

## Parte 3: Persistencia

### 36. Dispositivos de entrada y salida

Un programa sin entrada o salida es un dispositivo, ¿de qué nos sirve? El problema radica entonces en cómo integrar estos dispositivos al sistema de manera eficiente.

#### 36.1 Arquitectura del sistema

Los dispositivos se conectan de manera jerárquica a través de **buses**. Existen buses por ejemplo para la memoria, otros para dispositivos generales como la GPU o tarjetas de red (PCI) y otros para periféricos de entrada y salida, como USB, teclados y cosas) de esos. Mientras más rápido sea un bus, más cerca estará de la CPU.

#### Un dispositivo canónico

Imaginemos un dispositivo canónico, no uno real, que nos ayude a entender esta monda. Todo dispositivo de hardware debe tener una interfaz y un protocolo con el que el software lo pueda controlar. Lo único que deben tener es una estructura interna, que puede ir desde chips muy simples hasta incluso tener CPUs,

#### 36.3 El protocolo canónico

Un dispositivo simplificado expone tres registros con los que el OS lo puede controlar:

- Estado: para saber cómo está
- Comando: para decirle qué hacer
- Datos: para pasárselos o recuperarlos del dispositivo.

El protocolo tiene 4 pasos

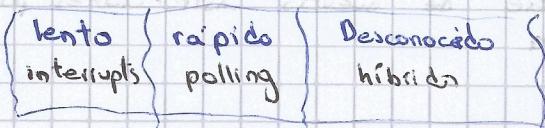
1. Esperar mientras el dispositivo esté ocupado, esto se conoce como **polling**
2. Establecer datos al registro de datos, esto puede tomar varias escrituras. Cuando la CPU está involucrada en la transferencia de datos, se conoce como **programmed I/O**
3. Escribir el registro de comando para que el dispositivo empiece a ejecutar su tarea
4. Esperar a que el dispositivo finalice la ejecución.

Este primer protocolo desperdicia mucho tiempo de CPU debido al polling, por lo que es necesario corregir esa inefficiencia.

#### 36.4 Usando interruptos

Esto ya lo conocímos. Al hacer un llamado a un dispositivo I/O, ponemos al llamador a dormir y cambiamos a otro proceso. Cuando el dispositivo termine, lanza una interrupción que es manejada por un handler, que le devuelve el control al proceso que llamó al I/O.

Esto puede ser malo cuando el dispositivo es muy rápido, pues completaría de una su tarea y nos demoraríamos más haciendo los cambios de contexto. Luego, si el dispositivo es



Puede ocurrir algo conocido como **live lock**, que es cuando se presentan muchas solicitudes a un dispositivo y el sistema se queda procesando únicamente las interrupciones. Esto se puede mitigar un poco con la **aproximación de coalescing**, que cuando se genera una interrupción solicitada a un dispositivo, espera otros poquito antes de posarle la instrucción al dispositivo, haciendo que varios interruptos se manejen en uno solo.

### 36.5 Más eficiencia con DMA

Es un dispositivo que se encarga de orquestar las transferencias entre dispositivo y la memoria principal, quitándole esa responsabilidad a la CPU.

### 36.6 Métodos de interacción con los dispositivos

Existen principalmente dos métodos para comunicar los dispositivos

- **Instrucciones I/O:** Son instrucciones, usualmente privilegiadas, que le permiten al OS pasar datos a un registro de un dispositivo específico.
- **Memory-mapped I/O:** El hardware hace que los registros estén disponibles como direcciones de memoria.

No hay ventajas entre uno y otro, aunque la 2da es buena porque no requiere de nuevas instrucciones para usarla.

### 36.7 El controlador de dispositivo

Para manejar cualquier dispositivo, debemos nuevamente realizar una abstracción; en este caso es el **controlador del dispositivo**, que expone unas interfaces genéricas que pueden ser usadas por el OS para controlar el dispositivo, el controlador se encarga de manejar los detalles de la solicitud que le llegó.

Dos cositas:

- Si el dispositivo tiene más "funcionalidades" que las que se le permite exponer, se pierden.
- Los controladores son como el 70% del código de un sistema operativo.

### 37. Discos duros (H.D.D)

Los discos duros han sido nuestro dispositivo de almacenamiento por décadas.

#### 37.1 La interfaz

Un disco es un arreglo de n bloques, cada uno con un tamaño de 512 bytes que pueden ser leídos o escritos. A pesar de que se pueda escribir en varios bloques a la vez, sólo se garantiza que una escritura de 512 bytes es atómica.

