BASH

THE BOURNE-AGAIN SHELL

Angel Luis Calvo

ESCUELA IT

En esta sesión:

- Uso de parámetros
- Condicionales
- Bucles
- Casos prácticos

Angel Luis Calvo

Angel Luis Calvo

angelonx@gmail.com

ESCUELA IT

Uso de parámetros

Angel Luis Calvo

ESCUELA IT

Uso de parámetros

Los parámetros son expresiones que acompañan a la ejecución de los scripts. También se les suele llamar “argumentos de entrada”.

Por ejemplo, tenemos un script llamado mi_script.sh.
Si al ejecutarlo escribimos ./mi_script.sh hola adiós
hola y *adiós* serían dos parámetros del script. Concretamente hola sería el primer parámetro y adiós el segundo.
¿Qué sentido tiene esto? hacer los scripts más flexibles.

Uso de parámetros

Los parámetros se utilizarán dentro de los scripts refiriéndose a ellos como variables, siendo la variable $1 la que se refiere al primer parámetro, $2 al segundo y así sucesivamente.

Ejemplo:

mi_script.sh

```
#!/bin/bash
echo $1
echo $2
```

Al ejecutarlo:

```
./mi_script hola adiós
hola
adiós
```

1_

Uso de parámetros

De hecho, $0, al que ya vimos, sería un parámetro especial, que hace referencia al nombre del script en sí.
Podemos poner hasta el parámetro $9.
Si queremos más, hay que usar llaves, ${10}, por ejemplo.
Un comando relacionado es shift. Este comando elimina el primer parámetro introducido, corriendo el resto a la izquierda (convirtiendo el $1 en $0, $2 en $1, el $3 en $2, etc.)

Uso de parámetros

Otras variables relacionadas son:
$* que muestra los valores de todos los parámetros introducidos
$@ hace lo mismo, pero los trata de forma separada.
$# indica el número de parámetros introducidos.
Ejemplo:

```
echo Has introducido $# parámetros
echo Éstos: $*
```

2_

En los parámetros, como en cualquier variable, se puede incluir un valor por defecto por si no se definió antes, por ejemplo: echo ${1-*lo_que_sea*}

Ejercicio

Lo que en el tema anterior, que hicimos con read, se puede hacer ahora con esto:
Hacer un script, sumar.sh, que sirva para calcular la suma de dos números, introducidos como parámetros.

```
#!/bin/bash
echo Los números a sumar son $1 y $2
suma=$(echo "scale=2; $1+$2" | bc)
echo La suma de ambos es: $suma
```

3_

Ejemplo:
./sumar.sh 2.5 3
Los números a sumar son 2.5 y 3
La suma de ambos es: 5.5

ESCUELA IT

Uso de parámetros

Una orden que se puede empezar a considerar es exit. Nos sirve para establecer el estado de salida del script.

Los estados de salida son valores numéricos; se utilizan en los comandos para indicar posibles salidas de error de dichos comandos.

Es arbitrario, pero normalmente, todos los comandos al finalizar correctamente muestran un estado de salida de valor cero.

Uso de parámetros

Cuando en el script se ejecuta un *exit* con un determinado valor, se detiene con ese valor de salida.
Ejemplo:

```
exit 2 #Si se ejecuta esta línea, el script finaliza con estado 2
```

El estado de salida se consulta con la orden `echo $?`

Ejercicio

Script para copiar un archivo en dos ubicaciones diferentes. El primer parámetro será el archivo y las dos ubicaciones los siguientes dos parámetros.

```
#!/bin/bash
echo Se va a copiar el archivo $1 en $2 y en $3
echo En 5 segundos comenzará la copia.
echo Puede cancelar pulsando CTRL-C #esta secuencia detiene cualquier script
sleep 5
cp $1 $2 &>> copia1.log || exit 1
cp $1 $3 &>> copia2.log || echo Falló la copia 2
echo Copias finalizadas
```


BASH
THE BOURNE-AGAIN SHELL

ESCUELA IT

Condicionales

Angel Luis Calvo

ESCUELA IT

Condicionales

Muchos de los scripts realizados hasta ahora estaban un tanto constreñidos. Se ejecutaban linealmente, sin bifurcación alguna, ni saltos, lo que los hace muy limitados.

