Práctica 1. Entrada/Salida utilizando interrupciones con lenguaje C

Duración: 2 sesiones

1. Objetivos de la práctica

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.

2. 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”).

3. 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!).

4. 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

Modo	Resolución	Colores	Tipo
AL = 00h	40x25	16	Texto
AL = 01h	40x25	16	Texto
AL = 02h	80x25	16	Texto
AL = 03h	80x25	16	Texto
AL = 04h	320x200	4	Gráfico
AL = 05h	320x200	4	Gráfico
AL = 06h	640x200	2	Gráfico
AL = 07h	80x25	2	Texto
AL = 0Dh	320x200	16	Gráfico
AL = 0Eh	640x200	16	Gráfico
AL = 0Fh	640x350	2	Gráfico
AL = 12h	640x480	16	Gráfico
AL = 13h	320x200	256	Gráfico

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 izq. 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.

5. Un par de ejemplos como punto de partida

En el caso de querer leer una pulsación de tecla, mostrarla a continuación, y cambiar el tipo de cursor, podemos usar:

```
#include <stdio.h>
#include <dos.h>

int mi_getchar(){
       union REGS inregs, outregs;
       int caracter;

       inregs.h.ah = 1;
       int86(0x21, &inregs, &outregs);

       caracter = outregs.h.al;
       return caracter;
}

void mi_putchar(char c){
       union REGS inregs, outregs;

       inregs.h.ah = 2;
       inregs.h.dl = c;
       int86(0x21, &inregs, &outregs);
}

void setcursortype(int tipo_cursor){
      union REGS inregs, outregs;
      inregs.h.ah = 0x01;
      switch(tipo_cursor){
            case 0: //invisible
                  inregs.h.ch = 010;
                  inregs.h.cl = 000;
                  break;
```

```
            case 1: //normal
                  inregs.h.ch = 010;
                  inregs.h.cl = 010;
                  break;
            case 2: //grueso
                  inregs.h.ch = 000;
                  inregs.h.cl = 010;
                  break;
      }
      int86(0x10, &inregs, &outregs);
}


int main(){
      int tmp;

      printf("\nPulsa una tecla...  ");
      tmp = mi_getchar();

      printf("\nHas pulsado: ");
      mi_putchar( (char)tmp );

      printf("\nCursor invisible: ");
      setcursortype(0);
      mi_pausa();
      printf("\nCursor grueso: ");
      setcursortype(2);
      mi_pausa();
      printf("\nCursor normal: ");
      setcursortype(1);
      mi_pausa();

      return 0;
}
```

Para trabajar en modo gráfico y crear dibujos a partir de pixels individuales, podemos usar:

```
#include <dos.h>

#define BYTE unsigned char

BYTE MODOTEXTO = 3;
BYTE MODOGRAFICO = 4;

// hace una pausa
void pausa(){
   union REGS inregs, outregs;
   inregs.h.ah = 0x00;
   int86(0x16, &inregs, &outregs);
}

// establece el modo de vídeo: 3-texto, 4-gráfico
void modovideo(BYTE modo){
   union REGS inregs, outregs;
   inregs.h.al = modo;
   inregs.h.ah = 0x00;
   int86(0x10, &inregs, &outregs);
}
```

```
// pone un pixel en la coordenada X,Y de color C
void pixel(int x, int y, BYTE C){
   union REGS inregs, outregs;
   inregs.x.cx = x;
   inregs.x.dx = y;
   inregs.h.al = C;
   inregs.h.ah = 0x0C;
   int86(0x10, &inregs, &outregs);
}

int main(){
   modovideo(MODOGRAFICO); //gráfico

   pixel(10,40,0);
   pixel(10,50,1);
   pixel(15,60,2);
   pixel(20,70,3);

   pausa();
   modovideo(MODOTEXTO); //texto

   return 0;
}
```

6. Cuestiones a resolver

Usando las subfunciones expuestas en este guión, hay que 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 la página 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.

