

# Sistemas Operativos

## Práctica 6 – Interfaz E/S y Sistema de archivos

### Notas preliminares

- Los ejercicios marcados con el símbolo ★ constituyen un subconjunto mínimo de ejercitación. Sin embargo, aconsejamos fuertemente hacer todos los ejercicios.

---

### Parte 1 – Interfaz de E/S

Un SO provee la siguiente API para operar con un dispositivo de E/S. Todas las operaciones retornan la constante `IO_OK` si fueron exitosas o la constante `IO_ERROR` si ocurrió algún error.

<code>int open(int device_id)</code>	Abre el dispositivo
<code>int close(int device_id)</code>	Cierra el dispositivo
<code>int read(int device_id, int* data)</code>	Lee el dispositivo <code>device_id</code>
<code>int write(int device_id, int* data)</code>	Escribe el valor en el dispositivo <code>device_id</code>

Para ser cargado como un Driver válido por el sistema operativo, el Driver debe implementar los siguientes procedimientos:

Función	Invocación
<code>int driver_init()</code>	Durante la carga del SO
<code>int driver_open()</code>	Al solicitarse un open
<code>int driver_close()</code>	Al solicitarse un close
<code>int driver_read(int *data)</code>	Al solicitarse un read
<code>int driver_write(int *data)</code>	Al solicitarse un write
<code>int driver_remove()</code>	Durante la descarga del SO

Para la programación de un driver, se dispone de las siguientes syscalls:

<code>void OUT(int IO_address, int data);</code>	Escribe data en el registro de E/S
<code>int IN(int IO_address);</code>	Retorna el valor almacenado en el registro de E/S
<code>int request_irq(int irq, void* handler)</code>	Permite asociar el procedimiento handler a la interrupción irq Retorna <code>IRQ_ERROR</code> si ya está asociada a otro handler
<code>int free_irq(int irq)</code>	Libera la interrupción irq del procedimiento asociado

### Ejercicio 1

Un cronómetro posee 2 registros de E/S:

- `CHRONO_CURRENT_TIME` que permite leer el tiempo medido,
- `CHRONO_CTRL` que permite ordenar al dispositivo que reinicie el contador.

El cronómetro reinicia su contador escribiendo la constante `CHRONO_RESET` en el registro de control.

Escribir un Driver para manejar este cronómetro. Este Driver debe retornar el tiempo actual cuando invoca la operación `read()`. Si el usuario invoca la operación `write()`, el cronómetro debe reiniciarse.

### Ejercicio 2

Una tecla posee un único registro de E/S : `BTN_STATUS`. Sólo el bit menos significativo y el segundo bit menos significativo son de interés:

- `BTN_STATUS0`: vale 0 si la tecla no fue pulsada, 1 si fue pulsada.
- `BTN_STATUS1`: escribir 0 en este bit para limpiar la memoria de la tecla.

Escribir un Driver para manejar este dispositivo de E/S. El Driver debe retornar la constante `BTN_PRESSED` cuando se presiona la tecla. Usar busy-waiting.

### Ejercicio 3 ★

Reescribir el Driver del ejercicio anterior para que utilice interrupciones en lugar de busy-waiting. Para ello, la tecla ha sido conectada a la línea de interrupción número 7.

Para indicar al dispositivo que debe efectuar una nueva interrupción al detectar una nueva pulsación de la tecla, debe guardar la constante `BTN_INT` en el registro de la tecla. (Ayuda: usar semáforos)

### Ejercicio 4

Indicar las acciones que debe tomar el administrador de E/S:

- cuando se efectúa un open
- cuando se efectúa un write

### Ejercicio 5

¿Cuál debería ser el nivel de acceso para las syscalls `IN`, `OUT`? ¿Por qué?

### Ejercicio 6

El siguiente es parte de un Driver para manejar una cinta:

```
#define TAPE_IRQ 11;
static semaphore motor_is_started = semaphore(0);
static semaphore motor_is_stoped  = semaphore(0);
static semaphore motor_is_full    = semaphore(0);
static enum tape_status { IS_STARTING, IS_READING, IS_STOPING }
static int buffer_index = -1;
static int buffer[BUFFER_SIZE];
static tape_position = -1;

int tape_init() {
    int irq_ret = request_irq(TAPE_IRQ, tape_int_handler);
    if irq_ret==IRQ_ERROR
        return IO_ERROR
    else
        return IO_OK
}

int tape_remove() {
    int irq_ret = free_irq(TAPE_IRQ);
    if irq_ret==IRQ_ERROR
        return IO_ERROR
    else
        return IO_OK
}

int tape_open() {
    OUT(TAPE_CTRL, TAPE_START);
    tape_status = IS_STOPING;
    wait(motor_is_started);
```

```
    return IO_OK;
}
int tape_close() {
    OUT(TAPE_CTRL, TAPE_STOP);
    tape_status = IS_STARTING;
    wait(motor_is_stoped);
    return IO_OK;
}
int tape_read(int *data) {
    if (buffer_index== -1) {
        OUT(TAPE_CTRL, TAPE_READ);
        wait(buffer_is_full);
    }
    (*data) = buffer[buffer_index]
    buffer_index++
    if (buffer_index==BUFFER_SIZE) {
        buffer_index=-1
    }
    return IO_OK
}
int tape_write(int *data) {
    return IO_ERROR;
}
```

