Periféricos y Dispositivos de Interfaz Humana
Grado en Ingeniería Informática

PRÁCTICA 1: PROGRAMACIÓN DE INTERRUPCIONES PARA PERIFÉRICOS DE ENTRADA/SALIDA EN MS-DOS

Objetivos

I. Aprender la forma de uso y programación de interrupciones software desde programas de usuario.

II. Comprender la interfaz que permite un sistema operativo para el acceso a los recursos de entrada/salida.

III. Crear una librería de funciones básicas de Entrada/Salida a través de llamadas a interrupciones.

Contenido

1. Introducción

En esta práctica se va a acceder a las rutinas de servicio a interrupción de la BIOS para teclado y vídeo desde el sistema operativo MS-DOS. Se va a utilizar el lenguaje de programación C para realizar interrupciones software a través de la función no estándar `int86()`, que viene definida en el fichero `dos.h` de varios compiladores de C:

```
#include <dos.h>
int int86(int intno, union REGS *inregs, union REGS *outregs);
```

El parámetro de entrada `intno` indica el número de interrupción que se desea ejecutar; en `inregs` se especifican los valores de los registros antes de la llamada, y en `outregs` se obtienen los valores de los registros tras ser ejecutada la rutina correspondiente. Tanto `inregs` como `outregs` se declaran como una unión de tipo REGS que está definida como sigue:

```
union REGS
{
struct WORDREGS x;
struct BYTEREGS h;
};
```

De esta forma se puede acceder a los registros internos bien como registros de 16 bits, bien como registros de 8 bits. Las estructuras BYTEREGS y WORDREGS se definen en `dos.h` como:

```
struct BYTEREGS  {
   unsigned char  al, ah, bl, bh;
   unsigned char  cl, ch, dl, dh;
};
struct WORDREGS  {
   unsigned int  ax, bx, cx, dx;
   unsigned int  si, di, cflag, flags;
};
```

Como punto de inicio, se propone partir del ejemplo usado en el seminario 1 para cambiar el modo de vídeo. La forma más estructurada de programar es crear una librería de funciones relacionadas con las llamadas a interrupciones de la BIOS (por ejemplo, “mi_io.h” y “mi_io.c”).

2. Interrupción BIOS del teclado

Las subfunciones para gestionar el teclado usan la interrupción de la BIOS 16h. En esta práctica usaremos la subfunción 0, que lee un carácter desde el teclado y la subfunción 1, que permite detectar la pulsación de una tecla.

Más información: http://es.wikipedia.org/wiki/Int_16h

Leer un carácter desde el teclado

Número de interrupción: 16h
Número de función: 0

Entrada: AH = 0

Salida: AL: código ASCII de la tecla pulsada
AH: BIOS SCAN CODE de la tecla pulsada

- Mediante esta función se capta un carácter del búfer del teclado, sin imprimirlo.
- Todas las teclas extendidas (F1, F2, etc) generan como código ASCII el 0. Para distinguirlas, hay que fijarse en el BIOS Scan Code.

Detección de tecla pulsada en búfer de teclado

Número de interrupción: 16h
Número de función: 1

Entrada: AH = 1
Salida: Zero-flag = 0 si hay una tecla en el búfer
Zero-flag = 1 el búfer está vacío
Nota: El flag del cero es el bit 6 del registro flags (¡el primer bit es el bit 0!).

Bit:	15	14	13	12	11	10	9	8	7	**6**	5	4	3	2	1	0
Flag:	-	-	-	-	O	D	I	T	S	**Z**	-	A	-	P	-	C

3. Interrupción BIOS de vídeo

La comunicación con la tarjeta de vídeo se puede realizar a través de la interrupción número 10h. Asignando distintos valores al registro *AH,* es posible acceder a diversas subfunciones que afectan a la salida de caracteres por pantalla.

Más información: http://es.wikipedia.org/wiki/Int_10h

Seleccionar el modo de vídeo

Número de interrupción: 10h
Número de función: 0

Entrada: AH = 0
AL = modo
Salida: No tiene

Tabla I. Modos de vídeo.

