Practica 6 - ISO

1) a) y b) Orientado a bloques: transmiten datos en bloques (paquetes) y por esa razón son usados a menudo para la transmisión paralela de datos. Estos dispositivos utilizan el Buffer de datos del sistema operativo. Ejemplos: discos, CD-ROM.

Orientado a flujos: transmiten solo un Bit o solo unByte a la vez, es decir, utilizan la transmisión serial de datos, sin usar buffer. Ejemplos: keyboards, mouse, serial ports.

c) Las diferencias que existen entre los dispositivos de E/S son las distintas velocidades en que se transmiten los datos, en todos los casos son mucho más lentos que la memoria y la CPU. El SO a través del uso de la multiprogramación permite que los procesos esperen por la finalización de la I/O mientras que otro se ejecuta.

2) E/S Programada: Los datos se intercambian entre el CPU y el módulo de E/S. El CPU ejecuta un programa que controla directamente la operación de E/S, incluyendo la comprobación del estado del dispositivo, el envío de la orden de lectura o escritura y la transferencia del dato. Cuando el CPU envía la orden debe esperar hasta que la operación de E/S concluya. Si el CPU es más rápido, éste estará ocioso. El CPU es el responsable de comprobar periódicamente el estado del módulo de E/S hasta que encuentre que la operación ha finalizado.

E/S dirigida por interrupciones:
• La CPU no tiene que esperar la finalización de la tarea de E/S, puede seguir procesando.
• No se repite la comprobación de los estados de los módulos
• El módulo envía un pedido de interrupción a la CPU cuando está listo nuevamente.

¿qué hace la CPU ???
• La CPU envía una orden de lectura (READ).
• El módulo E/S obtiene los datos del periférico mientras que la CPU realiza otro trabajo.
• La CPU chequea si hay pedidos de interrupciones pendientes al final de cada ciclo de instrucción.
• El módulo E/S emite un pedido de interrupción a la CPU.
• La CPU detecta el pedido, guarda el contexto, interrumpe el proceso y realiza la gestión de la interrupción.
• La CPU solicita los datos.
• El módulo E/S transfiere los datos.

DMA: Una transferencia DMA consiste principalmente en copiar un bloque de memoria de un dispositivo a otro. En lugar de que la CPU inicie la transferencia, esta se lleva a cabo por el controlador DMA. Solo se produce una interrupción por bloque en lugar de tener una interrupción por cada byte (o palabra). Un ejemplo típico es mover un bloque de memoria desde una memoria externa a una interna más rápida. Tal operación no ocupa al procesador y, por ende, éste puede efectuar otras tareas.

Ejemplo:

Imprimir un archivo de 10KB en una impresora láser de 20 páginas por minuto:
E/S programada La CPU entra en un bucle y envía un nuevo byte cada vez que la impresora está preparada. La impresora tarda 10s en imprimir 10 Kbytes. La CPU está ocupada 10 s con la operación de E/S (la CPU podría haber ejecutado 1000 millones de instrucciones).

E/S por interrupciones La impresora genera una interrupción cada vez que está preparada. Si la Rutina de Interrupción tiene 10 instrucciones (salvar contexto, comprobar estado, transferir byte, restaurar contexto, rti), para transferir 10 Kbytes se ejecuta 10.000 x 10 instrucciones. La CPU está ocupada 0,001 s con la operación de E/S.

Ejemplo 2:

Transferir un archivo de 10MB de la memoria a disco:

E/S programada La CPU entra en un bucle y envía un nuevo byte cada vez que el disco está preparado. El disco tarda 1s en recibir 10 Mbytes. La CPU está ocupada 1 s con la operación de E/S (la CPU podría haber ejecutado 100 millones de instrucciones).

E/S por interrupciones El disco genera una interrupción cada vez que está preparado. Si la Rutina de Interrupción tiene 10 instrucciones (salvar contexto, comprobar estado, transferir byte, restaurar contexto, rti), para transferir 10Mbytes se ejecuta 10.000000 x 10 instrucciones. La CPU está ocupada 1 s con la operación de E/S.

