir a servidor en este dominio
4 ningun tal mapa en dominio de servidor
5 ninguna tal llave en
6 interno yp error de cliente o servidor
7 fallo de asignacion de recurso
8 ningunos más registros en base de datos de mapa
9 no puede comunicar wiTh portmapper
10 no puede comunicar con ypbind
11 no puede comunicar con ypserv
12 nombre de dominio local no conjunto
13 yp base de datos es malo
14 yp La version mismatch
15 violacion de acceso
16 base de datos ocupar

Ver tambien: yp_err_string

yp_first (unknown)

devuelve la primera clave emparejada con el nombrado mapa.

```
string[]  yp_first  (cadena dominio, cadena mapa)
```

yp_first(nombre de la funcion)() Retorna la primera clave de valor pareada del mapa nombrado en el dominio, de otra manera FALSO.

Ejemplo 1. Ejemplo para el primer NIS

Ver tambien: yp_get_default_domain yp_errno y yp_err_string

yp_get_default_domain (PHP 3>= 3.0.7, PHP 4)

Trae el valor por omision de dominios de maquina NIS.

int **yp get_default_domain** (void) \linebreak

yp_ get_default_domain() Retorna el valor por omision del dominio del nodo o FALSO. Puede ser usado el parametro de dominio para sucesivas llamadas a NIS.

Un dominio de NIS puede ser descrito en un grupo de mapas NIS. Cada host necesita buscar uniones de informacion en un mismo dominio. Acudir a los documentos mencionados en el comienzo para mas informacion.

Ejemplo 1. Ejemplo para el dominio por omision

Ver tambien: yp_ errno (nombre de la funcion) y yp_err_string (nombre de la funcion)

yp_master (PHP 3>= 3.0.7, PHP 4)

Returns the machine name of the master NIS server for a map

string **yp_master** (string domain, string map) \linebreak

yp_master() returns the machine name of the master NIS server for a map.

Ejemplo 1. Example for the NIS master

```
<?php
$number = yp_master ($domain, $mapname);
echo "Master for this map is: " . $master;
?>
```

See also yp-get-default-domain().

yp_match (PHP 3>= 3.0.7, PHP 4)

Retorna la linea compañera (pareja).

cadena **yp match** (cadena dominio, cadena mapa, cadena teclea) \linebreak

yp_match(nombre de la funcion)() Retorna el valor asociado con la llave pasada fuera del mapa especificado o FALSO. esta llave tiene que ser exacta.

Ejemplo 1. Ejemplo para NIS parejo

En este caso esto puede ser: Joe:##joe:11111:100:joe usuario:/hogar/j/joe: User:/usr/local/bin/bash

Ver tambien: yp_get_default_domain yp_errno y yp_err_string

yp_next (PHP 3>= 3.0.7, PHP 4)

Devuelve la siguiente clave tecleada en el nombre de mapa

string[] **yp_next** (cadena dominio, cadena mapa, cadena teclea) \linebreak

yp_next() devuelve el siguiente par de valor tecleado en el mapa de nombres despues de la clave especificada o FALSO.

Ejemplo 1. Ejemplo para NIS siguiente

Ver tambien: yp_get_default_domain yp_errno y yp_err_string

yp_order (PHP 3>= 3.0.7, PHP 4)

Returns the order number for a map

int **yp_order** (string domain, string map) \linebreak

yp_order() returns the order number for a map or `FALSE`.

Ejemplo 1. Example for the NIS order

```
<?php
    $number = yp_order($domain,$mapname);
    echo "Order number for this map is: " . $number;
?>
```

See also yp-get-default-domain().

CX. Zip File Functions (Read Only Access)

This module uses the functions of the ZZIPlib (http://zziplib.sourceforge.net/) library by Guido Draheim to transparently read ZIP compressed archives and the files inside them.

Please note that ZZIPlib only provides a subset of functions provided in a full implementation of the ZIP compression algorithm and can only read ZIP file archives. A normal ZIP utility is needed to create the ZIP file archives read by this library.

Zip support in PHP is not enabled by default. You will need to use the --with-zip configuration option when compiling PHP to enable zip support. This module requires ZZIPlib version >= 0.10.6.

> **Nota:** Zip support before PHP 4.1.0 is experimental. This section reflects the Zip extension as it exists in PHP 4.1.0 and later.

Example Usage

This example opens a ZIP file archive, reads each file in the archive and prints out its contents. The `test2.zip` archive used in this example is one of the test archives in the ZZIPlib source distribution.

Ejemplo 1. Zip Usage Example

```
<?php

$zip = zip_open("/tmp/test2.zip");

if ($zip) {

    while ($zip_entry = zip_read($zip)) {
        echo "Name:               " . zip_entry_name($zip_entry) . "\n";
        echo "Actual Filesize:    " . zip_entry_filesize($zip_entry) . "\n";
        echo "Compressed Size:    " . zip_entry_compressedsize($zip_entry) . "\n";
        echo "Compression Method: " . zip_entry_compressionmethod($zip_entry) . "\n";

        if (zip_entry_open($zip, $zip_entry, "r")) {
            echo "File Contents:\n";
            $buf = zip_entry_read($zip_entry, zip_entry_filesize($zip_entry));
            echo "$buf\n";

            zip_entry_close($zip_entry);
        }
        echo "\n";

    }

    zip_close($zip);

}
```

```
?>
```

zip_close (PHP 4 >= 4.1.0)

Close a Zip File Archive

void **zip_close** (resource zip) \linebreak

Closes a zip file archive. The parameter *zip* must be a zip archive previously opened by zip_open().

This function has no return value.

See also zip_open() and zip_read().

zip_entry_close (PHP 4 >= 4.1.0)

Close a Directory Entry

void **zip_entry_close** (resource zip_entry) \linebreak

Closes a directory entry specified by *zip_entry*. The parameter *zip_entry* must be a valid directory entry opened by zip_entry_open().

This function has no return value.

See also zip_entry_open() and zip_entry_read().

zip_entry_compressedsize (PHP 4 >= 4.1.0)

Retrieve the Compressed Size of a Directory Entry

int **zip_entry_compressedsize** (resource zip_entry) \linebreak

Returns the compressed size of the directory entry specified by *zip_entry*. The parameter *zip_entry* is a valid directory entry returned by zip_read().

See also zip_open() and zip_read().

zip_entry_compressionmethod (PHP 4 >= 4.1.0)

Retrieve the Compression Method of a Directory Entry

string **zip_entry_compressionmethod** (resource zip_entry) \linebreak

Returns the compression method of the directory entry specified by *zip_entry*. The parameter *zip_entry* is a valid directory entry returned by zip_read().

See also zip_open() and zip_read().

zip_entry_filesize (PHP 4 >= 4.1.0)

Retrieve the Actual File Size of a Directory Entry

int **zip_entry_filesize** (resource zip_entry) \linebreak

Returns the actual size of the directory entry specified by *zip_entry*. The parameter *zip_entry* is a valid directory entry returned by zip_read().

See also zip_open() and zip_read().

zip_entry_name (PHP 4 >= 4.1.0)

Retrieve the Name of a Directory Entry

string **zip_entry_name** (resource zip_entry) \linebreak

Returns the name of the directory entry specified by *zip_entry*. The parameter *zip_entry* is a valid directory entry returned by zip_read().

See also zip_open() and zip_read().

zip_entry_open (PHP 4 >= 4.1.0)

Open a Directory Entry for Reading

bool **zip_entry_open** (resource zip, resource zip_entry [, string mode]) \linebreak

Opens a directory entry in a zip file for reading. The parameter *zip* is a valid resource handle returned by zip_open(). The parameter *zip_entry* is a directory entry resource returned by zip_read(). The optional parameter *mode* can be any of the modes specified in the documentaion for fopen().

Nota: Currently, *mode* is ignored and is always `"rb"`. This is due to the fact that zip support in PHP is read only access. Please see fopen() for an explanation of various modes, including `"rb"`.

Devuelve `TRUE` si todo fue bien, `FALSE` en caso de fallo.

Nota: Unlike fopen() and other similar functions, the return value of **zip_entry_open()** only indicates the result of the operation and is not needed for reading or closing the directory entry.

See also zip_entry_read() and zip_entry_close().

zip_entry_read (PHP 4 >= 4.1.0)

Read From an Open Directory Entry

string **zip_entry_read** (resource zip_entry [, int length]) \linebreak

Reads up to *length* bytes from an open directory entry. If *length* is not specified, then **zip_entry_read()** will attempt to read 1024 bytes. The parameter *zip_entry* is a valid directory entry returned by zip_read().

Nota: The *length* parameter should be the uncompressed length you wish to read.

Returns the data read, or FALSE if the end of the file is reached.

See also zip_entry_open(), zip_entry_close() and zip_entry_filesize().

zip_open (PHP 4 >= 4.1.0)

Open a Zip File Archive

resource **zip_open** (string filename) \linebreak

Opens a new zip archive for reading. The *filename* parameter is the filename of the zip archive to open.

Returns a resource handle for later use with zip_read() and zip_close() or returns FALSE if *filename* does not exist.

See also zip_read() and zip_close().

zip_read (PHP 4 >= 4.1.0)

Read Next Entry in a Zip File Archive

resource **zip_read** (resource zip) \linebreak

Reads the next entry in a zip file archive. The parameter *zip* must be a zip archive previously opened by zip_open().

Returns a directory entry resource for later use with the **zip_entry_...()** functions.

See also zip_open(), zip_close(), zip_entry_open(), and zip_entry_read().

CXI. Funciones de Compresión

Este módulo usa la función de zlib (http://www.gzip.org/zlib/) de Jean-loup Gailly y Mark Adler para leer y grabar archivos comprimidos .gz, de un modo transparente. Con este módulo, es requisito usar una versión de zlib igual o posterior a 1.0.9.

Este módulo contiene versiones de la mayoría de las funciones de Sistema de archivos que funcionan con los archivos comprimidos con gzip (y con los no-comprimidos tambien, pero no con conectores (sockets)).

Pequeño código de ejemlo

Abre un archivo temporal y escribe en él, una cadena de prueba, y luego presenta el cotenido del archivo dos veces

Ejemplo 1. Ejemplo de Zlib

```
<?php
  $filename = tempnam('/tmp', 'zlibtest').'.gz';
  print "<html>\n<head></head>\n<body>\n<pre>\n";
  $s = "Sólo es una prueba, prueba, prueba,prueba, prueba, prueba!\n";
  // Abre el archivo para escribirlo con máximo de compresión
  $zp = gzopen($filename, "w9");
  // Escribe la cadena en él
  gzwrite($zp, $s);
  // Cierra el fichero
  gzclose($zp);
  // Abre el fichero para lectura
  $zp = gzopen($filename, "r");
  // Lee 3 caracteres
  print gzread($zp, 3);
  // Salida hasta el final del fichero, para cerrarlo luego.
  gzpassthru($zp);
  print "\n";
  // Abre el fichero y muestra su contenido (por segunda vez).
  if (readgzfile($filename) != strlen($s)) {
          echo "Error con las funciones zlib!";
  }
  unlink($filename);
  print "<pre>\n</h1></body>\n</html>\n";
?>
```

gzclose (PHP 3, PHP 4)

cierra un puntero a archivo-gz abierto

int **gzclose** (int zp) \linebreak

El archivo-gz al que apunta zp se cierra.

Devuelve `TRUE` (verdadero) si fue exitoso, si hubo errores devuelve `FALSE`.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con gzopen().

gzcompress (PHP 4 >= 4.0.1)

Compress a string

string **gzcompress** (string data [, int level]) \linebreak

This function returns a compressed version of the input *`data`* using the ZLIB data format, or `FALSE` if an error is encountered. The optional parameter *`level`* can be given as 0 for no compression up to 9 for maximum compression.

For details on the ZLIB compression algorithm see the document "ZLIB Compressed Data Format Specification version 3.3 (http://www.ietf.org/rfc/rfc1950.txt)" (RFC 1950).

> **Nota:** This is *not* the same as gzip compression, which includes some header data. See gzencode() for gzip compression.

See also gzdeflate(), gzinflate(), gzuncompress(), gzencode().

gzdeflate (PHP 4 >= 4.0.4)

Deflate a string

string **gzdeflate** (string data [, int level]) \linebreak

This function returns a compressed version of the input *`data`* using the DEFLATE data format, or `FALSE` if an error is encountered. The optional parameter *`level`* can be given as 0 for no compression up to 9 for maximum compression.

For details on the DEFLATE compression algorithm see the document "DEFLATE Compressed Data Format Specification version 1.3 (http://www.ietf.org/rfc/rfc1951.txt)" (RFC 1951).

See also gzinflate(), gzcompress(), gzuncompress(), gzencode().

gzencode (PHP 4 >= 4.0.4)

Create a gzip compressed string

string **gzencode** (string data [, int level [, int encoding_mode]]) \linebreak

This function returns a compressed version of the input *data* compatible with the output of the **gzip** program, or FALSE if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression, if not given the default compression level will be the default compression level of the zlib library.

You can also give FORCE_GZIP (the default) or FORCE_DEFLATE as optional third paramter *encoding_mode*. If you use FORCE_DEFLATE, you get a standard zlib deflated string (inclusive zlib headers) after the gzip file header but without the trailing crc32 checksum.

Nota: *level* was added in PHP 4.2, before PHP 4.2 **gzencode()** only had the *data* and (optional) *encoding_mode* parameters..

The resulting data contains the appropriate headers and data structure to make a standard .gz file, e.g.:

Ejemplo 1. Creating a gzip file

```
<?php
    $data = implode("", file("bigfile.txt"));
    $gzdata = gzencode($data, 9);
    $fp = fopen("bigfile.txt.gz", "w");
    fwrite($fp, $gzdata);
    fclose($fp);
?>
```

For more information on the GZIP file format, see the document: GZIP file format specification version 4.3 (http://www.ietf.org/rfc/rfc1952.txt) (RFC 1952).

See also gzcompress(). gzuncompress(), gzdeflate(), gzinflate().

gzeof (PHP 3, PHP 4)

prueba el fin-de-archivo de un puntero de archivo-gz

int **gzeof** (int zp) \linebreak

Devuelve verdadero si el puntero-a-archivo está en el fin-de-archivo, o ha ocurrido un error. De otro modo devuelve falso.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con gzopen().

gzfile (PHP 3, PHP 4)

lee el archivo gz completo en un arreglo

array **gzfile** (string nombre_archivo [, int usar_include_path]) \linebreak

Identico a readgzfile(), solo que gzfile() devuelve el fichero en un arreglo.

