14.3.—Asociar scripts IDC con los atajos de teclado

En ocasiones podemos haber desarrollado un script tan útil que quisiéramos ejecutarlo con una o dos acciones. Llegado este caso, quisiéramos asignarle un atajo de teclado para activarlo rápidamente. Por fortuna IDA nos proporciona una forma simple para realizarlo. Cada vez que IDA es ejecutado, el script almacenado en **Archivos de programa\Ida\idc\ida.idc** se ejecuta. La versión por defecto de este script contiene una función **main** vacía que no realiza nada. Para asociar un atajo de teclado a uno de nuestros scripts, necesitaremos añadir dos líneas a **ida.idc**. La primera línea que se debe añadir es una directiva **include** la cual incluye nuestro archivo script en **ida.idc**. La segunda línea que se debe añadir es una llamada, dentro de **main** a la función **AddHotkey** la cual asocia nuestra atajo específico con nuestro maravilloso script. Por ejemplo, lo anterior nos mostraría **ida.idc** de esta forma:

```
#include <idc.idc>
#include <mi_script_maravilloso.idc>
static main () {
    AddHotkey ("z", "MiMaravillosaFunción");    //Ahora 'z' llamará a MiMaravillosaFunción
}
```

Si el atajo de teclado que queremos adjuntar a nuestro script, ya está asignado a otra acción de IDA, **AddHotkey** fallará sin que nos lo indique, la única forma de saberlo será cuando probemos nuestro atajo y veamos que no funciona.

Dos puntos importantes a tener en cuenta, uno el directorio estandar para nuestros scripts IDC es **Archivos de programa\Ida\idc** y otro que no debemos poner nombre a nuestra función **main**. Si queremos que IDA encuentre rápidamente nuestro script, podemos copiarlo dentro de la carpeta **idc**. Si pensamos dejarlo en otra ubicación, necesitaremos especificar la ruta completa de nuestro script en la declaración **include**. Para verificar nuestro script, deberemos ejecutarlo como un programa normal con una función **main**. Una vez funcione asociaremos nuestro script al atajo de teclado, sin embargo, no podremos utilizar el nombre **main**, ya que entraría en conflicto con la función **main** de **ida.idc**. Por lo tanto deberemos renombrar nuestra función **main** y utilizar el nombre nuevo en la llamada a **AddHotkey.**

14.4.—Utilizar funciones IDC

Llegados a este punto, tenemos toda la información necesaria para programar scripts IDC. Lo que nos falta es la habilidad para ejecutar cualquier interacción útil con IDA. IDC nos proporciona una gran lista de funciones las cuales nos ofrecen distintas formas de acceder a la base de datos. Todas estas funciones están documentadas en cierto grado en el IDA help bajo el título **Index of IDC functions**. En la mayoría de los casos la documentación no es más que la copia de las líneas más relevantes del archivo **idc.idc**. Con lo cual dicha información es muy débil para poder aprender IDC. Por regla general no hay ninguna forma fácil de poder responder a una pregunta tal ¿Cómo le hago hacer a **X** algo en IDC? La forma más común para deducir como poder hacer algo en IDC, es examinar su lista de funciones y buscar lo que basado en su nombre parezca realizar lo que necesitamos. Esto es así presumiendo, por supuesto, que las funciones estén nombradas de acuerdo a nuestro propósito, lo cual no siempre será obvio. Por ejemplo, en muchos casos, las funciones que recuperan información de la base de datos son nombradas

como **GetXXX**; sin embargo; en muchos otros casos, el prefijo **Get** no se utiliza. Las funciones para cambiar la base de datos son nombradas como **SetXXX**, **MakeXXX** o algo totalmente distinto. En resumen, si queremos utilizar IDC, nos tenemos que acostumbrar a examinar la lista de funciones y leer sus descripciones. Por otra parte si te encuentras totalmente perdido, pues no dudes de utilizar los foros en Internet entre ellos el de Hex-Rays.

En el resto de este escrito, se intentará señalar algunas de las funciones más útiles de Idc agrupadas por sus funcionalidades.

