Llamadas al sistema
Índice

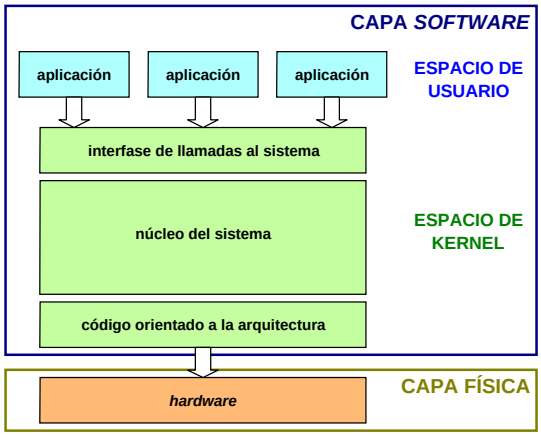
Llamadas al sistema
1. Fundamentos

1. Fundamentos

Índice

Llamadas al sistema
1.1. Concepto

- Mecanismo por el cual las aplicaciones solicitan servicios al sistema operativo



Llamadas al sistema
1.1. Concepto

- Las operaciones de comunicación con dispositivos periféricos o con controladores del sistema se podrían realizar en cada aplicación pero…
  - esta solución no genera aplicaciones “portables”
  - sobrecarga el desarrollo de aplicaciones
- Es más eficiente confiar estas operaciones al sistema operativo…
  - que ofrece una “máquina abstracta” que opera de una manera normalizada
  - funciona del mismo modo independientemente de la capa física ocultando los detalles de implementación

Llamadas al sistema
1.1. Concepto

- **Clases de servicios:**
  - **Control de procesos → creacíón, ejecución, sincronización, reserva/liberación de memoria…**
  - **Manejo de ficheros → creación, borrado, lectura, escritura…**
  - **Manejo de dispositivos → solicitud, configuración…**
  - **Información → fecha, hora, sistema, procesos…**
  - **Comunicación → creación de conexión, envío y recepción de mensajes…**

Llamadas al sistema
1.1. Concepto

- **Servicios típicos son:**
  - **`open`** → **abre (y crea) un descriptor de fichero[1]**
  - **`read`** → **lee un descriptor de fichero**
  - **`write`** → **escribe un descriptor de fichero**
  - **`close`** → **cierra un descriptor de fichero**
  - **`wait`** → **espera al cambio de estado de un proceso**
  - **`exec`** → **ejecuta un fichero**
  - **`fork`** → **crea un proceso hijo**
  - **`exit`** → **terminación normal de un proceso**
  - **`kill`** → **envía una señal a un proceso**

  - Existen cientos de servicios

[1] **incluye *stdin*, *stdout*, *stderr***

Llamadas al sistema
1.1. Concepto

- **Las llamadas al sistema implican:**

  1. **detención de la aplicación que invoca la llamada al sistema salvando su estado**
  2. **transferencia de control (salto) a código privilegiado (núcleo del sistema)**
  3. **una vez finalizada la llamada, devolución del control a la aplicación**

     ➡ conmutación de contexto y conmutación de modo de ejecución (entre modo usuario y modo supervisor)

Llamadas al sistema
1.2. Librerías

- **Generalmente los sistemas proporcionan una librería o API (*Application Programming Interface*) que permite relacionar las aplicaciones con el sistema**

  ➡ **La librería cuenta con funciones que empaquetan las llamadas al sistema con el fin de simplificar la escritura del código**

  ➡ **La función permite pasar los argumentos en los registros adecuados (y en la pila, en su caso) obviando detalles del ABI (*Application Binary Interface*) [1]**

[1] **ABI determina el orden en el que se pasan argumentos, entre otras cosas**

Llamadas al sistema
1.2. Librerías

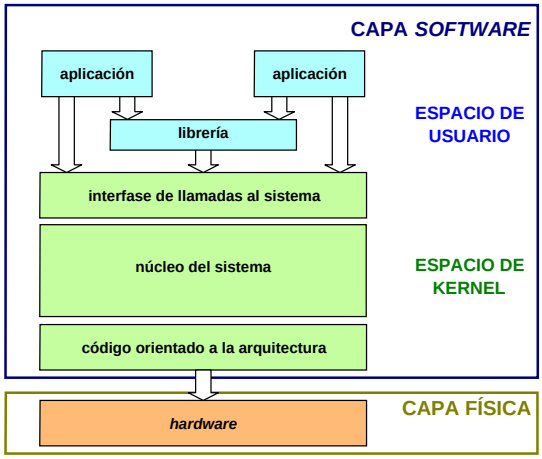
➡ Por ejemplo, leer carácter del teclado

`_getch()` → `int 21h` `AH = 08h` (DOS)

→ `int 80h` `EAX = 03h` (Linux)