Se puede usar el segundo parámetro opcional poniéndolo a "1", si se quiere que la función busque también el archivo en la trayectoria definida como include_path.

Vea también readgzfile(), y gzopen().

gzgetc (PHP 3, PHP 4)

toma caracteres de un archivo-gz

cadena **gzgetc** (int zp) \linebreak

Devuelve una cadena conteniendo un caracter en particular (sin comprimir) leído del archivo al que apunta zp. Devuelve `FALSE` cuando está al final del archivo (al igual que gzeof()).

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con gzopen().

Vea también gzopen(), y gzgets().

gzgets (PHP 3, PHP 4)

toma una linea del archivo apuntado

string **gzgets** (int zp, int longitud) \linebreak

Devuelve una cadena (descomprimida) con longitud - 1 bytes de largo, leída del archivo apuntado por fp. La lectura finaliza cuando se han leído longitud - 1 bytes, ante un salto de linea o un fin-de-archivo (lo que ocurra primero).

Si ocurre un error, devuelve falso.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con gzopen().

Vea también gzopen(), gzgetc(), y fgets().

gzgetss (PHP 3, PHP 4)

toma una linea del archivo-gz apuntado y le quita los tags HTML

string **gzgetss** (int zp, int longitud [, string tags_permitidos]) \linebreak

Idéntica a gzgets(), excepto que gzgetss intenta quitar cualquier "tag" HTML o PHP del texto que lee.

Se puede usar el tercer parámetro para indicar qué parametros no deben ser extraídos.

Nota: *tags_permitidos* fue agregado en la versión de PHP 3.0.13, PHP4B3.

Véase también gzgets(), gzopen(), y strip_tags().

gzinflate (PHP 4 >= 4.0.4)

Inflate a deflated string

string **gzinflate** (string data [, int length]) \linebreak

This function takes *data* compressed by gzdeflate() and returns the original uncompressed data or FALSE on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *data* or more than the optional parameter *length*.

See also gzcompress(). gzuncompress(), gzdeflate(), gzencode().

gzopen (PHP 3, PHP 4)

open gz-file

int **gzopen** (string nombre_fichero, string modo [, int use_include_path]) \linebreak

Abre un archivo gzip (.gz) para lectura o escritura. El parámetro modo es, como en fopen() ("rb" o "wb") pero puede incluir tambien el nivel de compresión ("wb9") o la estrategia: 'f' para filtrado de datos como en "wb6f", 'h' para comprimir solo por Huffman igual que en "wb1h". (Ver la descripción de deflateInit2 en zlib.h para más información sobre el parámetro de estrategia.)

Gzopen puede usarse para leer o escribir un fichero que no esté en formato gzip; en ese caso gzread() leerá el archivo directamente, sin descomprimirlo.

Gzopen devuelve un puntero al archivo abierto y luego, cualquier proceso de lectura o escritura relacionado con ese descriptor de archivo, será transparente: se comprimirá o descomprimirá los datos según la necesidad, de manera automática.

Si la apertura fallase, se devolverá falso.

Se puede usar el tercer parámetro opcional, poniéndolo a "1", si se quiere buscar también el fichero en la trayectoria include_path.

Ejemplo 1. ejemplo de gzopen()

```
$fp = gzopen("/tmp/file.gz", "r");
```

Vea también gzclose().

gzpassthru (PHP 3, PHP 4)

Devuelve el remanente de datos de un fichero-gz

int **gzpassthru** (int zp) \linebreak

Lee hasta el Fin-De-Archivo del archivo gz dado, y escribe los resultados (descomprimidos) en la salida standard.

Si ocurre un error, devuelve Falso.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con gzopen().

El archivo-gz es cerrado cuando **gzpassthru()** termina de leerlo (dejando *zp* sin utilidad).

gzputs (PHP 3, PHP 4)

escribe al fichero-gz que se apunta

int **gzputs** (int zp, string str [, int longitud]) \linebreak

gzputs() es un alias a gzwrite(), y es absolutamente idéntico.

gzread (PHP 3, PHP 4)

Lee archivos-gz en modo Binario

string **gzread** (int zp, int longitud) \linebreak

gzread() lee hasta *longitud* bytes del archivo-gz apuntado por el parámetro *zp*. La lectura termina cuando se han leído *longitud* bytes (descomprimidos) o se alcanza el fin-de-archivo, lo que sucediera primero.

```
// Pone los contenidos del gz, a una cadena
```

```
$filename = "/usr/local/algo.txt.gz";
$zd = gzopen( $filename, "r" );
$contents = gzread( $zd, 10000 );
gzclose( $zd );
```

Ver también gzwrite(), gzopen(), gzgets(), gzgetss(), gzfile(), y gzpassthru().

gzrewind (PHP 3, PHP 4)

Reposiciona al puntero de archivo-gz, al inicio de aquel

int **gzrewind** (int zp) \linebreak

Reubica el indicador de posición del archivo, al comienzo del mismo.

si surge un error, devuelve 0.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con gzopen().

Ver también: gzseek() y gztell().

gzseek (PHP 3, PHP 4)

Posiciona el puntero del archivo-gz

int **gzseek** (int zp, int offset) \linebreak

Busca la posición dentro del archivo zp, indicada en bytes por el parametro de desplazamiento offset. Es equivalente a llamar (en C) `gzseek( zp, offset, SEEK_SET )`.

Si el archivo se abre para lectura, la función será emulada, pero puede ponerse extremadamente lenta. Si se trata de escritura, solo está soportada la búsqueda hacia adelante; gzseek comprime entonces una secuencia de ceros hasta que alcanza la nueva ubicación.

Si se completa el pedido con éxito, devuelve 0; de lo contrario, devuelve -1. Note que la busqueda más allá del fin-de-archivo no se considera un error.

Vea también gztell() y gzrewind().

gztell (PHP 3, PHP 4)

Indica la posición de lecto-escritura en el archivo

int **gztell** (int zp) \linebreak

Devuelve la posición dentro del fichero referido por zp; p.e., su desplazamiento en el cuerpo del archivo.

Si hay algún error, devuelve falso.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con gzopen().

Ver también gzopen(), gzseek() y gzrewind().

gzuncompress (PHP 4 >= 4.0.1)

Uncompress a deflated string

string **gzuncompress** (string data [, int length]) \linebreak

This function takes *data* compressed by gzcompress() and returns the original uncompressed data or FALSE on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *data* or more than the optional parameter *length*.

See also gzdeflate(), gzinflate(), gzcompress(), gzencode().

gzwrite (PHP 3, PHP 4)

Escritura de ficheros gz en modo Binario

int **gzwrite** (int zp, string cadena [, int largo]) \linebreak

gzwrite() escribe el contenido de *cadena* al fichero gz referido por *zp*. Si el parámetro *largo* está presente, se detendrá la escritura luego de escribir *largo* bytes (descomprimidos) o al llegar el final de la *cadena*, lo que ocurriese primero.

Note que si se pasa el argumento *largo*, la opcion magic_quotes_runtime será ignorada y no se quitarán barras de la *cadena* en cuestión.

Ver también gzread(), gzopen(), y gzputs().

readgzfile (PHP 3, PHP 4)

devuelve el fichero-gz

int **readgzfile** (string nombre_archivo [, int usar_trayectoria_include]) \linebreak

Lee el archivo, lo descomprime y lo escribe en la salida estándar.

Readgzfile() puede usarse para leer un archivo comprimido o no con gzip; en cuyo caso readgzfile() leerá directamente el archivo, sin descomprimirlo.

Devuelve el numero de bytes (descomprimidos) leídos del archivo, si ocurre un error, se devuelve falso y hasta que se llame como @readgzfile, se imprime un mensaje de error.

El archivo *nombre_archivo* se abrirá en el sistema de archivos y su contenido enviado a la salida estándar.

Puede usarse el segundo paametro opcional dándole el valor "1", si se quiere que se busque el archivo también dentro de la trayectoria "include": include_path.

Ver también gzpassthru(), gzfile(), y gzopen().

Parte V. Extending PHP 4.0

Capítulo 25. Overview

"Extending PHP" is easier said than done. PHP has evolved to a full-fledged tool consisting of a few megabytes of source code, and to hack a system like this quite a few things have to be learned and considered. When structuring this chapter, we finally decided on the "learn by doing" approach. This is not the most scientific and professional approach, but the method that's the most fun and gives the best end results. In the following sections, you'll learn quickly how to get the most basic extensions to work almost instantly. After that, you'll learn about Zend's advanced API functionality. The alternative would have been to try to impart the functionality, design, tips, tricks, etc. as a whole, all at once, thus giving a complete look at the big picture before doing anything practical. Although this is the "better" method, as no dirty hacks have to be made, it can be very frustrating as well as energy- and time-consuming, which is why we've decided on the direct approach.

Note that even though this chapter tries to impart as much knowledge as possible about the inner workings of PHP, it's impossible to really give a complete guide to extending PHP that works 100% of the time in all cases. PHP is such a huge and complex package that its inner workings can only be understood if you make yourself familiar with it by practicing, so we encourage you to work with the source.

What Is Zend? and What Is PHP?

The name *Zend* refers to the language engine, PHP's core. The term *PHP* refers to the complete system as it appears from the outside. This might sound a bit confusing at first, but it's not that complicated (see Figura 25-1). To implement a Web script interpreter, you need three parts:

1. The *interpreter* part analyzes the input code, translates it, and executes it.
2. The *functionality* part implements the functionality of the language (its functions, etc.).
3. The *interface* part talks to the Web server, etc.

Zend takes part 1 completely and a bit of part 2; PHP takes parts 2 and 3. Together they form the complete PHP package. Zend itself really forms only the language core, implementing PHP at its very basics with some predefined functions. PHP contains all the modules that actually create the language's outstanding capabilities.

Figura 25-1. The internal structure of PHP.

The following sections discuss where PHP can be extended and how it's done.

Capítulo 26. Extension Possibilities

As shown in Figura 25-1 above, PHP can be extended primarily at three points: external modules, built-in modules, and the Zend engine. The following sections discuss these options.

External Modules

External modules can be loaded at script runtime using the function dl(). This function loads a shared object from disk and makes its functionality available to the script to which it's being bound. After the script is terminated, the external module is discarded from memory. This method has both advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
External modules don't require recompiling of PHP.	The shared objects need to be loaded every time a script is being executed (every hit), which is very slow.
The size of PHP remains small by "outsourcing" certain functionality.	External additional files clutter up the disk.
	Every script that wants to use an external module's functionality has to specifically include a call to dl(), or the `extension` tag in `php.ini` needs to be modified (which is not always a suitable solution).

To sum up, external modules are great for third-party products, small additions to PHP that are rarely used, or just for testing purposes. To develop additional functionality quickly, external modules provide the best results. For frequent usage, larger implementations, and complex code, the disadvantages outweigh the advantages.

Third parties might consider using the `extension` tag in `php.ini` to create additional external modules to PHP. These external modules are completely detached from the main package, which is a very handy feature in commercial environments. Commercial distributors can simply ship disks or archives containing only their additional modules, without the need to create fixed and solid PHP binaries that don't allow other modules to be bound to them.

Built-in Modules

Built-in modules are compiled directly into PHP and carried around with every PHP process; their functionality is instantly available to every script that's being run. Like external modules, built-in modules have advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
No need to load the module specifically; the functionality is instantly available.	Changes to built-in modules require recompiling of PHP.
No external files clutter up the disk; everything resides in the PHP binary.	The PHP binary grows and consumes more memory.

Built-in modules are best when you have a solid library of functions that remains relatively unchanged, requires better than poor-to-average performance, or is used frequently by many scripts on your site. The need to recompile PHP is quickly compensated by the benefit in speed and ease of use. However, built-in modules are not ideal when rapid development of small additions is required.

The Zend Engine

Of course, extensions can also be implemented directly in the Zend engine. This strategy is good if you need a change in the language behavior or require special functions to be built directly into the language core. In general, however, modifications to the Zend engine should be avoided. Changes here result in incompatibilities with the rest of the world, and hardly anyone will ever adapt to specially patched Zend engines. Modifications can’t be detached from the main PHP sources and are overridden with the next update using the "official" source repositories. Therefore, this method is generally considered bad practice and, due to its rarity, is not covered in this book.

Capítulo 27. Source Layout

> **Nota:** Prior to working through the rest of this chapter, you should retrieve clean, unmodified source trees of your favorite Web server. We're working with Apache (available at http://www.apache.org/) and, of course, with PHP (available at http://www.php.net/ - does it need to be said?).
>
> Make sure that you can compile a working PHP environment by yourself! We won't go into this issue here, however, as you should already have this most basic ability when studying this chapter.

Before we start discussing code issues, you should familiarize yourself with the source tree to be able to quickly navigate through PHP's files. This is a must-have ability to implement and debug extensions.

After extracting the PHP archive, you'll see a directory layout similar to that in Figura 27-1.

Figura 27-1. Main directory layout of the PHP source tree.

The following table describes the contents of the major directories.

Directory	Contents
`php-4`	Main PHP source files and main header files; here you'll find all of PHP's API definitions, macros, etc. (importa
`ext`	Repository for dynamic and built-in modules; by default, these are the "official" PHP modules that have been int
`pear`	Directory for the PHP class repository. At the time of this writing, this is still in the design phase, but it's being t
`sapi`	Contains the code for the different server abstraction layers.
`TSRM`	Location of the "Thread Safe Resource Manager" (TSRM) for Zend and PHP.
`Zend`	Location of Zend's file; here you'll find all of Zend's API definitions, macros, etc. (important).

Discussing all the files included in the PHP package is beyond the scope of this chapter. However, you should take a close look at the following files:

- `php.h`, located in the main PHP directory. This file contains most of PHP's macro and API definitions.
- `zend.h`, located in the main Zend directory. This file contains most of Zend's macros and definitions.
- `zend_API.h`, also located in the Zend directory, which defines Zend's API.

You should also follow some sub-inclusions from these files; for example, the ones relating to the Zend executor, the PHP initialization file support, and such. After reading these files, take the time to navigate around the package a little to see the interdependencies of all files and modules - how they relate to each

other and especially how they make use of each other. This also helps you to adapt to the coding style in which PHP is authored. To extend PHP, you should quickly adapt to this style.

