Universidad de Guadalajara.

Centro Universitario de Ciencias Exactas e Ingenierías.

DIVISIÓN DE TECNOLOGÍAS PARA LA INTEGRACIÓN CIBER-HUMANA.

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES.

TEMA: RESUMEN

NOMBRE DEL ESTUDIANTE: Padilla Perez Jorge Daray.

NOMBRE DE LA MATERIA: Sistemas operativos

NOMBRE DEL PROFESOR: Ramiro Lupercio Coronel

Table of Contents

2. **PROCESOS E HILOS**

En un sistema de **multiprogramación**, la CPU alterna rápidamente entre diferentes procesos, ejecutando cada uno durante fracciones de milisegundos. En cualquier momento, la CPU está ejecutando solo un proceso.

Cada **proceso** es una instancia de un programa en ejecución, que incluye valores actuales del contador de programa, registros y variables. La idea clave es que un proceso es una actividad con un programa, entrada, salida y estado. Varios procesos pueden compartir un solo procesador mediante un algoritmo de planificación que determina cuándo detener un proceso para dar servicio a otro.

Los cuatro eventos

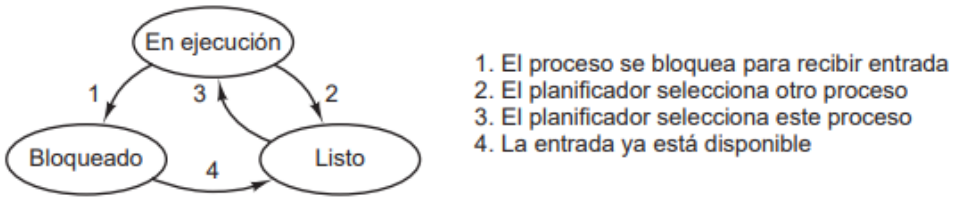

Figura 2-2. Un proceso puede encontrarse en estado "en ejecución", "bloqueado" o "listo". Las transiciones entre estos estados son como se muestran.

principales que provocan la creación de procesos son:

Arranque del sistema: Al iniciar el sistema, se crean procesos esenciales.

Llamada al sistema desde un proceso: Un proceso existente puede solicitar la creación de nuevos procesos.

Petición de usuario: Los usuarios pueden iniciar programas mediante comandos o haciendo clic en iconos.

Inicio de trabajos por lotes: Se crean procesos para manejar tareas en lotes.

Algunos procesos son en primer plano, interactúan con usuarios y realizan tareas visibles. Otros son en segundo plano, dedicados a funciones específicas. Los procesos en segundo plano que manejan actividades específicas se llaman demonios.

Un proceso en ejecución puede crear nuevos procesos mediante llamadas al sistema. Los usuarios pueden iniciar programas escribiendo comandos o haciendo

clic en iconos. Para crear un proceso, un proceso existente ejecuta una llamada al sistema de creación de procesos.

En UNIX, la llamada al sistema para crear un proceso es **fork()**, que crea un clon exacto del proceso que la invoca. En Windows, la función CreateProcess maneja la creación de procesos y carga el programa correcto en el nuevo proceso. Una vez creado, el padre y el hijo tienen espacios de direcciones separados, sin compartir memoria en la que se pueda escribir.

La **terminación** de un proceso puede ocurrir por:

Salida normal (voluntaria).

Salida por error (voluntaria).

Error fatal (involuntario), como una referencia a una parte de memoria inexistente.

Eliminación por otro proceso (involuntaria), mediante llamadas al sistema como kill en UNIX o TerminateProcess en Windows.

Para señalar al sistema operativo que ha concluido su ejecución, un proceso realiza una llamada al sistema. En **UNIX**, esta llamada es exit(), mientras que en Windows se utiliza ExitProcess.

En el entorno de UNIX, los procesos forman una estructura jerárquica donde todos los procesos descienden de un proceso raíz llamado init. En cambio, **Windows** no sigue una jerarquía de procesos; todos los procesos son independientes, aunque el proceso padre recibe un identificador especial, conocido como manejador, que le permite ejercer control sobre el proceso hijo.

Un proceso puede encontrarse en uno de los siguientes estados:

1. **En ejecución**: El proceso está activamente utilizando la CPU.
2. **Listo**: El proceso está listo para ejecutarse pero ha sido pausado para permitir la ejecución de otro proceso.
3. **Bloqueado**: El proceso está a la espera de que ocurra un evento externo para poder continuar, como la finalización de otro proceso.

Existen cuatro transiciones principales entre estos estados:

- **Transición 1**: Se da cuando el sistema operativo determina que un proceso no puede continuar en ese momento.
- **Transiciones 2 y 3**: Son gestionadas por el planificador de procesos, y el proceso afectado no tiene conocimiento de estas transiciones.
- **Transición 4**: Ocurre cuando el evento externo que el proceso estaba esperando sucede.

El **planificador** es el encargado de manejar todas las interrupciones y los detalles de la gestión de los procesos. El sistema operativo utiliza una tabla conocida como la tabla de procesos, que contiene una entrada por cada proceso. Estas entradas, a veces llamadas bloques de control de procesos, almacenan información vital sobre el estado del proceso, como el contador de programa, el apuntador de pila, la asignación de memoria, el estado de los archivos abiertos, datos de contabilidad y planificación, y otros detalles necesarios para reanudar el proceso más adelante como si nunca se hubiera detenido.

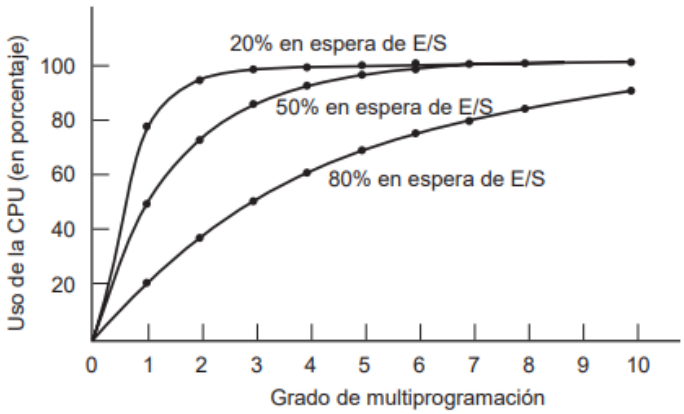

Figura 2-6. Uso de la CPU como una función del número de procesos en memoria.

El **vector de interrupción** es una ubicación específica que guarda la dirección del procedimiento de servicio de interrupciones, permitiendo al sistema operativo manejar adecuadamente las interrupciones que ocurren.

1. El hardware mete el contador del programa a la pila, etc.

2. El hardware carga el nuevo contador de programa del vector de interrupciones.

3. Procedimiento en lenguaje ensamblador guarda los registros.

4. Procedimiento en lenguaje ensamblador establece la nueva pila.

5. El servicio de interrupciones de C se ejecuta (por lo general lee y guarda la entrada en el búfer).

6. El planificador decide qué proceso se va a ejecutar a continuación.

7. Procedimiento en C regresa al código de ensamblador.

8. Procedimiento en lenguaje ensamblador inicia el nuevo proceso actual

2.2HILOS

Hilos: Los hilos son como mini-procesos. La razón principal para utilizar hilos es que tienen la capacidad de compartir un espacio de direcciones y todos sus datos entre sí de manera paralela. Además, son más fáciles de crear y destruir.

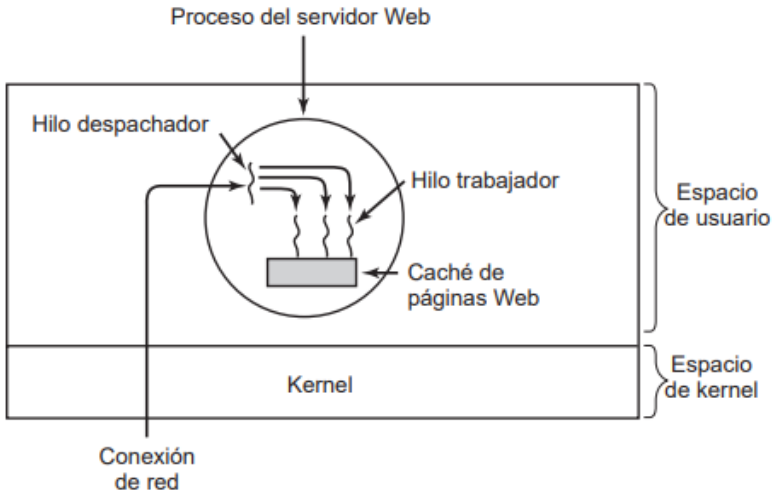

Figura 2-8. Un servidor Web con múltiples hilos.

Los hilos, que son **unidades de procesamiento más pequeñas** que los procesos completos, no mejoran el rendimiento por sí mismos cuando están limitados a una sola CPU. Sin embargo, son muy útiles en tareas que requieren muchos cálculos o en operaciones de entrada/salida. En tales escenarios, los hilos pueden ejecutarse de manera solapada, lo que resulta en una ejecución de la aplicación más eficiente.