3) La E/S mapeada en memoria usa el mismo bus de direcciones para memoria y dispositivos de E/S, y las instrucciones de la CPU usadas para acceder a la memoria son también usadas para acceder a los dispositivos. Para tener espacio para los dispositivos de E/S, las áreas del espacio direccionable por la CPU deben ser reservadas para E/S más que para memoria. Esta reserva puede ser temporal o permanente. Cada dispositivo de E/S monitoriza el bus de direcciones de la CPU y responde a cualquier acceso de esta al espacio de direcciones del dispositivo, conectando el bus de datos con la localización en memoria física del dispositivo deseado.

La E/S independiente usa un tipo especial de instrucciones de la CPU para implementar E/S. Principalmente en microprocesadores Intel encontramos las instrucciones IN y OUT, que pueden leer y escribir un único byte en un dispositivo de E/S. Estos tienen un espacio de direcciones separadas de la memoria, llevado a cabo o bien por un pin " E/S "extra en la CPU o bien por un bus entero dedicado a E/S.

4) Metas que debe perseguir un SO para la administración de la entrada salida:

Generalidad:
üEs deseable manejar todos los dispositivos de I/O de una manera uniforme, estandarizada.
üOcultar 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.

Eficiencia:
üLos dispositivos de I/O pueden resultar extremadamente lentos respecto a la memoria.
üEl uso de la multiprogramación permite que los procesos esperen por la finalización de la I/O mientras que otro se ejecuta.
üI/O no puede alcanzar la velocidad de la CPU.

5) a) Un driver, controlador de dispositivo o manejador de dispositivo es un programa informático que permite al sistema operativo interaccionar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz (posiblemente estandarizada) para utilizar el dispositivo.

- 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 de los dispositivos están asociadas a una función del driver.
- 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.

b) Debe tener al menos estas operaciones:
- init_module: Para instalarlo.
- cleanup_module: Para desinstalarlo.

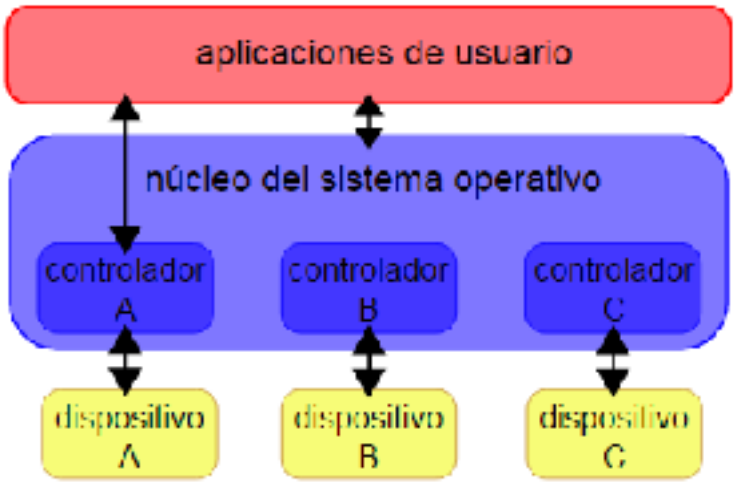
Operaciones que debe contener para I/O:
- open: abre el dispositivo
- release: cerrar el dispositivo
- read: leer bytes del dispositivo

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

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

c) Las funciones que deben disponer dependen en gran parte del dispositivo que se quiera controlar.

6)



Esquema del subsistema de controladores de dispositivos como parte del "núcleo del sistema operativo", actuando como interfaz (controlador A) entre las "aplicaciones de usuario" y un dispositivo externo (dispositivo A).

7) 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.
- Lectura física de los datos en la memoria.
- Marcar los datos como disponibles al proceso que realizo el requerimiento.
- Desbloquearlo
- Retornar el control al proceso

8) Servicios que provee el SO para administrar los dispositivos de E/S
Planificación:
- Organización de los requerimientos a los dispositivos.
- Ej: Planificación de requerimientos a disco para minimizar movimientos.

