*Introducción a los Sistemas Operativos*

Subsistema de Entrada / Salida

*I.S.O.*

Versión: Mayo 2020

Palabras Claves: Metas, Aspectos de dispositivos, Subsistema de IO

Algunas diapositivas/imágenes han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos), el de Silberschatz (Operating Systems Concepts) y Tenembaum (Sistemas Operativos Modernos 3er Edición) También se incluyen diapositivas cedidas por Microsoft S.A.

*Responsabilidades del SO*Controlar dispositivos de E/S

– Generar comandos

– Manejar interrupciones

– Manejar errores

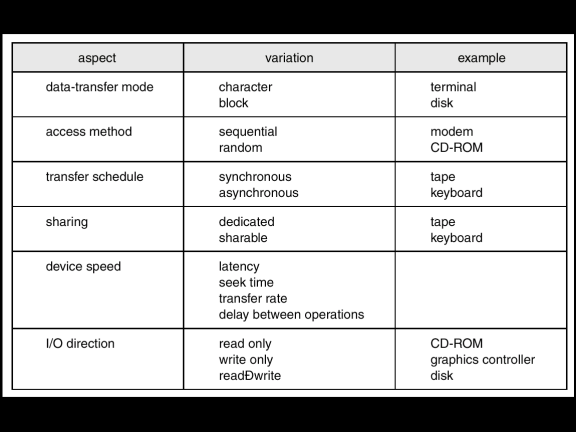
Proporcionar una interfaz de utilización

*Problemas*

Heterogeneidad de dispositivos Características de los dispositivos Velocidad

Nuevos tipos de dispositivos

Diferentes formas de realizar E/S (ver anexo)

*Aspectos de los dispositivos de I/O*

*Aspectos de los dispositivos de I/O (cont)*

Unidad de Transferencia

Dispositivos por bloques (discos):

Operaciones: Read, Write, Seek

Dispositivos por Caracter (keyboards, mouse, serial ports)

Operaciones: get, put

Formas de Acceso

Secuencial o Aleatorio

*Aspectos de los dispositivos de I/O (cont)* Tipo de acceso

• Acceso Compartido: Disco Rígido

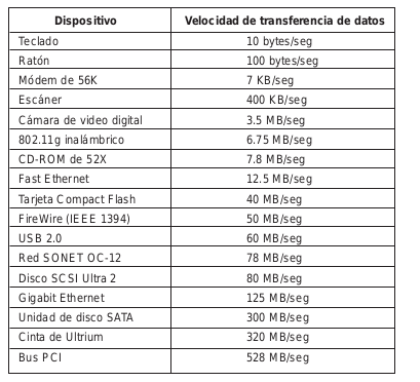
• Acceso Exclusivo: Impresora

Tipo de acceso:

• Read only: CDROM

• Write only: Pantalla

• Read/Write: Disco

*Aspectos de los dispositivos de I/O (cont)* Velocidad

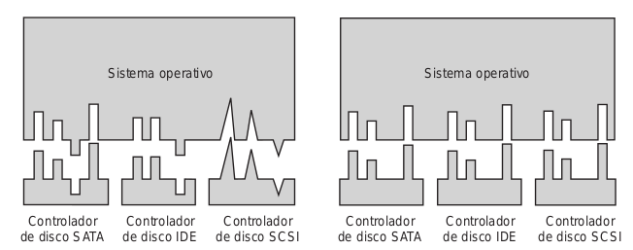
*Metas, Objetivos y Servicios*Generalidad:

Es deseable manejar todos los dispositivos de I/O de una manera uniforme,

estandarizada

Ocultar la mayoría de los detalles del dispositivo en las rutinas de niveles más “bajos” para que los procesos vean a los dispositivos, en términos de operaciones comunes como: read, write, open, close, lock, unlock

*Metas, Objetivos y Servicios*Interfaz Uniforme



*Metas, Objetivos y Servicios*

Eficiencia

Los dispositivos de I/O pueden resultar extremadamente lentos respecto a la memoria y la CPU

El uso de la multi-programación permite que un procesos espere por la finalización de su I/O mientras que otro proceso se ejecuta

*Metas, Objetivos y Servicios*

Planificación

Organización de los requerimientos a los dispositivos

Ej: Planificación de requerimientos a disco para minimizar tiempos

*Metas, Objetivos y Servicios*

Buffering – Almacenamiento de los datos en memoria mientras se transfieren

Solucionar problemas de velocidad entre los dispositivos

Solucionar problemas de tamaño y/o forma de los datos entre los dispositivos

*Metas, Objetivos y Servicios*

Caching – Mantener en memoria copia de los datos de reciente acceso para mejorar performance

Spooling – Administrar la cola de

requerimientos de un dispositivo

Algunos dispositivos de acceso exclusivo, no pueden atender distintos requerimientos al mismo tiempo: Por ej. Impresora

Spooling es un mecanismo para coordinar el acceso concurrente al dispositivo

*Metas, Objetivos y Servicios*

Reserva de Dispositivos: Acceso exclusivo

Manejo de Errores:

El S.O. debe administrar errores ocurridos (lectura de un disco, dispositivo no

disponible, errores de escritura)

La mayoría retorna un número de error o código cuando la I/O falla.