Llamadas al sistema
1.2. Librerías

- Esquema de uso



Llamadas al sistema
1.2. Librerías

- En sistemas DOS no existen librerías
  - …aunque algunos se refieren a la INT21h como API del DOS
- En sistemas Linux:
  - Librería de C → `glibc` (*gnu C library*)
- En sistemas Windows:
  - Win32 API → en librerías dinámicas `ntdll.dll`, `kernel32.dll`, `user32.dll` y `gdi32.dll` que incluye la librería *CRT* (*C-Run Time library*)

Llamadas al sistema
1.2. Librerías

- La librería GNU-C `glibc`
  - Es la biblioteca estándar de C de GNU (sistema operativo libre basado en el núcleo Linux)
  - Es muy portable → soporta gran cantidad de plataformas *hardware* y núcleos diferentes a Linux

Llamadas al sistema
1.3. Implementación

- Es necesario algún mecanismo de transferencia de control
  - …para ejecutar un código independiente de la aplicación
  - Normalmente se usa una interrupción
    - Involucra características específicas de la arquitectura
- Una vez transferido el control hay que cambiar a modo supervisor

Llamadas al sistema
1.4. Interrupciones

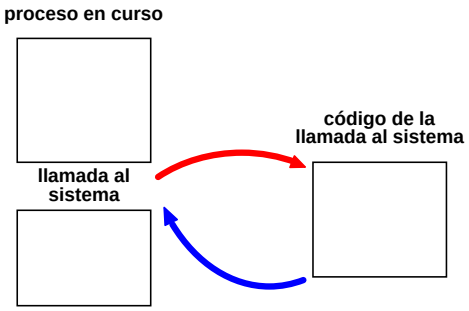
- Las interrupciones son procedimientos solicitados por número
  - …en lugar de por dirección
  - El número señala, dentro de una tabla, la dirección del procedimiento a ejecutar
  - El procedimiento o rutina de servicio es independiente del origen de la solicitud

- **Ventajas:**
  - **El uso de procedimientos numerados es muy flexible ya que cambiando la dirección de la tabla se puede cambiar la rutina de servicio**
  - **Los usuarios de las interrupciones no son responsables del código de las rutinas de servicio**
- **Inconvenientes:**
  - **Pueden ser lentas debido al cambio de contexto**

- **Tipos de interrupciones**
  - **Atendiendo al origen de la solicitud**
    - **Interrupciones *hardware*** → son asíncronas, es decir, se pueden disparar en cualquier instante; la solicitud se hace en hardware y suele provenir de dispositivos de E/S
    - **Excepciones** → son sincronas con la ejecución del código; suelen ser causadas por operaciones no permitidas (división por 0, desbordamiento, acceso no permitido) y disparadas por el procesador
    - **Interrupciones *software*** → las genera el propio código mediante la ejecución de una instrucción

Llamadas al sistema
1.4. Interrupciones

- **Las llamadas al sistema son interrupciones *software*, es decir, se invocan mediante la ejecución de una instrucción de llamada a interrupción dentro de la secuencia de código de la aplicación**



Llamadas al sistema
1.4. Interrupciones

- **Algunos ejemplos de instrucciones de interrupción:**
  - x86 → INT
  - x86-32 → SYSCALL/SYSRET (llamada rápida al sistema)
  - x86-32 → SYSENTER/SYSEXIT
  - IA64 → EPC (*Enter Privileged Code*)
  - Alpha → CALL PAL (*Privileged Architecture Library*)
  - *Call gate* → mecanismo (obsoleto) para cambiar a modo supervisor en el entorno x86 (no tuvo mucho éxito)

2. Llamadas al sistema bajo DOS
Índice

- **Se implementan usando la instrucción de interrupción *software* (INT *n*)**

    ➡ **donde *n* es un inmediato de tamaño *byte*, es decir, podemos tener hasta 256 interrupciones**

Llamadas al sistema
2.1. Soporte *hardware*

- La dirección de la rutina de servicio (ISR – *Interrupt Service Routine*) se encuentra en una tabla conocida como tabla de vectores de interrupción
- El número de la interrupción es el índice que sirve para acceder a cada dirección
- Cada dirección se denomina vector de interrupción y es un puntero de 32 bits (base y desplazamiento) al ISR correspondiente
- Ya que cada vector ocupa 4 bytes, el vector de la interrupción *n* estará en la posición *n* x 4