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.

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.

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.

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.

9) Un disco duro está compuesto por varios platos o discos que giran sobre un eje en el centro del cuál se unen. Pueden tener 2, 4, 6, o 7 platos por lo general. Cada plato tiene dos caras o superficies (una en la parte superior y otra en la inferior). Se poseen tantos cabezales como superficies. El cabezal está formado por un conjunto de brazos paralelos a los platos que se desplazan en sentido vertical (hacia el interior o exterior de los platos) y en su punta poseen las cabezas para realizar la lectura/escritura. A su vez cada superficie se divide en pistas que son circunferencias que van desde el exterior (circunferencias más grandes) al interior (circunferencias más chicas). Y las pistas se dividen en sectores. Existe la misma cantidad de sectores tanto para las pistas más externas como para las pistas más internas, pero cuando más adentro nos introducimos, más chico serán los tamaños de los sectores.

10) Seek Time (Posicionamiento): Tiempo que tarda en posicionarse la cabeza en el cilindro

Latency Time (Latencia): Tiempo que se sucede desde que la cabeza se posiciona en el cilindro hasta que el sector pasa por debajo de la misma.

Transfer Time (Transferencia):Tiempo de transferencia del sector (bloque) del disco a la memoria.

11) a) 7 discos * 2 caras * 1100 cilindros * 300 sectores * 512 bytes = 2365440000 bytes / 1024^3 = 18,19 Gb.

El disco es de 18,19 Gb.

b) 1 MiB (mebibyte) = 2^20 bytes.

3 MiB = (2^20) * 3 = 3145728 bytes / 512 bytes = 6144 sectores.

c) Archivo almacenado de manera secuencial:

Seek + Latency + (Tiempo_Trans_1_bloque * #Bloques)

10ms + (1/2 * 60000ms / 9000RPM)ms + (15Mib * 1seg / 10MiB)seg =

10ms + 3,3ms + 1,5seg (1500ms) = 1513,3ms

El archivo de 15MibB tarda 1513,3ms en grabarse. Nota: como se daba el tiempo de transferencia (10MiB/seg), y no nos especificaba la cantidad de bloques ni la tasa de transferencia de un bloque, aplicamos directamente regla de 3. (si 10MiB tardan 1 segundo en transferirse, entonces 15MiB tardan (15Mib * 1seg / 10MiB)seg.

d) Archivo almacenado de manera aleatoria:

(Seek + Latency + Tiempo_Trans_1_bloque) * #Bloques

En éste caso es obligatorio saber la cantidad de bloques que ocupa un archivo de 16MiB, y la tasa de transferencia por bloque.En primer lugar hay que saber que un bloque equivale a un sector. Entonces sacamos la cantidad de sectores que ocupa dicho archivo de 16MiB.

16 MiB = (2^20) * 16 = 16777216 bytes / 512 bytes = 32768 sectores.

Ahora la tasa de transferencia por bloque:

10MiB * 1024 = 10240KiB * 1024 = 10485760bytes

1seg = 1000ms

10485760bytes ----------→ 1000ms

512bytes ----------→ 512bytes * 1000ms / 10485760bytes = 0,05ms

La tasa de transferencia por bloque es de 0,032ms

Finalmente calculamos el tiempo que tarda en trasnferirse el archivo almacenado de manera aleatoria:

(10ms + 3,3ms + 0,05ms) * 32768 sectores =

13,35ms * 32768 sectores = 437452,8ms /1000 = 437,4528seg /60 = 7,29min

El archivo de 16MiB tarda 7,29 minutos en transferirse de forma aleatoria.

16) a) y b) Asignación contigua: los bloques que pertenecen a un mismo archivo se ubican de manera contigua. Cada directorio contiene para cada archivo la dirección del bloque en que comienza la longitud del área asignada a este archivo.

Ventajas:

- Fácil interpretación (solo es necesario el bloque inicial y la cantidad de bloques para llevar el registro de un archivo).