Extension Conventions

Zend is built using certain conventions; to avoid breaking its standards, you should follow the rules described in the following sections.

Macros

For almost every important task, Zend ships predefined macros that are extremely handy. The tables and figures in the following sections describe most of the basic functions, structures, and macros. The macro definitions can be found mainly in `zend.h` and `zend_API.h`. We suggest that you take a close look at these files after having studied this chapter. (Although you can go ahead and read them now, not everything will make sense to you yet.)

Memory Management

Resource management is a crucial issue, especially in server software. One of the most valuable resources is memory, and memory management should be handled with extreme care. Memory management has been partially abstracted in Zend, and you should stick to this abstraction for obvious reasons: Due to the abstraction, Zend gets full control over all memory allocations. Zend is able to determine whether a block is in use, automatically freeing unused blocks and blocks with lost references, and thus prevent memory leaks. The functions to be used are described in the following table:

Function	Description
emalloc()	Serves as replacement for **malloc()**.
efree()	Serves as replacement for **free()**.
estrdup()	Serves as replacement for **strdup()**.
estrndup()	Serves as replacement for **strndup()**. Faster than **estrdup()** and binary-safe. This is the recommended function t
ecalloc()	Serves as replacement for **calloc()**.
erealloc()	Serves as replacement for **realloc()**.

emalloc(), **estrdup()**, **estrndup()**, **ecalloc()**, and **erealloc()** allocate internal memory; **efree()** frees these previously allocated blocks. Memory handled by the **e*()** functions is considered local to the current process and is discarded as soon as the script executed by this process is terminated.

> **Aviso**
>
> To allocate resident memory that survives termination of the current script, you can use **malloc()** and **free()**. This should only be done with extreme care, however, and only in conjunction with demands of the Zend API; otherwise, you risk memory leaks.

Zend also features a thread-safe resource manager to provide better native support for multithreaded Web servers. This requires you to allocate local structures for all of your global variables to allow concurrent threads to be run. Because the thread-safe mode of Zend was not finished back when this was written, it is not yet extensively covered here.

Directory and File Functions

The following directory and file functions should be used in Zend modules. They behave exactly like their C counterparts, but provide virtual working directory support on the thread level.

Zend Function	Regular C Function
V_GETCWD()	getcwd()
V_FOPEN()	fopen()
V_OPEN()	**open()**
V_CHDIR()	chdir()
V_GETWD()	**getwd()**
V_CHDIR_FILE()	Takes a file path as an argument and changes the current working directory to that file's directory.
V_STAT()	stat()
V_LSTAT()	lstat()

String Handling

Strings are handled a bit differently by the Zend engine than other values such as integers, Booleans, etc., which don't require additional memory allocation for storing their values. If you want to return a string from a function, introduce a new string variable to the symbol table, or do something similar, you have to make sure that the memory the string will be occupying has previously been allocated, using the aforementioned **e*()** functions for allocation. (This might not make much sense to you yet; just keep it somewhere in your head for now - we'll get back to it shortly.)

Complex Types

Complex types such as arrays and objects require different treatment. Zend features a single API for these types - they’re stored using hash tables.

> **Nota:** To reduce complexity in the following source examples, we’re only working with simple types such as integers at first. A discussion about creating more advanced types follows later in this chapter.

Capítulo 28. PHP’s Automatic Build System

PHP 4 features an automatic build system that's very flexible. All modules reside in a subdirectory of the `ext` directory. In addition to its own sources, each module consists of an M4 file (for example, see http://www.gnu.org/manual/m4/html_mono/m4.html) for configuration and a Makefile.in file, which is responsible for compilation (the results of autoconf and automake; for example, see http://sourceware.cygnus.com/autoconf/autoconf.html and http://sourceware.cygnus.com/automake/automake.html).

Both files are generated automatically, along with `.cvsignore`, by a little shell script named `ext_skel` that resides in the `ext` directory. As argument it takes the name of the module that you want to create. The shell script then creates a directory of the same name, along with the appropriate `config.m4` and `Makefile.in` files.

Step by step, the process looks like this:

```
root@dev:/usr/local/src/php4/ext > ./ext_skel my_module
Creating directory
Creating basic files: config.m4 Makefile.in .cvsignore [done].
To use your new extension, you will have to execute the following steps:
    $ cd ..
    $ ./buildconf
    $ ./configure # (your extension is automatically enabled)
    $ vi ext/my_module/my_module.c
    $ make
Repeat the last two steps as often as necessary.
```

This instruction creates the aforementioned files. To include the new module in the automatic configuration and build process, you have to run `buildconf`, which regenerates the `configure` script by searching through the `ext` directory and including all found `config.m4` files.

Finally, running `configure` parses all configuration options and generates a makefile based on those options and the options you specify in `Makefile.in`.

Ejemplo 28-1shows the previously generated `Makefile.in`:

Ejemplo 28-1. The default `Makefile.in`.

```
# $Id: Extending_Zend_Build.xml,v 1.6 2002/03/25 08:13:46 hholzgra Exp $
LTLIBRARY_NAME        = libmy_module.la
LTLIBRARY_SOURCES     = my_module.c
LTLIBRARY_SHARED_NAME = my_module.la include
$(top_srcdir)/build/dynlib.mk
```

There's not much to tell about this one: It contains the names of the input and output files. You could also specify build instructions for other files if your module is built from multiple source files.

The default `config.m4` shown in Ejemplo 28-2'/> is a bit more complex:

Ejemplo 28-2. The default `config.m4`.

```
dnl $Id: Extending_Zend_Build.xml,v 1.6 2002/03/25 08:13:46 hholzgra Exp $
dnl config.m4 for extension my_module
dnl don't forget to call PHP_EXTENSION(my_module)
```

```
dnl If your extension references something external, use with:
PHP_ARG_WITH(my_module, for my_module support,
dnl Make sure that the comment is aligned:
    [  --with-my_module             Include my_module support])
dnl Otherwise use enable:
PHP_ARG_ENABLE(my_module, whether to enable my_module support,
dnl Make sure that the comment is aligned:
    [  --enable-my_module           Enable my_module support])
if test "$PHP_MY_MODULE" != "no"; then
dnl Action..
PHP_EXTENSION(my_module, $ext_shared)
fi
```

If you're unfamiliar with M4 files (now is certainly a good time to get familiar), this might be a bit confusing at first; but it's actually quite easy.

Note: Everything prefixed with `dnl` is treated as a comment and is not parsed.

The `config.m4` file is responsible for parsing the command-line options passed to `configure` at configuration time. This means that it has to check for required external files and do similar configuration and setup tasks.

The default file creates two configuration directives in the `configure` script: `--with-my_module` and `--enable-my_module`. Use the first option when referring external files (such as the `--with-apache` directive that refers to the Apache directory). Use the second option when the user simply has to decide whether to enable your extension. Regardless of which option you use, you should uncomment the other, unnecessary one; that is, if you're using `--enable-my_module`, you should remove support for `--with-my_module`, and vice versa.

By default, the `config.m4` file created by `ext_skel` accepts both directives and automatically enables your extension. Enabling the extension is done by using the `PHP_EXTENSION` macro. To change the default behavior to include your module into the PHP binary when desired by the user (by explicitly specifying `--enable-my_module` or `--with-my_module`), change the test for `$PHP_MY_MODULE` to `== "yes"`:

```
if test "$PHP_MY_MODULE" == "yes"; then dnl
    Action.. PHP_EXTENSION(my_module, $ext_shared)
    fi
```

This would require you to use `--enable-my_module` each time when reconfiguring and recompiling PHP.

Note: Be sure to run `buildconf` every time you change `config.m4`!

We'll go into more details on the M4 macros available to your configuration scripts later in this chapter. For now, we'll simply use the default files. The sample sources on the CD-ROM all have working `config.m4` files. To include them into the PHP build process, simply copy the source directories to your PHP `ext` directory, run `buildconf`, and then include the sample modules you want by using the appropriate `--enable-*` directives with `configure`.

Capítulo 29. Creating Extensions

We'll start with the creation of a very simple extension at first, which basically does nothing more than implement a function that returns the integer it receives as parameter. Ejemplo 29-1 shows the source.

Ejemplo 29-1. A simple extension.

```
/* include standard header */
#include "php.h"

/* declaration of functions to be exported */
ZEND_FUNCTION(first_module);

/* compiled function list so Zend knows what's in this module */
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};

/* compiled module information */
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES
};

/* implement standard "stub" routine to introduce ourselves to Zend */
#if COMPILE_DL_FIRST_MODULE
ZEND_GET_MODULE(firstmod)
#endif

/* implement function that is meant to be made available to PHP */
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAIL-
URE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

This code contains a complete PHP module. We'll explain the source code in detail shortly, but first we'd like to discuss the build process. (This will allow the impatient to experiment before we dive into API discussions.)

> **Nota:** The example source makes use of some features introduced with the Zend version used in PHP 4.1.0 and above, it won't compile with older PHP 4.0.x versions.

Compiling Modules

There are basically two ways to compile modules:

- Use the provided "make" mechanism in the `ext` directory, which also allows building of dynamic loadable modules.
- Compile the sources manually.

The first method should definitely be favored, since, as of PHP 4.0, this has been standardized into a sophisticated build process. The fact that it is so sophisticated is also its drawback, unfortunately - it's hard to understand at first. We'll provide a more detailed introduction to this later in the chapter, but first let's work with the default files.

The second method is good for those who (for some reason) don't have the full PHP source tree available, don't have access to all files, or just like to juggle with their keyboard. These cases should be extremely rare, but for the sake of completeness we'll also describe this method.

Compiling Using Make. To compile the sample sources using the standard mechanism, copy all their subdirectories to the `ext` directory of your PHP source tree. Then run `buildconf`, which will create an updated `configure` script containing appropriate options for the new extension. By default, all the sample sources are disabled, so you don't have to fear breaking your build process.

After you run `buildconf`, `configure --help` shows the following additional modules:

```
--enable-array_experiments   BOOK: Enables array experiments
--enable-call_userland       BOOK: Enables userland module
--enable-cross_conversion    BOOK: Enables cross-conversion module
--enable-first_module        BOOK: Enables first module
--enable-infoprint           BOOK: Enables infoprint module
--enable-reference_test      BOOK: Enables reference test module
--enable-resource_test       BOOK: Enables resource test module
--enable-variable_creation   BOOK: Enables variable-creation module
```

The module shown earlier in Ejemplo 29-1 can be enabled with `--enable-first_module` or `--enable-first_module=yes`.

Compiling Manually. To compile your modules manually, you need the following commands:

Action	Command
Compiling	cc -fpic -DCOMPILE_DL=1 -I/usr/local/include -I. -I.. -I../Zend -c -o `<your_object_file>` `<your_c_file>`

Linking	cc -shared -L/usr/local/lib -rdynamic -o `<your_module_file>` `<your_object_file(s)>`

The command to compile the module simply instructs the compiler to generate position-independent code (`-fpic` shouldn't be omitted) and additionally defines the constant `COMPILE_DL` to tell the module code that it's compiled as a dynamically loadable module (the test module above checks for this; we'll discuss it shortly). After these options, it specifies a number of standard include paths that should be used as the minimal set to compile the source files.

Note: All include paths in the example are relative to the directory `ext`. If you're compiling from another directory, change the pathnames accordingly. Required items are the PHP directory, the `Zend` directory, and (if necessary), the directory in which your module resides.

The link command is also a plain vanilla command instructing linkage as a dynamic module.

You can include optimization options in the compilation command, although these have been omitted in this example (but some are included in the makefile template described in an earlier section).

Note: Compiling and linking manually as a static module into the PHP binary involves very long instructions and thus is not discussed here. (It's not very efficient to type all those commands.)

Capítulo 30. Using Extensions

Depending on the build process you selected, you should either end up with a new PHP binary to be linked into your Web server (or run as CGI), or with an .so (shared object) file. If you compiled the example file `first_module.c` as a shared object, your result file should be `first_module.so`. To use it, you first have to copy it to a place from which it's accessible to PHP. For a simple test procedure, you can copy it to your `htdocs` directory and try it with the source in Ejemplo 30-1. If you compiled it into the PHP binary, omit the call to dl(), as the module's functionality is instantly available to your scripts.

> **Aviso**
>
> For security reasons, you *should not* put your dynamic modules into publicly accessible directories. Even though it *can* be done and it simplifies testing, you should put them into a separate directory in production environments.

Ejemplo 30-1. A test file for first_module.so.