Los hilos facilitan la **ejecución paralela** de tareas que de otro modo serían secuenciales, incluso cuando estas tareas involucran llamadas al sistema que

normalmente bloquearían la ejecución, como las operaciones de E/S de disco. El modelo de procesamiento se fundamenta en dos ideas clave: la agrupación de recursos y la ejecución en sí.

Además, cada proceso tiene lo que se conoce como un hilo de ejecución, comúnmente llamado simplemente hilo. Este hilo posee un contador de programa que indica cuál será la próxima instrucción para ejecutar. La introducción de hilos al modelo de procesos permite múltiples ejecuciones simultáneas dentro del mismo entorno de proceso, conocido como **multihilamiento**. Esto también hace posible que el cambio entre hilos se realice en nanosegundos.

En un sistema con una sola CPU, los hilos de un proceso multihilo se turnan para ejecutarse. Los hilos dentro de un mismo proceso no son tan independientes entre sí como lo son los procesos completos.

Un hilo puede encontrarse en uno de varios estados posibles:

- **En ejecución**: El hilo está activo y utilizando la CPU.
- **Bloqueado**: El hilo está en espera de que se resuelva un evento específico para poder continuar.
- **Listo**: El hilo está preparado para ejecutarse y lo hará en cuanto le llegue su turno.
- **Terminado**: El hilo ha completado su ejecución.

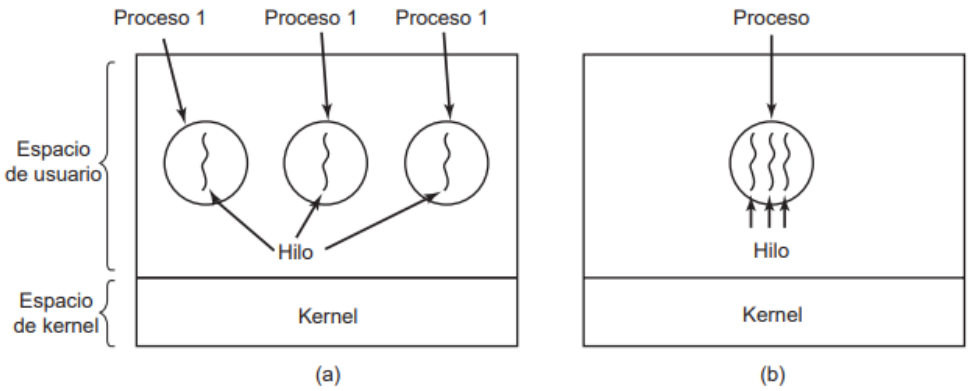

Figura 2-11. (a) Tres procesos, cada uno con un hilo. (b) Un proceso con tres hilos.

Existen dos enfoques principales para implementar paquetes de hilos:

1. **Espacio de Usuario**:
    - En este enfoque, los hilos se implementan completamente en el espacio de usuario, y el kernel no tiene conocimiento de ellos.
    - **Ventajas**:
        - Los paquetes de hilos de nivel usuario pueden implementarse en sistemas operativos que no admiten hilos nativos (implementados desde bibliotecas).
        - Cada proceso mantiene su propia tabla de hilos privada para llevar un registro de los hilos en ese proceso (administrada por el sistema en tiempo de ejecución).
        - La conmutación entre hilos es más rápida que con las interrupciones del kernel.
        - Permite que cada proceso tenga su propio algoritmo de planificación personalizado.
    - **Desventajas**:
        - Las llamadas al sistema de bloqueo se implementan de manera complicada (bloquean todos los hilos del proceso).

        - Los fallos de página pueden afectar a todo el proceso, incluso si otros hilos podrían ejecutarse.
        - Si un hilo comienza a ejecutarse, ningún otro hilo en ese proceso se ejecutará hasta que el primero renuncie voluntariamente a la CPU.

2. **Espacio del Kernel**:
    - En este método, el kernel mantiene una tabla de hilos que registra todos los hilos en el sistema.
    - Cuando un hilo desea crear o destruir otro hilo, realiza una llamada al kernel, que actualiza la tabla de hilos.
    - Para la conmutación de hilos, se utiliza principalmente el método de reciclaje debido al alto costo de la conmutación.
    - Si un hilo produce un fallo de página, el kernel puede verificar si el proceso tiene otros hilos que puedan ejecutarse y, de ser así, ejecutar uno de ellos mientras espera que se traiga la página requerida desde el disco.
3. **Método Híbrido**:
    - En este modelo, el kernel solo es consciente de los hilos de nivel kernel y los planifica.
    - Algunos de estos hilos de nivel kernel pueden tener varios hilos de nivel usuario multiplexados encima de ellos.
    - Los hilos de nivel usuario se crean, destruyen y planifican de manera similar a los hilos de nivel usuario en un sistema operativo sin capacidad de multihilamiento.
4. **Activaciones del Planificador**:

- El objetivo de las activaciones del planificador es imitar la funcionalidad de los hilos de kernel con mejor rendimiento y flexibilidad.
- Cuando el kernel detecta que un hilo se bloquea, notifica al sistema en tiempo de ejecución del proceso mediante una llamada ascendente (upcall).
- Esto permite que los hilos de nivel usuario se comporten de manera similar a los hilos de nivel kernel.

5. **Hilos Emergentes**:
    - Cuando llega un mensaje, el sistema crea un nuevo hilo para manejarlo (hilo emergente o pop-up thread).
    - Los hilos emergentes son rápidos porque no tienen historial que restaurar.
    - Pueden ejecutarse en espacio de kernel, lo que generalmente es más rápido y sencillo que colocarlos en espacio de usuario.
6. **Almacenamiento Compartido**:
    - En algunos sistemas operativos, los procesos que trabajan en conjunto pueden compartir espacio de almacenamiento.
    - Este almacenamiento compartido puede estar en la memoria principal o en archivos compartidos.

Cuando múltiples procesos interactúan con datos compartidos y los resultados varían según el orden y el momento de ejecución, esto se denomina **condiciones de carrera**, ya que los procesos compiten por recursos.

Para prevenir este problema, se utiliza la **exclusión mutua**, que impide que varios procesos lean o escriban datos simultáneamente.

La sección del programa que accede a la memoria compartida se llama región crítica.

Para una exclusión mutua efectiva, se deben cumplir cuatro condiciones:

1. Solo un proceso puede estar en su región crítica a la vez.
2. No se deben hacer suposiciones sobre la velocidad o la cantidad de CPUs.
3. Los procesos fuera de su región crítica no deben bloquear a otros.
4. Los procesos no deben esperar indefinidamente para entrar a su región crítica.

2.3 COMUNICACIÓN ENTRE PROCESOS

Métodos para manejar la exclusión mutua:

- **Desactivación de Interrupciones**: Un proceso desactiva las interrupciones al entrar en su región crítica y las reactiva al salir. Sin embargo, este método es poco práctico ya que otorga demasiado control a los procesos de usuario.
- **Variables de Candado**: Se utiliza una variable compartida inicializada en 0. Un proceso que quiera entrar a su región crítica verifica el candado; si es 0, lo establece en 1 y procede. Si ya es 1, espera. Este método puede fallar si dos procesos entran al mismo tiempo.
- **Candado de Giro**: Una variable entera indica qué proceso puede entrar a su región crítica. Los procesos esperan activamente su turno para entrar, lo que puede ser ineficiente.
- **Solución de Peterson**: Los procesos utilizan su número de identificación para solicitar entrada a la región crítica y esperan si es necesario. Al terminar, señalan su salida para permitir la entrada a otros.

- **Instrucción TSL**: Se usa una variable compartida para coordinar el acceso. Un proceso puede establecer esta variable usando la instrucción TSL y luego acceder a la memoria compartida. Al finalizar, restablece la variable a 0.
- **Sleep y Wakeup**: Sleep suspende un proceso hasta que otro lo despierta con Wakeup. Estas llamadas pueden asociarse a una dirección de memoria para sincronizar procesos.
- **Semáforos**: Un semáforo puede tener valor 0 o positivo, indicando si hay señales de despertar pendientes. Las operaciones down y up ajustan este valor y gestionan el acceso a la región crítica.
- **Mutexes**: Son variables que pueden estar bloqueadas o desbloqueadas. Un proceso solicita acceso a la región crítica con mutex_lock y, si está disponible, procede; de lo contrario, espera.

Llamada de hilo	Descripción
Pthread_mutex_init	Crea un mutex
Pthread_mutex_destroy	Destruye un mutex existente
Pthread_mutex_lock	Adquiere un mutex o se bloquea
Pthread_mutex_trylock	Adquiere un mutex o falla
Pthread_mutex_unlock	Libera un mutex

Figura 2-30. Algunas de las llamadas de Pthreads relacionadas con mutexes.

Las **variables de condición** son herramientas de sincronización que posibilitan el bloqueo de hilos cuando una condición específica aún no se ha cumplido. Existen funciones específicas para inicializar y eliminar estas variables de condición.