Modo	**Tipo**	**Resolución**	**Colores**
0h	texto	40x25	16 tonos de gris
1h	texto	40x25	16 colores
2h	texto	80x25	16 tonos de gris
3h	texto	80x25	16 colores

4h	gráfico	320x200	4 colores
5h	gráfico	320x200	4 colores
6h	gráfico	640x200	2 colores
7h	texto	80x25	monocromo
dh	gráfico	320x200	16 colores
eh	gráfico	640x200	16 colores
fh	gráfico	640x350	monocromo
10h	gráfico	640x350	16 colores
11h	gráfico	640x480	2 colores
12h	gráfico	640x480	16 colores
13h	gráfico	320x200	256 colores

Averiguar el modo de vídeo actual

Número de interrupción: 10h
Número de función: Fh

Entrada: AH = Fh
Salida: AL = modo actual
AH = número de columnas (sólo en los modos de texto)

Fijar el tamaño del cursor en modo texto

Número de interrupción: 10h
Número de función: 1

Entrada: AH = 1
CH = número de línea inicial (los 3 bits menos significativos)
CL = número de línea final (los 3 bits menos significativos)
Salida: No tiene
Nota: El número de línea se cuenta de arriba hacia abajo. Para hacer el cursor invisible, se debe poner el 6º bit de CH a 1.

Colocar el cursor en una posición determinada

Número de interrupción: 10h
Número de función: 2

Entrada: AH = 2
DH = número de fila (00h indica arriba del todo)
DL = número de columna (00h indica izquierda del todo)
BH = 0
Salida: No tiene

Obtener tamaño y posición del cursor

Número de interrupción: 10h
Número de función: 3

Entrada: AH = 3
BH = 0

Salida: CH = tamaño/número de línea inicial
CL = tamaño/número de línea final
DH = posición/fila (00h indica arriba del todo)
DL = posición/columna (00h indica izquierda del todo)

Escribir un carácter en pantalla

Número de interrupción: 10h
Número de función: 9

Entrada: AH = 9
AL = código ASCII del carácter
BL = color
BH = 0
CX = número de repeticiones

Salida: No tiene

Nota: Escribe un carácter en la posición actual del cursor. En cuanto al byte del color, el primer cuarteto fija el color de fondo y el segundo cuarteto el color del carácter (ver ayuda de Borland C sobre "textattr").

Colores de Fondo y de Texto			
Constante	**Valor**	**Significado**	**De Fondo o de Texto**
BLACK	0	Negro	Ambos
BLUE	1	Azul	Ambos
GREEN	2	Verde	Ambos
CYAN	3	Cían	Ambos
RED	4	Rojo	Ambos
MAGENTA	5	Magenta	Ambos
BROWN	6	Marrón	Ambos
LIGHTGRAY	7	Gris Claro	Ambos
DARKGRAY	8	Gris Oscuro	Sólo para texto
LIGHTBLUE	9	Azul Claro	Sólo para texto
LIGHTGREEN	10	Verde Claro	Sólo para texto
LIGHTCYAN	11	Cían Claro	Sólo para texto
LIGHTRED	12	Rojo Claro	Sólo para texto
LIGHTMAGENTA	13	Magenta Claro	Sólo para texto
YELLOW	14	Amarillo	Sólo para texto
WHITE	15	Blanco	Sólo para texto
BLINK	128	Parpadeo	Sólo para texto

Desplazar zona de pantalla hacia arriba (scroll vertical)

Número de interrupción: 10h
Número de función: 6

Entrada: AH = 6
AL = número de líneas a desplazar
BH = color para los espacios en blanco
CH = línea de la esquina superior izquierda
CL = columna de la esquina superior izquierda
DH = línea de la esquina inferior derecha
DL = columna de la esquina inferior derecha

Salida: No tiene

Nota: Desplaza una zona de la pantalla delimitada por los valores de los registros CH, CL, DH y DL tantas líneas hacia arriba como indique el registro AL. Si AL=0 entonces no se desplaza, sino que borra esa zona.

Desplazar zona de pantalla hacia abajo (scroll vertical)

Número de interrupción: 10h
Número de función: 7

Nota: Es análoga a la anterior variando únicamente el sentido del desplazamiento vertical.

4. Aplicación práctica

Usando las subfunciones expuestas en este guión, debe implementar en C/C++ un conjunto de funciones similares a las que ofrece la biblioteca conio.lib (*Console Input Output Library*), que permiten realizar tareas tales como cambiar la posición del cursor, borrar la pantalla, cambiar el color del texto, etc. Puede obtener más ayuda sobre las funciones de esta biblioteca desde el propio Borland C o en http://c.conclase.net/borland/?borlandlib=conio#inicio.