```
<?php

// remove next comment if necessary
// dl("first_module.so");

$param = 2;
$return = first_module($param);

print("We sent '$param' and got '$return'");

?>
```

Calling this PHP file in your Web browser should give you the output shown in Figura 30-1.

Figura 30-1. Output of first_module.php.

If required, the dynamic loadable module is loaded by calling the dl() function. This function looks for the specified shared object, loads it, and makes its functions available to PHP. The module exports the

function **first_module()**, which accepts a single parameter, converts it to an integer, and returns the result of the conversion.

If you've gotten this far, congratulations! You just built your first extension to PHP.

Capítulo 31. Troubleshooting

Actually, not much troubleshooting can be done when compiling static or dynamic modules. The only problem that could arise is that the compiler will complain about missing definitions or something similar. In this case, make sure that all header files are available and that you specified their path correctly in the compilation command. To be sure that everything is located correctly, extract a clean PHP source tree and use the automatic build in the `ext` directory with the fresh files; this will guarantee a safe compilation environment. If this fails, try manual compilation.

PHP might also complain about missing functions in your module. (This shouldn't happen with the sample sources if you didn't modify them.) If the names of external functions you're trying to access from your module are misspelled, they'll remain as "unlinked symbols" in the symbol table. During dynamic loading and linkage by PHP, they won't resolve because of the typing errors - there are no corresponding symbols in the main binary. Look for incorrect declarations in your module file or incorrectly written external references. Note that this problem is specific to dynamic loadable modules; it doesn't occur with static modules. Errors in static modules show up at compile time.

Capítulo 32. Source Discussion

Now that you've got a safe build environment and you're able to include the modules into PHP files, it's time to discuss how everything works.

Module Structure

All PHP modules follow a common structure:

- Header file inclusions (to include all required macros, API definitions, etc.)
- C declaration of exported functions (required to declare the Zend function block)
- Declaration of the Zend function block
- Declaration of the Zend module block
- Implementation of **get_module()**
- Implementation of all exported functions

Header File Inclusions

The only header file you really have to include for your modules is `php.h`, located in the PHP directory. This file makes all macros and API definitions required to build new modules available to your code.

Tip: It's good practice to create a separate header file for your module that contains module-specific definitions. This header file should contain all the forward definitions for exported functions and include `php.h`. If you created your module using `ext_skel` you already have such a header file prepared.

Declaring Exported Functions

To declare functions that are to be exported (i.e., made available to PHP as new native functions), Zend provides a set of macros. A sample declaration looks like this:

```
ZEND_FUNCTION ( my_function );
```

`ZEND_FUNCTION` declares a new C function that complies with Zend's internal API. This means that the function is of type `void` and accepts `INTERNAL_FUNCTION_PARAMETERS` (another macro) as parameters. Additionally, it prefixes the function name with `zif`. The immediately expanded version of the above definitions would look like this:

```
void zif_my_function ( INTERNAL_FUNCTION_PARAMETERS );
```

Expanding `INTERNAL_FUNCTION_PARAMETERS` results in the following:

```
void zif_my_function( int ht
                    , zval * return_value
                    , zval * this_ptr
                    , int return_value_used
                    , zend_executor_globals * executor_globals
                    );
```

Since the interpreter and executor core have been separated from the main PHP package, a second API defining macros and function sets has evolved: the Zend API. As the Zend API now handles quite a few of the responsibilities that previously belonged to PHP, a lot of PHP functions have been reduced to macros aliasing to calls into the Zend API. The recommended practice is to use the Zend API wherever possible, as the old API is only preserved for compatibility reasons. For example, the types zval and pval are identical. zval is Zend's definition; pval is PHP's definition (actually, pval is an alias for zval now). As the macro `INTERNAL_FUNCTION_PARAMETERS` is a Zend macro, the above declaration contains zval. When writing code, you should always use zval to conform to the new Zend API.

The parameter list of this declaration is very important; you should keep these parameters in mind (see Tabla 32-1 for descriptions).

Tabla 32-1. Zend's Parameters to Functions Called from PHP

Parameter	Description
ht	The number of arguments passed to the Zend function. You should not touch this directly, but instead us
return_value	This variable is used to pass any return values of your function back to PHP. Access to this variable is b
this_ptr	Using this variable, you can gain access to the object in which your function is contained, if it's used wi
return_value_used	This flag indicates whether an eventual return value from this function will actually be used by the calli
executor_globals	This variable points to global settings of the Zend engine. You'll find this useful when creating new vari

Declaration of the Zend Function Block

Now that you have declared the functions to be exported, you also have to introduce them to Zend. Introducing the list of functions is done by using an array of zend_function_entry. This array consecutively contains all functions that are to be made available externally, with the function's name as it should appear in PHP and its name as defined in the C source. Internally, zend_function_entry is defined as shown in Ejemplo 32-1.

Ejemplo 32-1. Internal declaration of zend_function_entry.

```
typedef struct _zend_function_entry {
    char *fname;
    void (*handler)(INTERNAL_FUNCTION_PARAMETERS);
    unsigned char *func_arg_types;
```

```
} zend_function_entry;
```

Entry	Description
fname	Denotes the function name as seen in PHP (for example, `fopen`, `mysql_connect`, or, in our example, `fir`
handler	Pointer to the C function responsible for handling calls to this function. For example, see the standard macr
func_arg_types	Allows you to mark certain parameters so that they're forced to be passed by reference. You usually should

In the example above, the declaration looks like this:

```
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};
```

You can see that the last entry in the list always has to be `{NULL, NULL, NULL}`. This marker has to be set for Zend to know when the end of the list of exported functions is reached.

> **Nota:** You *cannot* use the predefined macros for the end marker, as these would try to refer to a function named "NULL"!

The macro `ZEND_FE` (short for 'Zend Function Entry') simply expands to a structure entry in zend_function_entry. Note that these macros introduce a special naming scheme to your functions - your C functions will be prefixed with `zif_`, meaning that `ZEND_FE(first_module)` will refer to a C function **zif_first_module()**. If you want to mix macro usage with hand-coded entries (not a good practice), keep this in mind.

Tip: Compilation errors that refer to functions named **zif_*()** relate to functions defined with `ZEND_FE`.

Tabla 32-2 shows a list of all the macros that you can use to define functions.

Tabla 32-2. Macros for Defining Functions

Macro Name	Description
`ZEND_FE(name, arg_types)`	Defines a function entry of the name name in zend_function_en
`ZEND_NAMED_FE(php_name, name, arg_types)`	Defines a function that will be available to PHP by the name ph
`ZEND_FALIAS(name, alias, arg_types)`	Defines an alias named alias for name. arg_types needs to be se
`PHP_FE(name, arg_types)`	Old PHP API equivalent of `ZEND_FE`.
`PHP_NAMED_FE(runtime_name, name, arg_types)`	Old PHP API equivalent of `ZEND_NAMED_FE`.

Note: You can't use `ZEND_FE` in conjunction with `PHP_FUNCTION`, or `PHP_FE` in conjunction with `ZEND_FUNCTION`. However, it's perfectly legal to mix `ZEND_FE` and `ZEND_FUNCTION` with `PHP_FE` and `PHP_FUNCTION` when staying with the same macro set for each function to be declared. But mixing is *not* recommended; instead, you're advised to use the `ZEND_*` macros only.

Declaration of the Zend Module Block

This block is stored in the structure zend_module_entry and contains all necessary information to describe the contents of this module to Zend. You can see the internal definition of this module in Ejemplo 32-2.

Ejemplo 32-2. Internal declaration of zend_module_entry.

```
typedef struct _zend_module_entry zend_module_entry;

    struct _zend_module_entry {
    unsigned short size;
    unsigned int zend_api;
    unsigned char zend_debug;
    unsigned char zts;
    char *name;
    zend_function_entry *functions;
    int (*module_startup_func)(INIT_FUNC_ARGS);
    int (*module_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    int (*request_startup_func)(INIT_FUNC_ARGS);
    int (*request_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    void (*info_func)(ZEND_MODULE_INFO_FUNC_ARGS);
    char *version;
    int (*global_startup_func)(void);
    int (*global_shutdown_func)(void);

[ Rest of the structure is not interesting here ]

};
```

Entry	Description
size, zend_api, zend_debug and zts	Usually filled with the `"STANDARD_MODULE_HEADER"`, which fills these four membe
name	Contains the module name (for example, `"File functions"`, `"Socket function`
functions	Points to the Zend function block, discussed in the preceding section.
module_startup_func	This function is called once upon module initialization and can be used to do one-time
module_shutdown_func	This function is called once upon module shutdown and can be used to do one-time de
request_startup_func	This function is called once upon every page request and can be used to do one-time i
request_shutdown_func	This function is called once after every page request and works as counterpart to **requ**
info_func	When phpinfo() is called in a script, Zend cycles through all loaded modules and calls
version	The version of the module. You can use `NO_VERSION_YET` if you don't want to give t
Remaining structure elements	These are used internally and can be prefilled by using the macro `STANDARD_MODULE`

In our example, this structure is implemented as follows:

```
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
```

```
    "First Module",
    firstmod_functions,
    NULL, NULL, NULL, NULL, NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES,
};
```

This is basically the easiest and most minimal set of values you could ever use. The module name is set to `First Module`, then the function list is referenced, after which all startup and shutdown functions are marked as being unused.

For reference purposes, you can find a list of the macros involved in declared startup and shutdown functions in Tabla 32-3. These are not used in our basic example yet, but will be demonstrated later on. You should make use of these macros to declare your startup and shutdown functions, as these require special arguments to be passed (`INIT_FUNC_ARGS` and `SHUTDOWN_FUNC_ARGS`), which are automatically included into the function declaration when using the predefined macros. If you declare your functions manually and the PHP developers decide that a change in the argument list is necessary, you'll have to change your module sources to remain compatible.

Tabla 32-3. Macros to Declare Startup and Shutdown Functions

Macro	Description
`ZEND_MINIT(module)`	Declares a function for module startup. The generated name will be `zend_minit_<module`
`ZEND_MSHUTDOWN(module)`	Declares a function for module shutdown. The generated name will be `zend_mshutdown_<`
`ZEND_RINIT(module)`	Declares a function for request startup. The generated name will be `zend_rinit_<module`
`ZEND_RSHUTDOWN(module)`	Declares a function for request shutdown. The generated name will be `zend_rshutdown_<`
`ZEND_MINFO(module)`	Declares a function for printing module information, used when phpinfo() is called. The gen

Creation of get_module()

This function is special to all dynamic loadable modules. Take a look at the creation via the `ZEND_GET_MODULE` macro first:

```
#if COMPILE_DL_FIRSTMOD
     ZEND_GET_MODULE(firstmod)
#endif
```

The function implementation is surrounded by a conditional compilation statement. This is needed since the function **get_module()** is only required if your module is built as a dynamic extension. By specifying a definition of `COMPILE_DL_FIRSTMOD` in the compiler command (see above for a discussion of the compilation instructions required to build a dynamic extension), you can instruct your module whether you intend to build it as a dynamic extension or as a built-in module. If you want a built-in module, the implementation of **get_module()** is simply left out.

get_module() is called by Zend at load time of the module. You can think of it as being invoked by the dl() call in your script. Its purpose is to pass the module information block back to Zend in order to inform the engine about the module contents.

If you don't implement a **get_module()** function in your dynamic loadable module, Zend will compliment you with an error message when trying to access it.

Implementation of All Exported Functions

Implementing the exported functions is the final step. The example function in `first_module` looks like this:

```
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAIL-
URE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

The function declaration is done using `ZEND_FUNCTION`, which corresponds to `ZEND_FE` in the function entry table (discussed earlier).

After the declaration, code for checking and retrieving the function's arguments, argument conversion, and return value generation follows (more on this later).

Summary

That's it, basically - there's nothing more to implementing PHP modules. Built-in modules are structured similarly to dynamic modules, so, equipped with the information presented in the previous sections, you'll be able to fight the odds when encountering PHP module source files.

Now, in the following sections, read on about how to make use of PHP's internals to build powerful extensions.

Capítulo 33. Accepting Arguments

One of the most important issues for language extensions is accepting and dealing with data passed via arguments. Most extensions are built to deal with specific input data (or require parameters to perform their specific actions), and function arguments are the only real way to exchange data between the PHP level and the C level. Of course, there's also the possibility of exchanging data using predefined global values (which is also discussed later), but this should be avoided by all means, as it's extremely bad practice.

PHP doesn't make use of any formal function declarations; this is why call syntax is always completely dynamic and never checked for errors. Checking for correct call syntax is left to the user code. For example, it's possible to call a function using only one argument at one time and four arguments the next time - both invocations are syntactically absolutely correct.

Determining the Number of Arguments

Since PHP doesn't have formal function definitions with support for call syntax checking, and since PHP features variable arguments, sometimes you need to find out with how many arguments your function has been called. You can use the `ZEND_NUM_ARGS` macro in this case. In previous versions of PHP, this macro retrieved the number of arguments with which the function has been called based on the function's hash table entry, ht, which is passed in the `INTERNAL_FUNCTION_PARAMETERS` list. As ht itself now contains the number of arguments that have been passed to the function, `ZEND_NUM_ARGS` has been stripped down to a dummy macro (see its definition in `zend_API.h`). But it's still good practice to use it, to remain compatible with future changes in the call interface. *Note:* The old PHP equivalent of this macro is `ARG_COUNT`.

The following code checks for the correct number of arguments:

```
if(ZEND_NUM_ARGS() != 2) WRONG_PARAM_COUNT;
```

If the function is not called with two arguments, it exits with an error message. The code snippet above makes use of the tool macro `WRONG_PARAM_COUNT`, which can be used to generate a standard error message (see Figura 33-1).

Figura 33-1. `WRONG_PARAM_COUNT` in action.

This macro prints a default error message and then returns to the caller. Its definition can also be found in `zend_API.h` and looks like this:

```
ZEND_API void wrong_param_count(void);
```

```
#define WRONG_PARAM_COUNT { wrong_param_count(); return; }
```

As you can see, it calls an internal function named **wrong_param_count()** that's responsible for printing the warning. For details on generating customized error messages, see the later section "Printing Information."

Retrieving Arguments

> **New parameter parsing API:** This chapter documents the new Zend parameter parsing API introduced by Andrei Zmievski. It was introduced in the development stage between PHP 4.0.6 and 4.1.0 .

Parsing parameters is a very common operation and it may get a bit tedious. It would also be nice to have standardized error checking and error messages. Since PHP 4.1.0, there is a way to do just that by using the new parameter parsing API. It greatly simplifies the process of receiving parameteres, but it has a drawback in that it can't be used for functions that expect variable number of parameters. But since the vast majority of functions do not fall into those categories, this parsing API is recommended as the new standard way.

The prototype for parameter parsing function looks like this:

```
int zend_parse_parameters(int num_args TSRMLS_DC, char *type_spec, ...);
```

The first argument to this function is supposed to be the number of actual parameters passed to your function, so `ZEND_NUM_ARGS()` can be used for that. The second parameter should always be `TSRMLS_CC` macro. The third argument is a string that specifies the number and types of arguments your function is expecting, similar to how printf format string specifies the number and format of the output values it should operate on. And finally the rest of the arguments are pointers to variables which should receive the values from the parameters.

zend_parse_parameters() also performs type conversions whenever possible, so that you always receive the data in the format you asked for. Any type of scalar can be converted to another one, but conversions between complex types (arrays, objects, and resources) and scalar types are not allowed.

If the parameters could be obtained successfully and there were no errors during type conversion, the function will return `SUCCESS`, otherwise it will return `FAILURE`. The function will output informative error messages, if the number of received parameters does not match the requested number, or if type conversion could not be performed.

Here are some sample error messages:

```
Warning - ini_get_all() requires at most 1 parameter, 2 given
Warning - wddx_deserialize() expects parameter 1 to be string, array given
```

Of course each error message is accompanied by the filename and line number on which it occurs.

Here is the full list of type specifiers:

- `l` - long
- `d` - double
- `s` - string (with possible null bytes) and its length
- `b` - boolean
- `r` - resource, stored in `zval*`
- `a` - array, stored in `zval*`
- `o` - object (of any class), stored in `zval*`
- `O` - object (of class specified by class entry), stored in `zval*`
- `z` - the actual `zval*`

The following characters also have a meaning in the specifier string:

- `|` - indicates that the remaining parameters are optional. The storage variables corresponding to these parameters should be initialized to default values by the extension, since they will not be touched by the parsing function if the parameters are not passed.
- `/` - the parsing function will call **SEPARATE_ZVAL_IF_NOT_REF()** on the parameter it follows, to provide a copy of the parameter, unless it's a reference.
- `!` - the parameter it follows can be of specified type or `NULL` (only applies to a, o, O, r, and z). If `NULL` value is passed by the user, the storage pointer will be set to `NULL`.