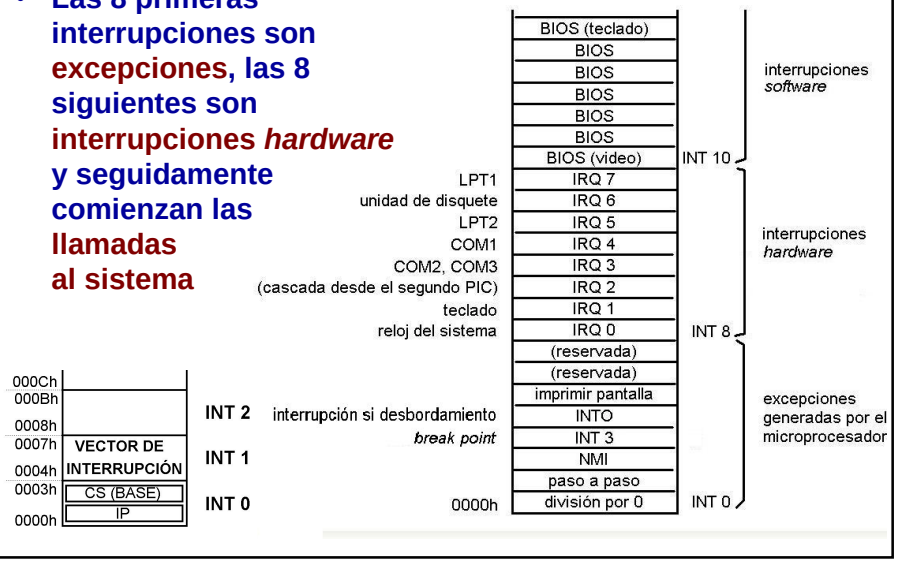
Llamadas al sistema
2.1. Soporte *hardware*

- La tabla de vectores de interrupción se sitúa en la parte más baja del mapa de memoria (00000h)
- Ocupa 1KB (4 bytes x 256)
- Inicialmente la carga el BIOS al arrancar el computador
- Posteriormente, se pueden cambiar los vectores

Mapa de memoria

Llamadas al sistema
2.1. Soporte *hardware*

- **Las 8 primeras interrupciones son excepciones, las 8 siguientes son interrupciones *hardware* y seguidamente comienzan las llamadas al sistema**



Llamadas al sistema
2.1. Soporte *hardware*

- **El mecanismo de tabla de vectores de interrupción es muy flexible ya que permite cambiar fácilmente las rutinas de servicio (ISR)**
  - **Para cambiar una rutina de servicio (ISR) basta con cargar un nuevo vector en la tabla apuntando al comienzo de dicha rutina que será**
    - o un **programa residente** (cargado en memoria)
    - o un procedimiento de un proceso en curso
  - **De esta manera, cada proceso puede disponer de ISRs diferentes**

Llamadas al sistema
2.1. Soporte *hardware*

- Cambio de vector de interrupción (I)
  - Método “manual” 1

```
xor ax, ax
mov es, ax
cli
mov es:4*n, offset rutina
mov es:4*n+2, seg rutina
sti
```


Llamadas al sistema
2.1. Soporte *hardware*

- Cambio de vector de interrupción (II)
  - Método “manual” 2 (transferencia atómica [1])

```
prutina dd rutina
xor di, di
mov es, di
mov di, 4*n
lds si, prutina
mov cx, 2
cld
cli
rep movsw
sti
```

[1] se llama atómica porque se hace en una única instrucción

- **Cambio de vector de interrupción (III)**
  - **Llamada al sistema**
    - Servicio 25h de la INT 21h
    - Es conveniente salvar el vector antiguo y volver a colocarlo cuando se finalice; para leer el vector antiguo se usa el servicio 35h de la INT 21h
    - Se pueden cambiar todos los vectores pero es usual hacerlo con 00h (división por 0), 04h (desbordamiento), 24h (error crítico), 23h (control-C)
    - Para dejar un programa residente se usa la llamada al sistema INT 27h

- **Cambio de vector de interrupción (ejemplo):**

```
            .DATA
mensaje     DB          "Desbordamiento",13,10,"$"
vector      DD          ?
            .CODE
inicio:     MOVE        AX, @DATA
            MOV         DS, AX

            MOV         AX, 3504h
            INT         21h
            MOV         WORD PTR VECTOR[2], ES
            MOV         WORD PTR VECTOR[0], BX

            PUSH        DS
            MOV         AX, CS
            MOV         DS, AX
            MOV         DX, OFFSET overflow
            MOV         AX, 2504h
            INT         21h
            POP         DS

            // TAREAS //

            LDS         DX, vector
            MOV         AX, 2504h
            INT         21h

            MOV         AX, 4C00h
            INT         21h
```

```
overflow    PROC        FAR
            STI

            MOV         AH, 09h
            MOV         DX, OFFSET mensaje
            INT         21h

            IRET
overflow    ENDP

            END INICIO
```

- **La nueva rutina de servicio a la INT 4 emite un mensaje**
- **Antes de terminar el programa se restaura el vector original**

Llamadas al sistema
2.1. Soporte *hardware*

- **Definición de rutinas de interrupción:**
  - **Una ISR es siempre un procedimiento far**
    - El vector tiene base y desplazamiento
  - **Termina con IRET ya que previamente salva el estado**

```
etiqueta    PROC FAR

      : : :       ;código

      IRET
etiqueta    ENDP
```


