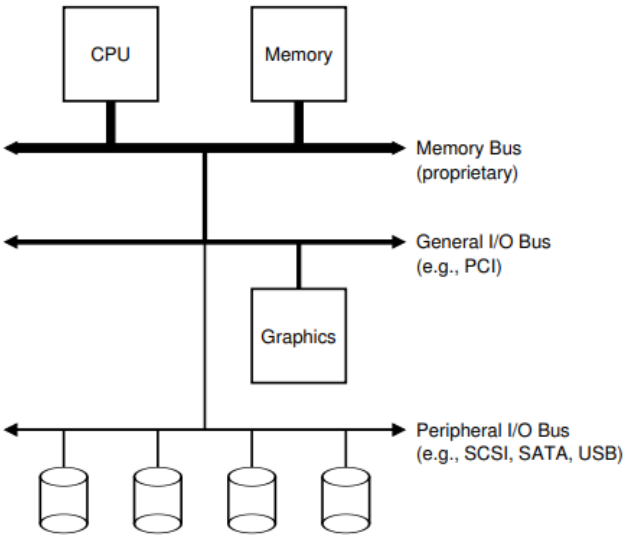
Capítulo 12: Dispositivos I/O

Antes de entrar en el contenido principal del tema, introduciremos el concepto de los dispositivos input/output y veremos cómo el S.O. interactúa con éstos. I/O es bastante crítico para sistemas informáticos

- Arquitectura del sistema



Esta imagen representa el clásico diagrama de un sistema. Vemos una sola CPU conectada a la memoria principal del sistema mediante un bus de memoria. Algunos de los dispositivos están conectados al sistema mediante buses de I/O. Incluso más abajo hay lo que es llamado bus de periféricos que conecta los SCSI, SATA o los USB. Esto conecta los dispositivos lentos al sistema, incluyendo los discos y los teclados

Más rápido un bus será cuanto más corto sea por lo que los componentes de alta demanda como las tarjetas gráficas están cerca de la CPU mientras que los menos demandados están lejos. Uno de los beneficios de colocar los discos en los buses periféricos es que puedes colocar un gran número de dispositivos

Los sistemas modernos usan chips y conexiones point-to-point más rápidas

La CPU se conecta a un chip I/O a través del DMI patentado de Intel y el resto de dispositivos se conectan al chip mediante diferentes conexiones. Uno o más discos duros se conectan al sistema mediante la interfaz eSATA

Debajo del chip I/O están las conexiones USB que se encargan, entre otras cosas, del ratón y del teclado conectados a un ordenador

También están las conexiones de alta demanda al sistema mediante PCIe

- Un dispositivo canónico

Estos dispositivos tienen dos componentes importantes. Una es la interfaz hardware que presenta al resto del sistema. Tiene una interfaz que permite al software del sistema controlar sus operaciones. Todos los dispositivos tienen una interfaz específica y un protocolo para cada interacción

La segunda parte es la estructura interna. Esta parte del dispositivo es la implementación específica y es responsable de implementar la abstracción del dispositivo al sistema. Los dispositivos más simples tienen pocos chips hardware que implementan su funcionalidad. Pero los sistemas modernos como los controladores RAID consisten en cientos de miles de líneas de firmware para implementar su funcionalidad

- El protocolo canónico

La interfaz del dispositivo está compuesta por tres registros. El registro de estado que puede ser leído para ver el estado del dispositivo, el registro de comandos para decirle al dispositivo lo que va a hacer y el registro de datos para pasarle datos al dispositivo

El protocolo tiene cuatro pasos. En el primero, el S.O. espera hasta que el dispositivo esté listo para recibir un comando mediante la lectura constante del registro de estado, llamamos a esto sondeo del dispositivo. En el segundo paso el S.O. envía datos al registro de datos, como si fuera el disco duro del dispositivo. Cuando la CPU está a cargo del movimiento de datos nos referimos a este como un programa I/O. En el tercer paso el S.O. escribe el comando en el registro de comandos para que el dispositivo sepa que tiene los datos necesarios para empezar a ejecutar el comando. Por último, en el cuarto paso el S.O. espera a que el dispositivo termine haciendo un nuevo sondeo en un bucle hasta que vea que ha terminado