The best way to illustrate the usage of this function is through examples:

```
/* Gets a long, a string and its length, and a zval. */
long l;
char *s;
int s_len;
zval *param;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "lsz", &l, &s, &s_len, &param) == FAILURE) {
    return;
}

/* Gets an object of class specified by my_ce, and an optional double. */
zval *obj;
double d = 0.5;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "O|d", &obj, my_ce, &d) == FAILURE) {
    return;
}

/* Gets an object or null, and an array.
   If null is passed for object, obj will be set to NULL. */
zval *obj;
```

```
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "O!a", &obj, &arr) == FAIL-
URE) {
    return;
}

/* Gets a separated array. */
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "a/", &arr) == FAILURE) {
    return;
}

/* Get only the first three parameters (useful for varargs functions). */
zval *z;
zend_bool b;
zval *r;
if (zend_parse_parameters(3, "zbr!", &z, &b, &r) == FAILURE) {
    return;
}
```

Note that in the last example we pass 3 for the number of received parameters, instead of **ZEND_NUM_ARGS()**. What this lets us do is receive the least number of parameters if our function expects a variable number of them. Of course, if you want to operate on the rest of the parameters, you will have to use **zend_get_parameters_array_ex()** to obtain them.

The parsing function has an extended version that allows for an additional flags argument that controls its actions.

```
int zend_parse_parameters_ex(int flags, int num_args TSRMLS_DC, char *type_spec, ...);
```

The only flag you can pass currently is `ZEND_PARSE_PARAMS_QUIET`, which instructs the function to not output any error messages during its operation. This is useful for functions that expect several sets of completely different arguments, but you will have to output your own error messages.

For example, here is how you would get either a set of three longs or a string:

```
long l1, l2, l3;
char *s;
if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                             ZEND_NUM_ARGS() TSRMLS_CC,
                             "lll", &l1, &l2, &l3) == SUCCESS) {
    /* manipulate longs */
} else if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                                    ZEND_NUM_ARGS(), "s", &s, &s_len) == SUCCESS) {
    /* manipulate string */
} else {
    php_error(E_WARNING, "%s() takes either three long values or a string as argument",
```

```
                get_active_function_name(TSRMLS_C));
    return;
}
```

With all the abovementioned ways of receiving function parameters you should have a good handle on this process. For even more example, look through the source code for extensions that are shipped with PHP - they illustrate every conceivable situation.

Old way of retrieving arguments (deprecated)

> **Deprecated parameter parsing API:** This API is deprecated and superseded by the new ZEND parameter parsing API.

After having checked the number of arguments, you need to get access to the arguments themselves. This is done with the help of **zend_get_parameters_ex()**:

```
zval **parameter;

if(zend_get_parameters_ex(1, &parameter) != SUCCESS)
  WRONG_PARAM_COUNT;
```

All arguments are stored in a zval container, which needs to be pointed to *twice*. The snippet above tries to retrieve one argument and make it available to us via the parameter pointer.

zend_get_parameters_ex() accepts at least two arguments. The first argument is the number of arguments to retrieve (which should match the number of arguments with which the function has been called; this is why it's important to check for correct call syntax). The second argument (and all following arguments) are pointers to pointers to pointers to zvals. (Confusing, isn't it?) All these pointers are required because Zend works internally with **zval; to adjust a local **zval in our function,**zend_get_parameters_ex()** requires a pointer to it.

The return value of **zend_get_parameters_ex()** can either be `SUCCESS` or `FAILURE`, indicating (unsurprisingly) success or failure of the argument processing. A failure is most likely related to an incorrect number of arguments being specified, in which case you should exit with `WRONG_PARAM_COUNT`.

To retrieve more than one argument, you can use a similar snippet:

```
zval **param1, **param2, **param3, **param4;

if(zend_get_parameters_ex(4, &param1, &param2, &param3, &param4) != SUCCESS)
    WRONG_PARAM_COUNT;
```

zend_get_parameters_ex() only checks whether you're trying to retrieve too many parameters. If the function is called with five arguments, but you're only retrieving three of them with **zend_get_parameters_ex()**, you won't get an error but will get the first three parameters instead. Subsequent calls of **zend_get_parameters_ex()** won't retrieve the remaining arguments, but will get the same arguments again.

Dealing with a Variable Number of Arguments/Optional Parameters

If your function is meant to accept a variable number of arguments, the snippets just described are sometimes suboptimal solutions. You have to create a line calling **zend_get_parameters_ex()** for every possible number of arguments, which is often unsatisfying.

For this case, you can use the function **zend_get_parameters_array_ex()**, which accepts the number of parameters to retrieve and an array in which to store them:

```
zval **parameter_array[4];

/* get the number of arguments */
argument_count = ZEND_NUM_ARGS();

/* see if it satisfies our minimal request (2 arguments) */
/* and our maximal acceptance (4 arguments) */
if(argument_count < 2 || argument_count > 5)
    WRONG_PARAM_COUNT;

/* argument count is correct, now retrieve arguments */
if(zend_get_parameters_array_ex(argument_count, parameter_array) != SUCCESS)
    WRONG_PARAM_COUNT;
```

First, the number of arguments is checked to make sure that it's in the accepted range. After that, **zend_get_parameters_array_ex()** is used to fill parameter_array with valid pointers to the argument values.

A very clever implementation of this can be found in the code handling PHP's fsockopen() located in `ext/standard/fsock.c`, as shown in Ejemplo 33-1. Don't worry if you don't know all the functions used in this source yet; we'll get to them shortly.

Ejemplo 33-1. PHP's implementation of variable arguments in fsockopen().

```
pval **args[5];
```

```
int *sock=emalloc(sizeof(int));
int *sockp;
int arg_count=ARG_COUNT(ht);
int socketd = -1;
unsigned char udp = 0;
struct timeval timeout = { 60, 0 };
unsigned short portno;
unsigned long conv;
char *key = NULL;
FLS_FETCH();

if (arg_count > 5 || arg_count < 2 || zend_get_parameters_array_ex(arg_count,args)==FAILURE)
    CLOSE_SOCK(1);
    WRONG_PARAM_COUNT;
}

switch(arg_count) {
    case 5:
        convert_to_double_ex(args[4]);
        conv = (unsigned long) (Z_DVAL_P(args[4]) * 1000000.0);
        timeout.tv_sec = conv / 1000000;
        timeout.tv_usec = conv % 1000000;
        /* fall-through */
    case 4:
        if (!PZVAL_IS_REF(*args[3])) {
            php_error(E_WARNING,"error string argument to fsockopen not passed by reference"
        }
        pval_copy_constructor(*args[3]);
        ZVAL_EMPTY_STRING(*args[3]);
        /* fall-through */
    case 3:
        if (!PZVAL_IS_REF(*args[2])) {
            php_error(E_WARNING,"error argument to fsockopen not passed by reference");
            return;
        }
        ZVAL_LONG(*args[2], 0);
        break;
}

convert_to_string_ex(args[0]);
convert_to_long_ex(args[1]);
portno = (unsigned short) Z_LVAL_P(args[1]);

key = emalloc(Z_STRLEN_P(args[0]) + 10);
```

fsockopen() accepts two, three, four, or five parameters. After the obligatory variable declarations, the function checks for the correct range of arguments. Then it uses a fall-through mechanism in a `switch()` statement to deal with all arguments. The `switch()` statement starts with the maximum number of arguments being passed (five). After that, it automatically processes the case of four arguments being passed, then three, by omitting the otherwise obligatory `break` keyword in all stages.

After having processed the last case, it exits the `switch()` statement and does the minimal argument processing needed if the function is invoked with only two arguments.

This multiple-stage type of processing, similar to a stairway, allows convenient processing of a variable number of arguments.

Accessing Arguments

To access arguments, it's necessary for each argument to have a clearly defined type. Again, PHP's extremely dynamic nature introduces some quirks. Because PHP never does any kind of type checking, it's possible for a caller to pass any kind of data to your functions, whether you want it or not. If you expect an integer, for example, the caller might pass an array, and vice versa - PHP simply won't notice.

To work around this, you have to use a set of API functions to force a type conversion on every argument that's being passed (see Tabla 33-1).

Note: All conversion functions expect a **zval as parameter.

Tabla 33-1. Argument Conversion Functions

Function	Description
convert_to_boolean_ex()	Forces conversion to a Boolean type. Boolean values remain untouched. Longs, doubl
convert_to_long_ex()	Forces conversion to a long, the default integer type. NULL values, Booleans, resourc
convert_to_double_ex()	Forces conversion to a double, the default floating-point type. NULL values, Booleans
convert_to_string_ex()	Forces conversion to a string. Strings remain untouched. NULL values are converted t
`convert_to_array_ex(value)`	Forces conversion to an array. Arrays remain untouched. Objects are converted to an a
`convert_to_object_ex(value)`	Forces conversion to an object. Objects remain untouched. NULL values are converte
`convert_to_null_ex(value)`	Forces the type to become a NULL value, meaning empty.

Nota: You can find a demonstration of the behavior in `cross_conversion.php` on the accompanying CD-ROM. Figura 33-2 shows the output.

Figura 33-2. Cross-conversion behavior of PHP.

Using these functions on your arguments will ensure type safety for all data that's passed to you. If the supplied type doesn't match the required type, PHP forces dummy contents on the resulting value (empty strings, arrays, or objects, `0` for numeric values, `FALSE` for Booleans) to ensure a defined state.

Following is a quote from the sample module discussed previously, which makes use of the conversion functions:

```
zval **parameter;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &parameter) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

convert_to_long_ex(parameter);

RETURN_LONG(Z_LVAL_P(parameter));
```

After retrieving the parameter pointer, the parameter value is converted to a long (an integer), which also forms the return value of this function. Understanding access to the contents of the value requires a short discussion of the zval type, whose definition is shown in Ejemplo 33-2.

Ejemplo 33-2. PHP/Zend zval type definition.

```
typedef pval zval;

typedef struct _zval_struct zval;

typedef union _zvalue_value {
long lval; /* long value */
double dval; /* double value */
struct {
char *val;
int len;
} str;
HashTable *ht; /* hash table value */
struct {
zend_class_entry *ce;
HashTable *properties;
} obj;
} zvalue_value;

struct _zval_struct {
/* Variable information */
zvalue_value value; /* value */
unsigned char type; /* active type */
unsigned char is_ref;
short refcount;
};
```

Actually, pval (defined in `php.h`) is only an alias of zval (defined in `zend.h`), which in turn refers to _zval_struct. This is a most interesting structure. _zval_struct is the "master" structure, containing the value structure, type, and reference information. The substructure zvalue_value is a union that contains the variable's contents. Depending on the variable's type, you'll have to access different members of this union. For a description of both structures, see Tabla 33-2, Tabla 33-3 and Tabla 33-4.

Tabla 33-2. Zend zval Structure

Entry	Description
value	Union containing this variable's contents. See Tabla 33-3 for a description.
type	Contains this variable's type. For a list of available types, see Tabla 33-4.
is_ref	0 means that this variable is not a reference; 1 means that this variable is a reference to another variable.
refcount	The number of references that exist for this variable. For every new reference to the value stored in this variable, t

Tabla 33-3. Zend zvalue_value Structure

Entry	Description
lval	Use this property if the variable is of the type `IS_LONG`, `IS_BOOLEAN`, or `IS_RESOURCE`.
dval	Use this property if the variable is of the type `IS_DOUBLE`.
str	This structure can be used to access variables of the type `IS_STRING`. The member len contains the string length; the
ht	This entry points to the variable's hash table entry if the variable is an array.
obj	Use this property if the variable is of the type `IS_OBJECT`.

Tabla 33-4. Zend Variable Type Constants

Constant	Description
`IS_NULL`	Denotes a NULL (empty) value.
`IS_LONG`	A long (integer) value.
`IS_DOUBLE`	A double (floating point) value.
`IS_STRING`	A string.
`IS_ARRAY`	Denotes an array.
`IS_OBJECT`	An object.
`IS_BOOL`	A Boolean value.
`IS_RESOURCE`	A resource (for a discussion of resources, see the appropriate section below).
`IS_CONSTANT`	A constant (defined) value.

To access a long you access zval.value.lval, to access a double you use zval.value.dval, and so on. Because all values are stored in a union, trying to access data with incorrect union members results in meaningless output.

Accessing arrays and objects is a bit more complicated and is discussed later.

Dealing with Arguments Passed by Reference

If your function accepts arguments passed by reference that you intend to modify, you need to take some precautions.

What we didn't say yet is that under the circumstances presented so far, you don't have write access to any zval containers designating function parameters that have been passed to you. Of course, you can change any zval containers that you created within your function, but you mustn't change any zvals that refer to Zend-internal data!

We've only discussed the so-called ***_ex()** API so far. You may have noticed that the API functions we've used are called **zend_get_parameters_ex()** instead of **zend_get_parameters()**, **convert_to_long_ex()** instead of **convert_to_long()**, etc. The ***_ex()** functions form the so-called new "extended" Zend API. They give a minor speed increase over the old API, but as a tradeoff are only meant for providing read-only access.

Because Zend works internally with references, different variables may reference the same value. Write access to a zval container requires this container to contain an isolated value, meaning a value that's not referenced by any other containers. If a zval container were referenced by other containers and you changed the referenced zval, you would automatically change the contents of the other containers referencing this zval (because they'd simply point to the changed value and thus change their own value as well).

zend_get_parameters_ex() doesn't care about this situation, but simply returns a pointer to the desired zval containers, whether they consist of references or not. Its corresponding function in the traditional API, **zend_get_parameters()**, immediately checks for referenced values. If it finds a reference, it creates a new, isolated zval container; copies the referenced data into this newly allocated space; and then returns a pointer to the new, isolated value.

This action is called *zval separation* (or pval separation). Because the ***_ex()** API doesn't perform zval separation, it's considerably faster, while at the same time disabling write access.