14.4.1.—Funciones para leer y modificar datos
Las siguientes funciones proporcionan acceso a **bytes, words** y **double words** de una base de datos.

long Byte (long addr)
Lee el valor de un byte de la dirección virtual **addr**.

long Word (long addr)
Lee el valor de un Word (2-byte) de la dirección virtual **addr**.

long Dword (long addr)
Lee el valor de un double Word (4-byte) de la dirección virtual **addr**.

void PatchByte (long addr, long val)
Coloca al valor de un byte en la dirección virtual **addr**.

void PatchWord (long addr, long val)
Coloca el valor de un Word en la dirección virtual **addr**.

void PatchDword (long addr, long val)
Coloca el valor de un double Word en la dirección virtual **addr**.

bool isLoaded (long addr)
Retorna **1** si **addr** contiene un dato válido y **0** si no es así.

Cada una de estas funciones toma el orden de byte (**little-endian o big-endian**) del actual módulo de procesador al leer o escribir en la base de datos. Las funciones **PatchXXX** arreglan el valor reemplazado a un tamaño apropiado según la función llamada utilizando solamente los bytes de orden bajo. Por ejemplo, una llamada **PatchByte (0x401010, 0x1234)** modificaría la ubicación **0x401010** con el byte de valor **0x34** (el byte de orden bajo de 0x1234). Si se suministra una dirección errónea mientras se está leyendo la base de datos con **Byte, Word y Dword**, se retornaran los valores **0xFF, 0xFFFF y 0xFFFFFFFF** respectivamente. Debido a que no hay forma de distinguir estos errores de los datos correctos almacenados en la base de datos, podemos llamar a **isLoaded** para determinar si una dirección de la base de datos contiene algún dato, antes de intentar leer esta dirección.

Debido a alguna cosa rara en IDA al refrescar la vista de desensamblado, podemos encontrarnos que el resultado de la operación de parcheo no se vea inmediatamente.

En dichos casos desplazándonos lejos de la ubicación del parcheo y volviendo luego a dicha ubicación veremos que se ha actualizado correctamente.

14.4.2.—Utilizar funciones interactivas

A fin de poder ejecutar una interacción total, es necesario familiarizarse con las funciones IDC de entrada y salida. Algunas de estas funciones más utilizadas son las siguientes:

void Message (string format, …)
Imprime un mensaje formateado en la ventana de mensajes. Esta función es análoga a la función de C **printf** y acepta el mismo formato de cadena que **printf.**

void Warning (string format, …)
Muestra un mensaje formateado en un diálogo.

string AskStr (string default, string prompt)
Muestra un diálogo de entrada preguntando al usuario si quiere introducir un valor de cadena. Retorna la cadena del usuario o **0** si el diálogo ha sido cancelado.

string AskFile (long doSave, string mask, string prompt)
Muestra un diálogo para seleccionar un archivo simplificando la tarea de elegir un archivo. Pueden crearse nuevos archivos para guardar sus datos (**doSave** = 1), o elegir los ya existentes para leer los datos (**doSave** = 0). La lista de archivos puede ser filtrada según indica **mask** (como *.* o *.idc). Retorna el nombre del archivo seleccionado o **0** si el diálogo es cancelado.

long AskYN (long default, string prompt)
Recuerda al usuario con una pregunta de **yes** o **no**, la respuesta puede ser (**1 = yes, 0 = no, -1 = cancel**). Retorna un entero representando la respuesta seleccionada.

long ScreenEA ( )
Retorna la dirección virtual de la posición actual del cursor.

bool Jump (long addr)
Salta a la dirección especificada en la ventana de desensamblado.

Debido a que IDC carece de cualquier facilidad para la depuración, podemos encontrar en la función **Message** la principal herramienta de depuración. Otras funciones tipo **AskXXX** existen para manejar de manera especial las entradas, como puede ser la entrada de un entero. Para una lista completa de dichas funciones tipo **AskXXX** deberemos referirnos a la ayuda de IDA. La función **ScreenEA** es muy útil para recoger la posición actual del cursor cuando deseamos crear un script que corta y pega y su comportamiento está basado en la ubicación del cursor. De forma similar, la función **Jump** es utilizada cuando tenemos un script que necesite llamar la atención del usuario en alguna ubicación específica dentro del desensamblado.