Generalmente, las variables de condición se emplean junto con los mutexes. El procedimiento habitual consiste en que un hilo adquiera un mutex y, si no logra conseguir lo que necesita, espere en una variable de condición.

Llamada de hilo	Descripción
Pthread_cond_init	Crea una variable de condición
Pthread_cond_destroy	Destruye una variable de condición
Pthread_cond_wait	Bloquea en espera de una señal
Pthread_cond_signal	Envia señal a otro hilo y lo despierta
Pthread_cond_broadcast	Envia señal a varios hilos y los despierta

Figura 2-31. Algunas de las llamadas a Pthreads que se relacionan con las variables de condición.

Monitores: Son módulos que encapsulan procedimientos, variables y estructuras de datos. Los procesos pueden invocar estos procedimientos pero no acceder directamente a las estructuras internas. Solo un proceso puede operar dentro de un monitor en un momento dado.

Los monitores emplean **variables de condición** y operaciones como **wait** y **signal**. Si un procedimiento no puede avanzar, por ejemplo, si un búfer está lleno,

utiliza wait para bloquearse, permitiendo la entrada a otro proceso. Signal se usa para indicar que un proceso ha terminado su tarea en el monitor.

Sin embargo, los monitores son más un concepto teórico de programación y no se utilizan ampliamente en la práctica.

Comunicación por Mensajes: Se basa en dos acciones fundamentales: **send** y **receive**. Send envía un mensaje a un destinatario, y receive recibe un mensaje de un remitente. Para evitar la pérdida de mensajes, se puede utilizar un sistema de acuse de recibo.

Buzones: Son espacios designados para almacenar mensajes. En lugar de dirigirse a procesos específicos, las llamadas a send y receive se dirigen a estos buzones.

Este método es común en la programación paralela.

Barreras: Se usan en aplicaciones que operan por fases, asegurando que ningún proceso avance a la siguiente fase hasta que todos estén listos.

2.4 PLANIFICACIÓN

Planificación de Procesos: Es la tarea de decidir qué proceso se ejecutará a continuación, optimizando el uso de la CPU.

Los procesos pueden ser:

- **Limitados a cálculos**: Con largas ráfagas de uso de CPU y esperas poco frecuentes por E/S.
- **Limitados a E/S**: Con ráfagas cortas de uso de CPU y esperas frecuentes por E/S.

La planificación ocurre:

1. Al crear un nuevo proceso.
2. Cuando un proceso termina.
3. Si un proceso se bloquea.
4. Durante una interrupción de E/S.

Algoritmos de Planificación:

- **No apropiativos**: Permiten que un proceso se ejecute hasta que se bloquee o ceda la CPU voluntariamente.
- **Apropiativos**: Asignan un tiempo fijo de ejecución a cada proceso y, si el proceso sigue activo al finalizar, se suspende para dar paso a otro.

Un algoritmo de planificador de procesos se caracteriza por:

1. Mantener ocupadas todas las partes del sistema siempre que sea posible y la equidad de procesos.

2. El rendimiento es el número de trabajos por hora que completa el Sistema.

3. El tiempo de retorno es el tiempo estadísticamente promedio desde el momento en que se envía un trabajo por lotes, hasta el momento en que se completa.

4. Para los sistemas interactivos se aplican distintas metas. La más importante es minimizar el tiempo de respuesta; es decir, el tiempo que transcurre entre emitir un comando y obtener el resultado.

Algoritmos de Planificación de Procesos:

1. **FCFS (Primero en entrar, primero en ser atendido)**:
    - No apropiativo.
    - Asigna la CPU a los procesos en el orden de llegada.
    - Única cola de procesos listos.
    - Si un proceso en ejecución se bloquea, el siguiente proceso en la cola toma la CPU.
2. **SJF (Shortest Job First)**:
    - No apropiativo.
    - Selecciona el trabajo más corto en la cola de entrada.
    - Óptimo cuando todos los trabajos están disponibles al mismo tiempo.

3. **SRTN (Shortest Remaining Time Next)**:
    - Apropiativo.
    - Selecciona el proceso con el tiempo restante más corto.
    - Requiere conocer los tiempos de ejecución de antemano.
4. **Round Robin**:
    - Asigna a cada proceso un intervalo de tiempo (quantum).
    - Si un proceso sigue activo al final del quantum, se suspende para dar paso a otro.
    - La longitud del quantum afecta la eficiencia y la respuesta a peticiones interactivas.
5. **Prioridades**:
    - Asigna prioridades a los procesos.
    - El proceso con la prioridad más alta se ejecuta.
    - Puede reducir la prioridad del proceso actual para evitar ejecución indefinida.
6. **Planificación por Sorteos**:
    - Asigna boletos de lotería a los procesos para recursos como la CPU.
    - Se selecciona un boleto al azar para decidir qué proceso obtiene el recurso.
7. **Planificación por Equidad de Partes**:
    - Asigna a cada usuario una fracción de la CPU.
    - El planificador selecciona procesos para cumplir con esta asignación.
8. **Planificación en Tiempo Real**:

- Categorías: tiempo real duro (con límites absolutos) y tiempo real suave (tolerancia a incumplimientos ocasionales).
- Divide el programa en procesos predecibles según eventos periódicos o aperiódicos.

3 ADMINISTRACIÓN DE MEMORIA

El Desafío del Crecimiento de los Programas y la Jerarquía de Memoria:

- **Jerarquía de Memoria**:
    - Las computadoras cuentan con diferentes niveles de memoria:
        - **Memoria Caché**: Es rápida, costosa y volátil, pero tiene poca capacidad (megabytes).
        - **Memoria Principal**: Es de velocidad media, precio moderado y volátil, pero tiene mayor capacidad (gigabytes).
        - **Almacenamiento en Disco**: Es lento, económico y no volátil, pero tiene una gran capacidad (terabytes).
- **Administrador de Memoria**:
    - Su función es gestionar la memoria eficientemente:
        - Registrar qué partes de la memoria están en uso.
        - Asignar memoria a los procesos cuando la necesitan.
        - Liberar memoria cuando los procesos terminan.

3.1 SIN ABSTRACCIÓN DE MEMORIA

- En este enfoque, los programas ven directamente la memoria física.
- No se pueden ejecutar varios programas simultáneamente.
- La solución es programar con múltiples hilos, pero esto no es ideal.

- **Espacio de Direcciones**:
    - Cada proceso tiene su propio conjunto de direcciones locales.
    - La reubicación dinámica asigna el espacio de direcciones de cada proceso a una parte diferente de la memoria física.

- **Registros Base y Límite**:
    - Estos registros permiten cargar programas en ubicaciones consecutivas de memoria sin reubicación durante la carga.
    - El registro base indica la dirección física donde comienza el programa, y el registro límite establece su longitud.

3.2ESPACIOS DE DIRECCIONES

Es el **conjunto de direcciones** que puede utilizar un proceso para direccionar la memoria. Cada proceso tiene su propio espacio de direcciones.

Reubicación dinámica: Lo que hace es asociar el espacio de direcciones de cada proceso sobre una parte distinta de la memoria física, de una manera simple.

Los registros base y límite. Cuando se utilizan estos registros, los programas se cargan en ubicaciones consecutivas de memoria en donde haya espacio y sin reubicación durante la carga. El registro base se carga con la dirección física donde empieza el programa en memoria y el registro límite se carga con la longitud del programa.

Cada vez que un proceso hace referencia a la memoria, el hardware de la CPU suma de manera automática el valor base a la dirección generada por el proceso. Al mismo tiempo comprueba si la dirección ofrecida es igual o mayor que el valor resultante de sumar los valores de los registros límite y base.

Una **desventaja** de la reubicación usando los registros base y límite es la necesidad de realizar una suma y una comparación en cada referencia a memoria

Sobrecarga de Memoria:

La cantidad total de RAM que requieren todos los procesos es a menudo mucho mayor de lo que puede acomodarse en memoria

Se han desarrollado dos esquemas generales para lidiar con la sobrecarga de memoria:

intercambio y memoria virtual

Intercambio

Consiste en llevar cada proceso completo a memoria, ejecutarlo durante cierto tiempo y después regresarlo al disco.

Cuando el intercambio crea varios huecos en la memoria, es posible combinarlos todos en uno grande desplazando los procesos lo más hacia abajo que sea posible. Esta técnica se conoce como compactación de memoria. Por lo general no se realiza debido a que requiere mucho tiempo de la CPU.

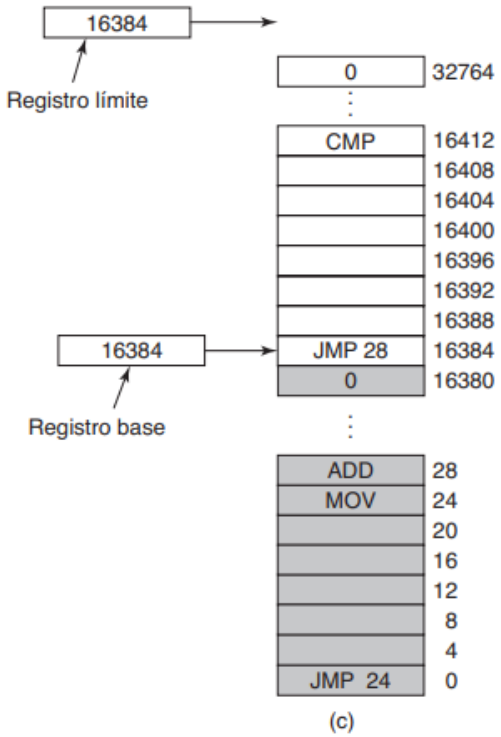

Figura 3-3. Se pueden utilizar registros base y límite para dar a cada proceso un espacio de direcciones separado.