To change parameters, however, write access is required. Zend deals with this situation in a special way: Whenever a parameter to a function is passed by reference, it performs automatic zval separation. This means that whenever you're calling a function like this in PHP, Zend will automatically ensure that $parameter is being passed as an isolated value, rendering it to a write-safe state:

```
my_function(&$parameter);
```

But this *is not* the case with regular parameters! All other parameters that are not passed by reference are in a read-only state.

This requires you to make sure that you're really working with a reference - otherwise you might produce unwanted results. To check for a parameter being passed by reference, you can use the macro `PZVAL_IS_REF`. This macro accepts a `zval*` to check if it is a reference or not. Examples are given in in Ejemplo 33-3.

Ejemplo 33-3. Testing for referenced parameter passing.

```
zval *parameter;
```

```
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
    return;

/* check for parameter being passed by reference */
if (!PZVAL_IS_REF(*parameter)) {
{
    zend_error(E_WARNING, "Parameter wasn't passed by reference");
    RETURN_NULL();
}

/* make changes to the parameter */
ZVAL_LONG(*parameter, 10);
```

Assuring Write Safety for Other Parameters

You might run into a situation in which you need write access to a parameter that's retrieved with **zend_get_parameters_ex()** but not passed by reference. For this case, you can use the macro `SEPARATE_ZVAL`, which does a zval separation on the provided container. The newly generated zval is detached from internal data and has only a local scope, meaning that it can be changed or destroyed without implying global changes in the script context:

```
zval **parameter;

/* retrieve parameter */
zend_get_parameters_ex(1, &parameter);

/* at this stage, <parameter> still is connected */
/* to Zend's internal data buffers */

/* make <parameter> write-safe */
SEPARATE_ZVAL(parameter);

/* now we can safely modify <parameter> */
/* without implying global changes */
```

`SEPARATE_ZVAL` uses **emalloc()** to allocate the new zval container, which means that even if you don't deallocate this memory yourself, it will be destroyed automatically upon script termination. However, doing a lot of calls to this macro without freeing the resulting containers will clutter up your RAM.

Note: As you can easily work around the lack of write access in the "traditional" API (with **zend_get_parameters()** and so on), this API seems to be obsolete, and is not discussed further in this chapter.

Capítulo 34. Creating Variables

When exchanging data from your own extensions with PHP scripts, one of the most important issues is the creation of variables. This section shows you how to deal with the variable types that PHP supports.

Overview

To create new variables that can be seen "from the outside" by the executing script, you need to allocate a new zval container, fill this container with meaningful values, and then introduce it to Zend's internal symbol table. This basic process is common to all variable creations:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol ta-
ble */
ZEND_SET_SYMBOL(EG(active_symbol_table), "new_variable_name", new_variable);

/* the variable is now accessible to the script by using $new_variable_name */
```

The macro `MAKE_STD_ZVAL` allocates a new zval container using `ALLOC_ZVAL` and initializes it using `INIT_ZVAL`. As implemented in Zend at the time of this writing, *initializing* means setting the reference count to `1` and clearing the is_ref flag, but this process could be extended later - this is why it's a good idea to keep using `MAKE_STD_ZVAL` instead of only using `ALLOC_ZVAL`. If you want to optimize for speed (and you don't have to explicitly initialize the zval container here), you can use `ALLOC_ZVAL`, but this isn't recommended because it doesn't ensure data integrity.

`ZEND_SET_SYMBOL` takes care of introducing the new variable to Zend's symbol table. This macro checks whether the value already exists in the symbol table and converts the new symbol to a reference if so (with automatic deallocation of the old zval container). This is the preferred method if speed is not a crucial issue and you'd like to keep memory usage low.

Note that `ZEND_SET_SYMBOL` makes use of the Zend executor globals via the macro `EG`. By specifying `EG(active_symbol_table)`, you get access to the currently active symbol table, dealing with the active, local scope. The local scope may differ depending on whether the function was invoked from within a function.

If you need to optimize for speed and don't care about optimal memory usage, you can omit the check for an existing variable with the same value and instead force insertion into the symbol table by using **zend_hash_update()**:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */
```

```
/* introduce this variable by the name "new_variable_name" into the symbol ta-
ble */
zend_hash_update(
    EG(active_symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

This is actually the standard method used in most modules.

The variables generated with the snippet above will always be of local scope, so they reside in the context in which the function has been called. To create new variables in the global scope, use the same method but refer to another symbol table:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global sym-
bol table
ZEND_SET_SYMBOL(&EG(symbol_table), "new_variable_name", new_variable);
```

The macro `ZEND_SET_SYMBOL` is now being called with a reference to the main, global symbol table by referring `EG(symbol_table)`.

Note: The active_symbol_table variable is a pointer, but symbol_table is not. This is why you have to use `EG(active_symbol_table)` and `&EG(symbol_table)` as parameters to `ZEND_SET_SYMBOL` - it requires a pointer.

Similarly, to get a more efficient version, you can hardcode the symbol table update:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global sym-
bol table
zend_hash_update(
    &EG(symbol_table),
```

```
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

Ejemplo 34-1 shows a sample source that creates two variables - local_variable with a local scope and global_variable with a global scope (see Figure 9.7). The full example can be found on the CD-ROM.

Note: You can see that the global variable is actually not accessible from within the function. This is because it's not imported into the local scope using `global $global_variable;` in the PHP source.

Ejemplo 34-1. Creating variables with different scopes.

```
ZEND_FUNCTION(variable_creation)
{
    zval *new_var1, *new_var2;

    MAKE_STD_ZVAL(new_var1);
    MAKE_STD_ZVAL(new_var2);

    ZVAL_LONG(new_var1, 10);
    ZVAL_LONG(new_var2, 5);

    ZEND_SET_SYMBOL(EG(active_symbol_table), "local_variable", new_var1);
    ZEND_SET_SYMBOL(&EG(symbol_table), "global_variable", new_var2);

    RETURN_NULL();

}
```

Longs (Integers)

Now let's get to the assignment of data to variables, starting with longs. Longs are PHP's integers and are very simple to store. Looking at the zval.value container structure discussed earlier in this chapter, you

can see that the long data type is directly contained in the union, namely in the lval field. The corresponding type value for longs is `IS_LONG` (see Ejemplo 34-2).

Ejemplo 34-2. Creation of a long.

```
zval *new_long;

MAKE_STD_ZVAL(new_long);

new_long->type = IS_LONG;
new_long->value.lval = 10;
```

Alternatively, you can use the macro `ZVAL_LONG`:

```
zval *new_long;

MAKE_STD_ZVAL(new_long);
ZVAL_LONG(new_long, 10);
```

Doubles (Floats)

Doubles are PHP's floats and are as easy to assign as longs, because their value is also contained directly in the union. The member in the zval.value container is dval; the corresponding type is `IS_DOUBLE`.

```
zval *new_double;

MAKE_STD_ZVAL(new_double);

new_double->type = IS_DOUBLE;
new_double->value.dval = 3.45;
```

Alternatively, you can use the macro `ZVAL_DOUBLE`:

```
zval *new_double;

MAKE_STD_ZVAL(new_double);
ZVAL_DOUBLE(new_double, 3.45);
```

Strings

Strings need slightly more effort. As mentioned earlier, all strings that will be associated with Zend's internal data structures need to be allocated using Zend's own memory-management functions. Referencing of static strings or strings allocated with standard routines is not allowed. To assign strings, you have to access the structure str in the zval.value container. The corresponding type is `IS_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);

new_string->type = IS_STRING;
new_string->value.str.len = strlen(string_contents);
new_string->value.str.val = estrdup(string_contents);
```

Note the usage of Zend's **estrdup()** here. Of course, you can also use the predefined macro `ZVAL_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);
ZVAL_STRING(new_string, string_contents, 1);
```

`ZVAL_STRING` accepts a third parameter that indicates whether the supplied string contents should be duplicated (using **estrdup()**). Setting this parameter to `1` causes the string to be duplicated; `0` simply uses the supplied pointer for the variable contents. This is most useful if you want to create a new variable referring to a string that's already allocated in Zend internal memory.

If you want to truncate the string at a certain position or you already know its length, you can use `ZVAL_STRINGL(zval, string, length, duplicate)`, which accepts an explicit string length to be set for the new string. This macro is faster than `ZVAL_STRING` and also binary-safe.

To create empty strings, set the string length to `0` and use `empty_string` as contents:

```
new_string->type = IS_STRING;
new_string->value.str.len = 0;
new_string->value.str.val = empty_string;
```

Of course, there's a macro for this as well (`ZVAL_EMPTY_STRING`):

```
MAKE_STD_ZVAL(new_string);
ZVAL_EMPTY_STRING(new_string);
```

Booleans

Booleans are created just like longs, but have the type `IS_BOOL`. Allowed values in lval are `0` and `1`:

```
zval *new_bool;

MAKE_STD_ZVAL(new_bool);

new_bool->type = IS_BOOL;
new_bool->value.lval = 1;
```

The corresponding macros for this type are `ZVAL_BOOL` (allowing specification of the value) as well as `ZVAL_TRUE` and `ZVAL_FALSE` (which explicitly set the value to `TRUE` and `FALSE`, respectively).

Arrays

Arrays are stored using Zend's internal hash tables, which can be accessed using the **zend_hash_*()** API. For every array that you want to create, you need a new hash table handle, which will be stored in the ht member of the zval.value container.

There's a whole API solely for the creation of arrays, which is extremely handy. To start a new array, you call **array_init()**.

```
zval *new_array;

MAKE_STD_ZVAL(new_array);

if(array_init(new_array) != SUCCESS)
{
    // do error handling here
}
```

If **array_init()** fails to create a new array, it returns `FAILURE`.

To add new elements to the array, you can use numerous functions, depending on what you want to do. Tabla 34-1, Tabla 34-2 and Tabla 34-3 describe these functions. All functions return `FAILURE` on failure and `SUCCESS` on success.

Tabla 34-1. Zend's API for Associative Arrays

Function	Description
add_assoc_long(zval *array, char *key, long n);()	Adds an element of type `long`.
add_assoc_unset(zval *array, char *key);()	Adds an unset element.
add_assoc_bool(zval *array, char *key, int b);()	Adds a Boolean element.
add_assoc_resource(zval *array, char *key, int r);()	Adds a resource to the array.

add_assoc_double(zval *array, char *key, double d);()	Adds a floating-point value.
add_assoc_string(zval *array, char *key, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_assoc_stringl(zval *array, char *key, char *str, uint length, int duplicate); ()	Adds a string with the desired length length to the array. Otherwise, behaves like **add_assoc_string()**.

Tabla 34-2. Zend's API for Indexed Arrays, Part 1

Function	Description
add_index_long(zval *array, uint idx, long n);()	Adds an element of type `long`.
add_index_unset(zval *array, uint idx);()	Adds an unset element.
add_index_bool(zval *array, uint idx, int b);()	Adds a Boolean element.
add_index_resource(zval *array, uint idx, int r);()	Adds a resource to the array.
add_index_double(zval *array, uint idx, double d);()	Adds a floating-point value.
add_index_string(zval *array, uint idx, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_index_stringl(zval *array, uint idx, char *str, uint length, int duplicate);()	Adds a string with the desired length length to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.

Tabla 34-3. Zend's API for Indexed Arrays, Part 2

Function	Description
add_next_index_long(zval *array, long n);()	Adds an element of type `long`.
add_next_index_unset(zval *array);()	Adds an unset element.
add_next_index_bool(zval *array, int b);()	Adds a Boolean element.
add_next_index_resource(zval *array, int r);()	Adds a resource to the array.
add_next_index_double(zval *array, double d);()	Adds a floating-point value.
add_next_index_string(zval *array, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_next_index_stringl(zval *array, char *str, uint length, int duplicate);()	Adds a string with the desired length length to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.

All these functions provide a handy abstraction to Zend's internal hash API. Of course, you can also use the hash functions directly - for example, if you already have a zval container allocated that you want to

insert into an array. This is done using **zend_hash_update()()** for associative arrays (see Ejemplo 34-3) and **zend_hash_index_update()** for indexed arrays (see Ejemplo 34-4):

Ejemplo 34-3. Adding an element to an associative array.

```
zval *new_array, *new_element;
char *key = "element_key";

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

if(array_init(new_array) == FAILURE)
{
    // do error handling here
}

ZVAL_LONG(new_element, 10);

if(zend_hash_update(new_array->value.ht, key, strlen(key) + 1, (void *)&new_element, sizeof(
{
    // do error handling here
}
```

Ejemplo 34-4. Adding an element to an indexed array.

```
zval *new_array, *new_element;
int key = 2;

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

if(array_init(new_array) == FAILURE)
{
    // do error handling here
}

ZVAL_LONG(new_element, 10);

if(zend_hash_index_update(new_array->value.ht, key, (void *)&new_element, sizeof(zval *), NU
{
    // do error handling here
}
```

To emulate the functionality of **add_next_index_*()**, you can use this:

```
zend_hash_next_index_insert(ht, zval **new_element, sizeof(zval *), NULL)
```

Note: To return arrays from a function, use **array_init()** and all following actions on the predefined variable return_value (given as argument to your exported function; see the earlier discussion of the call interface). You do not have to use `MAKE_STD_ZVAL` on this.

Tip: To avoid having to write `new_array->value.ht` every time, you can use `HASH_OF(new_array)`, which is also recommended for compatibility and style reasons.

Objects

Since objects can be converted to arrays (and vice versa), you might have already guessed that they have a lot of similarities to arrays in PHP. Objects are maintained with the same hash functions, but there's a different API for creating them.

To initialize an object, you use the function **object_init()**:

```
zval *new_object;

MAKE_STD_ZVAL(new_object);

if(object_init(new_object) != SUCCESS)
{
    // do error handling here
}
```

You can use the functions described in Tabla 34-4 to add members to your object.

Tabla 34-4. Zend's API for Object Creation

Function	Description
add_property_long(zval *object, char *key, long l);()	Adds a long to the object.
add_property_unset(zval *object, char *key);()	Adds an unset property to the obj
add_property_bool(zval *object, char *key, int b);()	Adds a Boolean to the object.
add_property_resource(zval *object, char *key, long r);()	Adds a resource to the object.
add_property_double(zval *object, char *key, double d);()	Adds a double to the object.
add_property_string(zval *object, char *key, char *str, int duplicate);()	Adds a string to the object.
add_property_stringl(zval *object, char *key, char *str, uint length, int duplicate);()	Adds a string of the specified leng
add_property_zval(zval *obect, char *key, zval *container):()	Adds a `zval` container to the obj

Resources

Resources are a special kind of data type in PHP. The term *resources* doesn't really refer to any special kind of data, but to an abstraction method for maintaining any kind of information. Resources are kept in a special resource list within Zend. Each entry in the list has a correspondending type definition that denotes the kind of resource to which it refers. Zend then internally manages all references to this

resource. Access to a resource is never possible directly - only via a provided API. As soon as all references to a specific resource are lost, a corresponding shutdown function is called.