Llamadas al sistema
2.1. Soporte *hardware*

INT *n*

- **Cuando se llama a una interrupción se siguen estos pasos:**

1. Búsqueda del vector en la tabla → *n* x 4
2. PUSH *flags*, CS e IP
3. TF = 0; IF = 0
4. Salta a la rutina de atención

SP = SP – 2
***flags* → pila**
IF = 0
TF = 0
SP = SP – 2
CS → pila
CS = *n* * 4 + 2
SP = SP – 2
IP → pila
IP = *n* * 4

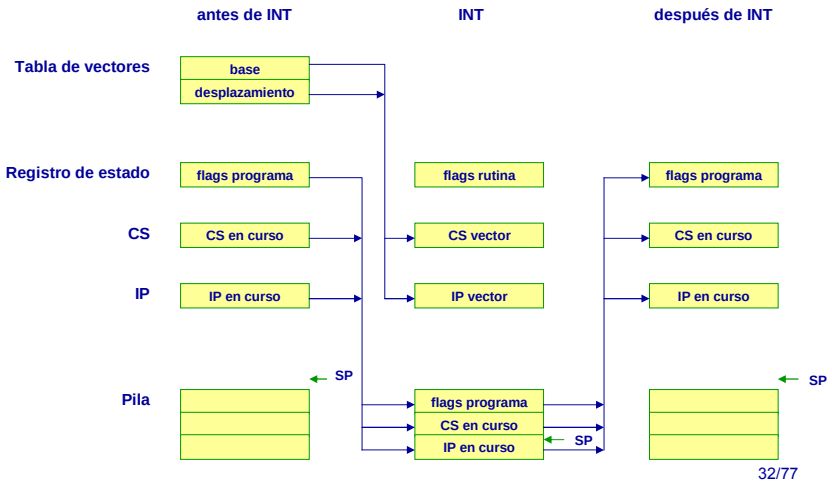
IRET

- **Cuando se regresa de una interrupción se siguen estos pasos:**

1. POP IP, CS y *flags*

pila → IP
SP = SP + 2
pila → CS
SP = SP + 2
pila → *flags*
SP = SP + 2

INT *n* – IRET



Llamadas al sistema
2.2. Paso de argumentos

- **Determinan el tipo de llamada al sistema, los datos de entrada y la ubicación de los valores devueltos**
- **Pueden ser valores o punteros a memoria**
- **Desde un punto de vista teórico, los argumentos se pueden pasar en registros o por la pila**
  - **Los registros ofrecen velocidad**
  - **La pila garantiza llamadas reentrantes**

Llamadas al sistema
2.2. Paso de argumentos

- **Con el fin de que las llamadas al sistema sean rápidas…**
  - **En la práctica el paso de argumentos se hace mediante registros**
  - **Por tanto, las llamadas al sistema no son reentrantes**
  - **Para que un sistema operativo sea reentrante, debemos deshabilitar las interrupciones durante la ejecución del código de cada llamada al sistema**
    - Esto aumenta la latencia (ralentiza el sistema)

Llamadas al sistema
2.2. Paso de argumentos

- Cada llamada al sistema (nivel de interrupción) cuenta, normalmente, con múltiples servicios
- El número de servicio se suele pasar en el registro AH
- El resto de argumentos se pasan en otros registros, ya sea de tamaño byte o *word*

Llamadas al sistema
2.3. Tipos de llamadas

- Las llamadas al sistema bajo DOS se articulan alrededor de dos tipos
  - BIOS → (*Basic Input Output System*) son rutinas básicas de entrada/salida
  - DOS → (*Disk Operating System*) son rutinas del sistema operativo con cierto nivel de abstracción que invocan en último término a los servicios BIOS

Llamadas al sistema
2.3.1. Interrupciones BIOS

- **El BIOS (*Basic Input Output System*) es un programa escrito en memoria ROM al que se transfiere el control desde la posición de arranque (`FFFF:0`)**
  - **Realiza la inicialización del sistema e invoca al cargador del sistema operativo**
  - **Ejecuta el *Power On Self Test* (POST)**
  - **Carga la tabla de vectores de interrupción**
  - **Carga el área de datos de la BIOS**
  - **Instala todo el código de las llamadas al sistema BIOS**
  - **Instala el código de algunas interrupciones *hardware***
  - **Transfiere el control al cargador del sistema operativo**

Llamadas al sistema
2.3.1. Interrupciones BIOS

- **Área de datos del BIOS**
  - **Se encuentra justo a continuación de la tabla de vectores de interrupción, entre `0040:0` y `0040:0FF`, es decir, 256 bytes**
  - **Contiene la siguiente información:**
    - Cantidad de memoria RAM
    - *Hardware* presente (puertos, direcciones, etc.)
    - *Buffer* y estado del teclado
    - Datos de video
    - Contador de *ticks* (reloj)
    - Datos del sistema (reset, *Ctrl-Break*, etc.)

