VIII

Traduciendo…

Puesto que lo más común es utilizar roms en inglés, es importante aprender a traducir. No obstante, también habrá que cambiar parte de los textos aun estando en español si escogemos una rom de dicho idioma. Sea como sea, tarde o temprano deberemos cambiar algún texto. Esto lo podemos hacer de muchas formas y aquí explicaremos tres de sus procedimientos. En este tema aprenderemos a traducir con Advance-Text, XSE y con un editor hexadecimal. Para esta última, aprenderemos a repointear los offsets de los textos.

·Traducir con Advance-Text

Es la manera más rápida y sencilla de traducir pero la de menos calidad al tratarse de textos en español. Nos podrá dar problemas con tildes u otros símbolos no utilizados en inglés. Pero para lo que pueda servir es conveniente usarla. Como siempre y por si las moscas, es recomendable hacer una copia de seguridad de nuestro hack antes de empezar a traducir. Empecemos.

Lo primero es abrir el programa. Es posible que nos de error diciendo que necesitamos un componente y nos impida abrirlo. Para arreglarlo, simplemente buscad el componente que pide en Google y no habrá problemas en descargarlo rápidamente. Cuando lo tengáis, simplemente debéis pegarlo en la carpeta *System32*. Con eso debería bastar. En caso de no servir, buscad soporte en algún foro dedicado al rom hacking de pokémon.

Una vez abierto el programa, cargamos el rom. Dependiendo de la versión puede que tengas textos ya guardados preparados para ser editados. Pero es importante aprender a hacerlo, así que vamos a llevar a cabo todo el procedimiento. Para ponerlo a modo de ejemplo, vamos a editar la primera frase que nos dice Oak en el rom *Pokémon Fire Red USA*:

Hello, there!
Glad to meet you!

Necesitamos la frase exacta para poder encontrarla usando el buscador de Advance-Text. Vamos a *Search* y se abrirá una ventana. Ahí escribimos parte del texto. Vamos a poner "Hello" y le damos a *Search* para buscar. Si todo sale como debería, en la parte derecha saldrá un texto que no es el que buscamos. Si le damos a *Resume* se buscará el siguiente texto. Pero mejor

probemos, para más exactitud, con la segunda frase: "Glad to meet you!". Esta vez si se encuentra el texto, pero no desde el principio, así que seleccionamos la opción *Go to beginning text* para que se busque el texto completo. Ahora volvemos a buscar. Debería ir bien y salir todo el texto. Pero Advance-Text puede fallar y no salir. En ese caso, busca todo el texto poniendo:

Hello, there!
Glad to meet you!

Lo importante, sea como sea, es localizar el texto desde que empieza y no a la mitad. Debemos editar el texto completo y no una parte. Localizado el texto, vamos a editarlo dándole a *Write to INI*. Ahora pondremos en *Section name* el nombre de nuestra categoría para crear una nueva o seleccionamos debajo una ya existente para guardarlo en dicha categoría. Luego vamos a *Text name* y le ponemos un nombre a nuestro texto. Esto es una forma de ordenar los textos para luego poder editarlos de forma rápida y sencilla. En este ejemplo, crearemos una categoría llamada *introducción* y el texto se llamará *texto 1*. Ahora cerramos la ventana y volvemos a la ventana principal del programa. En el menú buscamos nuestra categoría y seleccionamos nuestro texto. El texto que estamos usando para el ejemplo y el que deberíamos haber guardado es este:

Hello, there!
Glad to meet you!

Welcome to the world of POKéMON!

My name is OAK.

People affectionately refer to me
as the POKéMON PROFESSOR.

Los saltos dobles de linea representan lo mismo que si en XSE pusieramos un \p. Pero nosotros editaremos las dos primeras frases (las que buscamos) para hacer aprender. Las cambiamos por otras dos frases poniendo lo que queramos. Cuando acabemos, nos fijamos en la frase que hay debajo de la caja del texto. Podrá poner dos cosas:

1. **X Characters left until repoint:** Quiere decir que el texto es igual (si X es 0) de largo al texto original o que es menor (si X es mayor que 0). En este caso, el texto se podrá guardar en el espacio original ya que será igual o menor y no sobrescribirá ningún dato.