Con las órdenes condicionales se puede dotar al script de más flexibilidad y agilidad.
Una de las cosas que permitirán es controlar mejor la entrada de opciones, vigilando posibles errores de usuario.

Condicionales: if

Quizá el condicional más utilizado sea: “Si se produce una determinada situación, qué decisión tomar”

Ese planteamiento se realiza con la combinación de las sentencias *if* y *then*.
Lo que hace es evaluar la condición, si es verdadera entra por el then, si no lo es sale. Para saber hasta donde se ejecuta el condicional hay que cerrar con *fi*.
La comparación básica se indica en *if* con [(es la orden test)

Condicionales: if

Es decir, if [*condición*] es equivalente a if test *condición*.
Ojo: existe la opción de usar doble corchete [[, lo que aporta más funcionalidades, pero no está disponible en todas las distribuciones.

Muchas veces será necesario recortar las variables para manejarlas, ya sea en condicionales o en otros sitios.
Se puede hacer usando el formato ${variable:inicio:longitud}

Condicionales: if

Por ejemplo:
Siendo var=elefante
${var:4:3} → sería equivalente a “ant”
${var:1:-1} → sería equivalente a “lefant”
${var:0:2} → sería equivalente a “el”
${var:2} → sería equivalente a “efante”

Otra utilidad aprovechable para mejorar los scripts, es el formato de texto de eco.

Condicionales: if

A la orden echo con el parámetro *-e* le podemos pedir que dé un formato especial al texto de salida (color, cursiva, resaltado, etc.) pudiendo aprovechar más códigos de escape
Los formatos se establecen con la sintaxis \e[*nº*m para indicar donde empieza, y \e[0m para indicar donde termina. El nº codifica lo que se quiere aplicar.

Más info: http://manpages.ubuntu.com/manpages/focal/es/man1/echo.1.html
y en: https://robologs.net/2016/03/31/como-colorear-el-output-de-la-terminal-en-linux/

Condicionales: if

Es muy importante el uso de expresiones regulares, que ya hemos estado utilizando:
Los caracteres de escape (\ y comillas simples)
// delimitadores de expresiones
. cualquier carácter
etc.

Más info: https://www.atareao.es/tutorial/terminal/comodines-y-expresiones-regulares/#

Ejercicio

Haremos un script para borrar todos los archivos de la carpeta actual que empiecen por una determinada letra, que el usuario indicará como parámetro. Controlaremos si el usuario introduce el parámetro. Antes de eliminar esos archivos se mostrará un listado de los mismos pidiendo confirmación.

```
#!/bin/bash
#Para eliminar archivos cuya inicial se introduzca como parámetro
if [ "$1" = "" ]
        then
                echo Falta el parámetro
                echo -e "Escriba ./5_if_borrar.sh \e[3mInicial del archivo\e[0m" #ponemos códigos de cursiva
                exit 1 #Terminamos mal
fi
```

5_

------ Continúa en la siguiente -------

Ejercicio

---------- Viene de la anterior ----------

```
lista=${1:0:1} #cogemos solo el primer carácter del parámetro 1 (posición cero, longitud 1)
ls -l $lista* #Podría ser bueno comprobar que hay archivos que cumplan con esto.
echo Borrar todo? s/n
read si
if [ "$si" = "s" ]
        then
                rm $lista*
                echo Borrado
                exit 0 #Terminamos bien
fi
echo "Cancelado"
```

ESCUELA IT

Condicionales: if

La estructura if…then puede ampliarse con if…then…else (si no se cumple *then* se ejecuta *else*, y sale)
También se puede utilizar una estructura más elaborada, con varias condiciones posibles, revisadas secuencialmente con el comando elif.
Ejemplo:

```
if [ “$1” = A ]
then echo Escribiste A
elif [ “$1” = B ]
then echo Escribiste B
else echo No has escrito ni A ni B
fi
```

Condicionales: if

La condición dentro de los corchetes admite muchas cosas, tiene sus propios parámetros. Algunos de ellos:
-d → la condición es cierta si es un directorio.
-f → la condición es cierta si es un fichero.
-s → la condición es cierta si es un fichero no vacío.
Hay muchos otros parámetros:
-r, -w, -x, y alguno más para verificar los permisos.
-z y -n si se trata de una cadena de longitud nula o no nula, respectivamente.