For example, resources are used to store database links and file descriptors. The *de facto* standard implementation can be found in the MySQL module, but other modules such as the Oracle module also make use of resources.

> **Nota:** In fact, a resource can be a pointer to anything you need to handle in your functions (e.g. pointer to a structure) and the user only has to pass a single resource variable to your function.

To create a new resource you need to register a resource destruction handler for it. Since you can store any kind of data as a resource, Zend needs to know how to free this resource if its not longer needed. This works by registering your own resource destruction handler to Zend which in turn gets called by Zend whenever your resource can be freed (whether manually or automatically). Registering your resource handler within Zend returns you the **resource type handle** for that resource. This handle is needed whenever you want to access a resource of this type later and is most of time stored in a global static variable within your extension. There is no need to worry about thread safety here because you only register your resource handler once during module initialization.

The Zend function to register your resource handler is defined as:

```
ZEND_API int zend_register_list_destructors_ex(rsrc_dtor_func_t ld, rsrc_dtor_func_t pld, ch
```

There are two different kinds of resource destruction handlers you can pass to this function: a handler for normal resources and a handler for persistent resources. Persistent resources are for example used for database connection. When registering a resource, either of these handlers must be given. For the other handler just pass `NULL`.

zend_register_list_destructors_ex() accepts the following parameters:

`ld`	Normal resource destruction handler callback
`pld`	Pesistent resource destruction handler callback
`type_name`	A string specifying the name of your resource. It’s always a good thing to specify an unique name within
`module_number`	The `module_number` is automatically available in your `PHP_MINIT_FUNCTION` function and therefore y

The return value is an unique integer ID for your **resource type**.

The resource destruction handler (either normal or persistent resources) has the following prototype:

```
void resource_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC);
```

The passed `rsrc` is a pointer to the following structure:

```
typedef struct _zend_rsrc_list_entry {

    void *ptr;
    int type;
    int refcount;
```

```
} zend_rsrc_list_entry;
```

The member `void *ptr` is the actual pointer to your resource.

Now we know how to start things, we define our own resource we want register within Zend. It is only a simple structure with two integer members:

```
typedef struct {

    int resource_link;
    int resource_type;

} my_resource;
```

Our resource destruction handler is probably going to look something like this:

```
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    // You most likely cast the void pointer to your structure type

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;

    // Now do whatever needs to be done with you resource. Closing
    // Files, Sockets, freeing additional memory, etc.
    // Also, don't forget to actually free the memory for your resource too!

    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}
```

> **Nota:** One important thing to mention: If your resource is a rather complex structure which also contains pointers to memory you allocated during runtime you have to free them **before** freeing the resource itself!

Now that we have defined

1. what our resource is and
2. our resource destruction handler

we can go on and do the rest of the steps:

1. create a global variable within the extension holding the resource ID so it can be accessed from every function which needs it
2. define the resource name
3. write the resource destruction handler

4. and finally register the handler

```
// Somewhere in your extension, define the variable for your registered resources.
// If you wondered what 'le' stands for: it simply means 'list entry'.
static int le_myresource;

// It's nice to define your resource name somewhere
#define le_myresource_name  "My type of resource"

[...]

// Now actually define our resource destruction handler
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;
    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}

[...]

PHP_MINIT_FUNCTION(my_extension) {

    // Note that 'module_number' is already provided through the
    // PHP_MINIT_FUNCTION() function definition.

    le_myresource = zend_register_resource_destructors_ex(my_destruction_handler, NULL,

    // You can register additional resources, initialize
    // your global vars, constants, whatever.
}
```

To actually register a new resource you use can either use the **zend_register_resource()** function or the **ZEND_REGISTER_RESOURE()** macro, both defined in zend_list.h . Although the arguments for both map 1:1 it's a good idea to always use macros to be upwards compatible:

```
int ZEND_REGISTER_RESOURCE(zval *rsrc_result, void *rsrc_pointer, int rsrc_type);
```

`rsrc_result`	This is an already initialized `zval *` container.
`rsrc_pointer`	Your resource pointer you want to store.
`rsrc_type`	The type which you received when you registered the resource destruction handler. If you followed the nar

The return value is an unique integer identifier for that resource.

What is really going on when you register a new resource is it gets inserted in an internal list in Zend and the result is just stored in the given `zval *` container:

```
rsrc_id = zend_list_insert(rsrc_pointer, rsrc_type);
```

```
if (rsrc_result) {
    rsrc_result->value.lval = rsrc_id;
    rsrc_result->type = IS_RESOURCE;
}

return rsrc_id;
```

The returned `rsrc_id` uniquly identifies the newly registered resource. You can use the macro `RETURN_RESOURE` to return it to the user:

```
RETURN_RESOURCE(rsrc_id)
```

> **Nota:** It is common practice that if you want to return the resource immidiately to the user you specify the `return_value` as the `zval *` container.

Zend now keeps track of all references to this resource. As soon as all references to the resource are lost, the destructor that you previously registered for this resource is called. The nice thing about this setup is that you don't have to worry about memory leakages introduced by allocations in your module - just register all memory allocations that your calling script will refer to as resources. As soon as the script decides it doesn't need them anymore, Zend will find out and tell you.

Now that the user got his resource, at some point he is passing it back to one of your functions. The value.lval inside the `zval *` container contains the key to your resource and thus can be used to fetch the resource with the following macro: `ZEND_FETCH_RESOURCE`:

```
ZEND_FETCH_RESOURCE(rsrc, rsrc_type, rsrc_id, default_rsrc_id, resource_type_name, resource_
```

`rsrc`	This is your pointer which will point to your previously registered resource.
`rsrc_type`	This is the typecast argument for your pointer, e.g. `myresource *`.
`rsrc_id`	This is the address of the `zval *`container the user passed to your function, e.g. `&z_resource` if
`default_rsrc_id`	This integer specifies the default resource `ID` if no resource could be fetched or -1.
`resource_type_name`	This is the name of the requested resource. It's a string and is used when the resource can't be four
`resource_type`	The `resource_type` you got back when registering the resource destruction handler. In our exam

This macro has no return value. It is for the developers convenience and takes care of TSRMLS arguments passing and also does check if the resource could be fetched. It throws a warning message and returns the current PHP function with `NULL` if there was a problem retrieving the resource.

To force removal of a resource from the list, use the function **zend_list_delete()**. You can also force the reference count to increase if you know that you're creating another reference for a previously allocated value (for example, if you're automatically reusing a default database link). For this case, use the function **zend_list_addref()**. To search for previously allocated resource entries, use **zend_list_find()**. The complete API can be found in `zend_list.h`.

Macros for Automatic Global Variable Creation

In addition to the macros discussed earlier, a few macros allow easy creation of simple global variables. These are nice to know in case you want to introduce global flags, for example. This is somewhat bad practice, but Table Tabla 34-5 describes macros that do exactly this task. They don't need any zval allocation; you simply have to supply a variable name and value.

Tabla 34-5. Macros for Global Variable Creation

Macro	Description
`SET_VAR_STRING(name, value)`	Creates a new string.
`SET_VAR_STRINGL(name, value, length)`	Creates a new string of the specified length. This macro is faster than `SET_VAR_STRING` and also binary-safe.
`SET_VAR_LONG(name, value)`	Creates a new long.
`SET_VAR_DOUBLE(name, value)`	Creates a new double.

Creating Constants

Zend supports the creation of true constants (as opposed to regular variables). Constants are accessed without the typical dollar sign (`$`) prefix and are available in all scopes. Examples include `TRUE` and `FALSE`, to name just two.

To create your own constants, you can use the macros in Tabla 34-6. All the macros create a constant with the specified name and value.

You can also specify flags for each constant:

- `CONST_CS` - This constant's name is to be treated as case sensitive.
- `CONST_PERSISTENT` - This constant is persistent and won't be "forgotten" when the current process carrying this constant shuts down.

To use the flags, combine them using a inary OR:

```
// register a new constant of type "long"
    REGISTER_LONG_CONSTANT("NEW_MEANINGFUL_CONSTANT", 324, CONST_CS |
    CONST_PERSISTENT);
```

There are two types of macros - `REGISTER_*_CONSTANT` and`REGISTER_MAIN_*_CONSTANT`. The first type creates constants bound to the current module. These constants are dumped from the symbol table as soon as the module that registered the constant is unloaded from memory. The second type creates constants that remain in the symbol table independently of the module.

Tabla 34-6. Macros for Creating Constants

Macro
`REGISTER_LONG_CONSTANT(name, value, flags)` `REGISTER_MAIN_LONG_CONSTANT(name, value, flags)`

`REGISTER_DOUBLE_CONSTANT(name, value, flags)`	`REGISTER_MAIN_DOUBLE_CONSTANT(name, value, flag`
`REGISTER_STRING_CONSTANT(name, value, flags)`	`REGISTER_MAIN_STRING_CONSTANT(name, value, flag`
`REGISTER_STRINGL_CONSTANT(name, value, length, flags)`	`REGISTER_MAIN_STRINGL_CONSTANT(name, v`

Capítulo 35. Duplicating Variable Contents: The Copy Constructor

Sooner or later, you may need to assign the contents of one zval container to another. This is easier said than done, since the zval container doesn't contain only type information, but also references to places in Zend's internal data. For example, depending on their size, arrays and objects may be nested with lots of hash table entries. By assigning one zval to another, you avoid duplicating the hash table entries, using only a reference to them (at most).

To copy this complex kind of data, use the *copy constructor*. Copy constructors are typically defined in languages that support operator overloading, with the express purpose of copying complex types. If you define an object in such a language, you have the possibility of overloading the "=" operator, which is usually responsible for assigning the contents of the lvalue (result of the evaluation of the left side of the operator) to the rvalue (same for the right side).

Overloading means assigning a different meaning to this operator, and is usually used to assign a function call to an operator. Whenever this operator would be used on such an object in a program, this function would be called with the lvalue and rvalue as parameters. Equipped with that information, it can perform the operation it intends the "=" operator to have (usually an extended form of copying).

This same form of "extended copying" is also necessary for PHP's zval containers. Again, in the case of an array, this extended copying would imply re-creation of all hash table entries relating to this array. For strings, proper memory allocation would have to be assured, and so on.

Zend ships with such a function, called **zend_copy_ctor()** (the previous PHP equivalent was **pval_copy_constructor()**).

A most useful demonstration is a function that accepts a complex type as argument, modifies it, and then returns the argument:

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
   return;
}

// do modifications to the parameter here

// now we want to return the modified container:
*return_value == *parameter;
zval_copy_ctor(return_value);
```

The first part of the function is plain-vanilla argument retrieval. After the (left out) modifications, however, it gets interesting: The container of parameter is assigned to the (predefined) return_value container. Now, in order to effectively duplicate its contents, the copy constructor is called. The copy constructor works directly with the supplied argument, and the standard return values are `FAILURE` on failure and `SUCCESS` on success.

If you omit the call to the copy constructor in this example, both parameter and return_value would point to the same internal data, meaning that return_value would be an illegal additional reference to the same data structures. Whenever changes occurred in the data that parameter points to, return_value might be affected. Thus, in order to create separate copies, the copy constructor must be used.

The copy constructor's counterpart in the Zend API, the destructor **zval_dtor()**, does the opposite of the constructor.

Capítulo 36. Returning Values

Returning values from your functions to PHP was described briefly in an earlier section; this section gives the details. Return values are passed via the return_value variable, which is passed to your functions as argument. The return_value argument consists of a zval container (see the earlier discussion of the call interface) that you can freely modify. The container itself is already allocated, so you don't have to run `MAKE_STD_ZVAL` on it. Instead, you can access its members directly.

To make returning values from functions easier and to prevent hassles with accessing the internal structures of the zval container, a set of predefined macros is available (as usual). These macros automatically set the correspondent type and value, as described in Tabla 36-1 and Tabla 36-2.

> **Nota:** The macros in Tabla 36-1 automatically *return* from your function, those in Tabla 36-2 only *set* the return value; they don't return from your function.

Tabla 36-1. Predefined Macros for Returning Values from a Function

Macro	Description
`RETURN_RESOURCE(resource)`	Returns a resource.
`RETURN_BOOL(bool)`	Returns a Boolean.
`RETURN_NULL()`	Returns nothing (a NULL value).
`RETURN_LONG(long)`	Returns a long.
`RETURN_DOUBLE(double)`	Returns a double.
`RETURN_STRING(string, duplicate)`	Returns a string. The duplicate flag indicates whether the string should be duplicated using **estrdup()**.
`RETURN_STRINGL(string, length, duplicate)`	Returns a string of the specified length; otherwise, behaves like `RETURN_STRING`. This macro is faster and binary-safe, however.
`RETURN_EMPTY_STRING()`	Returns an empty string.
`RETURN_FALSE`	Returns Boolean false.
`RETURN_TRUE`	Returns Boolean true.

Tabla 36-2. Predefined Macros for Setting the Return Value of a Function

Macro	Description
`RETVAL_RESOURCE(resource)`	Sets the return value to the specified resource.
`RETVAL_BOOL(bool)`	Sets the return value to the specified Boolean value.
`RETVAL_NULL`	Sets the return value to NULL.
`RETVAL_LONG(long)`	Sets the return value to the specified long.
`RETVAL_DOUBLE(double)`	Sets the return value to the specified double.
`RETVAL_STRING(string, duplicate)`	Sets the return value to the specified string and duplicates it to Zend internal memory if desired (see also `RETURN_STRING`).

`RETVAL_STRINGL(string, length, duplicate)`	Sets the return value to the specified string and forces the length to become length (see also `RETVAL_STRING`). This macro is faster and binary-safe, and should be used whenever the string length is known.
`RETVAL_EMPTY_STRING`	Sets the return value to an empty string.
`RETVAL_FALSE`	Sets the return value to Boolean false.
`RETVAL_TRUE`	Sets the return value to Boolean true.

Complex types such as arrays and objects can be returned by using **array_init()** and **object_init()**, as well as the corresponding hash functions on return_value. Since these types cannot be constructed of trivial information, there are no predefined macros for them.

Capítulo 37. Printing Information

Often it's necessary to print messages to the output stream from your module, just as print() would be used within a script. PHP offers functions for most generic tasks, such as printing warning messages, generating output for phpinfo(), and so on. The following sections provide more details. Examples of these functions can be found on the CD-ROM.

zend_printf()

zend_printf() works like the standard printf(), except that it prints to Zend's output stream.

zend_error()

zend_error() can be used to generate error messages. This function accepts two arguments; the first is the error type (see `zend_errors.h`), and the second is the error message.