Requisitos mínimos:
Realizar las siguientes 10 funciones:

- gotoxy(): coloca el cursor en una posición determinada
- setcursortype(): fijar el aspecto del cursor, debe admitir tres valores: INVISIBLE, NORMAL y GRUESO.
- setvideomode(): fija el modo de video deseado
- getvideomode(): obtiene el modo de video actual
- textcolor(): modifica el color de primer plano con que se mostrarán los caracteres
- textbackground(): modifica el color de fondo con que se mostrarán los caracteres
- clrscr(): borra toda la pantalla
- cputchar(): escribe un carácter en pantalla con el color indicado actualmente
- getche(): obtiene un carácter de teclado y lo muestra en pantalla
- pixel(): dibujar un pixel en modo gráfico (la función recibirá la coordinada x,y y el color del punto.

Estas funciones deben diseñarse e implementarse de forma que se puedan reutilizar fácilmente en otros programas. El programa debe utilizar de todas esas funciones para comprobar su correcto funcionamiento.

Requisitos ampliados (opcionales para subir nota):

- Implementar una función que permita dibujar un recuadro en la pantalla en modo texto. Recibirá como parámetros las coordenadas superior izquierda e inferior derecha del recuadro, el color de primer plano y el color de fondo.
- Implementar en lenguaje C un programa que establezca modo gráfico CGA (modo=4) para crear dibujos sencillos en pantalla.
- Implementar un programa sencillo que realice un dibujo sencillo de tipo "ascii art". En el ANEXO al final de este guión se proponen algunos diseños.

Normas de entrega

La práctica/seminario podrá realizarse de manera individual o por grupos de hasta 2 personas.

Se entregará como un archivo de texto en el que se muestre la información requerida. También se puede utilizar la sintaxis de Markdown para conseguir una mejor presentación e incluso integrar imágenes o capturas de pantalla. La entrega se realizará subiendo los archivos necesarios al repositorio "**PDIH**" en la cuenta de GitHub del estudiante, a una carpeta llamada "**P1**".

Toda la documentación y material exigidos se entregarán en la fecha indicada por el profesor. No se recogerá ni admitirá la entrega posterior de las prácticas/seminarios ni de parte de los mismos.

La detección de prácticas copiadas implicará el suspenso inmediato de todos los implicados en la copia (tanto de quien realizó el trabajo como de quien lo copió).

Las faltas de ortografía se penalizarán con hasta 1 punto de la nota de la práctica/seminario.

Referencias

https://www.dosbox.com/DOSBoxManual.html
https://www.linuxadictos.com/dosbox-en-linux.html
https://es.wikihow.com/usar-DOSBox
http://ubuntudriver.blogspot.com/2011/09/instalacion-basica-de-dosbox-en-ubuntu.html
https://fresh2refresh.com/c-programming/c-file-handling/putchar-getchar-function-c/
https://www.dosgamers.com/es/dos/dosbox-dos-emulator/screen-resolution
https://www.enmimaquinafunciona.com/pregunta/168127/aumentar-el-tamano-de-la-ventana-de-dosbox

Anexo. Funciones de E/S utilizando servicios de la BIOS o del DOS

Colocar el cursor en modo texto en (X,Y):

```
mov dl,x  ; dl=columna
mov dh,x  ; dl=fila
mov bh,0
mov ah,2  ;función para posicionar el cursor
int 10h   ;interrupción BIOS para pantalla
```

Escribir un carácter en pantalla (int 10h):

```
mov al,CARACTER  ;código ASCII del carácter a escribir
mov bx,0
mov ah,0Eh  ;función para escribir un carácter
int 10h     ;interrupción BIOS para pantalla
```

Escribir un carácter en pantalla (int 21h):

```
mov dl,CARACTER  ;código ASCII del carácter a escribir
mov ah,2    ;función para escribir un carácter
int 21h     ;interrupción MSDOS para pantalla
```

Esperar la pulsación de una tecla (int 16h):

```
mov ah,0   ;función para leer una tecla
int 16h    ;interrupción BIOS para teclado
 ;en AH devuelve el identificador de la tecla
 ;en AL devuelve el codigo ASCII de la tecla pulsada
```