- Es de fácil acceso.

- Excelente desempeño de lectura (el archivo entero puede ser leído en una única operación, no hay retrasos rotacionales de la cabeza).

Desventajas:

- Genera fragmentación externa.

Asignación enlazada: cada archivo es una lista ligada de bloques de disco, en cada bloque existe un puntero que direcciona al bloque siguiente, el resto del bloque es usado para almacenar datos, de ésta forma todos los bloques del disco pueden ser usados.

Ventajas:

- Solo se necesita la dirección del bloque inicial, los demás pueden ser encontrados a partir del primero.

- No desperdicia espacio (no requiere dejar espacios vacíos para ubicación, reubicación o el crecimiento de archivos, lo que conlleva a un buen manejo del espacio libre).

- No produce fragmentación externa.

Desventajas:

- Solo es eficiente para archivos de acceso secuencial, el acceso a archivo de forma aleatoria o directa es bastante lenta (para leer el bloque n, se deben leer n-1 bloques antes de poder acceder al bloque deseado, lo que conlleva muchas lecturas y por consiguiente mucho tiempo.

- Si se pierde un puntero al siguiente bloque se pierde acceso al resto del archivo.

- Perdida de espacio debido al tamaño que ocupan los punteros.

- Perdida de eficiencia, en la lectura datos en múltiplos del tamaño del bloque, pues la cantidad de datos guardados ya no será una potencia de 2 debido a los punteros, por ende la cantidad de datos que caben en un bloque completo se deben distribuir entre 2, lo que requiere una posterior concatenación generando un overhead debido a la copia necesaria en la operación.

Asignación indexada: También conocido como asignación por "I-Nodes" éste método lleva el rastro de que bloque pertenece a cada archivo, asignando un I-node con todos los punteros hacia los demás bloques en el orden correspondiente, para cada archivo existe un I-node. El i-node lista los atributos y las direcciones de bloques del archivo asociado por lo que dado su I-nodo, es posible encontrar todos los bloques de un archivo.

Ventajas:

- No requiere una gran tabla en memoria como en FAT: simplemente se carga el I-nodo de el/los archivos abiertos en cada momento.

- Permite el acceso específico del archivo.

- Acceso aleatorio.

- No produce fragmentación externa.

Desventajas:

- Permite fragmentación externa, pero causa sobrecarga en el I-nodo.

- Desperdicio de espacio para archivos muy pequeños, ya que sin importar el contenido del archivo, además hay que incluir su respectivo I-nodo, que puede llegar a ocupar bastante espacio.

- Tamaño de archivo limitado debido al tamaño prefijado del I-nodo.

17) Mapa de Bits: Cada bloque se representa con un bit, el cual es: 0 si está libre, 1 si está ocupado.

Ventaja: resulta fácil encontrar bloques consecutivos.

Desventaja: No resulta eficiente si en la mayoría de los accesos el mapa (o vector) no está en memoria principal.

Lista Ligada: Enlaza los bloques libres del disco manteniendo un apuntador al siguiente bloque libre.

Desventaja: No es eficiente ya que para recorrer la lista tenemos que leer cada uno de los bloques, lo que representa tiempo de E/S.

Agrupamiento: Almacena en el primer bloque las direcciones de n bloques libre, los primeros n-1 están libres y el último contiene las direcciones de otros n bloques libres.

Ventaja: Se pueden encontrar rápidamente bloques libres.

Recuento: Aprovecha varios bloques contiguos que se pueden asignar simultáneamente.

Ventaja: Se puede tener la dirección del primer bloque libre, y los n bloques contiguos.

Recuento (sacado de otra fuente): Consiste en aprovechar que, en general, varios bloques contiguos se pueden asignar o liberar simultáneamente, especialmente en asignación contigua. Así podemos tener la dirección del primer bloque libre y los n bloques libres contiguos que le siguen. La lista sería más pequeña, siempre y cuando el recuento sea generalmente mayor que uno.