Ejercicio

Plantearemos un script que permita conocer cuál es la puerta de enlace de nuestro equipo y que compruebe que funcione.

```
#!/bin/bash
enlace=$(ip route | grep default | cut -d " " -f 3)
#enlace=172.16.45.55 #una ip que sirva para hacer pruebas
echo La puerta de enlace es $enlace
falla=$(ping -c 1 $enlace | grep Unreachable | cut -d " " -f 6)
if [ -n “$falla” ]
        then
                echo Mensaje de vuelta: $falla
                echo “¡La puerta de enlace falla!”
        else
                echo “¡La puerta de enlace funciona!”
fi
```

6_

ESCUELA IT

Condicionales: if

También se pueden usar operadores de todo tipo:
Comparadores numéricos:
-eq, -ne, -lt, -gt (igual, no igual, menor que y mayor que)
Comparadores de cadenas:
= igual, != no igual (se puede utilizar ! para invertir otros parámetros)
Y muchos más.

Para más info:
https://linuxacademy.com/blog/linux/conditions-in-bash-scripting-if-statements/
https://www.imd.guru/sistemas/bash/operadores-de-comparacion.html

Ejercicio

Crearemos un script que copie archivos de una determinada extensión, que se indicará como primer parámetro, en una carpeta que se indicará como segundo parámetro. Si esa carpeta no existe debe crearse. Deberán controlarse los posibles errores del usuario.

```
#!/bin/bash
#Para copiar archivos cuya extensión se introduzca como parámetro
if [ $# -eq 0 ]
        then
                echo Faltan parámetros
                echo -e "Escriba .$0 \"Estensión que quiere copiar\" \"Directorio\""
                exit 1 #Terminamos mal
 elif [ "$2" = "" ]
        then
                echo Falta el segundo parámetro, el directorio
                echo -e "Escriba .$0 \"Estensión que quiere copiar\" \"Directorio\""
                exit 2 # Salimos mal
```

---------- Continúa en la siguiente----------

BASH
THE BOURNE-AGAIN SHELL

ESCUELA IT

Ejercicio

---------- Viene de la anterior ----------

```
fi
ls -l *$1 &> /dev/null
if [ $? != 0 ]
   then
        echo No hay archivos de extensión $1
        exit 3 # Salimos mal
fi
if [ -d $2 ]
  then
        echo Copiando archivos en $2
        cp -v *$1 $2
        echo Terminado
        exit 0 # Salimos bien
  else
```

---------- Continúa en la siguiente----------

BASH
THE BOURNE-AGAIN SHELL

ESCUELA IT

Ejercicio

---------- Viene de la anterior ----------

```
        echo La carpeta no existe, la crearemos.
        mkdir $2
        echo Creada
        echo Copiando archivos en $2
        cp -v *$1 $2
        echo Terminado
        exit 0 # Salimos bien
fi      #¿Este último if no es muy largo?
```

ESCUELA IT

Condicionales: if

Consideraciones:

- Dentro de los corchetes es muy crítica la sintaxis, deben respetarse escrupulosamente los espacios.
- Es conveniente escalonar los comandos para clarificar lo que hace el código.

Otro ejemplo con un script que pide contraseña.
Si usamos contraseñas, si se quiere se puede ofuscar el código.
Una utilidad para ofuscar es SHC. https://github.com/existz/shc-3.8.9

Ejercicio

Crearemos un script que copie un archivo, que se indicará como tercer parámetro, en una carpeta que se indicará como segundo parámetro. Si esa carpeta no existe debe crearse. Para que se ejecute debe pedirse una contraseña como primer parámetro.

```
if [ "$1" != miclave ]
        then
                exit 100 #Fallo de contraseña
elif [ "$2" = "" ]
        then
                echo Falta el segundo parámetro, el directorio
                echo -e "Escriba $0 \"Clave\" \"Directorio\" \"Archivos que quiere copiar\""
                exit 1 # Salimos mal
elif [ "$3" = "" ]
        then
                echo Falta el tercer parámetro, el archivo
```