14.4.3.—Funciones para manipulación de strings.

Aunque la asignación y concatenación simple de una **string** en IDC está a cargo de operadores básicos, las operaciones más complejas se deben ejecutar utilizando

funciones disponibles para manipulación de **strings**, algunas las detallamos seguidamente:

string form (string format, …)
Retorna una nueva **string** formateada de acuerdo al formato y valores suministrados. Esta función equivale a **sprintf** en C.

long atol (string val)
Convierte el valor decimal **val** a su correspondiente representación entera.

long xtol (string val)
Convierte el valor hexadecimal **val** (opcionalmente puede empezar con **0x**) a su correspondiente representación decimal.

string ltoa (long val, long radix)
Retorna como **string** la representación de **val** con el **radix** especificado (2,8,10,16)

long ord (string ch)
Retorna el valor ASCII, de un carácter **ch**.

long strlen (string str)
Retorna la longitud de la string proporcionada **str**.

long strstr (string str, string substr)
Retorna el índice **substr** de **str** o **-1** si la substring no existe.

string substr (string str, long start, long end)
Retorna una substring, conteniendo los caracteres de **str**, desde **start** hasta **end-1**.

Si queremos iterar de carácter en carácter dentro de una cadena, deberemos tomar sucesivas subcadenas de un carácter, por cada carácter de la cadena.

14.4.4.—Funciones de archivo Input/Output
La ventana de mensajes de Ida no siempre será ideal para enviar las salidas de nuestros scripts. Para scripts que generan una gran cantidad de texto o scripts que generan datos binarios, la mejor salida es al disco. Ya hemos hablado de utilizar la función **AskFile** como un medio de preguntar al usuario sobre el nombre de un archivo. Sin embargo, **AskFile** retorna solamente un valor cadena conteniendo el nombre del archivo. Las funciones para tratar con archivos las detallamos seguidamente:

long fopen (string filename, string mode)
Retorna un manejador del archivo (**handle**) o **0** si hay error, para utilizar con todas las funciones de archivo **I/O** de IDC. El parámetro **mode** es similar a los modos utilizados en C por **fopen** (**r** para leer, **w** para escribir, y demás).

void fclose (long handle)
Cierra el archivo abierto por **fopen**, que su manejador coincida con **handle**.

long filelength (long handle)
Retorna el largo del archivo indicado o **-1** si da error.

long fgetc (long handle)
Lee un byte del archivo dado. Retorna **-1** si hay error.

long fputc (long val, long handle)
Escribe un byte en el archivo dado. Retorna **0** si ocurre o **-1** si hay error.

long fprintf (long handle, string format, …)
Escribe una **string** formateada al archivo dado.

long writestr (long handle, string str)
Escribe la **string** especificada al archivo dado.

string/long readstr (long handle)
Lee una **string** del archivo dado. Esta función lee todos los caracteres, incluidos los que no son ASCII hasta e incluyendo a éste, el carácter de siguiente línea (**ASCII 0xA**). Retorna la **string** si ocurre o **-1** al llegar al final del archivo.

long writelong (long handle, long val, long bigendian)
Escribe un entero de 4-byte al archivo dado utilizando uno de los siguientes órden de byte, big-endian (**bigendian =1**) o little-endian (**bigendian = 0**).

long readlong (long handle, long bigendian)
Lee un entero de 4-byte del archivo dado utilizando uno de los siguientes orden de byte, big-endian (**bigendian = 1**) o little-endian (**bigendian = 0**).

long writeshort (long handle, long val, long bigendian)
Escribe un entero de 2-byte del archivo dado utilizando uno de los siguientes orden de byte, big-endian (**bigendian = 1**) o little-endian (**bigendian = 0**).

long readshort (long handle, long bigendian)
Lee un entero de 2-byte del archivo dado utilizando uno de los siguientes orden de byte, big-endian (**bigendian = 1**) o little-endian (**bigendian = 0**).

bool loadfile (long handle, long pos, long addr, long length)
Lee **length** número de bytes desde la posición pos del archivo dado y escribe estos bytes en la base de datos iniciando su escritura en la dirección **addr**.

bool savefile (long handle, long pos, long addr, long length)
Escribe **length** numero de bytes, desde la base de datos al archivo dado, desde la dirección **addr** hasta la posición **pos**.