- **Asignación Estática**:
    - El sistema operativo asigna exactamente la cantidad necesaria de memoria, sin excederse.
    - Si se espera que la mayoría de los procesos crezcan durante su ejecución, puede ser conveniente asignar un poco de memoria adicional al intercambiar o mover un proceso.

- Sin embargo, al intercambiar procesos al disco, solo se debe intercambiar la memoria en uso; no tiene sentido intercambiar la memoria adicional.

- **Asignación Dinámica**:
  - En este caso, el sistema operativo debe administrar la memoria.
  - Hay dos métodos para llevar un registro del uso de la memoria:
    - **Mapas de Bits**: Se utilizan para rastrear qué partes de la memoria están ocupadas o libres.
    - **Listas Libres**: Mantienen una lista de bloques de memoria disponibles para asignación.

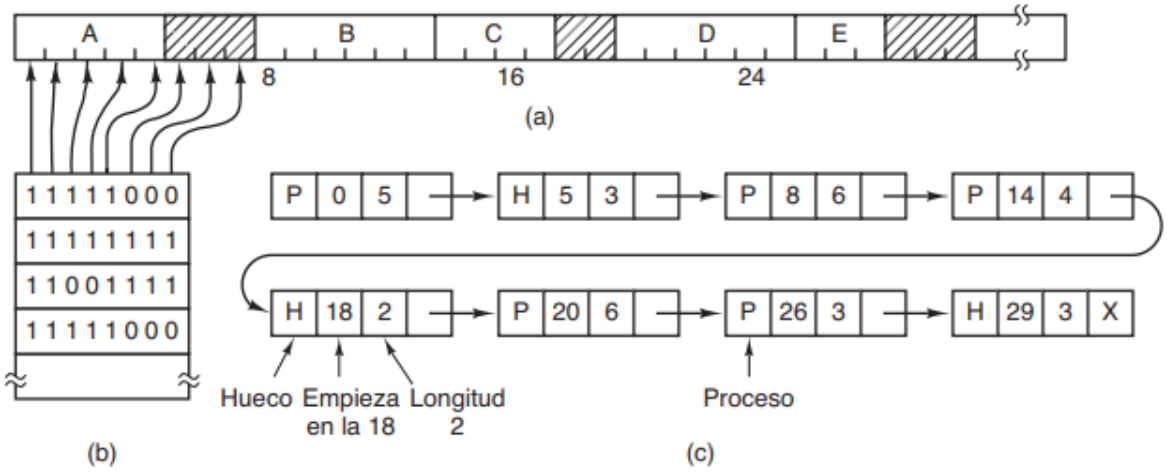

Figura 3-6. (a) Una parte de la memoria con cinco procesos y tres huecos. Las marcas de graduación muestran las unidades de asignación de memoria. Las regiones sombreadas (0 en el mapa de bits) están libres. (b) El mapa de bits correspondiente. (c) La misma información en forma de lista.

Mapa de Bits: Para cada unidad de asignación hay un bit correspondiente en el mapa de bits, que es 0 si la unidad está libre y 1 si está ocupada.

Un mapa de bits proporciona una manera simple de llevar el registro de las palabras de memoria en una cantidad fija de memoria, debido a que el tamaño del mapa de bits sólo depende del tamaño de la memoria y el tamaño de la unidad de asignación.

- **Mapa de Bits**:

    - Cada unidad de asignación tiene un bit correspondiente en el mapa de bits.
    - El bit es 0 si la unidad está libre y 1 si está ocupada.
    - El mapa de bits es simple y su tamaño depende solo de la memoria y la unidad de asignación.
- **Lista Ligada**:
    - Es una lista de segmentos de memoria asignados y libres.
    - Cada entrada especifica un hueco (H) o un proceso (P), con dirección de inicio, longitud y un apuntador a la siguiente entrada.
    - Las combinaciones de la figura 3-7 incluyen fusionar entradas y reducir la lista.
- **Algoritmos de Asignación de Memoria**:
    - **Primer Ajuste**: Busca un hueco lo suficientemente grande y lo divide en partes para el proceso y la memoria no utilizada.
    - **Siguiente Ajuste**: Similar al primer ajuste, pero mantiene un registro de la última posición explorada.
    - **Mejor Ajuste**: Busca el hueco más pequeño adecuado en toda la lista.
    - **Peor Ajuste**: Siempre toma el hueco más grande disponible.
    - **Ajuste Rápido**: Mantiene listas separadas para tamaños comunes solicitados.

3.3 MEMORIA VIRTUAL

La necesidad de ejecutar programas que exceden la capacidad de memoria y la posibilidad de tener varios programas en ejecución al mismo tiempo llevaron a soluciones como los **sobrepuestos**. Estos consistían en dividir los programas en partes más pequeñas y mantenerlas en el disco, intercambiándolas según fuera necesario mediante un administrador de sobrepuestos.

Aunque el sistema operativo se encargaba del intercambio de sobrepuestos en la memoria, la división inicial del programa en partes debía ser realizada manualmente por el programador.

La **memoria virtual** surgió como una solución definitiva. Cada programa tiene su propio espacio de direcciones, dividido en páginas contiguas. Estas páginas se asocian a la memoria física, pero no todas deben estar en ella para ejecutar el programa. Cuando el programa hace referencia a una parte en memoria física, el hardware realiza la asociación instantáneamente. Si la parte no está en memoria física, el sistema operativo busca la información faltante y reanuda la instrucción que falló.

Las tres técnicas para manejar la memoria virtual son: **paginación**, **segmentación** y **segmentación paginada**.

1. **Paginación**:
    - Los programas en una computadora utilizan un conjunto de direcciones conocidas como direcciones virtuales, que componen el espacio de direcciones virtuales.
    - Estas direcciones se dividen en un número de página y un desplazamiento dentro de esa página.
    - El número de página sirve como un índice en la tabla de páginas, que apunta al marco de página correspondiente en la memoria física.
    - La MMU (Unidad de Administración de Memoria) es la encargada de convertir las direcciones virtuales en direcciones físicas.
    - El espacio de direcciones virtuales está organizado en páginas, que se mapean a marcos de página en la memoria física, generalmente del mismo tamaño.
    - Un bit de presente/ausente en el hardware indica si una página está en la memoria física.

- Si una página no está presente, se produce un fallo de página, y el sistema operativo interviene para cargar la página necesaria y continuar la ejecución.
- **Tabla de Páginas**:
    - Utiliza el número de página virtual para encontrar la entrada correspondiente y el número de marco de página asociado.
    - La dirección física se forma combinando el número del marco de página con el desplazamiento.
- **Estructura de la Tabla de Páginas**:
    - El número de marco de página es el dato más crucial.
    - El bit de presente/ausente valida la entrada.
    - Los bits de protección definen los tipos de acceso permitidos (lectura, escritura, ejecución).
    - Los bits de modificada y referenciada se actualizan automáticamente al escribir o acceder a una página.
    - Un bit adicional puede desactivar la caché para páginas asociadas con registros de dispositivos.
- **Optimización de la Paginación**:
    - La conversión de direcciones virtuales a físicas debe ser eficiente.
    - Si el espacio de direcciones virtuales es extenso, la tabla de páginas puede ser muy grande.
    - **Búferes de Traducción Adelantada (TBL)**:
        - Basados en la tendencia de los programas a acceder frecuentemente a un número limitado de páginas.

- El TBL, ubicado en la MMU, es una tabla pequeña que acelera la traducción de direcciones sin consultar la tabla de páginas completa.

Cuando se presenta una **dirección virtual a la MMU** para que la traduzca, el hardware primero comprueba si su número de página virtual está presente en el TLB al compararla con todas las entradas en forma simultánea (es decir, en paralelo). Si se encuentra una coincidencia válida y el acceso no viola los bits de protección, el marco de página se toma directamente del TLB. Cuando el número de página virtual no está en el TLB. La **MMU** detecta que no está y realiza una búsqueda ordinaria en la tabla de páginas. Después desaloja una de las entradas del TLB y la reemplaza con la entrada en la tabla de páginas que acaba de buscar.

También se puede implementar **TBL** desde software, cargando la tabla de TBL de manera explícita en el Sistema Operativo. Esto tiene la ventaja de quitar carga a la MMU y CPU, ya que todo es administrado por el SO.

Paginación de Múltiples Niveles: La estrategia clave de la paginación multinivel es no retener todas las tablas de página en la memoria constantemente, especialmente las que no son utilizadas frecuentemente.