2. **X Characters to repoint:** Quiere decir que el texto es más grande que el original y que necesitará ser repointeado. Es decir, tendrá que buscar un lugar en el rom con suficiente espacio libre como para guardar el texto sin sobrescribir nada. Por suerte, la herramienta lo hace por nosotros, así que no tenemos que preocuparnos por nada.

Cuando tengamos el texto cambiado le damos a *Save text*. Y ya está. El texto se edita. Pero debemos aclarar algo. Advance-Text no es una herramienta que destaque por sus pocos errores. Cuando el texto sea igual o menor al original, no habrá problemas. Pero cuando sea más grande tendrá que repointear y hay veces que falla. Si esto ocurre, que no debería, habrá que cambiar el texto por uno más pequeño o utilizar otro método de traducción.

·Traducir con XSE

Editar textos con XSE es fácil. Es editar los scripts del juego. El problema es que no podremos cambiar textos que no estén en scipts. Es decir, la introducción no es un script, no podremos editarla con XSE, pero él lo que nos dicen en un centro pokémon cuando curamos a nuestro equipo sí que es un script, así que ese sí podemos traducirlo. No es necesario extenderse demasiado con esto. Simplemente abrimos el script con XSE y ponemos el *#dynamic 0x800000* para que el programa busque espacios libres. Esto ya lo sabemos. Ahora dejamos el script tal como está, pero editamos los msgbox cambiando los offsets por pointers. Ahora escribimos el nuevo texto y el programa lo cambiará automáticamente. Por ejemplo. Descompilamos este hipotético script:

```
#org 0x8000DF
lock
faceplayer
msgbox 0x80140A 0x6
release
end

#org 0x80140A
= Hello, friend!
```

Es un simple script que hemos descompilado. Ahora bien, como hemos dicho, vamos a poner el *dynamic* para que XSE busque espacios vacíos para nuestro nuevo texto:

```
#dynamic 0x800000

#org 0x8000DF
lock
faceplayer
msgbox 0x80140A 0x6
release
end

#org 0x80140A
= Hello, friend!
```

Ahora vemos que lo que queremos cambiar es el msgbox, por lo que cambiamos los offsets por pointers para que se compilen de nuevo:

```
#dynamic 0x800000

#org 0x8000DF
lock
faceplayer
msgbox @texto 0x6
release
end

#org @texto
= ¡Hola, amigo!
```

Ahora, al compilar, el texto se escribirá en un offset libre y el resto del script, puesto que no hemos cambiado el offset, quedará intacto. Pero el texto, que es lo que nos importa, sí se editará.

·Traducir con hexadecimal

Cuando Advance-Text nos falle y no podamos traducir con XSE por no tratarse de un script, un editor hexadecimal será lo único que no nos dé la espalda. Siempre recomendaré que el editor usado para modificar el rom de manera hexadecimal sea **HxD**, pero en este caso usaremos también **Thingy32**, ya que nos permitirá cargar las tablas del juego y buscar fácilmente los textos.

Lo primero que debes hacer es abrir Thingy32 y cargar el rom. Automáticamente te pedirá que cargues la tabla que quieras. Tendremos que cargar la tabla que mostrará a que carácter corresponde cada byte. Por ejemplo, la hache mayúscula (*H*) se corresponde con el valor hexadecimal *C2*. Entonces, lo que hará la tabla será sustituir los *C2* por *H*. Y así con todo. En la parte izquierda de la herramienta veremos los bytes del rom, mientras que en la derecha, la equivalencia dependiendo de la tabla que hayamos cargado.

Ahora hay dos opciones. O poner el nuevo texto sobrescribiendo el original, de manera que no haya que repuntear; o compilar el nuevo texto en un espacio vacío y repointear. Recordemos que cada texto es llamado por un pointer, por lo tanto, un texto como tal en alguna parte del rom no sirve para nada, tiene que llamarse desde un script o una rutina. Entonces, siempre que cambiemos el lugar del texto, habrá que cambiar el pointer para que apunte hacia el nuevo texto y no siga apuntando al antiguo. Cuando el nuevo texto quepa en el espacio del antiguo, puedes usar dicho espacio para el nuevo y evitas tener que repuntear. Pero muchas veces hay que repuntear porque no hay espacio o, simplemente, para evitarnos errores.