14.4.5,-- Manipulación de los “Names” de la base de datos
La necesidad de manipular ubicaciones nombradas sucede bastante a menudo en los scripts. Las funciones siguientes IDC, son aptas para trabajar con ubicaciones nombradas de una base de datos de IDA.

string Name (long addr)
Retorna el nombre asociado a la dirección dada, o retorna una **string** vacía si dicha ubicación no tiene nombre. Esta función no retornará nombres asignados por el usuario cuando los nombres están marcados como **local**.

string Names (long from, long addr)
Retorna el nombre asociado con **addr**. Retorna una **string** vacía si la ubicación no tiene nombre. Esta función retorna nombres **local** definidos por el usuario si **from** es cualquier dirección dentro de una función que contenga la dirección **addr**.

bool MakeNameEx (long addr, string name, long flags)
Asigna el nombre dado a la dirección dada. El nombre es creado con atributos especificados en la máscara de bit **flags.** Estos **flags** están descritos en la documentación del archivo de ayuda en **MakeNameEx** y se utilizan para especificar atributos como si el nombre es **local** o **public** o si debe ser listado en la ventana **Names**.

long LocByName (string name)
Retorna la dirección de la ubicación con el nombre dado. Retorna **BADADDR (-1)** si tal nombre no existe en la base de datos.

long LocByNameEx (long funcaddr, string localname)
Busca en el archivo dado el **localname** dentro de la función contenida en **funcaddr**. Retorna **BADADDR (-1)** si tal nombre no existe en la base de datos.

14.4.6.—Funciones que trabajan con funciones
Algunos scripts están diseñados para ejecutar análisis de funciones dentro de la base de datos. IDA asigna a las funciones desensambladas un cierto número de atributos, como el tamaño del área de la variable local de la función o el tamaño de la pila para los argumentos de la función. Las funciones IDC siguientes pueden utilizarse para acceder a la información de las funciones dentro de la base de datos.

long GetFunctionAttr (long addr, long attrib)
Retorna el atributo pedido de la función ubicada en la dirección dada. Para la lista de atributos constantes tenemos que referirnos a la documentación de ayuda IDC. Como ejemplo, para hallar la dirección final de una función, utilizamos **GetFunctionAttr (addr, FUNCATTR_END);**

string GetFunctionName (long addr)
Retorna el nombre de la función la cual contiene la dirección dada o una **string** vacía si la dirección dada no pertenece a una función.

long NextFunction (long addr)
Retorna la dirección de inicio de la siguiente función que sigue a la dirección dada. Retorna **-1** si no existen más funciones, después de esta, en la base de datos.

long PrevFunction (long addr)
Retorna la dirección de inicio de la función anterior a la dirección dada. Retorna **-1** si no existe una función anterior a la dirección dada.

Utilizaremos la función **Name** para hallar la dirección de inicio de una función dada por el nombre de la función.

14.4.7.—Funciones para referencias cruzadas de código
De las referencias cruzadas ya hablamos en el escrito 8. IDC nos proporciona funciones para acceder a la información de las referencias cruzadas asociadas a cualquier instrucción. Decidir qué función realizará mejor nuestras necesidades con respecto al script puede ser un poco desconcertante. Requiere comprender si nos interesa seguir el flujo de una dirección dada o si nos interesa iterar sobre todas las ubicaciones que hagan referencia a la dirección dada. Funciones que realizan los dos tipos de tares las vamos a describir seguidamente. Varias de estas funciones están diseñadas para soportar la iteración sobre un conjunto de referencias cruzadas. Tales funciones soportan la noción de una secuencia de referencias cruzadas y requiere ejecutar un **current** de referencia cruzada con el fin de retornar un **next** de referencia cruzada.

long Rfirst (long from)
Retorna la primera ubicación a la que la dirección dada transfiere el control. Retorna **BADADDR (-1)** si la dirección dada no tiene referencia a otra dirección.