Tablas de Páginas Invertidas: Para manejar espacios de direcciones virtuales de 64 bits paginados, es necesario adoptar un enfoque distinto. Una solución es la implementación de tablas de páginas invertidas. Este método consiste en tener una entrada para cada marco de página física, en lugar de una por cada página del espacio de direcciones virtuales, lo cual es lo opuesto a las tablas de páginas convencionales. La principal ventaja de este sistema es el significativo ahorro de espacio. Sin embargo, la desventaja es que la conversión de direcciones virtuales a físicas se complica más. Para resolver este problema, se puede recurrir al uso del **TLB (Translation Lookaside Buffer)**. Si el TLB logra almacenar todas las páginas que se acceden con regularidad, la conversión de direcciones se puede realizar tan eficientemente como con las tablas de páginas estándar.

3.4 ALGORITMOS DE REEMPLAZO DE PÁGINAS

Cuando ocurre un fallo de página, el sistema operativo tiene que elegir una página para desalojarla (eliminarla de memoria) y hacer espacio para la página entrante. Si la página a eliminar se modificó mientras estaba en memoria, debe volver a escribirse en el disco para actualizar la copia de este.

Si se elimina una página de uso frecuente, tal vez tenga que traerse de vuelta rápidamente, lo cual produce una sobrecarga adiciona.

Existen diferentes tipos de algoritmos de reemplazo:

1. **Algoritmo de Reemplazo Óptimo**: El algoritmo óptimo de reemplazo de páginas es fácil de describir, pero imposible de implementar. Se basa en etiquetar cada página con el número de instrucciones que se ejecutarán antes de que se haga referencia por primera vez a esa página. Según este algoritmo, la página con la etiqueta más alta debería eliminarse. Sin embargo, el problema radica en que, al momento del fallo de página, el sistema operativo no puede predecir cuándo ocurrirá la próxima referencia a cada página.
2. **No Usadas Recientemente (NRU)**: En este enfoque, cada página en memoria virtual tiene dos bits de estado asociados: R (referenciada) y M (modificada). Cuando se inicia un proceso, ambos bits de página para todas sus páginas se establecen en 0. Tras un fallo de página, el sistema operativo inspecciona todas las páginas y las divide en cuatro categorías según los valores actuales de los bits R y M:
    - Clase 0: No ha sido referenciada ni modificada.
    - Clase 1: No ha sido referenciada pero ha sido modificada.
    - Clase 2: Ha sido referenciada pero no modificada.
    - Clase 3: Ha sido referenciada y modificada. NRU elimina una página al azar de la clase con menor nivel.

3. **Primera en Entrar, Primera en Salir (FIFO)**: En este algoritmo, el sistema operativo mantiene una lista de todas las páginas en memoria, donde la llegada más reciente está al final y la menos reciente al principio. Sin embargo, FIFO no garantiza que la página más antigua sea la menos utilizada.

4. **Segunda Oportunidad**: Una modificación simple de FIFO consiste en inspeccionar el bit R de la página más antigua. Si es 0, la página es antigua y no se ha utilizado, por lo que se sustituye de inmediato. Si el bit R es 1, se borra y la página se coloca al final de la lista como si acabara de llegar a la memoria. El objetivo es encontrar una página antigua que no se haya referenciado recientemente.

5. **Clock**: Aunque el algoritmo de segunda oportunidad es razonable, mueve constantemente páginas en la lista. Una mejora es mantener todos los marcos de página en una lista circular en forma de reloj, donde la manecilla apunta a la página más antigua.

6. **Menos Usadas Recientemente (LRU)**: LRU sugiere descartar la página que no se ha utilizado durante el mayor tiempo. Aunque teóricamente es realizable, su implementación completa requiere mantener una lista enlazada de todas las páginas en memoria, actualizándola en cada referencia.

7. **No Usadas Frecuentemente (NFU)**: Este algoritmo es una implementación de LRU por software. Cada página tiene un contador asociado que se actualiza en cada interrupción de reloj. Sin embargo, el problema es que nunca olvida nada.

8. **Envejecimiento (Aging)**: Modifica NFU al desplazar los contadores a la derecha antes de agregar el bit R. Luego, el bit R se agrega al bit más a la izquierda en lugar de al más a la derecha.

9. **Modelo de Conjunto de Trabajo (WS**): En su forma más elemental, la paginación comienza con los procesos sin páginas en memoria. La primera instrucción solicitada por la CPU resulta en un fallo de página. Con el tiempo,

el proceso acumula la mayoría de sus páginas necesarias, permitiéndole operar con mínimos fallos de página, conocido como paginación bajo demanda. Los procesos muestran localidad de referencia, indicando que solo una pequeña porción de sus páginas es referenciada en cualquier fase de ejecución. Este subconjunto de páginas activas se denomina conjunto de trabajo. Si este conjunto está completamente en memoria, el proceso corre fluidamente hasta que transita a otra fase. Los sistemas de paginación intentan monitorear y mantener en memoria el conjunto de trabajo de cada proceso. Este enfoque se llama modelo de conjunto de trabajo. Dado que el conjunto de trabajo cambia gradualmente, se pueden predecir las páginas necesarias al reiniciar un programa basándose en su conjunto de trabajo previo. La pre-paginación carga estas páginas anticipadamente. El sistema operativo lleva registro del conjunto de trabajo y, ante un fallo de página, decide qué página desalojar. El conjunto de trabajo se define por las páginas usadas en los últimos n milisegundos de tiempo de ejecución del proceso. El tiempo total de CPU usado por un proceso se llama tiempo virtual actual. Bajo este modelo, el conjunto de trabajo de un proceso comprende las páginas referenciadas en los últimos τ segundos de tiempo virtual. Funcionamiento de WS: Cada entrada en la tabla de páginas incluye el tiempo aproximado del último uso y el bit R. En un fallo de página, se busca una página para desalojar. Si el bit R es 1, se actualiza el tiempo de último uso. Si es 0, se evalúa su edad comparándola con τ para determinar si sigue en el conjunto de trabajo.

10. **WS Clock**: El método básico de conjunto de trabajo puede ser tedioso al requerir revisar toda la tabla de páginas por un candidato en cada fallo de página. Una mejora es el algoritmo WSClock, que combina el enfoque de reloj con la noción de conjunto de trabajo. Se utiliza una lista circular de marcos de página, cada uno con el tiempo de último uso y los bits R y M. En un fallo de página, se revisa la página indicada por la manecilla. Si R es 1, se descarta como candidata y se avanza la manecilla. Si R es 0 y la página

es mayor que τ y no está modificada, se reemplaza. Si está modificada, se programa su escritura en disco y se continúa buscando una página antigua y no modificada para reemplazo inmediato.

3.5 CUESTIONES DE DISEÑO PARA LOS SISTEMAS DE PAGINACIÓN

1. **Políticas de asignación**:
    - Los algoritmos locales asignan una fracción fija de memoria a cada proceso.
    - Los algoritmos globales asignan marcos de página dinámicamente entre procesos ejecutables.
    - Los algoritmos globales suelen funcionar mejor cuando el tamaño del conjunto de trabajo varía durante la vida de un proceso.
    - El algoritmo de frecuencia de fallo de páginas (PFF) ajusta la asignación de páginas a un proceso según su comportamiento.
2. **Control de cargas**:
    - Intercambiar procesos al disco y liberar sus páginas ayuda a reducir la competencia por la memoria.
    - Periódicamente, algunos procesos se traen del disco y otros se intercambian.
3. **Tamaño de página**:
    - No hay un tamaño óptimo universal.
    - Páginas grandes pueden generar fragmentación interna.
    - Páginas pequeñas pueden requerir una tabla de páginas grande y aumentar las transferencias hacia/desde el disco.
4. **Espacio separado de direcciones**:

- Espacios I y D separados para instrucciones y datos.
- Cada espacio tiene su propia tabla de páginas y asignación de páginas virtuales a marcos físicos.

5. **Páginas compartidas**:
    - En sistemas de multiprogramación, varios procesos pueden compartir código.
    - En sistemas paginados, se comparte una página en SOLO LECTURA.
    - Si se actualiza, se copia para tener una versión LECTURA-ESCRITURA privada.
6. **Bibliotecas compartidas**:
    - Vinculación con rutinas auxiliares en tiempo de ejecución.
    - Se cargan páginas según sea necesario.
    - Actualizaciones no requieren recompilar programas que las utilizan.
7. **Archivos asociados**:
    - Asociar archivos a porciones de espacio de direcciones virtuales.
    - Paginación bajo demanda desde el archivo de disco.
    - Escritura de páginas modificadas al archivo al desasociar.
8. **Política de limpieza**:
    - El demonio de paginación selecciona páginas para desalojar según un algoritmo de reemplazo.
    - Si las páginas han sido modificadas, se escriben en disco.
    - El demonio recuerda el contenido anterior de las páginas desalojadas.