Como cambiar un texto repointeando es más difícil que no repointeando, será lo que expliquemos. Un ejemplo claro es si abrimos el menú. Podremos ver la palabra “BAG”. Pero claro, esa palabra tiene tres letras, mientras que “MOCHILA” tiene seis. La nueva palabra no cabría en el espacio original, por lo que lo tendremos que compilar en un espacio vacío del rom y cambiar el pointer.

Vamos a Thingy32 y buscamos “BAG$” ¿Por qué “$” al final? Porque la palabra “BAG” está en un montón de textos, por lo que nos costaría encontrarla. Sin embargo, sabemos que todo texto en hexadecimal acaba con “FF”, que es el byte que simboliza precisamente eso, el fin del texto. Entonces, como la palabra es “BAG” y ahí acaba, podemos suponer que después de ella habrá un “FF”. Y como en Thingy32, el carácter que representa FF es $, lo usaremos.

Nota: Cuando busquemos textos más largos, basta con buscar una parte de él, no es necesario buscarlo entero.

Ahora tenemos que fijarnos que encontramos la palabra que buscamos. Lo recomendable es alterarla levemente y comprobar el resultado en el rom. Por ejemplo, “BAG” la cambiamos por “BEG” y abrimos el rom. Si vamos al menú y aparece “BEG”, era la que buscábamos. Si no, reestablecemos la palabra y buscamos la siguiente.

Yo he encontrado la palabra “BAG” que buscaba en el offset 0x416285. Recordamos ese offset. Ahora abrimos XSE y compilamos el nuevo texto de la siguiente manera:

```
#dynamic 0x800000

#org @texto
= MOCHILA
```

Así compilaremos simplemente el texto que queramos. En el *dynamic* pondremos la dirección donde queramos compilar el texto. Por supuesto, se compilará en una zona vacía, como si de cuando compilamos un script se tratara. En este caso, el texto se ha compilado en la dirección 0x8016A0. Recordamos también este offset.

Ahora cogemos el offset del texto original (“BAG”), que era el 0x416285. Como lo que hay que buscar es el puntero, le añadimos “08” (08416285) y lo permutamos (85624108). Ahora tenemos que buscar esos bytes en el rom. Se puede hacer con Thingy32, aunque para esto yo ya prefiero usar HxD. Pero lo que cada uno quiera.

Buscamos el puntero 85624108 (puede haber más de uno, pero los repunteamos todos, ya que apuntan al mismo texto). Cuando lo localicemos, tendremos el puntero. Ahora lo que hay que hacer es repuntear, es decir, cambiar el pointer para que apunte al nuevo offset. Nuestro nuevo offset (donde se encuentra el nuevo texto) es 0x8016A0. Lo convertiremos en puntero añadiendo 08 (088016A0) y lo permutaremos (A0168008). Ahora lo único que tenemos que hacer es sobrescribir el puntero antiguo con el nuevo para que ahora el juego apunte a nuestro nuevo texto.

Adicionalmente podemos borrar el texto original. Si repointeamos todo y ya no queda nada apuntando al texto antiguo, podemos eliminarlo para que simplemente no nos confundamos al editar textos que contengan la misma palabra, por ejemplo. Eso se puede hacer directamente en Thingy32 cambiando el texto por caracteres $. Por ejemplo, la palabra “BAG” la cambiaríamos por “$$$”.

No obstante, esto es opcional. Así que debería quedar a elección del usuario.

Este procedimiento recordemos que es para textos que haya que repointear porque así lo queramos o porque sean más largos que el original. Pero cuando los textos sean iguales o más cortos, podemos usar el espacio original editándolo directamente en Thingy32, así nos evitaremos repointear.

“Traduciendo”

Manual redactado por **Javi4315**.