Los requisitos mínimos se valorarán sobre 7 puntos como máximo, los ampliados se valorarán con 3 puntos más como máximo.

R**equisitos mínimos:**

Realizar las siguientes funciones:

1. *kbhit()*: indica si se ha pulsado alguna tecla
2. *wherex()*: indica la posición x actual del cursor
3. *wherey()*: indica la posición y actual del cursor
4. *gotoxy():* Mueve el cursor a la posición según las coordenadas especificadas por los argumentos x e y.

5. *setcursortype()*: fijar el aspecto del cursor, debe admitir tres valores: INVISIBLE, NORMAL y GRUESO.
6. *setvideomode()*: fija el modo de video deseado
7. *getvideomode()*: obtiene el modo de video actual
8. *textcolor()*: modifica el color de primer plano con que se mostrarán los caracteres
9. *textbackground()*: modifica el color de fondo con que se mostrarán los caracteres
10. *clrscr()*: borra toda la pantalla
11. *clreol()*: borra una línea desde la posición actual del cursor hasta el final de la misma
12. *scrollup()*: desplazar toda la pantalla una línea hacia arriba
13. *scrolldown()*: desplazar toda la pantalla una línea hacia abajo
14. *cputchar()*: escribe un carácter en pantalla con el color indicado actualmente
15. *getche():* obtiene un carácter de teclado y lo muestra en pantalla
16. cputs(): imprime una cadena de caracteres en pantalla

Estas funciones debe incluirlas en un fichero distinto del que contenga a la función principal, con objeto de que sea reutilizable por otros programas. De esta forma, se creará una biblioteca de funciones que pueda enlazar con otros programas (mi_io.c y mi_io.h).

El programa principal debe utilizar de forma secuencial todas las funciones anteriores y permitir comprobar su correcto funcionamiento de la forma más interactiva posible.

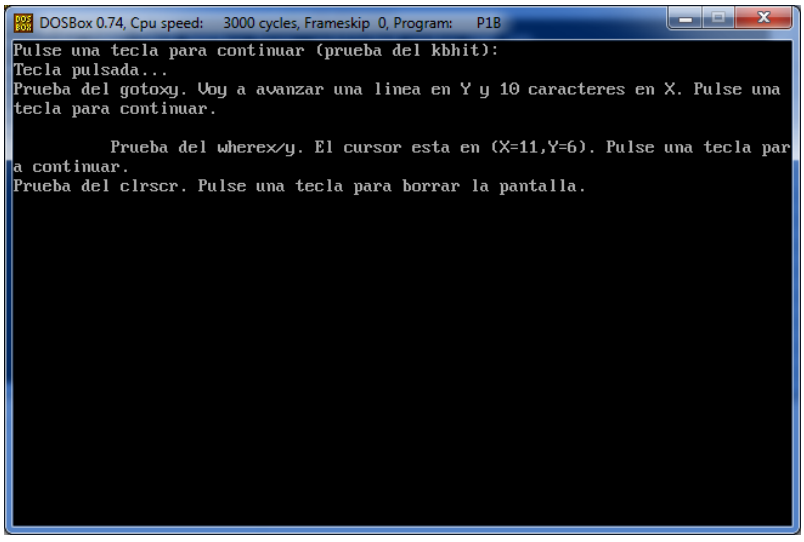

Figura 1. Ejemplo de interfaz secuencial para cubrir los requisitos básicos.

Requisitos ampliados:

1. Implementar una función que permita dibujar un recuadro en la pantalla. Recibirá como parámetros las coordenadas superior izquierda e inferior derecha del recuadro, el color de primer plano y el color de fondo. La impresión de cada carácter se realizará mediante la función *cputchar*. Se usará esta función en la realización de esta práctica para dividir la pantalla y crear una interfaz agradable.

2. Programar un menú de opciones navegable mediante las teclas de cursor (u otras), destacando la opción seleccionada en cada momento, que es la que se activará al pulsar ENTER. Utilizar funciones modulares para la programación del menú. Usar este menú para demostrar la ejecución de las funciones de los requisitos mínimos.