Interfaz de memoria virtual: Una razón por la que se otorga a los programadores el control sobre su mapa de memoria es para permitir que dos o más procesos

compartan la misma memoria. Participación del sistema operativo en la paginación: Hay cuatro ocasiones en las que el sistema operativo tiene que realizar trabajo relacionado con la paginación: al crear un proceso, al ejecutar un proceso, al ocurrir un fallo de página y al terminar un proceso.

Crear un proceso: Determinar su tamaño inicial, crear una TDP, asignarla a memoria y determinar el espacio de intercambio en disco

Cuando se ejecuta un proceso: MMU se tiene que restablecer para el nuevo proceso y el TLB se vacía para deshacerse de los restos del proceso que se estaba ejecutando antes.

Cuando ocurre un fallo de página: Lee los registros del hardware para determinar qué dirección virtual produjo un fallo, calcular la página que necesita, localizarla, buscar su marco de página y colocarla en la tabla eliminando alguna página en caso de que no haya espacio.

Cuando finaliza un proceso: Debe liberar su tabla de páginas, sus páginas y el espacio en disco que ocupan las páginas cuando están en disco. Si sus páginas están compartidas, se mantienen hasta que nadie más las use.

Manejo de fallos de página:

1. El hardware hace un trap al kernel, guardando el contador de programa en la pila.

2. Se inicia una rutina en código ensamblador para guardar los registros generales y demás información volátil, para evitar que el sistema operativo la destruya.

3. El sistema operativo descubre que ha ocurrido un fallo de página y trata de descubrir cuál página virtual se necesita.

4. Una vez que se conoce la dirección virtual que produjo el fallo, el sistema comprueba si esta dirección es válida y si la protección es consistente con el acceso. De no ser así, el proceso recibe una señal o es eliminado. Si es válida, comprueba si hay un marco de página disponible. Si no hay marcos disponibles, se ejecuta el algoritmo de reemplazo de páginas para seleccionar una víctima.

5. Si el marco de página seleccionado está sucio, la página se planifica para transferirla al disco y se realiza una conmutación de contexto, suspendiendo el proceso fallido y dejando que se ejecute otro hasta que se haya completado la transferencia al disco.

6. Tan pronto como el marco de página esté limpio, el sistema operativo busca la dirección de disco en donde se encuentra la página necesaria, y planifica una operación de disco para llevarla a memoria. Mientras se está cargando la página, el proceso fallido sigue suspendido y se ejecuta otro proceso de usuario, si hay uno disponible.

7. Cuando la interrupción de disco indica que la página ha llegado, las tablas de páginas se actualizan para reflejar su posición y el marco se marca como en estado normal.

8. La instrucción fallida se respalda al estado en que tenía cuando empezó, y el contador de programa se restablece para apuntar a esa instrucción.

9. El proceso fallido se planifica y el sistema operativo regresa a la rutina.

10. Esta rutina recarga los registros y demás información de estado, regresando al espacio de usuario para continuar la ejecución, como si no hubiera ocurrido el fallo.

Bloqueo de páginas en memoria: La memoria virtual y la E/S interactúan en formas sutiles. Existe la posibilidad de que en algún momento el algoritmo de reemplazo elimine las páginas de un proceso suspendido en espera a un dispositivo de E/S.

Una solución a este problema es bloquear las páginas involucradas en operaciones de E/S en memoria, de manera que no se eliminen. Bloquear una página se conoce a menudo como fijada en memoria.

Almacén de respaldo: El algoritmo más simple para asignar espacio de página en el disco es tener una partición de intercambio especial en el disco. Esta partición no tiene un sistema de archivos normal, lo cual elimina la sobrecarga.

La partición de intercambio se administra como una lista de trozos libres. Con cada proceso está asociada la dirección de disco de su área de intercambio; es decir, en

qué parte de la partición de intercambio se mantiene su imagen. Esta información se mantiene en la tabla de procesos.

Este simple modelo tiene un problema: los procesos pueden incrementar su tamaño antes de empezar. En consecuencia, podría ser mejor reservar áreas de intercambio separadas para el texto, los datos y la pila, permitiendo que cada una de estas áreas consista en más de un trozo en el disco.

3.7 SEGMENTACIÓN

La memoria virtual que hemos analizado hasta ahora es unidimensional, debido a que las direcciones virtuales van desde 0 hasta cierta dirección máxima, una dirección después de la otra.

Por lo general un programa se puede dividir en muchas tablas a medida que se compila, por ejemplo: El texto del código Fuente, La tabla de símbolos, La tabla que contiene todas las constantes enteros y flotantes, El árbol de análisis sintáctico, La pila utilizada para las llamadas a procedimientos, etc.

Cada una de las primeras cuatro tablas crece en forma continua a medida que procede la compilación.

Lo que se necesita realmente es una forma de liberar al programador de tener que administrar las tablas en expansión y contracción, de la misma forma que la memoria virtual elimina la preocupación de tener que organizar el programa en sobrepuestos.

Una solución es proporcionar la máquina con muchos espacios de direcciones por completo independientes,

llamados segmentos. Cada segmento consiste en una secuencia lineal de direcciones, desde 0 hasta cierto valor máximo. Los distintos segmentos pueden tener distintas longitudes. Además, las longitudes de los segmentos pueden cambiar durante la ejecución.

Por lo general los segmentos son muy grandes. Para especificar una dirección, el programa debe suministrar una dirección en dos partes, un número de segmento y una dirección dentro del segmento.

Un segmento es una entidad lógica, de la cual el programador está consciente y la cual no contiene una mezcla de distintos tipos.

La segmentación también facilita la compartición de procedimientos o datos entre varios procesos. Un ejemplo común es la biblioteca compartida.

Los distintos segmentos pueden tener diferentes tipos de protección.

En una memoria segmentada, el usuario está consciente de lo que hay en cada segmento. Como cada segmento contiene sólo un tipo de objeto, puede tener la protección apropiada para ese tipo específico.

Una vez que el sistema haya estado en ejecución por un tiempo, la memoria se dividirá en un número de trozos, de los cuales algunos contendrán segmentos y otros huecos. Este fenómeno, llamado efecto fragmentación externa.

SEGMENTACIÓN - PÁGINADA:

Si los segmentos son extensos, puede ser inconveniente (o incluso imposible) mantenerlos completos en memoria principal. Esto nos lleva a la idea de paginarlos, de manera que sólo las páginas que realmente se necesiten tengan que estar presentes.

MULTICS: Proveía a cada programa con una memoria virtual de hasta 218 segmentos, cada uno hasta 65,536 palabras (36 bits) de longitud. Considerar a cada segmento como una memoria virtual y la paginaron, combinando las ventajas de la paginación (tamaño de página uniforme y no tener que mantener todo el segmento en la memoria, si sólo se está utilizando parte de él) con las ventajas de la segmentación (facilidad de programación, modularidad, protección, compartición).

Tiene una tabla de segmentos, con un descriptor por segmento. La tabla de segmentos es en sí un segmento y se página. Un descriptor de segmentos contiene una indicación acerca de si el segmento está

en memoria principal o no, gracias a un apuntador de 18 bits a su tabla de páginas. El descriptor también contiene el tamaño del segmento, los bits de protección y unos cuantos elementos más.

Cada segmento se página de la misma forma que la memoria paginada no segmentada.

Una dirección en MULTICS consiste en dos partes: el segmento (s) y la dirección dentro del segmento.

La dirección dentro del segmento se divide aún más en un número de página (p) y en una palabra dentro de la página (d).

Se utiliza un registro (el registro base del descriptor) para localizar la tabla de páginas del segmento del descriptor.

INTEL PENTIUM: El Pentium tiene 16K segmentos independientes, cada uno de los cuales contiene hasta unos mil millones de palabras de 32 bits.

La memoria virtual del Pentium consiste en dos tablas, llamadas LDT (Local Descriptor Table, Tabla de descriptores locales) y GDT (Global Descriptor Table, Tabla de descriptores globales). Cada programa tiene su propia LDT, pero hay una sola GDT compartida por todos los programas en la computadora. La LDT describe los segmentos locales para cada programa, incluyendo su código, datos, pila, etcétera, mientras que la GDT describe los segmentos del sistema, incluyendo el sistema operativo en sí.

El registro CS contiene el selector para el segmento de código y el registro DS contiene el selector para el segmento de datos. Uno de los bits del selector indica si el segmento es local o global.

Tan pronto como el microprograma sabe cuál registro de segmento se está utilizando, puede encontrar el descriptor completo que corresponde a ese selector en sus registros internos.

Después, el hardware utiliza el campo Límite para comprobar si el desplazamiento está más allá del final del segmento. El campo Límite es el tamaño de segmento exacto.

El campo Base permite que cada segmento empiece en un lugar arbitrario dentro del espacio de direcciones lineal de 32 bits.

Cada programa en ejecución tiene un directorio de páginas que consiste de 1024 entradas de 32 bits.

Dir se utiliza para indexar en el directorio de páginas y localizar un apuntador a la tabla de páginas apropiada

Para evitar realizar referencias repetidas a memoria, el Pentium (al igual que MULTICS) tiene un pequeño

TLB que asigna directamente las combinaciones Dir-Página de uso más reciente a la dirección física del marco de página.

