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.

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

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

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

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

void OUT(int IO_address, int data); int IN(int IO_address);	Escribe data en el registro de E/S Retorna el valor almacenado en el registro de E/S
int request_irq(int irq, void* handler) int free_irq(int irq)	Permite asociar el procedimiento handler a la interrupción irq Retorna IRQ_ERROR si ya está asoacida a otro handler 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_STATUS`$_0$: vale 0 si la tecla no fue pulsada, 1 si fue pulsada.
- `BTN_STATUS`$_1$: 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 ★

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 7 ★

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) ¿Cuántos accesos al disco son necesarios para llegar hasta el último bloque de un archivo de N bloques?

Ejercicio 8 ★

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 9

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.

Ejercicio 10 ★

Se tiene un disco rígido de 16GB de espacio con sectores de 1 KB. 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 cuentas 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, para cada hash posible, el identificador del bloque inicial y el tamaño en bytes del archivo correspondiente a dicho hash.

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 24 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.