long Rnext (long from, long current)
Retorna la siguiente ubicación que transfiere el control de la dirección dada **from**, suponiendo que **current** se haya retornado por una llamada previa a **Rfirst o Rnext**. Retorna **BADADDR** si no existen más referencias.

long XrefType ( )
Retorna una constante indicando el tipo de la última referencia cruzada retornada por una función de búsqueda a referencia cruzada como **Rfirst**. Para las referencias cruzadas de código, estas constantes son **fl_CN** (llamada cercana), **fl_CF** (llamada lejana), **fl_JN** (salto cercano), **fl_JF** (salto lejano) y **fl_F** (flujo secuencias ordinario).

long RfirstB (long to)
Retorna la primera ubicación a la cual trasfiere el control la dirección dada. Retorna **BADADDR (-1)** si no existe ninguna referencia a la dirección dada.

long RnextB (long to, long current)
Retorna la siguiente ubicación a la cual transfiere el control la dirección dada **to**, suponiendo que **current** haya sido retornado por una llamada previa a **RfirstB** o **RnextB.** Retorna **BADADDR** si no existen más referencias cruzadas a la ubicación dada.

Cada vez que es llamada una función de referencias cruzadas, una variable de estado interna de IDC se habilita para indicar el tipo de la última referencia cruzada que ha sido retornada. Si necesitamos conocer el tipo de la referencia cruzada que hemos recibido, deberemos llamar a **XrefType** antes de llamar a otra función de mirada a la referencia cruzada.

14.4.8.—Funciones para referencias cruzadas de datos
Las funciones para acceder a la información de las referencias cruzadas de datos son muy similares a las funciones utilizadas para acceder a la información de las referencias cruzadas de código. Estas funciones las describimos seguidamente:

long Dfirst (long from)
Retorna la primera ubicación con valor de dato referenciada a la dirección dada. Retorna **BADADDR (-1)** si la dirección dada no está referenciada por otras direcciones.

long Dnext (long from, long current)
Retorna la siguiente ubicación con valor de dato referida a la dirección dada **from,** suponiendo que **current** haya sido retornado por una llamada previa a **Dfirst** o **Dnext**. Retorna **BADADDR** si no existen más referencias cruzadas.

long XrefType ( )
Retorna una constante indicando el tipo de la última referencia cruzada retornada por una función de búsqueda a la referencia cruzada como **Dfirst**. Para las referencias cruzadas de datos, estas constantes son **dr_O** (toma offset), **dr_W** (escribe dato) y **dr_R** (lee dato).

long DfirstB (long to)
Retorna la primera ubicación dato la cual refiere a la dirección dada. Retorna **BADADDR (-1)** si no existe ninguna referencia a la dirección dada.

long DnextB (long to, long current)
Retorna la siguiente ubicación como dato que refiere a la dirección dada **to**, suponiendo que **current** haya sido retornado por una llamada previa a **DfirstB** o **DnextB**. Retorna **BADADDR** si no existen más referencias a la ubicación dada.

Como con las referencias cruzadas de código, si necesitamos conocer el tipo de la referencia cruzada que hemos recibido, deberemos llamar a **XrefType** antes de llamar a otra función de mirada a la referencia cruzada.

14.4.9.—Funciones para la manipulación de la base de datos.
Existen varias funciones para formatear el contenido de una base de datos. Unas cuantas de estas se describen seguidamente:

void MakeUnkn (long addr, long flags)
Interpreta el elemento de la dirección especificada. Los **flags**, ver documentación de IDC para **MakeUnkn**, dictan si los elementos siguientes también serán interpretados y si todos los nombres asociados con los elementos interpretados serán borrados.

long MakeCode (long addr)
Convierte los byte de la dirección especificada en una instrucción. Retorna la longitud de la instrucción o **0** si la operación falla.

bool MakeByte (long addr)
Convierte el elemento de la dirección especificada en un byte de dato. **MakeWord** y **MakeDword** también están disponibles.

bool MakeComm (long addr, string comment)
Añade un comentario ordinario a la dirección dada.

bool Make Function (long begin, long end)
Convierte el rango de instrucciones desde **begin** hasta **end** en una función. Si **end** se especifica como **BADADDR (-1)**, IDA intentará de forma automática identificar el fin de la función localizando la instrucción de retorno de función.

bool MakeStr (long begin, long end)
Crea una **string** del tipo de la cadena actual, retornada por **GetStringType**, con los byte desde **begin** hasta **end – 1**. Si **end** es especificado como **BADADDR**, IDA intentará automáticamente identificar el fin de la **string**.