Logs de errores

*Metas, Objetivos y Servicios*

Formas de realizar I/O

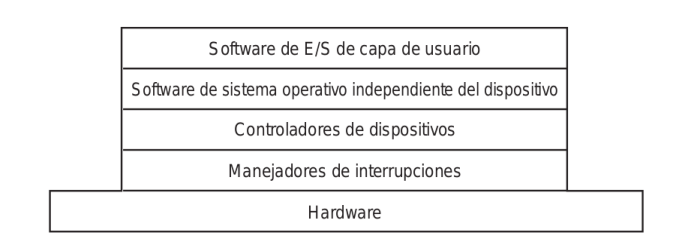
Bloqueante: El proceso se suspende hasta que el requerimiento de I/O se completa Fácil de usar y entender

No es suficiente bajo algunas necesidades No Bloqueante: El requerimiento de I/O retorna en cuanto es posible

 Ejemplo: Interfaz de usuario que recibe input desde el teclado/mouse y se muestra en el screen.

 Ejemplo: Aplicación de video que lee frames desde un archivo mientras va mostrandolo en pantalla.

*Diseño*

**

*Diseño – Software capa de usuario*

Librerías de funciones

– Permiten acceso a SysCalls

– Implementan servicios que no dependen del Kernel

Procesos de apoyo

– Demonio de Impresión (spooling)

*Diseño – Software independiente SO*

Brinda los principales servicios de E/S antes vistos

– Interfaz uniforme

– Manejo de errores

– Buffer

– Asignación de Recursos

– Planificación

*Diseño – Software independiente SO*

Brinda los principales servicios de E/S antes vistos

– Interfaz uniforme

– Manejo de errores

– Buffer

– Asignación de Recursos

– Planificación

*Diseño – Software independiente SO*

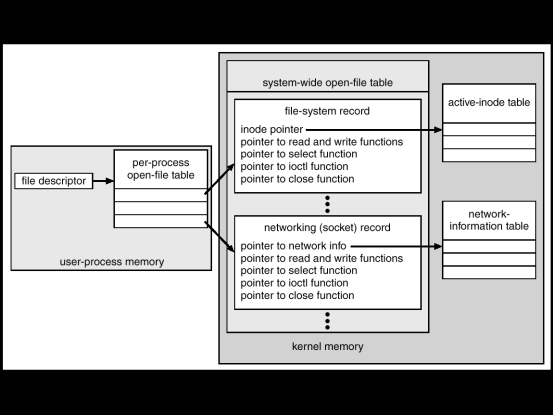
El Kernel mantiene la información de estado de cada dispositivo o componente

Archivos abiertos

Conexiones de red

Etc.

Hay varias estructuras complejas que representan buffers, utilización de la memoria, disco, etc.

*Diseño – Software independiente SO* UNIX I/O Kernel Structure

*Diseño – Controladores (Drivers)*

Contienen el código dependiente del dispositivo

Manejan un tipo dispositivo

Traducen los requerimientos abstractos en los comandos para el dispositivo

Escribe sobre los registros del controlador Acceso a la memoria mapeada

Encola requerimientos

Comúnmente las interrupciones generadas por los dispositivos son atendidas por funciones provistas por el driver

*Diseño – Controladores (Drivers)*

Interfaz entre el SO y el HARD

Forman parte del espacio de memoria del Kernel

En general se cargan como módulos

Los fabricantes de HW implementan el driver en función de una API especificada por el SO open(), close(), read(), write(), etc

Para agregar nuevo HW sólo basta indicar el driver correspondiente sin necesidad de cambios en el Kernel

*Driver - Ejemplo en Linux*

Linux distingue 3 tipos de dispositivos Carácter: I/O programada o por interrupciones Bloque: DMA

Red: Ports de comunicaciones

Los Drivers se implementan como módulos Se cargan dinámicamente

Debe tener al menos estas operaciones: init\_module: Para instalarlo

cleanup\_module: Para desinstalarlo.