Hay algunos supuestos que los clientes de los discos tienen. Uno es que es más rápido acceder a bloques cercanos que lejanos y otro es que acceder a bloques contiguos es el modo más rápido posible, incluso mucho más rápido que un patrón de acceso aleatorio.

## 37.2 Geometría básica

Un disco moderno se compone de varias cosas

- Un plato, que es una superficie dura circular que tiene dos caras. Sobre ellos reposa una capa magnética que permite almacenar los datos.
- Un spindle, al que se conectan todos los platos y que un motor hace girar a una velocidad fija, medida en RPMs. Usualmente van de 7200 a 15000. Es de interés el tiempo de una sola rotación.
- Un track, que son círculos concéntricos donde se almacenan los datos. Hay miles en un disco.
- Una cabeza que lee los patrones magnéticos del disco. Hay una cabeza por superficie.
- Un brazo, que mueve la cabeza al sector de la superficie donde está el track buscado.

## 37.3 Un disco simple

Primero trabajaremos con un disco de un track

**Rotational delay** Es el tiempo que se demora el disco en ir de un bloque inicial a un bloque final

**seek time** Es el tiempo que se demora en mover el brazo y posicionar la cabeza correctamente. Es de las cosas más costosas, estando en el orden de los ms.

Otras detalles, los discos tienen como un delay para acelerar los cambios entre track y no tener mucho

delay rotacional, los tracks exteriores tienen más bloques que los internos, se dividen por zonas. Otra parte importante es el cache

## 37.4 I/O Time: Doing the math

El tiempo de entrada/salida se puede representar como la suma de tres componentes

$$T_{IO} = T_{seek} + T_{rotation} + T_{transfer}$$

El ratio se usa más a menudo para realizar comparaciones

$$Ratio = \frac{\text{Size transfer}}{T_{IO}}$$

Pueden existir diferentes cargas de trabajo, como accesos aleatorios o secuenciales, y hay discos que se desempeñan mejor en una cosa que en otra, como velocidad vs. capacidad. Lo ideal sería evitar las pequeñas transferencias aleatorias e intentar hacerlas secuenciales.

**Nota:** El promedio del tiempo de rotación es dividido a la mitad

## 37.5 Disk Scheduling

El sistema operativo puede estimar cuánto va a durar una transferencia para luego ejecutar la más corta (SJF)

**SSTF: Shortest Seek Time First**

El scheduler ordena la cola de solicitudes de I/O, eligiendo los tracks más cercanos primero. Sin embargo es mejor buscar el bloque más cercano (para el OS). Se puede

introduci el problema de la inanición

El raid se expone de manera transparente al sistema.

### Elevators (SCAN o C-SCAN)

Es una forma de evitar la inanición. El algoritmo se mueve al traves del disco, sirviendo peticiones en orden al traves de los track. Cuando se ha un paso desde dentro hacia afuera del disco, lo llamamos un barrido. Si una petición a un bloque llega y ya se ha servido ese bloque, se encola la petición hasta completar el barrido.

La familia SCAN no es la mejor, pues no se acerca mucho al SJF.

Shortest positioning first Aquí el movimiento depende de si es más rápido el tiempo de seek o el tiempo de rotación, sin embargo, puede ser difícil de implementar.

### Otros problemas de scheduling

Hay otros problemas como dónde poner la información. Antes la hacía únicamente el OS, pero ahora también puede intervenir el controlador. También cuánto debe esperar el OS al dispositivo. También mezclar las solicitudes.

## 38. RAID's

Es una técnica que permite crear discos más rápidos, con más capacidad y más confiables. Los controladores son dispositivos complejos que tienen memoria y cosas así.

Son más rápidos porque se puede escribir en paralelo

Si se daña un disco se puede recuperar por la redundancia

### 38.1 Interfaces y cosas internas

Para el sistema es sólo un atlejo lineal de bloques. Cuando se pide una solicitud de I/O lógico, el RAID debe calcular dónde está físicamente, esto depende del nivel del RAID

Aunque se vea como una caja, un controlador de RAID es un sistema especializado en operar el atlejo, con su procesador y su RAM

### 38.2 Modelo de fallos

Por ahora sólo consideraremos el fail-stop, que pues, dice si un disco está bueno o está malo. Y supondremos que es fácil detectar si el disco está malo

### 38.3 Cómo evaluar un RAID?

Hay varias formas de crear un RAID, cada una con sus ventajas y desventajas.

Evaluaremos

- Capacidad ¿cuanta capacidad útil hay?
- Confiabilidad ¿cuantos fallos de disco puedes tolerar?
- Desempeño: depende de la carga de trabajo