8_

---------- Continúa en la siguiente----------

BASH
THE BOURNE-AGAIN SHELL

ESCUELA IT

Ejercicio

---------- Viene del anterior ----------

```
        echo -e "Escriba $0 \"Clave\" \"Directorio\" \"Archivo que quiere copiar\""
        exit 2 # Salimos mal
fi

if [ ! -f “$3” ]
   then
        echo No se encuentra el archivo $3
        exit 3 # Salimos mal
fi

if [ ! -d $2 ]
  then
        echo La carpeta no existe, la crearemos.
```

---------- Continúa en la siguiente----------

ESCUELA IT

Ejercicio

```
---------- Viene del anterior ----------
        mkdir $2
        echo Creada
fi
echo Copiando archivo
cp -v $3 $2
echo Terminado
```

ESCUELA IT

Condicionales: case

Otro condicional es case, cuya sintaxis es:

case *expresión* in
caso1) *comandos* ;;
caso2) *comandos* ;;
…
esac

La expresión indica el caso que se debe ejecutar, es cómo un pequeño menú.

Condicionales: case

Normalmente se puede sustituir por un if encadenado, pero resulta más visual y efectivo, sobre todo cuando hay muchos casos.
Ejemplo:

```
read -n 1 -p “Elige el 1 ó el 2” num #La n limita el número de caracteres
case $num in
1)    echo Pulsaste el 1 ;;
2)    echo Pulsaste el 2 ;;
*)    echo No pulsaste ni 1 ni 2 ;;
esac
```

Ejercicio

Desarrollaremos un script para que un usuario pueda consultar los archivos /var/log/auth.log y /var/log/syslog, introduciendo un criterio de búsqueda inicialmente, y luego eligiendo el archivo. Para terminar se le debe ofrecer al usuario la opción de guardar el resultado de la búsqueda.

```
#!/bin/bash
#Para consultar diversos archivos diferentes condiciones
read -p "Indique el criterio de búsqueda: " criterio
echo Para buscar en /var/log/auth.log pulse 1
echo Para buscar en /var/log/syslog pulse 2
read pulsar
case $pulsar in
 1) cat /var/log/auth.log | grep "$criterio"
        read -n 1 -p "¿Desea guardar el resultado? s/n " si
        if [ "$si" = s ]
          then
```

9_

---------- Continúa en la siguiente----------

ESCUELA IT

Ejercicio

```
---------- Viene del anterior ----------
                cat /var/log/auth.log | grep "$criterio" > consulta_$USERNAME.txt
        fi ;;
 2) cat /var/log/syslog | grep "$criterio"
        read -n 1 -p "¿Desea guardar el resultado? s/n " si
        case $si in
                [Ss]) cat /var/log/syslog | grep "$criterio" > consulta_$USERNAME.txt
        ;;
        esac
esac
echo Finalizado
```

ESCUELA IT

Bucles

Angel Luis Calvo

ESCUELA IT

Bucles

Es muy habitual que en un script se necesite ejecutar la misma acción de forma iterativa, como en un ciclo.
Es ahí donde entran en juego los bucles.
En ellos se pueden utilizar los operadores lógicos y aritméticos vistos (gt, &&, !, etc.)
Un bucle debe tener siempre una condición de salida, para evitar los llamados bucles infinitos.

Recuerde: para detener un script durante su ejecución, se puede utilizar la combinación **CTRL-C**

Bucles: for

La sintaxis principal es:

for *variable* [in *lista*] ; do
comandos
done

Ejemplo:

```
for trimestre in Enero Febrero Marzo
do
echo $trimestre
done
```

Bucles: for

Otra sintaxis puede ser:
for *variable* in {*inicio*.. *final*.. *incremento*} ; do
comandos
done

Ejemplos:

```
for num in {0..10..2}
do   #saldrán los pares
echo -n $num
done
```

```
for num in {1..5}
do
echo Escribiendo líneas repetidas
done
```

Bucles: for

Y otra sintaxis, para expresiones aritméticas es:

```
for (( expresión1; expresión2; expresión3 )) ; do
comandos
done
```

Se verifica la 1 si la 2 es cierta, luego la 3 mientras la 2 sea cierta.