Llamadas al sistema
2.3.1. Interrupciones BIOS

- **Tipos de interrupciones BIOS**
  - **Llamadas al sistema → entre INT 10h e INT 1Ah**
  - **Rutinas de usuario → entre INT 1Bh e INT 1Ch**
  - **Parámetros del BIOS → entre INT 1Dh e INT 1Fh**

  - **Las rutinas de usuario deben ser programadas por el usuario**
    - INT 1Bh → *Ctrl-Break* (por defecto IRET)
    - INT 1Ch → tic del reloj (por defecto IRET)

  - **Los parámetros del BIOS no son código, sino punteros a datos del BIOS**

Llamadas al sistema
2.3.1. Interrupciones BIOS

- **Llamadas al sistema BIOS:**
  - **INT 10h → acceso a la pantalla**
  - **INT 11h → información sobre el equipo físico**
  - **INT 12h → tamaño de memoria**
  - **INT 13h → acceso al diskette**
  - **INT 14h → acceso al puerto serie**
  - **INT 15h → cassette (hoy también servicios de teclado)**
  - **INT 16h → acceso al teclado**
  - **INT 17h → acceso a la impresora**
  - **INT 18h → BASIC**
  - **INT 19h → restaurar sistema**
  - **INT 1Ah → temporizador (reloj)**

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **INT 10h**
  - **Servicio AH = 00h → selecciona modo de video**
  - **Servicio AH = 02h → sitúa posición del cursor**
  - **Servicio AH = 03h → lee posición del cursor**
  - **Servicio AH = 06h y 07h → *scroll* arriba y abajo**
  - **Servicio AH = 08h → lee carácter y atributo en posición del cursor**
  - **Servicio AH = 09h → escribe carácter y atributo *n* veces a partir de la posición del cursor (modo texto)**
  - **Servicio AH = 0Ah → escribe carácter *n* veces a partir de la posición del cursor (modo texto)**
  - **Servicio AH = 0Ch → escribe un pixel en la posición indicada**
  - **Servicio AH = 0Dh → lee un pixel en la posición indicada**
  - **Servicio AH = 13h → escribe una cadena en la posición indicada**

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Selección de modo de video → `INT 10h` `AH=00h`**
  - **Argumentos**
    - AH = `00h`
    - AL = modo de video
  - **Devuelve**
    - nada

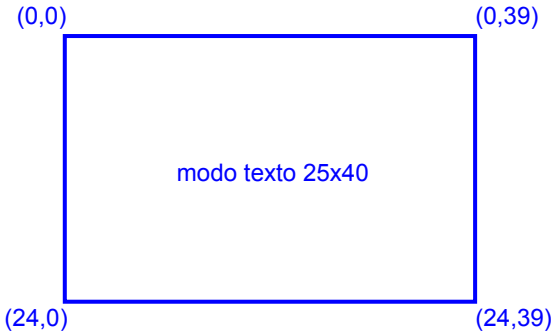
modo	resolución	colores	video
00h	40 x 25	16	texto
01h	40 x 25	16	texto
02h	80 x 25	16	texto
03h	80 x 25	16	texto
04h	20 x 200	4	gráfico
05h	20 x 200	4	gráfico
06h	640 x 200	2	gráfico
07h	80 x 25	2	texto
0Dh	20 x 200	16	gráfico
0Eh	640 x 200	16	gráfico
0Fh	640 x 350	2	gráfico
10h	640 x 350	4	gráfico
10h	640 x 350	16	gráfico
11h	640 x 480	2	gráfico
12h	640 x 480	16	gráfico
13h	20 x 200	256	gráfico

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Establece tamaño del cursor → `INT 10h` `AH=01h`**
  - **Argumentos**
    - `AH` = `01h`
    - `CH` = línea inicial del cursor (0 – 15)
    - `CL` = línea final del cursor (0 – 15)
  - **Devuelve**
    - nada

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Sitúa posición del cursor → `INT 10h` `AH=02h`**
  - **Argumentos**
    - `AH` = `02h`
    - `BH` = página de video
    - `DH` = fila
    - `DL` = columna
  - **Devuelve**
    - nada



Llamadas al sistema
2.3.1.1. Servicios BIOS de video

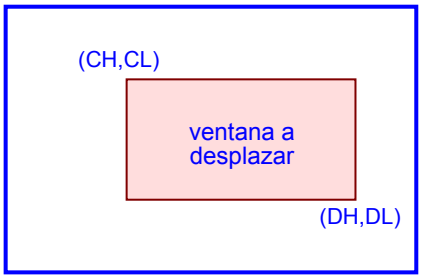
- **Lee posición del cursor →** `INT 10h` `AH=03h`
  - **Argumentos**
    - `AH` = `03h`
    - `BH` = página de video
  - **Devuelve**
    - `CH` = línea inicial del cursor (tamaño del cursor)
    - `CL` = línea final del cursor
    - `DH` = fila donde se encuentra el cursor
    - `DL` = columna donde se encuentra el cursor