El Pentium admite cuatro niveles de protección, donde el nivel 0 es el más privilegiado y el 3 el menos privilegiado. Cada segmento en el sistema también tiene un nivel. Un programa puede utilizar segmentos de su propio nivel o superior, nunca inferior.

4. SISTEMA DE ARCHIVOS

En consecuencia, tenemos tres requerimientos esenciales para el almacenamiento de información a largo plazo:

1. Debe ser posible almacenar una cantidad muy grande de información.

2. La información debe sobrevivir a la terminación del proceso que la utilice.

3. Múltiples procesos deben ser capaces de acceder a la información concurrentemente

Los archivos son unidades lógicas de información creada por los procesos. Los procesos pueden leer los archivos existentes y crear otros si es necesario. La información que se almacena en los archivos debe ser persistente, es decir, no debe ser afectada por la creación y terminación de los procesos.

La parte del sistema operativo que trata con los archivos se conoce como sistema de archivos.

Los archivos son un mecanismo de abstracción. Proporcionan una manera de almacenar información en el disco y leerla después. La parte que va después del punto se conoce como la extensión del archivo y por lo general indica algo acerca de su naturaleza.

Los archivos se pueden estructurar en una de varias formas:

a. Como una secuencia de bytes sin estructura: el sistema operativo no sabe, ni le importa, qué hay en el archivo. Cualquier significado debe ser impuesto por los programas a nivel usuario (más usada).

b. Como un archivo es una secuencia de registros de longitud fija, cada uno con cierta estructura interna, con la idea de que la operación de lectura devuelva un registro y la operación de escritura sobrescriba o agregue un registro.

c. En esta organización, un archivo consiste de un árbol de registros, donde no todos son necesariamente de la misma longitud; cada uno de ellos contiene un campo

llave en una posición fija dentro del registro. El árbol se ordena con base en el campo llave para permitir una búsqueda rápida por una llave específica.

Los archivos regulares son los que contienen información del usuario. Estos pueden ser en código ASCII o binario.

4.2 Directorios

Los directorios son sistemas de archivos para mantener la estructura del sistema de archivos.

Los archivos especiales de caracteres se relacionan con los dispositivos de E/S en serie, y los archivos especiales de bloques se utilizan para modelar discos.

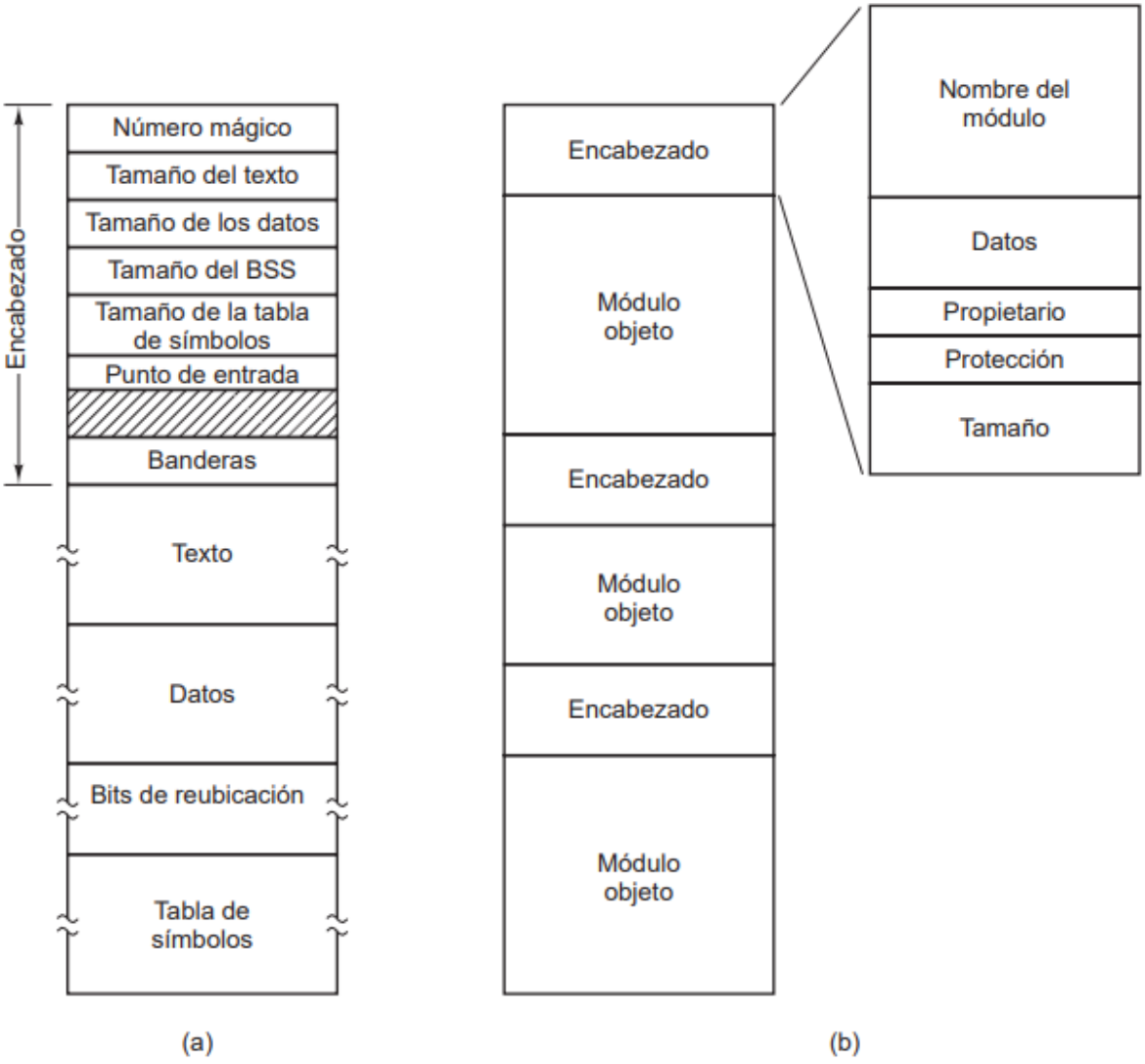

Figura 4-3. (a) Un archivo ejecutable. (b) Un archivo.

El número mágico, que identifica al archivo como un archivo ejecutable.

Tipos de acceso a archivos:

Acceso secuencial: Un proceso podía leer todos los bytes o registros en un archivo en orden, empezando desde el principio, pero no podía saltar algunos y leerlos fuera de orden. Permitía rebobinarse para poder ser releídos.

Los archivos cuyos bytes o registros se pueden leer en cualquier orden se llaman archivos de acceso aleatorio.

Cada archivo tiene un conjunto de datos conocidos como metadatos o atributos.

Las llamadas al sistema más comunes relacionadas con los archivos:

1. **Creación**: Se genera un archivo vacío, anunciando su existencia y definiendo sus propiedades iniciales.
2. **Eliminación**: Para liberar espacio en disco, se borra el archivo cuando ya no es necesario mediante una función específica.
3. **Apertura**: Un proceso debe activar el archivo para usarlo, lo que permite al sistema preparar los atributos y direcciones para un acceso rápido posteriormente.
4. **Cierre**: Finalizado el uso, se cierra el archivo para desocupar recursos, lo que puede incluir la escritura del último bloque de datos, aunque no esté completo.
5. **Lectura**: Se recuperan datos del archivo, normalmente desde la posición actual, y se requiere especificar la cantidad y proporcionar un espacio para almacenarlos.
6. **Escritura**: Se añaden o sobrescriben datos en el archivo, lo que puede alterar su tamaño o eliminar información existente.
7. **Añadir**: Función especializada de escritura que solo permite agregar datos al final del archivo.

8. **Búsqueda**: Para archivos con acceso no secuencial, se usa esta función para posicionar el punto de lectura/escritura en un lugar específico.
9. **Obtener atributos**: Los procesos pueden necesitar consultar las propiedades de un archivo, como los tiempos de modificación, para gestionar tareas como compilaciones.
10. **Establecer atributos**: Permite al usuario modificar ciertas propiedades del archivo después de su creación, como los permisos de acceso.
11. **Renombrar**: Cambia el nombre de un archivo existente, una acción que puede ser reemplazada por copiar y eliminar el archivo original si es necesario.

Para llevar el registro de los archivos, los sistemas de archivos por lo general tienen directorios o carpetas. La forma más común es usar directorios en forma de jerarquía. Con este esquema, puede haber tantos directorios como se necesite para agrupar los archivos en formas naturales.

Además, si varios usuarios comparten un servidor de archivos común, como se da el caso en muchas redes de empresas, cada usuario puede tener un directorio raíz privado para su propia jerarquía.

Nombre de ruta absoluto que consiste en la ruta desde el directorio raíz al archive al final. En cambio, el nombre de ruta relativa. Éste se utiliza en conjunto con el concepto del directorio de trabajo (también llamado directorio actual).

Cada proceso tiene su propio directorio de trabajo. Los procedimientos de biblioteca raras veces cambian el directorio de trabajo y cuando deben hacerlo siempre lo devuelven a su posición original antes de regresar.

