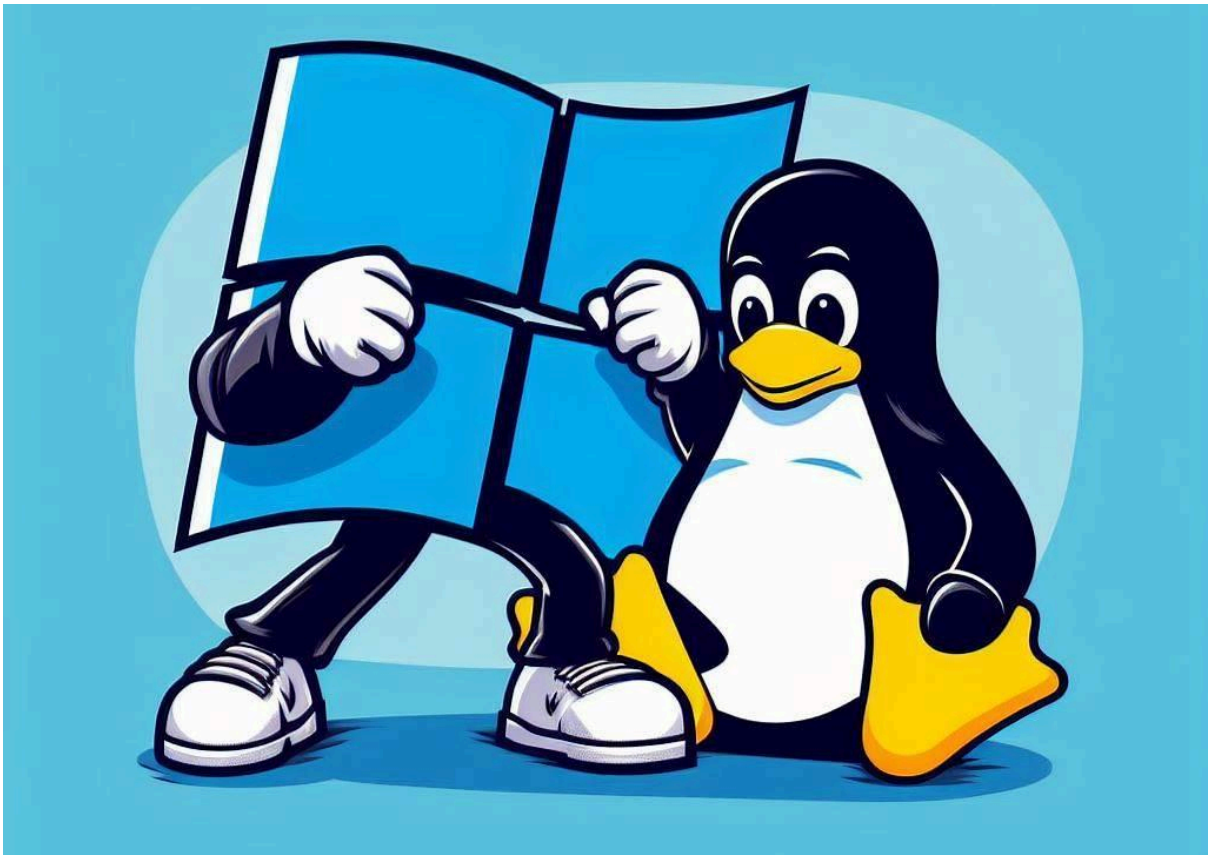
INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

SUBSISTEMA DE ENTRADA / SALIDA

Resumen teoría



Caporal Nicolás

Facultad de Informática

UNLP

2023

Tkm profe Lia Molinari

ÍNDICE

RESPONSABILIDADES DEL SISTEMA OPERATIVO

La comunicación desde el SO hacia los dispositivos de E/S se realiza a través de comandos especificos que entiende el dispositivo.

La comunicación desde el los dispositivos hacia el SO se realiza a través de interrupciones.