- Escribir el código del procedimiento `void tape_int_handler()` de modo que la cinta funcione correctamente.
- ¿Qué tipo de método de acceso utiliza este Driver ?
- Modificar el driver para que retorne `TAPE_EOF` a partir que se alcanza la posición 65535 en la cinta.
- Explicar brevemente cómo modificaría este Driver si quisiera usarlo para manejar K cintas distintas.

### Ejercicio 7 ★

Se desea implementar el driver de una controladora de una vieja unidad de discos ópticos que requiere controlar manualmente el motor de la misma. Esta controladora posee 3 registros de lectura y 3 de escritura. Los registros de escritura son:

- `DOR_IO`: enciende (escribiendo 1) o apaga (escribiendo 0) el motor de la unidad.
- `ARM`: número de pista a seleccionar.
- `SEEK_SECTOR`: número de sector a seleccionar dentro de la pista.

Los registros de lectura son:

- `DOR_STATUS`: contiene el valor 0 si el motor está apagado (o en proceso de apagarse), 1 si está encendido. Un valor 1 en este registro no garantiza que la velocidad rotacional del motor sea la suficiente como para realizar exitosamente una operación en el disco.
- `ARM_STATUS`: contiene el valor 0 si el brazo se está moviendo, 1 si se ubica en la pista indicada en el registro `ARM`.
- `DATA_READY`: contiene el valor 1 cuando el dato ya fue enviado.

Además, se cuenta con las siguientes funciones auxiliares (ya implementadas):

- `int cantidadSectoresPorPista()`: Devuelve la cantidad de sectores por cada pista del disco. El sector 0 es el primer sector de la pista.
- `void escribirDatos(void * src)`: Escribe los datos apuntados por `src` en el último sector seleccionado.
- `void sleep(int ms)`: Espera durante `ms` milisegundos.

Antes de escribir un sector, el driver debe asegurarse que el motor se encuentre encendido. Si no lo está, debe encenderlo, y para garantizar que la velocidad rotacional sea suficiente, debe esperar al menos 50 ms antes de realizar cualquier operación. A su vez, para conservar energía, una vez que finalice una operación en el disco, el motor debe ser apagado. El proceso de apagado demora como máximo 200 ms, tiempo antes del cual no es posible comenzar nuevas operaciones.

1. Implemente la función `write(int sector, void* data)` del driver, que escriba los datos apuntados por `data` en el sector en formato LBA indicado por `sector`. Para esta primera implementación no use interrupciones.
2. Modifique la función del inciso anterior utilizando interrupciones. La controladora del disco realiza una interrupción en el IRQ 6 cada vez que los registros `ARM_STATUS` o `DATA_READY` toman el valor 1. Además, el sistema ofrece un *timer* que realiza una interrupción en el IRQ 7 una vez cada 50 ms. Para este inciso no puede utilizar la función `sleep`.

## Parte 2 – Sistemas de archivos

### Ejercicio 8 ★

Suponer una computadora cuyo disco se accede sin memoria caché y un sistema de archivos FAT. Además, en este sistema, la FAT no queda almacenada en la memoria (recordar que lo normal es que la FAT se cargue en memoria).

- a) ¿Cuántos accesos al disco son necesarios para llegar hasta el último bloque de un archivo de  $N$  bloques?
- b) Si todos los bloques del archivo se encuentran cerca del final del disco, ¿qué característica tendrá la lista de accesos al disco? ¿Nota algún problema de eficiencia?

### Ejercicio 9 ★

Suponer que se cuenta con un sistema de archivos basado en inodos y bloques de 4 KB.

- a) Si se tiene un archivo de 40 KB, ¿cuántos bloques es necesario leer para procesarlo completamente?
- b) ¿Cuántos bloques es necesario leer si el archivo tiene 80 KB?

