

PRÁCTICA 2 PDIH

MANUEL FERNÁNDEZ LA-CHICA

Ejercicios obligatorios

Realizar las siguientes 9 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
- getch(): obtiene un carácter de teclado y lo muestra en pantalla

pausa()

Usamos esta función implementada en uno de los ejercicios de ejemplo de la “P2” para usarlo cada vez que ejecutamos a una de las funciones implementadas en esta práctica.

```
void pausa(){
    union REGS inregs, outregs;
    inregs.h.ah = 8;
    int86(0x21, &inregs, &outregs);
}
```

En esta función de pausa se usa la función de leer una tecla sin mostrarla en pantalla $AH = 8$

setcursortype()

Pasaremos como parámetro un entero para seleccionar desde el método el tipo de cursor que queremos utilizar.

```
void setcursortype(int cursor){
    union REGS inregs, outregs;
    inregs.h.ah = 0x01;

    switch(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);
}
```

*Este método viene implementado en el PDF de la P2

Consiste en utilizar la función 1 de entrada en *AH*, en *CH* será el número de línea inicial y en *CL* el número de línea final.

Ejecución:

```
C:\PRACTI~1>P2.EXE
Cursor invisible
Cursor normal

DOSBox 0.74, Cpu speed
HAVE FUN!
The DOSBox Team http://www.dos
Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount C /home/manuel/Escrito
Drive C is mounted as local dire
Z:\>keyb sp
Keyboard layout sp loaded for co
Z:\>path c:\bc\bin
Z:\>c:
C:\>cd PRACTI~1
C:\PRACTI~1>P2.EXE
Cursor invisible
Cursor normal
Cursor grueso
```

El cursor grueso nos lo sitúa arriba por haber ejecutado la instrucción `gotoxy(0,30)` justo después de utilizar el cursor grueso.

gotoxy()

Posicionaremos el cursor en unas coordenadas dadas por dos parámetros enteros.

```
void gotoxy(int posicionX, int posicionY){
    union REGS inregs, outregs;
    inregs.h.dh = posicionX;
    inregs.h.dl = posicionY;
    inregs.h.bh = 0;
    inregs.h.ah = 2;          //función para posicionar el cursor
    int86(0x10, &inregs, &outregs); //interrupción BIOS para pantalla
}
```

Utilizamos el número de función = 2

setvideomode()

```
void setvideomode(unsigned char modo){  
    union REGS inregs, outregs;  
    inregs.h.ah = 0x00;  
    inregs.h.al = modo;  
    int86(0x10,&inregs,&outregs);  
}
```

La función de número 0, usada para elegir el modo de vídeo deseado, en mi caso he seleccionado el 3-texto y el 4-grafico.

Ejecución:

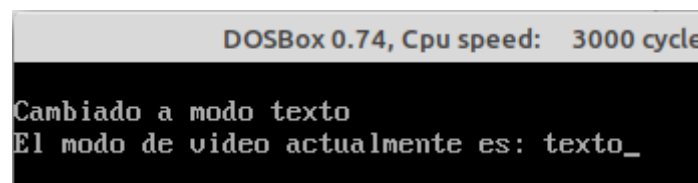


getvideomode()

```
int getvideomode(){  
    union REGS inregs, outregs;  
    int modo;  
    inregs.h.ah = 0xF;  
    int86(0x10,&inregs,&outregs);  
    modo = outregs.h.al;  
    return modo;  
}
```

La función para averiguar el modo de vídeo actual usa el número de función “Fh”, en *outregs* obtendremos la salida y en la variable *AL* obtendremos el identificador exacto del modo de vídeo actual.

Ejecución:



textcolor()* y *textbackground()

Disponemos de dos variables globales en el código llamadas “color_texto y color_fondo”, en estos métodos lo que haremos básicamente es sobrescribir esas variables con los colores que les pasemos como parámetros..

```
void textcolor(int color){
    color_texto = color;
}

//textbackground(): modifica el c
void textbackground(int color){
    color_fondo = color;
}
```

Ejecución:

```
//textcolor() y textbackground()
textcolor(5); //magenta
textbackground(2); //verde
printf("\nSe ha modificado el color del texto y color de fondo");
```

clrscr()

Limpia la pantalla, usamos según nos indica en el PDF de la práctica, usamos este scroll en concreto(hacia arriba)