Esperar la pulsación de una tecla sin mostrarla por pantalla (controla Crtl-Break):

```
mov ah,08h   ;función para leer una tecla
int 21h      ;interrupción DOS para teclado
 ;en AL devuelve el carácter tecleado
```

Esperar la pulsación de una tecla mostrándola por pantalla (controla Crtl-Break):

```
mov ah,01h   ;función para leer una tecla
int 21h      ;interrupción DOS para teclado
 ;en AL devuelve el carácter tecleado
```

Poner el MODO gráfico y pintar un punto en X,Y de COLOR específico:

0 - texto 40x25 b/n
1 - texto 40x25 color
2 – texto 80x25 b/n
3 – texto 80x25 color
4 – gráfico 320x200 color
5 – gráfico 320x200 b/n
6 – gráfico 640x200 b/n

Poner el modo texto o gráfico deseado (int 10h):

```
mov al,MODO
mov ah,0     ;función para establecer el modo de pantalla
int 10h      ;interrupción BIOS para pantalla
```

Pintar un pixel en modo gráfico (previamente debemos haber puesto el modo gráfico):

```
mov cx,X     ;columna
mov dx,Y     ;fila
mov al,COLOR  ;color del pixel iluminado
mov ah,0Ch    ;función para iluminar un pixel
int 10h      ;interrupción BIOS para pantalla
```

programando en C usando la librería dos.h

```c
#include <stdio.h>
#include <dos.h>

void mi_getchar(){
     union REGS inregs, outregs;
     //int caracter;

     inregs.h.ah = 1;
     int86(0x21, &inregs, &outregs);

     //caracter = outregs.h.al;
     //return caracter;
}

void mi_exit(){
     union REGS inregs, outregs;
     inregs.x.ax = 0x4C00;
     int86(0x21, &inregs, &outregs);
}

void main(){
    printf("\nPulsa una tecla para terminar...  ");

    mi_getchar();

   mi_exit();
}
```

programando en ensamblador (x86, 16bits)

Esperar la pulsación de una tecla

```
mov ah,1     ;función para leer una tecla
int 21h      ;interrupción DOS
```

Terminar el programa

```
mov ax,4C00h   ;función para terminar
int 21h        ;interrupción DOS
```

1

programando en C usando la librería dos.h

```c
int mi_getchar(){
    union REGS inregs, outregs;
    int caracter;

    inregs.h.ah = 1;
    int86(0x21, &inregs, &outregs);

    caracter = outregs.h.al;

    return caracter;
}

void mi_putchar(char c){
    union REGS inregs, outregs;

    inregs.h.ah = 2;
    inregs.h.dl = c;

    int86(0x21, &inregs, &outregs);
}
```

programando en ensamblador (x86, 16bits)

Esperar la pulsación de una tecla mostrándola por pantalla

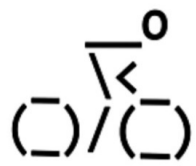
```
mov ah,1          ;función para leer una tecla
int 21h           ;interrupción MSDOS

;en AL devuelve el carácter tecleado
```

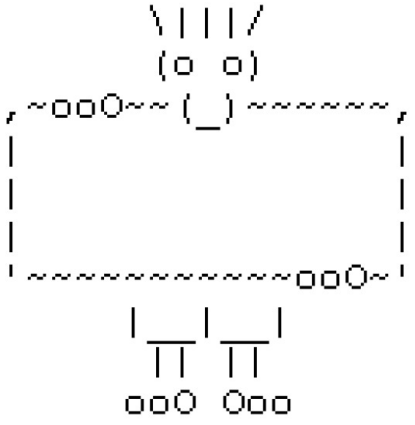
Escribir un carácter en pantalla

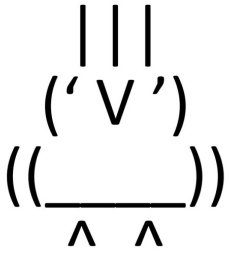
```
mov dl,CARACTER   ;código ASCII del carácter a escribir
mov ah,2          ;función para escribir un carácter
int 21h           ;interrupción MSDOS
```