*Driver - Ejemplo en Linux (cont.)*

Operaciones que debe contener para I/ O

open: abre el dispositivo

release: cerrar el dispositivo

read: leer bytes del dispositivo

write: escribir bytes en el dispositivo ioctl: orden de control sobre el dispositivo

*Driver - Ejemplo en Linux (cont.)*

Otras operaciones menos comunes llseek: posicionar el puntero de lectura/escritura flush: volcar los búferes al dispositivo

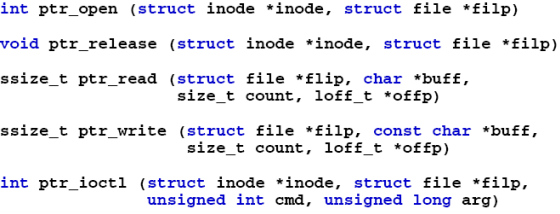
poll: preguntar si se puede leer o escribir mmap: mapear el dispositivo en memoria fsync: sincronizar el dispositivo

fasync: notificación de operación asíncrona lock: reservar el dispositivo

…..

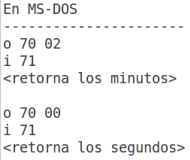
*Driver - Ejemplo en Linux (cont.)*

Por convención, los nombres de las operaciones comienzan con el nombre del dispositivo Por ejemplo, para /dev/ptr



*Driver - Ejemplo en Linux (cont.)* Acceso al hardware

Funciones para acceso a los puertos de I/ O <asm/io.h>

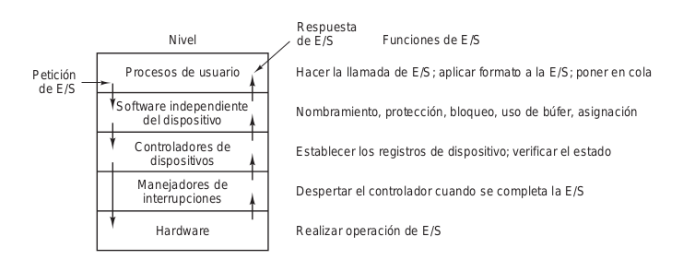
Leen o Escriben un byte en el puerto de E/S indicado

*Diseño – Gestor de interrupciones*

Atiende todas las interrupciones del HW Deriva al driver correspondiente según interrupción

Resguarda información

Independiente del Driver

*Ciclo de atención de un Requerimiento*

*Desde el Requerimiento de I/O hasta el Hardware*

Consideremos la lectura sobre un archivo en un disco:

Determinar el dispositivo que almacena los datos • Traducir el nombre del archivo en la

representación del dispositivo.

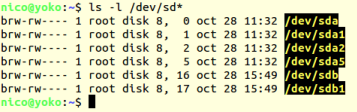
Traducir requerimiento abstracto en bloques de disco (Filesystem)

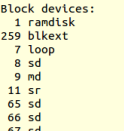
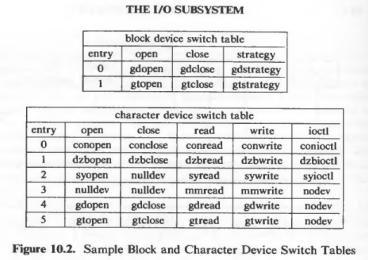
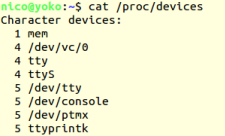
Realizar la lectura física de los datos (bloques) en la memoria

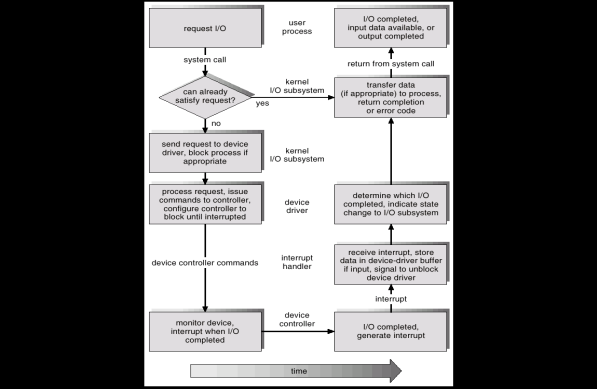
Marcar los datos como disponibles al proceso que realizo el requerimiento

• Desbloquearlo

Retornar el control al proceso

*Desde el Requerimiento de I/O hasta el Hardware*

**

*Ciclo de vida de un requerimiento de I/O*

*Performance*

I/O es uno de los factores que mas afectan a la performance del sistema: Utiliza mucho la CPU para executar los drivers y el codigo del subsistema de I/O Provoca Context switches ante las interrupciones y bloqueos de los procesos Utiliza el bus de mem. en copia de datos: • Aplicaciones (espacio usuario) – Kernel • Kernel (memoria fisica) - Controladora