Existen muchas otras funciones tipo **MakeXXX** que ofrecen un comportamiento similar a las funciones descritas. Si quieres la lista completa búscala en la documentación IDC.

14.4.10.—Funciones de búsqueda en la base de datos
La mayoría de capacidades de búsqueda de IDA son accesibles a través de varias funciones tipo **FindXXX** de IDC, algunas las describiremos a continuación. El parámetro **flags** es utilizado en las funciones **FindXXX** como una máscara de bit para especificar el comportamiento de la operación buscada. Tres de los **flags** más utilizados son **SEARCH_DOWN**, produce la búsqueda empezando a examinar las direcciones más altas; **SEARCH_NEXT**, la cual salta de la ocurrencia actual a la siguiente ocurrencia; y **SEARCH_CASE**, produce la búsqueda de texto en el binario ejecutándose dentro del marco de casos.

long FindCode (long addr, long flags)
Búsqueda de una instrucción en la dirección dada.

long FindData (long addr, long flags)
Búsqueda de un elemento dato en la dirección dada.

long FindBinary (long addr, long flags, string binary)
Búsqueda de una sucesión de byte en la dirección dada. La cadena **binary** especifica una secuencia de byte con valores hexadecimales. Si **SEARCH_CASE** no se especifica y los byte especifican valores de letras ASCII en mayúsculas o minúsculas, entonces la búsqueda también confrontará los valores correspondientes a cada caso. Por ejemplo, **“41 42”** coincidirá con **“61 62”** a menos que esté habilitada la bandera (flag) **SEARCH_CASE**.

long FindText (long addr, long flags, long row, long columna, string text)
Búsqueda de una cadena **text** de la **column** dada en la **row** dada en la dirección dada. Observemos que el texto del desensamblado en la dirección dada puede tener varias líneas, por lo tanto es necesario especificar en qué línea debería empezar la búsqueda.

Observemos también que **SEARCH_NEXT** no define la dirección de búsqueda, la cual puede ser hacía arriba (direcciones superiores) o hacia abajo (direcciones

inferiores) de acuerdo con la bandera **SEARCH_DOWN**. Además cuando no se especifica **SEARCH_NEXT**, es perfectamente razonable que una función **FindXXX** retorne la misma dirección que ha sido pasada como argumento **addr** cuando el elemento en **addr** cumple los requisitos de búsqueda.

14.4.11.—Componentes de la línea de desensamblado
A veces, es útil extraer el texto o partes de texto, de las líneas de un listado de desensamblado. Las siguientes funciones proporcionan acceso a distintos componentes de las líneas de desensamblado:

string GetDisasm (long addr)
Retorna el texto del desensamblado de la dirección dada. El texto retornado incluye cualquier comentario pero no información de la dirección.

string GetMnem (long addr)
Retorna la parte mnemónica de la instrucción de la dirección dada.

string GetOpnd (long addr, long opnum)
Retorna el texto representando al operando especificado de la dirección dada. Los operandos son numerados desde cero empezando por el de más a la izquierda.

long GetOpType (long addr, long opnum)
Retorna un entero representando el tipo del operando dado en la dirección dada. Tendremos que referirnos a la documentación IDC respecto a **GetOpType** para tener una lista completa de los códigos de tipo de operando.

long GetOperandValue (long addr, long opnum)
Retorna el valor entero asociado con el operando dado en la dirección dada. La naturaleza del valor retornado dependerá del tipo del operando dado, especificado por **GetOpType.**

string CommentEx (long addr, long type)
Retorna el texto de cualquier comentario presente en la dirección dada. Si **type** es **0**, se retorna el texto de un comentario ordinario. Si **type** es **1**, se retorna un comentario repetitivo. Si no existe ningún comentario en la dirección dada, se retorna una cadena vacía.

Performance Bigundill@