UNIVERSIDAD DE GRANADA

Periféricos y Dispositivos de Interfaz Humana

Práctica 1 – Entrada y salida utilizando interrupciones con lenguaje C

Arturo Alonso Carbonero

ÍNDICE

1. Introducción

El objetivo de esta práctica es desarrollar una serie de funciones en lenguaje C que empleen interrupciones software para realizar una serie de acciones, tales como modificar el estado del cursor, modificar el modo de vídeo o cambiar el color de los caracteres que se muestran por pantalla.

A lo largo de esta memoria, las imágenes contendrán la función correspondiente a cada apartado, donde se muestra el código de la misma y algún que otro comentario con información útil. En el directorio P1 del repositorio de la asignatura aparecerán el archivo “practica1.C” y esta memoria. Dentro del fichero del código, se encuentra la estructura de la interrupción 86 (int86) que se emplea en las funciones. Algunas de las funciones no requieren de explicación por su simplicidad.

2. Requisitos mínimos

2.1. Colocar el cursor en una posición determinada

Función que recibe como parámetros las coordenadas (x, y) donde se desea colocar el cursor.

```
54  // gotoxy() -> Función para colocar el cursor en una posición determinada
55
56  void gotoxy(int x, int y){
57    union REGS inregs, outregs;
58
59    inregs.h.ah=0x02;
60    inregs.h.bh=0x00;
61    inregs.h.dh=x; // Coordenada x
62    inregs.h.dl=y; // Coordenada y
63
64    int86(0x10, &inregs, &outregs);
65  }
```

2.2. Fijar el aspecto del cursor

Función que recibe como parámetro el tipo de cursor que se desea aplicar. Compara el valor del parámetro y establece el tipo del cursor según el mismo.

```
69  // setcursortype() -> Función para fijar el aspecto del cursor (INVISIBLE, NORMAL y GRUESO)
70
71  void setcursortype(int tipo_cursor){
72    union REGS inregs, outregs;
73    inregs.h.ah=0x01;
74
75    switch(tipo_cursor){
76      case 0: // Invisible
77        inregs.h.ch=010;
78        inregs.h.cl=000;
79        break;
80      case 1: // Normal
81        inregs.h.ch=010;
82        inregs.h.cl=010;
83        break;
84      case 2: // Grueso
85        inregs.h.ch=000;
86        inregs.h.cl=010;
87        break;
88    }
89
90    int86(0x10, &inregs, &outregs);
91  }
```

2.3. Fijar el modo de vídeo

Función que recibe como parámetro el tipo de vídeo que se desea fijar.

```
95  // setvideomode() -> Función para fijar el modo de vídeo deseado
96
97  void setvideomode(int x){
98    union REGS inregs, outregs;
99
100   inregs.h.ah=0x00;
101   inregs.h.al=x;
102
103   int86(0x10, &inregs, &outregs);
104 }
```

2.4. Obtener el modo de vídeo actual

Función que recoge el tipo de vídeo actual tras la llamada a la interrupción 86, almacenado en ‘al’, comprueba si es de tipo texto o tipo gráfico y muestra el resultado.

```
108 // getvideomode() -> Función para obtener el modo de vídeo actual
109
110 void getvideomode(){
111   union REGS inregs, outregs;
112   int modo;
113   int numColumnas;
114
115   inregs.h.ah=0xF;
116
117   int86(0x10, &inregs, &outregs);
118
119   modo=outregs.h.al; // Modo actual
120   numColumnas=outregs.h.ah; // Número de columnas en los modos de texto
121
122   if(modo<=3 || modo==7){ // Según la tabla
123     printf("Texto");
124   }else{
125     printf("Grafico");
126   }
127 }
```

2.5. Modificar el color de primer plano

Función que recibe como parámetro el color con el que se mostrarán los caracteres en primer plano, es decir, el color de los propios caracteres. Además, muestra el carácter indicado un número de repeticiones (en este caso, el carácter ‘a’ 10 veces, aunque podría recibirlos como parámetros).

```
131 // textcolor() -> Función para modificar el color de primer plano con el que se mostrarán los caracteres
132
133 void textcolor(int colorTexto){
134   union REGS inregs, outregs;
135
136   inregs.h.ah=0x09;
137   inregs.h.al=97; // 'a'
138   inregs.h.bl=colorTexto;
139   inregs.h.bh=0x00;
140   inregs.x.cx=10; // Número de repeticiones
141
142   int86(0x10, &inregs, &outregs);
143 }
```