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Establece página de video →** `INT 10h` `AH=05h`
  - **Argumentos**
    - `AH` = `05h`
    - `AL` = página de video
  - **Devuelve**
    - nada
  - **Observaciones**
    - El número de páginas de video depende del modo

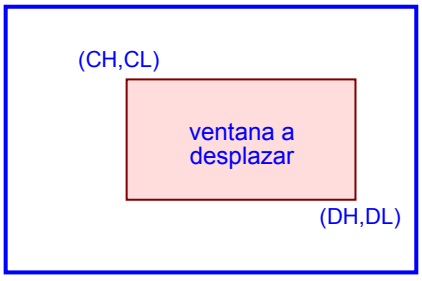
Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Desplaza texto hacía arriba → `INT 10h` `AH=06h`**
  - **Argumentos**
    - AH = 06h
    - AL = nº de filas a desplazar; si 0 se borra toda la ventana
    - BH = atributo a usar en líneas borradas
    - CH = fila de inicio de la ventana
    - CL = columna de inicio de la ventana
    - DH = fila final de la ventana
    - DL = columna final de la ventana
  - **Devuelve**
    - Nada



Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Desplaza texto hacía abajo → `INT 10h` `AH=07h`**
  - **Argumentos**
    - AH = 07h
    - AL = nº de filas a desplazar; si 0 se borra toda la ventana
    - BH = atributo a usar en líneas borradas
    - CH = fila de inicio de la ventana
    - CL = columna de inicio de la ventana
    - DH = fila final de la ventana
    - DL = columna final de la ventana
  - **Devuelve**
    - Nada



Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Lee carácter y atributo →** `INT 10h` `AH=08h`
  - **Argumentos**
    - AH = `08h`
    - BH = página de video
  - **Devuelve**
    - AH = atributo
    - AL = código ASCII del carácter

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Escribe carácter y atributo →** `INT 10h` `AH=09h`
  - **Argumentos**
    - AH = `09h`
    - AL = código ASCII del carácter
    - BH = página de video
    - BL = atributo
    - CX = cantidad de veces que se repite el carácter
  - **Devuelve**
    - nada

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Escribe carácter →** `INT 10h` `AH=0Ah`
  - **Argumentos**
    - AH = 0Ah
    - AL = código ASCII del carácter
    - BH = página de video
    - CX = cantidad de veces que se repite el carácter
  - **Devuelve**
    - nada
  - **Observaciones**
    - El atributo no se modifica

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Escribe un pixel →** `INT 10h` `AH=0Ch`
  - **Argumentos**
    - AH = 0Ch
    - AL = color del pixel
    - BH = página de video
    - CX = columna del pixel
    - DX = fila del pixel
  - **Devuelve**
    - nada
  - **Observaciones**
    - Sólo válido en modo gráfico

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Lee un pixel →** `INT 10h` `AH=0Dh`
  - **Argumentos**
    - AH = 0Dh
    - BH = página de video
    - CX = columna del pixel
    - DX = fila del pixel
  - **Devuelve**
    - AL = color del pixel
  - **Observaciones**
    - Sólo válido en modo gráfico

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Lee el modo de video actual →** `INT 10h` `AH=0Fh`
  - **Argumentos**
    - AH = 0Fh
  - **Devuelve**
    - AL = modo de video actual
    - AH = cantidad de caracteres por línea en el modo actual
    - BH = página de video activa
  - **Observaciones**
    - El interés de esta función es facilitar una respuesta adecuada cuando un programa tiene que dar un servicio gráfico pero no controla el modo de video (p.ej. un *driver*)

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Escribe cadena →** **`INT 10h AH=13h`**
  - **Argumentos**
    - AH = 13h
    - AL = modo
      - bit 0: 0 no mueve cursor; 1 mueve cursor
      - bit 1: 0 BL es el atributo; 1 el atributo en la cadena
    - BH = página de video
    - BL = atributo si es el caso
    - CX = longitud de la cadena
    - DH = fila del pixel
    - DL = columna del pixel
    - ES:BP = puntero a la cadena
  - **Devuelve**
    - nada

Llamadas al sistema
2.3.1.1. Servicios BIOS de video

- **Ejemplos:**

```
mov ah, 02h     ;servicio 02h (sitúa el cursor)
mov bh, 00h     ;página 0
mov dh, fila
mov dl, columna
int 10h

mov ah, 09h     ;servicio 09h (escribe carácter y
                               atributo n veces)
mov bh, 00h     ;página 0
mov al, char    ;carácter
mov bl, attr    ;atributo
mov cx, n       ;repetición
int 10h
```