Este protocolo es bastante simple y funcional pero también tiene sus inconvenientes. Un problema es que sondear es ineficiente, una espera activa gasta mucho tiempo de CPU en vez de estar ejecutando otro proceso mientras se espera un cambio en el estado del dispositivo

- Reducción de la sobrecarga de la CPU con interrupciones

En vez de sondear el dispositivo repetidamente, el S.O. puede lanzar una solicitud, poniendo el proceso a dormir y con un cambio de contexto ejecutar una nueva tarea. Cuando el dispositivo ha terminado las operaciones, lanza una interrupción hardware causando que la CPU salte al S.O. a una rutina de servicio de interrupciones. Este es solo una parte del código del S.O. que termina la solicitud y despierta el proceso esperando a la I/O

Las interrupciones nos permiten la superposición del cómputo y de la I/O. En los diagramas, el proceso 1 se está ejecutando en la CPU durante un tiempo cuando se emite una solicitud de I/O al disco para leer datos. Sin interrupciones, el sistema simplemente sondea el estado del dispositivo repetidamente hasta que se completa la I/O. El servicio del disco termina y el proceso 1 vuelve a ejecutarse en la CPU. En ese tiempo en el que el proceso está en el disco, el S.O. puede hacer algo mientras espera como ejecutar otro proceso

El uso de interrupciones no es siempre la mejor opción. En un sistema muy rápido puede reducir su velocidad cuando se cambia entre procesos, se maneja la interrupción y se vuelve a poner el proceso anterior en la CPU

Lo mejor sería usar un sistema híbrido que realice solicitudes para poco tiempo y después, si el dispositivo aún no ha terminado, usa interrupciones

Otro motivo para no usar el lanzamiento de interrupciones sale en las redes. Cuando, en una gran cantidad de paquetes entrantes, cada uno genera una interrupción, es posible que el S.O. se bloquee ya que estaría constantemente encontrando interrupciones de procesos y nunca permitirá a procesos de usuario ejecutarse

- Movimiento de datos más eficiente con DMA

Desafortunadamente hay una parte del protocolo canónico que requiere nuestra atención. En particular, cuando usamos programas de I/O para mover una gran cantidad de datos, la CPU se sobrecarga con una tarea trivial y pierde tiempo y esfuerzo que podría haber usado en ejecutar otros procesos

Supongamos que un proceso 1 está ejecutándose en la CPU y quiere escribir datos en el disco duro. Inicia la I/O que copiará la palabra de su memoria al dispositivo. Entonces se carga en la CPU la palabra y la I/O empieza en el disco para que en la CPU se pueda usar para otra cosa

Una solución para el problema de cargar la palabra en la CPU es algo llamado Acceso Directo a Memoria (DMA). Es un dispositivo específico que se encarga de las transferencias entre dispositivos y la memoria principal sin la intervención de la CPU. Para transferir los datos el S.O. programa la DMA para que sepa dónde se encuentran los datos que va a copiar, cuántos datos va a copiar y a dónde. En este punto el S.O. ha terminado con la transferencia y puede empezarse a ejecutar otro proceso mientras la DMA trabaja en la copia de los datos. Cuando esta termina lanza una interrupción y el S.O. sabe que la transferencia se ha completado y vuelve a ejecución el proceso 1

- Métodos de interacción del dispositivo

Un problema que habremos podido detectar es que no hemos dicho nada de cómo realmente el S.O. se comunica con el dispositivo. Se han desarrollado dos métodos, el primero, el más antiguo, es tener instrucciones explícitas de I/O. Estas instrucciones especifican una manera del S.O. para enviar datos a los registros de los dispositivos y entonces construir los protocolos descritos anteriormente

Estas instrucciones son privilegiadas. El S.O. controla los dispositivos y es el único que puede directamente comunicarse con los programas. Sería un completo caos si éstos pudieran leer o escribir directamente en el disco

El segundo método para interactuar con dispositivos es conocido como memoria asignada a I/O. El hardware permite registros de dispositivos como si estuvieran localizados en la memoria

- Fitting into The OS: The Device Driver

El ultimo problema que veremos será el de cómo encajar los dispositivos, ya que cada uno tiene sus interfaces específicas, en el S.O.

El problema se puede solucionar con la técnica de la abstracción. En el nivel más bajo, una parte del software en el S.O. debe saber con detalle cómo el dispositivo funciona. Llamaremos a esa parte driver del dispositivo y ahí encontraremos también las especificaciones del dispositivo