2

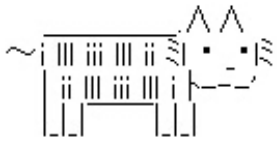
ANEXO. EJEMPLOS DE DIBUJOS “ascii art”







Rat



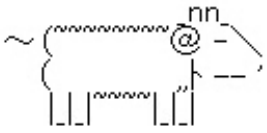
Tiger

Rabbit

Dragon

Serpent

Horse

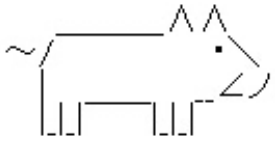


Sheep

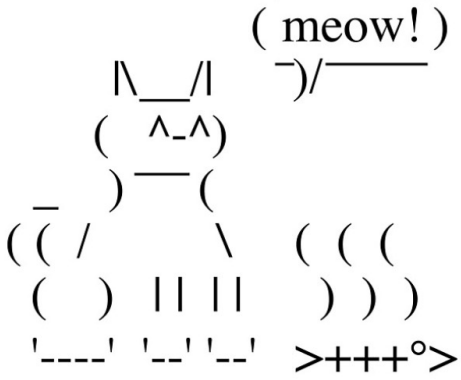
Monkey

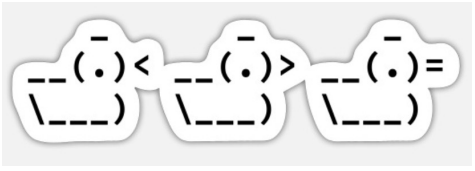
Cock

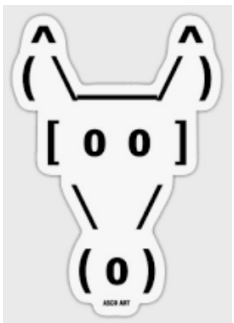
Dog

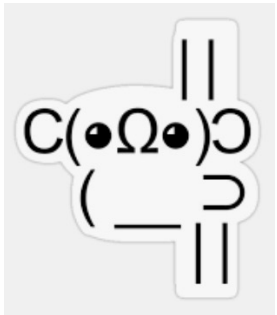


Boar









Rúbrica para evaluar la práctica 1

El objetivo es usar las subfunciones expuestas en el guión de la práctica 1 e implementar en C/C++ un conjunto de funciones para realizar tareas tales como cambiar la posición del cursor, borrar la pantalla, cambiar el color del texto, etc. La idea es que esas funciones se puedan usar fácilmente en otros programas. El programa principal (*main*) debe utilizar las funciones implementadas para comprobar su correcto funcionamiento.

Esta primera parte (*implementación de funciones*) contará un **máximo de 7 puntos**. La siguiente tabla detalla cuánto contará cada función implementada, según la implementación realizada y su correcto funcionamiento:

Función a implementar	0,7 puntos	0,5 puntos	0,3 puntos	0 puntos
gotoxy()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
setcursortype()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
setvideomode()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
getvideomode()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
textcolor()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
textbackground()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
clrscr()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
cputchar()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
getche()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
pixel()	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
alguna función adicional no indicada más arriba	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar

Para **subir nota** se proponen **tres pequeños ejercicios adicionales**. Cada uno contará un máximo de 1 punto, de forma que haciendo los tres se puede llegar a la máxima nota en la práctica:

Ejercicio adicional	1 punto	0,5	0,3	0 puntos
Implementar una función para dibujar recuadros en la pantalla (en modo texto)	Función implementada adecuadamente y funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar
Programa que realice dibujos sencillos en pantalla (en modo gráfico)	El programa funciona correctamente y muestra al menos un dibujo con varias formas	El programa funciona correctamente. Sólo muestra líneas simples o puntos	La implementación no es adecuada y da errores	No se ha intentado implementar
Programa que realice un dibujo sencillo de tipo "ascii art" (en modo texto)	El programa funciona correctamente	La implementación de la función no es adecuada, aunque funciona	La implementación no es adecuada y da errores	No se ha intentado implementar