Llamadas al sistema
2.3.1.2. Servicios BIOS de teclado

- **INT 16h**
  - **Servicio AH = 00h → lee *buffer* de teclado y avanza puntero; si no hay tecla pulsada espera (bloqueante)**
  - **Servicio AH = 01h → lee estado del *buffer* (vacío/lleno) y si hay tecla pulsada devuelve sus códigos (no bloqueante)**
  - **Servicio AH = 02h → lee byte de estado del teclado**
  - **Servicios para teclados de 107 teclas:**
  - **Servicio AH = 10h → igual a 00h**
  - **Servicio AH = 11h → igual a 01h**
  - **Servicio AH = 12h → igual a 02h**

Llamadas al sistema
2.3.1.2. Servicios BIOS de teclado

- **Lee *buffer* de teclado →** `INT 16h` `AH=10h`
  - **Argumentos**
    - AH = `10h`
  - **Devuelve**
    - AH = *scan code*
    - AL = código ASCII de la tecla pulsada
  - **Observaciones**
    - Teclado expandido (107 teclas o más)
    - Para teclado antiguo usar el servicio AH = `00h`

Llamadas al sistema
2.3.1.2. Servicios BIOS de teclado

- **Lee estado del *buffer* → `INT 16h` `AH=11h`**
  - **Argumentos**
    - AH = 11h
  - **Devuelve**
    - ZF = 1 si no hay pulsación
    - ZF = 0 si se ha pulsado una tecla
    - AH = *scan code*
    - AL = código ASCII de la tecla pulsada
  - **Observaciones**
    - Comprueba si hay pulsación pero la tecla pulsada no se saca del *buffer* de teclado

Llamadas al sistema
2.3.1.2. Servicios BIOS de teclado

- **Lee bytes de estado → `INT 16h` `AH=12h`**
  - **Argumentos**
    - AH = 12h
  - **Devuelve**
    - AL = byte de estado 1
    - AH = byte de estado 2
  - **Observaciones**
    - Los bytes de estado indican si ciertas teclas están pulsadas o no

Llamadas al sistema
2.3.1.2. Servicios BIOS de teclado

- **Bytes de estado de teclado**

byte de estado 1	significado
x x x x x x x 1	*Shift* derecha pulsado
x x x x x x 1 x	*Shift* izquierda pulsado
x x x x x 1 x x	*Ctrl* pulsado
x x x x 1 x x x	*Alt* pulsado
x x x 1 x x x x	*Scroll lock* activado
x x 1 x x x x x	*Num lock* activado
x 1 x x x x x x	*Caps lock* activado
1 x x x x x x x	*Insert lock* activado

byte de estado 2	significado
x x x x x x x 1	*Ctrl* izquierda pulsado
x x x x x x 1 x	*Alt* izquierda pulsado
x x x x x 1 x x	*Sys req* pulsado
x x x x 1 x x x	*Pause* activado
x x x 1 x x x x	*Scroll lock* pulsado
x x 1 x x x x x	*Num lock* pulsado
x 1 x x x x x x	*Caps lock* pulsado
1 x x x x x x x	*Insert lock* pulsado

Llamadas al sistema
2.3.1.2. Servicios BIOS de teclado

- **Ejemplo:**

```
                ;teclado expandido (107 teclas o más)

mov ah, 10h     ;servicio 10h (lee buffer teclado)
int 16h         ;devuelve ASCII en AL y
                ;scan code en AH
```

Llamadas al sistema
2.3.2. Interrupciones DOS

- El sistema operativo DOS carece de interfase gráfica y es incapaz de detectar el *hardware* (eso se lo facilita el BIOS)
- Dispone de una serie de llamadas al sistema que suponen una capa de abstracción superior a las llamadas de la BIOS
- Algunos consideran a la INT 21h como la API del DOS

Llamadas al sistema
2.3.2. Interrupciones DOS

- Tipos de interrupciones DOS
  - INT 20h → terminar programa
  - INT 21h → petición de servicio DOS
  - INT 22h → direccíón de terminación
  - INT 23h → dirección de rutina *Ctrl-Break*
  - INT 24h → error crítico
  - INT 25h → lectura de disco (por sectores)
  - INT 26h → escritura en disco (por sectores)
  - INT 27h → terminar y dejar residente

Llamadas al sistema
2.3.2. Interrupciones DOS

- **INT 21h**
  - **Servicios 00 a 0Ch → entrada/salida de carácter**
  - **Servicios 0D a 46h → sistema de ficheros**
  - **Servicio AH = 02h → escribe carácter por pantalla**
  - **Servicio AH = 09h → escribe cadena terminada en '$' por pantalla**
  - **Servicio AH = 01h → espera a leer carácter del teclado**
  - **Servicio AH = 0Ah → lee cadena de caracteres**
  - **Servicio AH = 05h → imprimir carácter**
  - **Servicio AH = 0Fh → abrir fichero existente**
  - **Servicio AH = 1Bh → obtener la dirección de la FAT**
  - **Servicio AH = 39h → crear directorio**