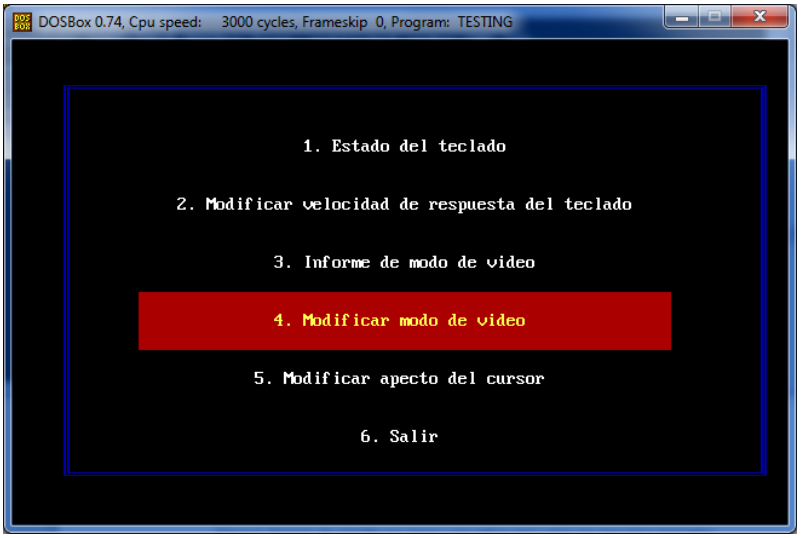

Figura 2. Ejemplo de interfaz con menú y recuadros para cubrir los requisitos ampliados.

5. Consejos

1. Tratar de implementar las funciones lo más similares posibles a las análogas de la librería del C. En caso de duda, leer la ayuda referente a cada una de ellas para ver el número y tipo de parámetros que requieren, así como los valores que devuelven. Es más, se puede empezar probando la función original y después la propia para verificar que la implementada la sustituye perfectamente.

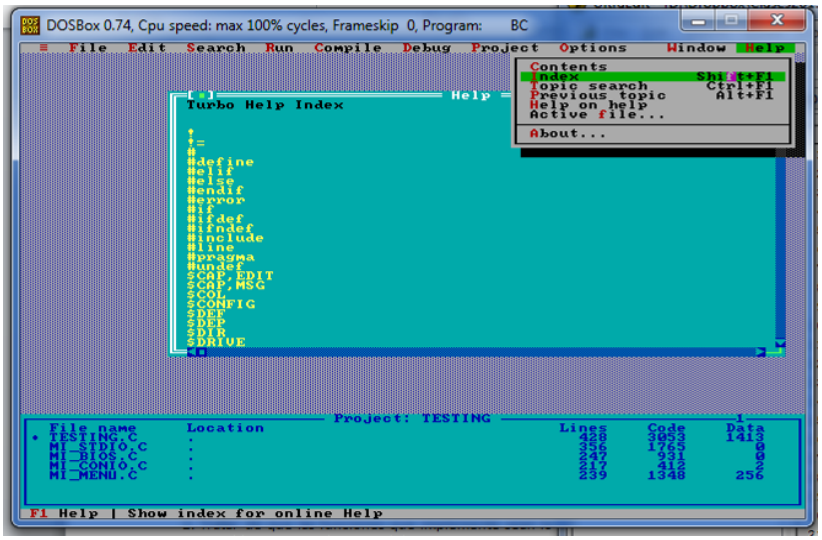

Figura 3. Ayuda contextual de Borland C

2. Las funciones implementadas deben ser lo más independientes del contexto posible. Por ejemplo, no se deben usar instrucciones de E/S, mensajes ni cosas similares en el código de las funciones de la biblioteca que se va a programar.
3. No implementar funciones complejas que sirvan para varias cosas a la vez. En su lugar, se debe Implementar una función distinta por cada una de las que se requieren.
4. No usar los mismos identificadores que los que use el compilador de C/C++ para nombrar a las funciones.
5. Usar el depurador integrado de Borland C para corregir los errores de programación.

6. Evaluación y entrega

- **Número de sesiones para esta práctica:** 3
- **Fecha límite de entrega y defensa:** 26 de marzo de 2019.
- **Material a entregar:** código fuente documentado de la librería (mi_io.h, mi_io.c) y del programa principal. Se realizará una defensa en el propio PC del alumno de la práctica realizada.