Ejemplo:

```
for (( num=1; num<=5; num++ ))
do
echo $num
done
```

Ojo con el bucle infinito:

```
for ((;;))
do lo que sea
done
```

Angel Luis Calvo

ESCUELA IT

Ejercicio

Script para que se copien archivos que coincidan con un determinado patrón que se indicará como segundo parámetro. La carpeta de destino será el primer parámetro.

```
#!/bin/bash
if [ $# -eq 0 ]
        then
                echo Faltan parámetros
                echo -e "Escriba $0 \"Directorio de destino\" \"Patrón de archivo\""
                exit 1 #Terminamos mal
 elif [ "$2" = "" ]
        then
                echo Falta el segundo parámetro
                echo -e "Escriba $0 \"Directorio de destino\" \"Patrón de archivo\""
                exit 2 #Terminamos mal
fi
```


---------- Continúa en la siguiente----------

ESCUELA IT

Ejercicio

---------- Viene del anterior ----------

```
patron="*$2*"
ls $patron &> /dev/null
if [ $? != 0 ]
   then
        echo No existe ningún fichero así
        exit 3
fi

if [ ! -d $1 ]
        then
                echo La carpeta $1 no existe
                mkdir $1
                echo Creada
fi
```

---------- Continúa en la siguiente----------

BASH
THE BOURNE-AGAIN SHELL

ESCUELA IT

Ejercicio

---------- Viene del anterior ----------

```
for copi in *
  do
        cp $patron $1
done
echo Terminado
exit 0 # Esto es siempre por defecto, no es necesario
```

ESCUELA IT

Bucles: for

Hay dos órdenes que se pueden utilizar en toda clase de bucles.

break → Se interrumpe el bucle

continue → Continúa el bucle, saltándose una iteración.

Ejemplo:

```
read -p “Hasta qué número deseas contar: ” cont
for (( num=1; num<=10; num++ )); do
        echo Llegamos hasta el $num
        if [ $num -eq $cont ]; then
                break
        fi
done
```


Bucles: for

Ejemplo de *continue*:

```
for ((num = 1; num < 10; num++)); do
   if [ $num -gt 0 ] && [ $num -lt 5 ]; then
       continue # El efecto es como si retuviese la iteración
   fi               #no muestra los menores de 5
   echo $num
done
```


Bucles: for

En los bucles se puede hacer referencia de archivos externos, para extraer información o lo que se quiera:
Ejemplo, para mostrar un listado almacenado en un archivo:

```
#!/bin/bash
clear
for var in $(cat usuarios.txt); do
        echo "$var"
done
```

Bucles: while

El comando while (mientras) va ejecutando el bucle *mientras* se cumpla la condición de una determinada expresión. Pueden usarse expresiones semejantes a las del comando *if*.

Sintaxis:

```
while Expresión
do
Comandos
done
```

Ejemplo

```
while [ $# !=0 ]
do
echo Los parámetros son $1
shift
done
```

Angel Luis Calvo

Ejercicio

Script para copiar en una carpeta, introducida como primer parámetro, el archivo o archivos (pudiendo usar caracteres comodín) del segundo o más parámetros.

```
#!/bin/bash
if [ $# -eq 0 ]
        then
                echo Faltan parámetros
                echo -e "Escriba $0 \"Directorio destino\" \"Archivos a copiar\""
                exit 1 #Terminamos mal

 elif [ "$2" = "" ]
        then
                echo Falta el segundo parámetro, el archivo/s
                echo -e "Escriba $0 \"Directorio destino\" \"Archivos a copiar\""
                exit 2 # Salimos mal
fi
```


---------- Continúa en la siguiente----------

Ejercicio

---------- Viene del anterior ----------

```
carpeta=”$1”
ls $2 &> /dev/null
if [ $? != 0 ]
   then
        echo no hay archivos $2
        exit 3 #Salimos mal
 elif [ ! -d $carpeta ]
   then
        echo no existe la carpeta $carpeta
        echo Creándola
        mkdir $carpeta
fi
```

---------- Continúa en la siguiente----------

ESCUELA IT

Ejercicio

---------- Viene del anterior ----------

```
while [ $# -gt 1 ]
  do
        echo Copiando archivo
        cp -v “$2” $carpeta
        shift
done
```

ESCUELA IT

Bucles: while

Podemos usar la expresión *true*, que haría que while se verificase siempre y podría provocar un bucle infinito:
Ejemplos:

```
while true
do
sleep 1
echo pulsa CTRL-C para parar
done
```

```
while : ; do #expresión equivalente
read -p “Adivina la clave” clave
if [ $clave = “no” ] ; then
echo Acertaste ; exit
else
echo No acertaste
fi
done
```

Bucles: while

En estos bucles podemos redirigir la salida de un archivo hacia el bucle. Un ejemplo, en el que salen por pantalla las líneas de un archivo:

```
while read salida
do
echo $salida
done < usuarios.txt
```

Bucles: until

En este caso, el bucle se ejecuta *hasta* que se cumpla una condición.