```
zend_error(E_WARNING, "This function has been called with empty arguments");
```

Tabla 37-1 shows a list of possible values (see Figura 37-1). These values are also referred to in `php.ini`. Depending on which error type you choose, your messages will be logged.

Tabla 37-1. Zend's Predefined Error Messages.

Error	Description
`E_ERROR`	Signals an error and terminates execution of the script immediately .
`E_WARNING`	Signals a generic warning. Execution continues.
`E_PARSE`	Signals a parser error. Execution continues.
`E_NOTICE`	Signals a notice. Execution continues. Note that by default the display of this type of error message
`E_CORE_ERROR`	Internal error by the core; shouldn't be used by user-written modules.
`E_COMPILE_ERROR`	Internal error by the compiler; shouldn't be used by user-written modules.
`E_COMPILE_WARNING`	Internal warning by the compiler; shouldn't be used by user-written modules.

Figura 37-1. Display of warning messages in the browser.

Including Output in phpinfo()

After creating a real module, you'll want to show information about the module in phpinfo() (in addition to the module name, which appears in the module list by default). PHP allows you to create your own section in the phpinfo() output with the `ZEND_MINFO()` function. This function should be placed in the module descriptor block (discussed earlier) and is always called whenever a script calls phpinfo().

PHP automatically prints a section in phpinfo() for you if you specify the `ZEND_MINFO` function, including the module name in the heading. Everything else must be formatted and printed by you.

Typically, you can print an HTML table header using **php_info_print_table_start()** and then use the standard functions **php_info_print_table_header()** and **php_info_print_table_row()**. As arguments, both take the number of columns (as integers) and the column contents (as strings). Ejemplo 37-1 shows a source example and its output. To print the table footer, use **php_info_print_table_end()**.

Ejemplo 37-1. Source code and screenshot for output in phpinfo().

```
php_info_print_table_start();
php_info_print_table_header(2, "First column", "Second column");
php_info_print_table_row(2, "Entry in first row", "Another entry");
php_info_print_table_row(2, "Just to fill", "another row here");
php_info_print_table_end();
```

Execution Information

You can also print execution information, such as the current file being executed. The name of the function currently being executed can be retrieved using the function **get_active_function_name()**. This function returns a pointer to the function name and doesn't accept any arguments. To retrieve the name of the file currently being executed, use **zend_get_executed_filename()**. This function accesses the executor globals, which are passed to it using the `TSRMLS_C` macro. The executor globals are automatically available to every function that's called directly by Zend (they're part of the `INTERNAL_FUNCTION_PARAMETERS` described earlier in this chapter). If you want to access the executor globals in another function that doesn't have them available automatically, call the macro `TSRMLS_FETCH()` once in that function; this will introduce them to your local scope.

Finally, the line number currently being executed can be retrieved using the function **zend_get_executed_lineno()**. This function also requires the executor globals as arguments. For examples of these functions, see Ejemplo 37-2.

Ejemplo 37-2. Printing execution information.

```
zend_printf("The name of the current function is %s<br>", get_active_function_name(TSRMLS_C)
zend_printf("The file currently executed is %s<br>", zend_get_executed_filename(TSRMLS_C));
zend_printf("The current line being executed is %i<br>", zend_get_executed_lineno(TSRMLS_C))
```

Capítulo 38. Startup and Shutdown Functions

Startup and shutdown functions can be used for one-time initialization and deinitialization of your modules. As discussed earlier in this chapter (see the description of the Zend module descriptor block), there are global, module, and request startup and shutdown events.

The global startup functions are called once when PHP starts up; similarly, the global shutdown functions are called once when PHP shuts down. Please note that they’re really only called *once*, not when a new Apache process is being created!

The module startup and shutdown functions are called whenever a module is loaded and needs initialization; the request startup and shutdown functions are called every time a request is processed (meaning that a file is being executed).

For dynamic extensions, module and request startup/shutdown events happen at the same time.

Declaration and implementation of these functions can be done with macros; see the earlier section "Declaration of the Zend Module Block" for details.

Capítulo 39. Calling User Functions

You can call user functions from your own modules, which is very handy when implementing callbacks; for example, for array walking, searching, or simply for event-based programs.

User functions can be called with the function **call_user_function_ex()**. It requires a hash value for the function table you want to access, a pointer to an object (if you want to call a method), the function name, return value, number of arguments, argument array, and a flag indicating whether you want to perform zval separation.

```
ZEND_API int call_user_function_ex(HashTable *function_table, zval *object,
zval *function_name, zval **retval_ptr_ptr,
int param_count, zval **params[],
int no_separation);
```

Note that you don't have to specify both function_table and object; either will do. If you want to call a method, you have to supply the object that contains this method, in which case **call_user_function()**automatically sets the function table to this object's function table. Otherwise, you only need to specify function_table and can set object to `NULL`.

Usually, the default function table is the "root" function table containing all function entries. This function table is part of the compiler globals and can be accessed using the macro `CG`. To introduce the compiler globals to your function, call the macro `TSRMLS_FETCH` once.

The function name is specified in a zval container. This might be a bit surprising at first, but is quite a logical step, since most of the time you'll accept function names as parameters from calling functions within your script, which in turn are contained in zval containers again. Thus, you only have to pass your arguments through to this function. This zval must be of type `IS_STRING`.

The next argument consists of a pointer to the return value. You don't have to allocate memory for this container; the function will do so by itself. However, you have to destroy this container (using **zval_dtor()**) afterward!

Next is the parameter count as integer and an array containing all necessary parameters. The last argument specifies whether the function should perform zval separation - this should always be set to `0`. If set to `1`, the function consumes less memory but fails if any of the parameters need separation.

Ejemplo 39-1 shows a small demonstration of calling a user function. The code calls a function that's supplied to it as argument and directly passes this function's return value through as its own return value. Note the use of the constructor and destructor calls at the end - it might not be necessary to do it this way here (since they should be separate values, the assignment might be safe), but this is bulletproof.

Ejemplo 39-1. Calling user functions.

```
zval **function_name;
zval *retval;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &function_name) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

if((*function_name)->type != IS_STRING)
{
    zend_error(E_ERROR, "Function requires string argument");
```

```
}

TSRMSLS_FETCH();

if(call_user_function_ex(CG(function_table), NULL, *function_name, &retval, 0, NULL, 0) != S
{
    zend_error(E_ERROR, "Function call failed");
}

zend_printf("We have %i as type<br>", retval->type);

*return_value = *retval;
zval_copy_ctor(return_value);
zval_ptr_dtor(&retval);



<?php

dl("call_userland.so");

function test_function()
{

    print("We are in the test function!<br>");

    return("hello");

}

$return_value = call_userland("test_function");

print("Return value: \"$return_value\"<br>");
?>
```

Capítulo 40. Initialization File Support

PHP 4 features a redesigned initialization file support. It's now possible to specify default initialization entries directly in your code, read and change these values at runtime, and create message handlers for change notifications.

To create an .ini section in your own module, use the macros `PHP_INI_BEGIN()` to mark the beginning of such a section and `PHP_INI_END()` to mark its end. In between you can use `PHP_INI_ENTRY()` to create entries.

```
PHP_INI_BEGIN()
PHP_INI_ENTRY("first_ini_entry",  "has_string_value", PHP_INI_ALL, NULL)
PHP_INI_ENTRY("second_ini_entry", "2",                PHP_INI_SYSTEM, OnChangeSecond)
PHP_INI_ENTRY("third_ini_entry",  "xyz",              PHP_INI_USER, NULL)
PHP_INI_END()
```

The `PHP_INI_ENTRY()` macro accepts four parameters: the entry name, the entry value, its change permissions, and a pointer to a change-notification handler. Both entry name and value must be specified as strings, regardless of whether they really are strings or integers.

The permissions are grouped into three sections:`PHP_INI_SYSTEM` allows a change only directly in the `php3.ini` file; `PHP_INI_USER` allows a change to be overridden by a user at runtime using additional configuration files, such as `.htaccess`; and `PHP_INI_ALL` allows changes to be made without restrictions. There's also a fourth level, `PHP_INI_PERDIR`, for which we couldn't verify its behavior yet.

The fourth parameter consists of a pointer to a change-notification handler. Whenever one of these initialization entries is changed, this handler is called. Such a handler can be declared using the `PHP_INI_MH` macro:

```
PHP_INI_MH(OnChangeSecond);             // handler for ini-entry "second_ini_entry"

// specify ini-entries here

PHP_INI_MH(OnChangeSecond)
{

    zend_printf("Message caught, our ini entry has been changed to %s<br>", new_value);

    return(SUCCESS);

}
```

The new value is given to the change handler as string in the variable new_value. When looking at the definition of `PHP_INI_MH`, you actually have a few parameters to use:

```
#define PHP_INI_MH(name) int name(php_ini_entry *entry, char *new_value,
                                  uint new_value_length, void *mh_arg1,
                                  void *mh_arg2, void *mh_arg3)
```

All these definitions can be found in `php_ini.h`. Your message handler will have access to a structure that contains the full entry, the new value, its length, and three optional arguments. These optional arguments can be specified with the additional macros `PHP_INI_ENTRY1` (allowing one additional

argument), `PHP_INI_ENTRY2` (allowing two additional arguments), and `PHP_INI_ENTRY3` (allowing three additional arguments).

The change-notification handlers should be used to cache initialization entries locally for faster access or to perform certain tasks that are required if a value changes. For example, if a constant connection to a certain host is required by a module and someone changes the hostname, automatically terminate the old connection and attempt a new one.

Access to initialization entries can also be handled with the macros shown in Tabla 40-1.

Tabla 40-1. Macros to Access Initialization Entries in PHP

Macro	Description
`INI_INT(name)`	Returns the current value of entry `name` as integer (long).
`INI_FLT(name)`	Returns the current value of entry `name` as float (double).
`INI_STR(name)`	Returns the current value of entry `name` as string. *Note:* This string is not duplicated, but instead
`INI_BOOL(name)`	Returns the current value of entry `name` as Boolean (defined as zend_bool, which currently means
`INI_ORIG_INT(name)`	Returns the original value of entry `name` as integer (long).
`INI_ORIG_FLT(name)`	Returns the original value of entry `name` as float (double).
`INI_ORIG_STR(name)`	Returns the original value of entry `name` as string. Note: This string is not duplicated, but instead
`INI_ORIG_BOOL(name)`	Returns the original value of entry `name` as Boolean (defined as zend_bool, which currently mean

Finally, you have to introduce your initialization entries to PHP. This can be done in the module startup and shutdown functions, using the macros `REGISTER_INI_ENTRIES()` and `UNREGISTER_INI_ENTRIES()`:

```
ZEND_MINIT_FUNCTION(mymodule)
{

    REGISTER_INI_ENTRIES();

}

ZEND_MSHUTDOWN_FUNCTION(mymodule)
{

    UNREGISTER_INI_ENTRIES();

}
```

Capítulo 41. Where to Go from Here

You’ve learned a lot about PHP. You now know how to create dynamic loadable modules and statically linked extensions. You’ve learned how PHP and Zend deal with internal storage of variables and how you can create and access these variables. You know quite a set of tool functions that do a lot of routine tasks such as printing informational texts, automatically introducing variables to the symbol table, and so on.

Even though this chapter often had a mostly "referential" character, we hope that it gave you insight on how to start writing your own extensions. For the sake of space, we had to leave out a lot; we suggest that you take the time to study the header files and some modules (especially the ones in the `ext/standard` directory and the MySQL module, as these implement commonly known functionality). This will give you an idea of how other people have used the API functions - particularly those that didn’t make it into this chapter.

Capítulo 42. Reference: Some Configuration Macros

config.m4

The file `config.m4` is processed by `buildconf` and must contain all the instructions to be executed during configuration. For example, these can include tests for required external files, such as header files, libraries, and so on. PHP defines a set of macros that can be used in this process, the most useful of which are described in Tabla 42-1.

Tabla 42-1. M4 Macros for `config.m4`

Macro	Description
`AC_MSG_CHECKING(message)`	Prints a "checking <message>
`AC_MSG_RESULT(value)`	Gives the result to `AC_MSG_C`
`AC_MSG_ERROR(message)`	Prints message as error messa
`AC_DEFINE(name,value,description)`	Adds `#define` to `php_conf`
`AC_ADD_INCLUDE(path)`	Adds a compiler include path
`AC_ADD_LIBRARY_WITH_PATH(libraryname,librarypath)`	Specifies an additional library
`AC_ARG_WITH(modulename,description,unconditionaltest,conditionaltest)`	Quite a powerful macro, addi
`PHP_EXTENSION(modulename, [shared])`	This macro is a *must* to call fo

Capítulo 43. API Macros

A set of macros was introduced into Zend's API that simplify access to zval containers (see Tabla 43-1).

Tabla 43-1. API Macros for Accessing zval Containers

Macro	Refers to
`Z_LVAL(zval)`	(zval).value.lval
`Z_DVAL(zval)`	(zval).value.dval
`Z_STRVAL(zval)`	(zval).value.str.val
`Z_STRLEN(zval)`	(zval).value.str.len
`Z_ARRVAL(zval)`	(zval).value.ht
`Z_LVAL_P(zval)`	(*zval).value.lval
`Z_DVAL_P(zval)`	(*zval).value.dval
`Z_STRVAL_P(zval_p)`	(*zval).value.str.val
`Z_STRLEN_P(zval_p)`	(*zval).value.str.len
`Z_ARRVAL_P(zval_p)`	(*zval).value.ht
`Z_LVAL_PP(zval_pp)`	(**zval).value.lval
`Z_DVAL_PP(zval_pp)`	(**zval).value.dval
`Z_STRVAL_PP(zval_pp)`	(**zval).value.str.val
`Z_STRLEN_PP(zval_pp)`	(**zval).value.str.len
`Z_ARRVAL_PP(zval_pp)`	(**zval).value.ht