

Introducción al manejo de archivos¹

El lenguaje ensamblador posee un conjunto de instrucciones para el manejo de archivos (y carpetas) que son útiles para abrir, crear, borrar y mover tanto archivos de texto como binarios. La lectura y escritura se puede hacer de forma secuencial o aleatoria en función de las necesidades de la solución a codificar.

En muchas ocasiones se requiere conservar los resultados de un programa más allá del tiempo en que lo ejecutamos, por ejemplo, al editar una imagen o un texto. A veces lo que queremos es procesar un largo conjunto de datos, que seguramente no vamos a introducir por teclado sino a través de algún tipo de archivo.

Entonces, podemos decir que el manejo de archivos en un programa amplía sus capacidades para manejar datos, administrando la información que se desea conservar en conjuntos complejos y personalizados.

El manejo elemental del sistema de archivos se hace por lo general, y para no causar conflictos, a través de servicios de la interrupción **21H** del sistema operativo. Pero también es posible realizarlo mediante **interrupciones directas**.

Para abrir un archivo se utiliza el servicio **0716CH** de la interrupción 21H. BX tendrá un valor de **0** si se abre sólo para lectura, **1** si es para escritura y **2** cuando haremos lectura **7escritura**. DX se deberá tener un 1 para abrir, un 2 para bloquear y un 10H si se quiere crear (si se intenta crear y ya existe, regresa una bandera). El nombre de archivo es una cadena de caracteres que debe estar apuntada en **DS (segmento)** y **DI (el offset)** en el momento de hacer la interrupción. Si la apertura o creación es exitosa la bandera CF tendrá 1, en caso contrario, señalará con un 0.

- El servicio **3EH** de la interrupción 21H es quien se encarga de hacer el cierre del archivo. Hasta ese momento vaciará el contenido en el buffer de memoria en el sistema de archivo (salvar). Utilizará la misma bandera para comunicar el estado final de la acción con la bandera **CF=1** si no tuvo éxito y **CF=0** si cerró bien el archivo.

Para moverse entre la información contenida en el archivo se moviliza el apuntador de archivo con la ayuda del servicio **42H** de la interrupción 21H. El registro **AL** indica la opción de movimiento. **AL=0** indica un desplazamiento relativo al inicio del archivo. **AL=1** indica un movimiento a partir de la posición actual. **AL=2** señala un desplazamiento a partir de la posición final del archivo. **CX:DX** deberán tener el valor de posiciones en que se desplazará el apuntador de archivo.

¹ Documento elaborado por el Dr. Carlos Jiménez Gallegos.

A continuación se reproduce para mayor referencia el ejemplo del libro (Irvine, 2007, pág. 435) de lenguaje ensamblador.

TITLE Chapter 13 Exercise 1 (ch13_01.asm)

Comment !

Description: Open a file for input, read the file, and display its contents on the screen in Hexadecimal. Make the input buffer smaller than the file and use a loop to repeat the call to Function 3Fh as many times as necessary until the entire file has been processed.

Implementation note: We will borrow code from the ReadFile program in Section 13.3.4, and use DumpMem from the link library to display the buffer.

Difficulty level: 2/5

Last update: 05/14/2002

!

INCLUDE Irvine16.inc

.data

BufSize = 256

infile BYTE "infile.txt",0

inHandle WORD ?

buffer BYTE BufSize DUP(?)

bytesRead WORD ?

.code

main PROC

mov ax,@data

mov ds,ax

; Open the input file

```
mov ax,716Ch      ; extended create or open
mov bx,0          ; mode = read-only
mov cx,0          ; normal attribute
mov dx,1          ; action: open
mov si,OFFSET infile
int 21h           ; call MS-DOS
jc  quit ; quit if error
mov inHandle,ax
```

Read_File_Into_Buffer:

```
mov ah,3Fh      ; read file or device
mov bx,inHandle ; file handle
mov cx,BufSize  ; max bytes to read
mov dx,OFFSET buffer ; buffer pointer
int 21h
jc Close_File ; quit if error
cmp ax,0        ; end of file?
je Close_File ; yes: close the file
mov bytesRead,ax
```

; Display the buffer in hexadecimal

```
mov esi,OFFSET buffer ; address of buffer
movzx ecx,bytesRead    ; number of units to display
mov ebx,1 ; unit size (byte)
call DumpMem

jmp Read_File_Into_Buffer ; read more data
```

Close_File:

```
mov ah,3Eh      ; function: close file
mov bx,inHandle ; input file handle
```

```
int 21h      ; call MS-DOS

quit:
    call Crlf
    exit

main ENDP
END main
```

Nota: para aprender más acerca del método FCB, consulte el siguiente enlace

:<http://es.scribd.com/doc/97567087/84/Metodo-FCB>

Y para el Método de canales de comunicación:

<http://www.slideshare.net/andressteve/curso-bsico-de-ensamblador>