Sintaxis:

```
until Expresión
do
Comandos
done
```

Ejemplo

```
until [ $# = 0 ]
do
echo Los parámetros son $1
shift
done
```

Bucles: until

También se podría provocar un bucle infinito:

```
until false
do
sleep 1
echo pulsa CTRL-C para parar
done
```

Ejercicio

Realizar con until un bucle infinito que se pare cuando un archivo de control tenga una línea con la expresión “parar”.

```
#!/bin/bash
until ! :
do
        echo Esto se para con CTRL-C
        echo O poniendo “parar” en el archivo “control.txt”
        if [ $(cat control.txt) = “parar” ]
           then
                echo Por fin parar
                exit
        fi
done
```


Casos prácticos

Angel Luis Calvo

ESCUELA IT

Casos prácticos

Ya hemos visto muchos comandos que nos permiten operar con cadenas de caracteres, archivos y variables (como grep, cat, cut,...)
Esos comandos incorporan parámetros para operaciones más avanzadas, e incluso comandos aparentemente sencillos, como ls, también
Hay otros comandos interesantes como wc.
wc nos informa del número de líneas, palabras y bytes de un archivo:
wc /etc/passwd
46 79 2728 -etc/passwd → en este caso da una idea del nº de usuarios

Casos prácticos

Otro comando es sed, que nos permitirá cambiar unas palabras por otras dentro de archivos, o eliminar líneas, por ejemplo. Ojo, hay que usar el parámetro -i para que lo que se muestre se aplique en el archivo, el comando no pretende modificar, solo mostrar.
Eliminación de la 2ª línea: sed '2d' fichero.txt
Eliminación permanente de las líneas que contengan la cadena *hola*:
sed -i '/hola/d' fichero.txt

Ejercicio

Script que copia todos los archivos del directorio actual en varias carpetas, repartiéndose entre ellas.
Las carpetas están creadas previamente (carpeta1,..., carpeta10)
Controlar que se avise cuando haya más de 100 archivos por copiar.

```
#!/bin/bash
#Queremos copiar los archivos del directorio actual de 10 en 10 en diferentes sitios
#Primero obtenemos el listado de archivos a copiar.
ls -p | grep -v / > listado.borrable #-p añade símbolo / a los directorios -v filtra por ese símbolo
cont=$(wc listado.borrable | cut -d " " -f 2) #para saber el nº de archivos
if [ $cont -gt 100 ]
   then
        echo Hay más de 100 ficheros, solo se copiarán los primeros 100.
        read -n 1 -p "¿Desea continuar? s/n " si
        if [[ $si != [sS] ]]; then
        echo Anulado
        exit
```

---------- Continúa en la siguiente----------

Ejercicio

```
---------- Viene del anterior ----------
        fi
fi
#El while dura mientras haya archivos que copiar
while [ -s listado.borrable ]; do #En lugar de esto, es mejor un if dentro del for
  for num in {1..10}
  do
        echo copiando
        copias=$(head -n 1 listado.borrable)
        #A medida de que se va copiando uno, se elimina de la lista
        sed -i '1d' listado.borrable
        cp  -v "$copias" "carpeta$num"
  done
done
echo Terminado
```

ESCUELA IT

Casos prácticos

En el script anterior, para crear las carpetas se podría haber hecho un pequeño script, o incluir su código, así:
Estableciendo un índice para hacerlo decrementar, por probar con *until*

```
i=10
until [ $i -lt 1 ]
do
   mkdir carpeta$i
   ((i--))
done
```


Casos prácticos

Otro comando muy usado es tail, “gemelo” a head, para ver las últimas líneas de un archivo.
De gran utilidad es que podemos visualizar, en tiempo real con el parámetro -f, el “movimiento” de un archivo log, por ejemplo:
tail -f /var/log/syslog
(Ejecutar y probar haciendo logger hola)