### Ejercicio 10

Un archivo está almacenado en un disco de 16 MB que tiene bloques de 1 KB. El archivo ocupa un bloque en cada una de las siguientes pistas: 20, 500, 10 y 900, en ese orden. El último acceso al disco fue en un bloque ubicado en la pista 51 y la entrada de directorio para el archivo está contenida por completo en un bloque en la pista 50. El primer bloque del disco es el 0. La política de E/S de disco es FCFS.

1. Suponga que se utiliza una lista enlazada como método de representación de archivos. ¿Cuál es la *distancia de búsqueda* o *seek distance* para leer el archivo completo?
2. Suponga que se utilizase una FAT almacenada al comienzo del disco, donde cada entrada ocupa dos bytes.

- a) ¿Cuál es el tamaño máximo de disco (en bytes) direccionable con esta FAT?
- b) Se desea agregar un bloque al final del archivo, utilizando para ello un bloque en la pista 600. ¿Cuál es la *distancia de búsqueda* para esta operación?
3. Si se utilizase una política SCAN (comienza ascendente) para el manejo de disco, ¿cambiarían las respuestas a las preguntas anteriores? Justifique (no es necesario rehacer las cuentas).
4. ¿Cuántos accesos a discos serían necesarios para resolver el punto 2 si se utilizasen 4 discos en RAID 0, 1 y 0+1?

### Ejercicio 11

Una compañía que fabrica discos rígidos decide, como parte de cierta estrategia comercial, emprender la creación de un nuevo *filesystem*. Durante la fase de diseño preliminar los ingenieros a cargo del proyecto discuten acaloradamente la conveniencia de adoptar un enfoque inspirado en FAT vs. la de uno basado en inodos. Ayúdelos a desempatar indicando cuál de las dos opciones recomendaría, y por qué, para cada uno de los siguientes requerimientos:

1. Es importante que puedan crearse enlaces simbólicos.
2. Es importante que la cantidad de sectores utilizados para guardar estructuras auxiliares sea acotada, independientemente del tamaño del disco.
3. Es importante que el tamaño máximo de archivo sólo esté limitado por el tamaño del disco.
4. Es importante que la cantidad de memoria principal ocupada por estructuras del *filesystem* en un instante dado sea (a lo sumo) lineal en la cantidad de archivos abiertos en ese momento.

Se tiene un disco rígido de 2 platos, dos caras por plato, 2048 pistas y 4096 sectores de 1 KB por pista. Se desea dar formato al disco usando un sistema de archivos de uso específico llamado HashFS basado en FAT. La idea es que no existen directorios ni archivos. Dado un path, se calcula el *Hash* del nombre y este indica cual es el archivo buscado. En resumen, este sistema de archivo cuenta con dos tablas:

- Una única FAT que guarda las entradas correspondientes al próximo bloque, indicando el final de un archivo cuando estos valores coinciden.
- Una única tabla de Hash que contiene los números de bloques iniciales y el tamaño en bytes para cada hash posible.

La novedad, es que este sistema de archivos permite configurar los siguientes elementos:

- Tamaño del bloque: 2, 4 o 8 sectores.
- Tamaño de identificadores de bloque: 8, 16, 24 o 32 bits.
- Tamaño del hash: 8, 16, 24 o 32 bits.

### Ejercicio 12 ★

Se tiene un disco rígido de 2 platos, dos caras por plato, 2048 pistas y 4096 sectores de 1 KB por pista. Se desea dar formato al disco usando un sistema de archivos de uso específico llamado HashFS basado en FAT. La idea es que no existen directorios ni archivos. Dado un path, se calcula el *Hash* del nombre y este indica cual es el archivo buscado. En resumen, este sistema de archivo cuenta con dos tablas:

- Una única FAT que guarda las entradas correspondientes al próximo bloque, indicando el final de un archivo cuando estos valores coinciden.

- Una única tabla de Hash que contiene los números de bloques iniciales y el tamaño en bytes para cada hash posible.

La novedad, es que este sistema de archivos permite configurar los siguientes elementos:

- Tamaño del bloque: 2, 4 o 8 sectores.
- Tamaño de identificadores de bloque: 8, 16, 24 o 32 bits.
- Tamaño del hash: 8, 16, 24 o 32 bits.

1. Suponiendo que se configura con 2 sectores por bloque, identificadores de bloque de 16 bits, y hash de 16 bits. ¿Cuál es el tamaño que ocupa la FAT? ¿Cuál es el tamaño de la tabla de archivos? ¿Cuál es el espacio que queda en disco para archivos?
2. Sabiendo que se desea maximizar la cantidad de archivos que el sistema soporta y que, además, en promedio los archivos tendrán un tamaño de 1 KB. ¿cuál sería la configuración óptima del sistema de archivos? Justifique.
3. ¿Cómo lo configuraría si el promedio de tamaño de archivos es de 16 Mb? Justifique.