*Mejorar la Performance*

Reducir el número de context switches Reducir la cantidad de copias de los datos mientras se pasan del dispositivo a la aplicación

Reducir la frecuencia de las interrupciones, utilizando:

– Transferencias de gran cantidad de datos – Controladoras mas inteligentes

– Polling, si se minimiza la espera activa. Utilizar DMA

*Introducción a los Sistemas Operativos*

Anexo I

Arquitectura de Entrada/Salida

*I.S.O.*

Versión: Octubre 2017

Palabras Claves: Dispositivos de IO, Hardware de IO, IO programada, Polling, Interrupciones, DMA

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz

(Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.

*Variedad en los dispositivos de I/O*

Legible para el usuario

Usados para comunicarse con el usuario

 Impresoras, Terminales: Pantalla, Teclado, Mouse Legible para la máquina

Utilizados para comunicarse con los componentes electrónicos

 Discos, Cintas, Sensores, etc.

Comunicación

Usados para comunicarse con dispositivos remotos

 Líneas Digitales, Modems, Interfaces de red, etc.

*Problemas que surgen*

Amplia Variedad

Manejan diferentes cantidad de datos En Velocidades Diferentes

En Formatos Diferentes

La gran mayoría de los dispositivos de E/S son más lentos que la CPU y la RAM

*Hardware y software involucrado*

Buses

Controladores

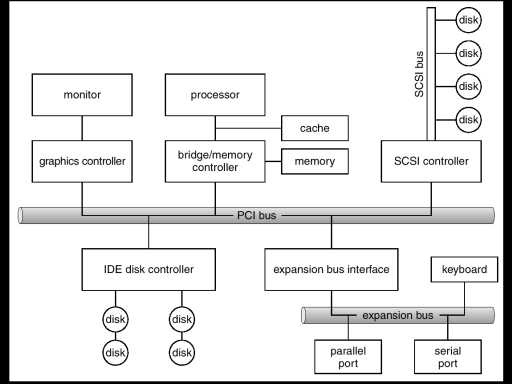
Dispositivos

Puertos de E/S – Registros

Drivers

Comunicación con controlador del dispositivo: I/O Programada,

Interrupciones, DMA

*Estructura de Bus de una PC*

*Comunicación: CPU - Controladora*

¿Cómo puede la CPU ejecutar

comandos o enviar/recibir datos de una controladora de un dispositivo? La controladora tiene uno o mas registros: • Registros para señales de control

• Registros para datos

La CPU se comunica con la

controladora escribiendo y leyendo en dichos registros

*Comandos de I/O*

CPU emite direcciones

Para identificar el dispositivo

CPU emite comandos

Control – Que hacer?

Ej. Girar el disco

Test – Controlar el estado

Ej. power? Error?

Read/Write

Transferir información desde/hacia el dispositivo

*Mapeo de E/S y E/S aislada*

 Correspondencia en memoria (Memory mapped I/O)  Dispositivos y memoria comparten el espacio de direcciones.

 I/O es como escribir/leer en la memoria.

 No hay instrucciones especiales para I/O

 Ya se dispone de muchas instrucciones para la memoria Isolated I/O (Aislada, uso de Puertos de E/S)  Espacio separado de direcciones

 Se necesitan líneas de I/O. Puertos de E/S

 Instrucciones especiales

 Conjunto Limitado

*Memory Mapped and Isolated I/O*

*Técnicas de I/O - Programada*

CPU tiene control directo sobre la I/O 

Controla el estado

Comandos para leer y escribir

Transfiere los datos

CPU espera que el componente de I/O complete la operación

Se desperdician ciclos de CPU

*Polling*

En la I/O Programada, es necesario hacer polling del dispositivo para determinar el estado del mismo

Listo para recibir comandos

Ocupado

Error

Ciclo de “Busy-wait” para realizar la I/O Puede ser muy costoso si la espera es muy larga

*Técnicas de I/O - Manejada por Interrupciones*

Soluciona el problema de la 

espera de la CPU

La CPU no repite el chequeo

sobre el dispositivo

El procesador continúa la

ejecución de instrucciones

El componente de I/O envía una

interrupción cuando termina

*Técnicas de I/O - DMA*

DMA (Direct Memory Access) Un componente de DMA controla el intercambio de datos entre la memoria 

principal y el dispositivo

El procesador es

interrumpido luego de que el bloque entero fue

transferido.

*Pasos para una transferencia DMA*

**