Casos prácticos

Existen comandos más complejos:
awk → es un lenguaje de script en sí mismo, podemos aprovecharlo para procesar patrones sobre líneas de texto.
Ejemplos:
ls -l | awk '{ print $5 }' #nos muestra la 5ª columna de la salida, los bytes
awk '{ print $2, $1 }' *fichero* #muestra las columnas en otro orden
awk 'length > 60' *fichero* #muestra solo las líneas de más de 60 caracteres
awk '/^wiki/ {print $0}' *fichero* #muestra las líneas que empiezan con ese patrón (aprovechando la expresión regular con ^)

Más info:
https://www.gnu.org/software/gawk/manual/gawk.html

Casos prácticos

Con el comando xargs pasa algo parecido.
Suele utilizarse para procesar la salida de otros comandos, como una tubería. A veces una tubería simple no cumple con lo que se necesita, sobre todo cuando el número de argumentos es muy elevado.
Ejemplos:
ls * | xargs wc #muestra estadísticas por cada archivo, sin atascar el buffer de ls, ya que xargs usará un proceso diferente para cada argumento.
cat lista.txt | xargs -I% bash -c 'touch %; echo Creado %' #crea con touch un archivo con cada nombre almacenado en lista.txt, usando un subshell.

Más info:
http://man7.org/linux/man-pages/man1/xargs.1.html

Ejercicios

1.- Script para que se copien archivos que coincidan con una lista de nombres almacenados en un archivo. La carpeta de destino será un parámetro. En caso de no poner el parámetro se debe solicitar al usuario que lo ponga o terminar la ejecución.

16_

2.- Script para que se muestren por pantalla, y se almacenen en un archivo, los mensajes de /var/log/syslog que contienen warnings del mes indicado como parámetro.

17_

3.- Realizar un script que busque entre los scripts del directorio del curso, aquellos que tengan una determinada expresión, introducida como parámetro. Esos archivos deberán ser copiados a la carpeta “seleccionados” en el escritorio.

18_

ESCUELA IT

1.-

```
#!/bin/bash
#Para copiar archivos de una lista de nombres
#en una carpeta que ya exista
if [ $# -eq 0 ]
        then
        echo Falta el parámetro
        read -p "¿Desea introducirlo ahora? s/n " si
        if [ "$si" = s ]
        then
                read -p "Bien, introdúzcalo ahora: " kk
                set $1 $kk
        else
                echo Adiós
                exit 1
        fi
fi
```

---------- Continúa en la siguiente----------

ESCUELA IT

---------- Viene del anterior ----------

```
while read orig
do
        cp -v "$orig" "$1"/
done < archivos.txt

echo Terminado
```

ESCUELA IT

2.-

```
#!/bin/bash
#Para copiar los warnings de syslog
if [ $# -eq 0 ]
        then
        echo Falta el parámetro
        echo Ejecute de nuevo $0 indicando el Mes de búsqueda
        exit 1
fi

sel=$(grep warning /var/log/syslog | grep "$1")

echo -e "$sel \n"
echo -e "$sel \n" > warnings.txt
# el \n es para introducir intro
```

ESCUELA IT

3.-

```
#!/bin/bash
#Para copiar los sh de la carpeta actual
# buscando los que cumplan un criterio
if [ $# -eq 0 ]
        then
        echo Falta el parámetro
        echo Ejecute de nuevo $0 indicando el criterio de búsqueda
        exit 1
fi

#uniq, es un comando para eliminar repeticiones en las cadenas de caracteres
busco=$(grep "$1" ~/curso/*sh | cut -d ":" -f 1 | uniq)
#Ojo, no poner entre comillas $busco
for i in $busco
do
        ---------- Continúa en la siguiente----------
```

---------- Viene del anterior ----------

```
        #con -p no da error si ya existe el directorio
        mkdir -p /home/$USERNAME/Escritorio/seleccionados
        cp -v "$i" /home/$USERNAME/Escritorio/seleccionados
done
echo Terminado
exit
```

ESCUELA IT

BASH

THE BOURNE-AGAIN SHELL

GRACIAS!!

Angel Luis Calvo

ESCUELA IT