Por tanto, el SO debe:

Para 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

ASPECTOS DE LOS DISPOSITIVOS DE I/O

Unidad de Transferencia:

- Dispositivos por bloques (discos):

  Operaciones: Read, Write, Seek

- Dispositivos por Carácter (keyboards, mouse, serial port)

  Operaciones: get, put

Formas de Acceso:

- Secuencial
- Aleatorio

Tipo de acceso:

- Acceso Compartido: Disco Rígido
- Acceso Exclusivo: Impresora

Dirección E/S:

- Read only: CDROM
- Write only: Pantalla
- Read/Write: Disco

Velocidad

METAS, OBJETIVOS Y SERVICIOS DEL SO

Generalidad:

Es deseable manejar todos los dispositivos de I/O de una manera uniforme, estandarizada (el USB fue un gran avance)

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 y estandarizadas: read, write, open, close, lock, unlock

Eficiencia

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

El uso de la multi-programación (tener más de un proceso en memoria) permite que un procesos espere por la finalización de su I/O mientras que otro proceso se ejecuta

Planificación

Organización de los requerimientos a los dispositivos: Multiples usuarios o procesos pueden realizar una operación de entrada (o salida) sobre el mismo dispositivo al mismo tiempo y hay que planificar en función del dispositivo para minimizar tiempos

Ej: Planificación de requerimientos a disco para minimizar tiempos como vimos en la práctica

Buffering

Almacenamiento de los datos en memoria mientras se transfieren

Ejemplo: el buffer de la impresora en MSX88 o VonSim

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

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.

Se organiza una cola en disco, para mantener almacenada la informacion e ir administrando el envío de los datos al dispositivo

Reserva de Dispositivos:

Acceso exclusivo

Ejemplo: como los robots de Concurrente de R-Info en Taller de Programación

Manejo de Errores:

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

Los debe detectar y comunicarles al proceso. La mayoría retorna un número de error o código cuando la I/O falla. (como en Scripting en Bash de la práctica)

Logs de errores: Es importante tener un registro de la actividad para saber qué falló

Formas de realizar I/O

- **Bloqueante:** El proceso se suspende hasta que el requerimiento de I/O se complete. Fácil de usar y entender. No es suficiente bajo algunas necesidades
- **No Bloqueante:** El requerimiento de I/O retorna en cuanto es posible.
  Ejemplo: Aplicación de video que lee frames desde un archivo mientras va mostrandolo en pantalla.

DISEÑO

Todo esto debe ser implementado en código por el SO

SOFTWARE CAPA DE USUARIO

Es la manera que tiene el proceso de comunicarse con los dispositivos de I/O

Librerías de funciones:

Debe haber librerias que tengan funciones ya definidas para poder interactuar con los dispositivos de I/O. Las librerías se deben implementar teniendo en cuenta las caracteristicas de los dispositivos

- Permiten acceso a SysCalls
- Implementan servicios que no dependen del Kernel

Procesos de apoyo:

- Demonio de Impresión (Spooling)

SOFTWARE INDEPENDIENTE DEL SO

Son softwares orientados al manejo específico de cada dispositivo, que los toma el SO para saber cómo trabajar con ese dispositivo.

- Interfaz uniforme
- Manejo de errores
- Buffer
- Asignación de Recursos
- Planificación

Además, el **kernel** debe mantener 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. que tiene el SO para tomar decisiones

CONTROLADORES (DRIVERS)

Son software que se encargan de decir **cómo** trabajar con el dispositivo.

Contienen el código (instrucciones) dependiente del dispositivo

Manejan un tipo dispositivo específico

Traducen los requerimientos abstractos en los comandos para el dispositivo

Escribe sobre los registros del controlador

Acceso a la memoria mapeada (hay un area de memoria asignada al dispositivo)

Encola requerimientos

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

Funcionan como Interfaz entre el SO y el HW

Forman parte del espacio de memoria del Kernel y 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

GESTOR DE INTERRUPCIONES

Atiende todas las interrupciones del HW

Deriva al driver correspondiente según interrupción

Resguarda información

Es independiente del Driver

EJEMPLO DESDE EL REQUERIMIENTO HASTA DE I/O HASTA EL HARDWARE

Consideremos la lectura sobre un archivo en un disco:

- Determinar en que dispositivo están almacena los datos de ese archivo
- Traducir el nombre del archivo en la representación del dispositivo.
- Traducir requerimiento abstracto (read) en bloques de disco (Filesystem)
- Realizar la lectura física de los datos (bloques) en la memoria
- Marcar los datos como disponibles al proceso que realizó el requerimiento
- Desbloquearlo
- Comunicarle al proceso que ya tiene los datos en memoria y retornarle el control

PERFORMANCE

La I/O es uno de los factores que mas afectan a la performance general del sistema:

- Utiliza mucho la CPU para ejecutar 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 (cuando es posible), utilizando:
    - Transferencias de gran cantidad de datos
    - Controladoras más inteligentes
    - Polling, si se minimiza la espera activa.
- Utilizar DMA (*direct memory access*)