Llamadas al sistema
2.3.2.1. Servicios DOS de E/S de caracteres

- **Entrada bloqueante de carácter con eco → `INT 21h AH=01h`**
  - **Argumentos**
    - AH = `01h`
  - **Devuelve**
    - AL = código ASCII del carácter leído

Llamadas al sistema
2.3.2.1. Servicios DOS de E/S de caracteres

- **Salida de carácter →** `INT 21h` `AH=02h`
  - **Argumentos**
    - AH = 02h
    - DL = código ASCII del carácter
  - **Devuelve**
    - nada

Llamadas al sistema
2.3.2.1. Servicios DOS de E/S de caracteres

- **Carácter a la impresora →** `INT 21h` `AH=05h`
  - **Argumentos**
    - AH = 05h
    - DL = código ASCII del carácter
  - **Devuelve**
    - nada
  - **Observaciones**
    - Se envía el carácter al puerto paralelo

Llamadas al sistema
2.3.2.1. Servicios DOS de E/S de caracteres

- **E/S de carácter →** **`INT 21h`** **`AH=06h`**
  - **Argumentos**
    - `AH = 06h`
    - `DL` = código ASCII del carácter (si salida)
    - `DL = 0FFh` (si entrada)
  - **Devuelve**
    - `ZF = 1` si no hay pulsación
    - `ZF = 0` si se ha pulsado una tecla
    - `AL` = código ASCII leído (`DL = FFh`) o escrito (`DL ≠ FFh`)
  - **Observaciones**
    - La entrada es no bloqueante

Llamadas al sistema
2.3.2.1. Servicios DOS de E/S de caracteres

- **Salida de cadena →** **`INT 21h`** **`AH=09h`**
  - **Argumentos**
    - `AH = 09h`
    - `DS` = segmento de la cadena
    - `DX` = *offset* de la cadena
  - **Devuelve**
    - nada
  - **Observaciones**
    - Muestra una cadena en la salida estándar

Llamadas al sistema
2.3.2.1. Servicios DOS de E/S de caracteres

- **Lee cadena →** **`INT 21h`** **`AH=0Ah`**
  - **Argumentos**
    - `AH = 0Ah`
    - `DS` = segmento del *buffer*
    - `DX` = *offset* del *buffer*
  - **Devuelve**
    - `buffer[1]` = número de caracteres de la cadena
    - `buffer[2]` = cadena
  - **Observaciones**
    - Declaración del *buffer* → `buffer DB n,?,n DUP(?)`
    - El tamaño máximo de la cadena incluyendo ENTER es *n*
    - Si intento escribir más de *n* caracteres, se ignoran

Llamadas al sistema
2.3.2.1. Servicios DOS de E/S de caracteres

- **Lee estado de la entrada →** **`INT 21h`** **`AH=0Bh`**
  - **Argumentos**
    - `AH = 0Bh`
  - **Devuelve**
    - `AL = 00h` si no hay carácter disponible
    - `AL = 0FFh` si hay carácter disponible
  - **Observaciones**
    - Lee la entrada estándar

Llamadas al sistema
2.3.2.1. Servicios DOS de E/S de caracteres

- **Ejemplo:**

```
                        ;asumo que DS está iniciado

mov ah, 09h             ;servicio 09h (escribe cadena)
mov dx, offset cadena   ;offset de cadena
int 21h
```


Llamadas al sistema
2.3.2.2. Servicios DOS de sistema de ficheros

- **Descripción →** `INT 21h  AH=00h`
  - **Argumentos**
    - AH = `00h`
  - **Devuelve**
    - AL = byte
    - AH = byte
  - **Observaciones**
    - Los

Llamadas al sistema
2.3.2.2. Servicios DOS de sistema de ficheros

- **Descripción →** `INT 21h AH=00h`
  - **Argumentos**
    - AH = `00h`
  - **Devuelve**
    - AL = byte
    - AH = byte
  - **Observaciones**
    - Los

Llamadas al sistema
2.3.2.3. Servicios DOS de acceso a disco

- **Descripción →** `INT 21h AH=00h`
  - **Argumentos**
    - AH = `00h`
  - **Devuelve**
    - AL = byte
    - AH = byte
  - **Observaciones**
    - Los

Llamadas al sistema
2.3.2.4. Servicios DOS de TSR

- **Descripción →** `INT 21h  AH=00h`
  - **Argumentos**
    - AH = `00h`
  - **Devuelve**
    - AL = byte
    - AH = byte
  - **Observaciones**
    - Los