2.6. Modificar el color de fondo

Función que recibe como parámetros el color con el que se desean mostrar los caracteres en primer plano y el color de fondo. En cuanto a los caracteres a mostrar y al número de repeticiones, es equivalente a la función anterior (en este caso, se muestra ‘b’ 10 veces).

```
147  // textbackground() -> Función para modificar el color de fondo con el que se mostrarán los caracteres
148
149  void textbackground(int colorTexto, int colorFondo){
150    union REGS inregs, outregs;
151    int color=colorFondo << 4 | colorTexto;
152
153    inregs.h.ah=0x09;
154    inregs.h.al=98; // 'b'
155    inregs.h.bl=color;
156    inregs.h.bh=0x00;
157    inregs.x.cx=10; // Número de repeticiones
158
159    int86(0x10, &inregs, &outregs);
160  }
```

2.7. Borrar toda la pantalla

Función para eliminar el contenido de la pantalla.

```
164  // clrscr() -> Borrar toda la pantalla
165
166  void clrscr(){
167    union REGS inregs, outregs;
168
169    inregs.h.ah=0x15;
170    int86(0x10, &inregs, &outregs);
171    inregs.h.ah=0x00;
172    int86(0x10, &inregs, &outregs);
173  }
```

2.8. Escribir en pantalla un carácter con el color actual

Función que recibe como parámetro un carácter que, posteriormente, se mostrará por pantalla.

```
177  // cputchar() -> Escribe un carácter en pantalla con el color acual
178
179  void cputchar(char c){
180    union REGS inregs, outregs;
181
182    inregs.h.ah=2;
183    inregs.h.dl=c;
184
185    int86(0x21, &inregs, &outregs);
186  }
```

2.9. Obtener un carácter del teclado y mostrarlo en pantalla

Función que devuelve un carácter recibido por teclado.

```
190  // getche() -> Obtiene un carácter de teclado y lo muestra en pantalla
191
192  int getche(){
193    union REGS inregs, outregs;
194    int caracter;
195
196    inregs.h.ah=0x01;
197
198    int86(0x21, &inregs, &outregs);
199
200    caracter=outregs.h.al; // Código ASCII del carácter
201    return caracter;
202  }
```

3. Requisitos ampliados

3.1. Dibujar un cuadrado en modo texto

```
208  // dibujarCuadradoModoTexto() -> Función para dibujar un cuadrado en pantalla. recibe como parámetros
209  //                               las coordenadas de las esquinas superior izquierda e inferior derecha
210  //                               del cuadrado, el color de primer plano y el color de fondo
211
212  void dibujarCuadradoModoTexto(int f1, int c1, int f2, int c2, int primerPlano, int fondo){
213    union REGS inregs, outregs;
214
215    inregs.h.ah=0x06;
216    inregs.h.al=0; // Número de líneas a desplazar. Si al=0 => borra la zona
217    inregs.h.bh=fondo << 4 | primerPlano; // Color para los espacios en blanco
218    inregs.h.ch=f1; // Fila de la esquina superior izquierda
219    inregs.h.cl=c1; // Columna de la esquina superior izquierda
220    inregs.h.dh=f2; // Fila de la esquina inferior derecha
221    inregs.h.dl=c2; // Columna de la esquina inferior derecha
222
223    int86(0x10, &inregs, &outregs);
224  }
```

3.2. Establecer modo gráfico CGA para crear un dibujo

Función para mostrar por pantalla un píxel iluminado en las coordenadas y del color recibidos como parámetros. Para crear un dibujo, se entra previamente en modo gráfico (mediante la función setvideomode()) y se llama a la función de la imagen para representar los píxeles deseados.

```
228  // modoGraficoCGA() -> Función para crear dibujos sencillos en pantalla
229
230  void modoGraficoCGA(int x, int y, int color){
231    union REGS inregs, outregs;
232
233    inregs.h.ah=0x0C;
234    inregs.x.cx=x;
235    inregs.x.dx=y;
236    inregs.h.al=color;
237
238    int86(0x10, &inregs, &outregs);
239  }
```