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

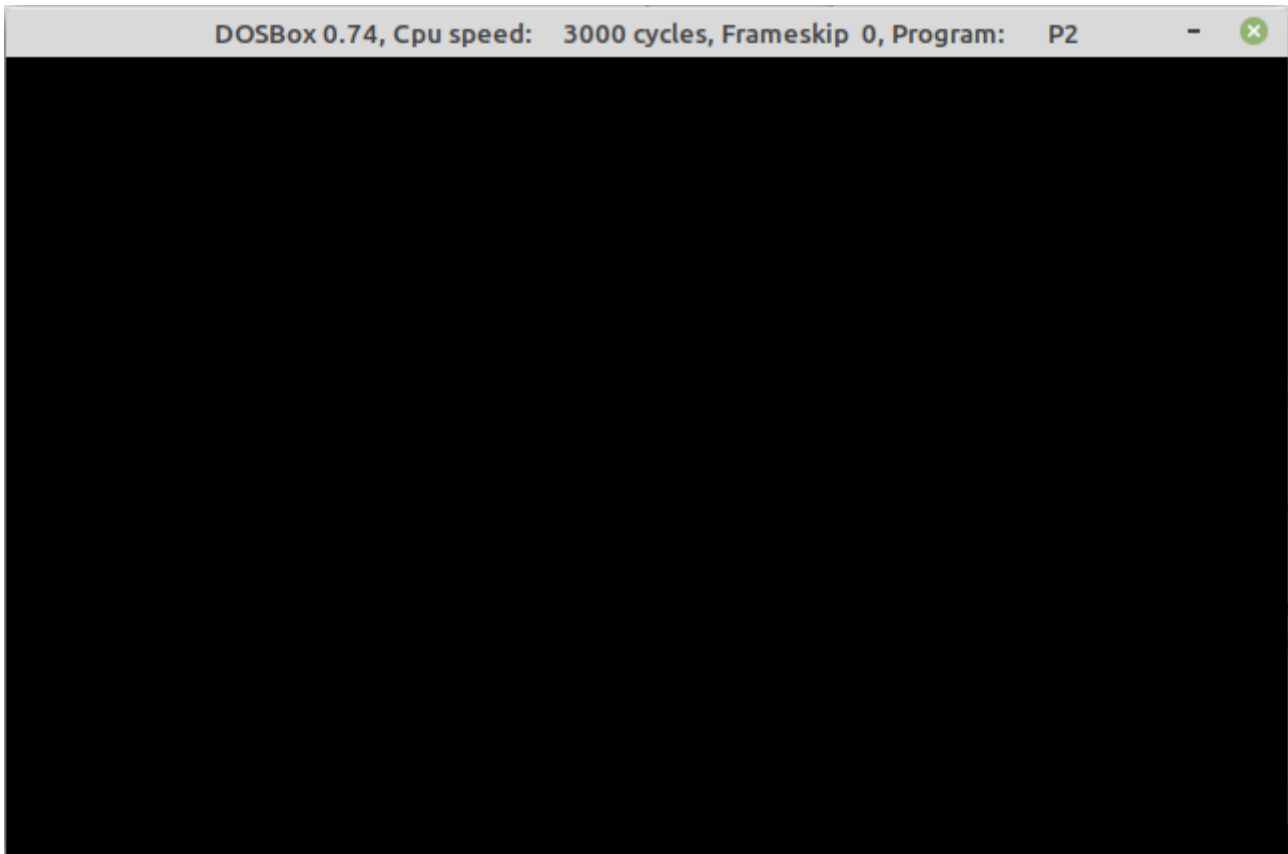
DL = columna de la esquina inferior derecha

~ ~ ~ ~ ~

y en código sería:

```
void clrscr(){
    union REGS inregs, outregs;
    inregs.h.ah = 6;
    inregs.h.al = 0;
    inregs.h.bh = 0;
    inregs.h.ch = 0;
    inregs.h.cl = 0;
    inregs.h.dh = 100;
    inregs.h.dl = 100;
    int86(0x10, &inregs, &outregs);
}
```

Ejecución:



cputchar()

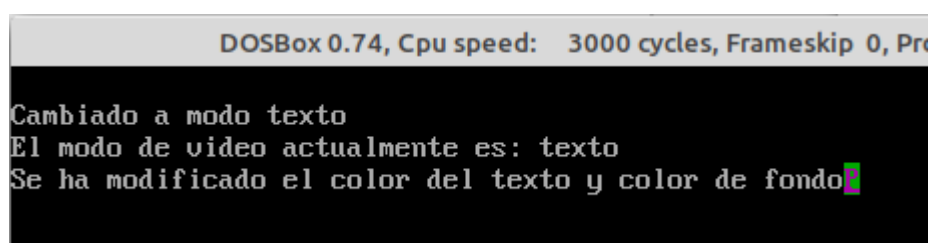
Escribiremos un carácter pasado como parámetro, este carácter se escribirá con el color de texto y fondo indicado establecido con las funciones anteriormente descritas.

```
void cputchar(char character){
    union REGS inregs, outregs;
    //int codigoASCII = int (character);
    inregs.h.ah = 9;
    inregs.h.al = character;
    inregs.h.bl = color_fondo<<4 | color_texto;
    inregs.h.bh = 0x00;
    inregs.x.cx = 1;
    int86(0x10, &inregs, &outregs);
}
```

Usamos la función 9.

En el byte del color, los 4 primeros bits fijan el color de fondo y los 4 últimos bits fijan el color de texto del carácter. Por eso utilizamos *color_fondo<<4* para desplazar esos 4 bits al principio.

Ejecución:

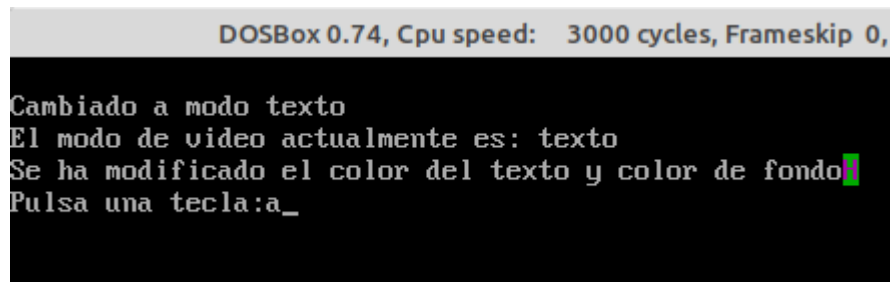


getche()

```
int getche(){ //Devuelvo un entero porque el valor almacenado en al es el codigo ASCII
    union REGS inregs, outregs;
    int character;
    inregs.h.ah = 1;
    int86(0x21, &inregs, &outregs);
    character = outregs.h.al;
    return character;
}
```

Se usa la función 1, el método devuelve un entero porque el valor almacenado en *al* es el código ASCII del carácter. Este método espera la pulsación de una tecla mostrándola por pantalla.

Ejecución:



DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0,

```
Cambiado a modo texto
El modo de video actualmente es: texto
Se ha modificado el color del texto y color de fondo
Pulsa una tecla:a_
```