Las llamadas al sistema permitidas para administrar directorios:

1. **Crear**: Se crea un directorio vacío, con las entradas estándar “.” y “…” (que se añaden automáticamente).

2. **Eliminar**: Solo se puede eliminar un directorio vacío. Si solo contiene las entradas “.” y “…”, se considera vacío.
3. **Abrir directorio**: Permite leer los nombres de archivos en un directorio. Similar a abrir y leer un archivo.
4. **Cerrar directorio**: Después de leer un directorio, se cierra para liberar recursos.
5. **Leer entrada**: Devuelve la siguiente entrada en un directorio abierto. Evita tratar con la estructura interna del directorio.
6. **Cambiar nombre**: Los directorios se pueden renombrar como los archivos.
7. **Vinculación**: Permite que un archivo aparezca en varios directorios. Los vínculos duros incrementan el contador en el nodo-i del archivo.
8. **Eliminar entrada**: Elimina una entrada de directorio. Si el archivo está en varios directorios, solo se elimina la ruta especificada.

4.4 ADMINISTRACIÓN Y OPTIMIZACIÓN DE SISTEMAS DE ARCHIVOS

Además, los discos se pueden dividir en particiones con sistemas de archivos independientes, y el MBR contiene la tabla de particiones.

Asignación contigua: El esquema de asignación más simple es almacenar cada archivo como una serie contigua de bloques de disco. Cada archivo empieza al inicio de un nuevo bloque. Es simple de implementar, el rendimiento de lectura es excelente, pero tiene el gran problema que con el transcurso del tiempo los discos se fragmentan.

Asignación de lista enlazada (ligada): El segundo método para almacenar archivos es mantener cada uno como una lista enlazada de bloques de disco. La primera palabra de cada bloque se utiliza como apuntador al siguiente. El resto del bloque es para los datos. este método se puede utilizar cada bloque del disco. No se pierde espacio debido a la fragmentación del disco, pero tiene la desventaja de

que la cantidad de almacenamiento de datos en un bloque ya no es una potencia de dos, debido a que el apuntador ocupa unos cuantos bytes.

Asignación de lista enlazada utilizando una tabla en memoria: Consiste en tomar la palabra del apuntador de cada bloque de disco y la colocamos en una tabla en memoria. Para finalizar la cadena se ciloca un marcador espesalr distinto de un bloque valido. Dicha tabla en memoria principal se conoce como FAT. Utilizando esta organización, el bloque completo está disponiblepara los datos, pero tiene la desventaja de que toda la tabla debe estar en memoria todo el tiempo para que funcione.

Nodos-i: Consiste en asociar con cada archivo una estructura de datos conocida como nodo-i (nodo-índice), la cual lista los atributos y las direcciones de disco de los bloques del archivo. La gran ventaja de este esquema es que el nodo-i necesita estar en memoria sólo cuando está abierto el archivo correspondiente.

La función principal del sistema de directorios es asociar el nombre ASCII del archivo a la información necesaria para localizar los datos. Los atributos son almacenados directamente en la entrada de directorio.

Un directorio consiste en una **lista de entradas de tamaño fijo**, una por archivo, que contienen un nombre de archivo (de longitud fija), una estructura de los atributos del archivo y una o más direcciones de disco (hasta cierto máximo) que indique en dónde se encuentran los bloques de disco.

Para los sistemas que utilizan nodos-i, existe otra posibilidad para almacenar los atributos en los nodos-i, en vez de hacerlo en las entradas de directorio. En ese caso, la entrada de directorio puede ser más corta: sólo un nombre de archivo y un número de nodo-i.

A. Este enfoque asigna un espacio fijo a cada entrada de directorio. Sin embargo, el inconveniente es que al borrar un archivo, se crea un espacio vacío que puede no ajustarse al tamaño del próximo archivo a añadir.

B. En este sistema, todas las entradas de directorio tienen un tamaño constante y los nombres de los archivos se almacenan juntos en una zona

común al final del directorio. La ventaja es que, al eliminar una entrada, siempre habrá espacio suficiente para el archivo que se agregue después. Cuando un archivo se usa simultáneamente en varios directorios de diferentes usuarios, la relación entre el directorio y el archivo se denomina vínculo.

4.5 EJEMPLOS DE SISTEMAS DE ARCHIVOS

A. **LFS (Log-Structured File System)**: Es un sistema de archivos que organiza los datos en un registro secuencial. Regularmente, o cuando es necesario, las operaciones de escritura que están en espera en la memoria se agrupan y se escriben juntas en el disco en una secuencia continua al final del registro. Los nodos-i son parte de este sistema y se distribuyen a lo largo del registro, localizables mediante un mapa de nodos-i. LFS emplea un proceso de limpieza que recorre el registro para optimizar el espacio.

B. **Sistema de Archivos con Bitácora**: Este sistema registra anticipadamente las acciones que realizará, de modo que si hay una falla antes de completarlas, al reiniciar, el sistema puede consultar el registro para finalizar las tareas pendientes. Las operaciones y estructuras de datos en este sistema están diseñadas para ser idempotentes, es decir, pueden ejecutarse varias veces sin efectos adicionales.

C. **Sistema de Archivos Virtuales (VFS)**: Existen múltiples sistemas de archivos que pueden coexistir en una misma máquina, incluso bajo el mismo sistema operativo. Para el usuario, se presenta como una única jerarquía de archivos, ocultando la complejidad de los distintos sistemas subyacentes. El VFS se estructura en dos niveles: uno superior que interactúa con los procesos de usuario y otro inferior que se comunica con los sistemas de archivos específicos.

Hay dos estrategias generales posibles para almacenar un archivo de n bytes: se asignan n bytes consecutivos de espacio en disco o el archivo se divide en varios bloques (no necesariamente) contiguos. Casi todos los sistemas de archivos dividen los archivos en bloques de tamaño fijo que no necesitan ser adyacentes.

Para calcular la eficiencia del espacio, necesitamos hacer una **suposición acerca del tamaño de archivo promedio**, ya que, si la unidad de asignación es demasiado grande, desperdiciamos espacio; si es demasiado pequeña, desperdiciamos tiempo. Mayormente se utiliza un rango de 1 KB a 4 KB.

Hay dos métodos para llevar registro de los bloques libres:

El primero consiste en utilizar una **lista enlazada de bloques de disco**, donde cada bloque contiene tantos números de bloques de disco libres como pueda.

La otra técnica de administración del espacio libre es **el mapa de bits**. Un disco con n bloques requiere un mapa de bits con n bits. Los bloques libres se representan mediante 1s en el mapa, los bloques asignados mediante 0s (o viceversa).

Un método alternativo que evita la mayor parte de estas operaciones de E/S de disco es **dividir el bloque completo de apuntadores**. Ahora el sistema puede manejar una serie de archivos temporales sin realizar ninguna operación de E/S. Si el bloque en memoria se llena, se escribe en el disco y se lee el bloque medio lleno del disco. La idea aquí es mantener la mayor parte de los bloques de apuntadores en el disco llenos (para minimizar el uso del disco), pero mantener el que está en memoria lleno a la mitad, para que pueda manejar tanto la creación como la remoción de archivos sin necesidad de operaciones de E/S de disco en la lista de bloques libres.

Para evitar que los usuarios ocupen demasiado espacio en disco, los sistemas operativos multiusuario proporcionan un mecanismo para **imponer las cuotas de disco**. Existen límites duro y suave. Se puede exceder el límite suave, pero el límite duro no. Si el usuario excede el límite suave, se muestra una advertencia y se reduce la cuenta en uno. Si llega a 0, no se le permite iniciar sesión.

Un **vaciado físico** empieza en el bloque 0 del disco, escribe todos los bloques del disco en la cinta de salida en orden y se detiene cuando acaba de copiar el último (copia tanto bloque de datos, defectuosos y vacíos).

Un **vaciado lógico** empieza en uno o más directorios especificados y vacía en forma recursiva todos los archivos y directorios que se encuentran ahí que hayan sido modificados desde cierta fecha base dada.

Todos los verificadores de sistema de archivos comprueban cada sistema de archivos (**partición de disco**) de manera independiente de los demás. Se pueden realizar dos tipos de verificaciones de consistencia: archivos y bloques.

La verificación de bloques consiste en dos contadores: en la primera tabla llevan el registro de

cuántas veces está presente cada bloque en un archivo; los contadores en la segunda tabla registran con qué frecuencia está presente cada bloque en la lista de bloques libres.

La verificación de archivos también utiliza una tabla de contadores. Para cada nodo-i en cada directorio, incrementa un contador para la cuenta de uso de ese archivo.

Cuando el verificador termina, tiene una **lista indexada por número de nodo-**i, que indica cuántos directorios contienen cada archivo. Después el programa lee todos los nodos-i utilizando un dispositivo puro, y aquí pueden ocurrir dos tipos de errores: que la cuenta de vínculos en el nodo-i sea demasiado alta o demasiado baja.

La solución es tan sólo obligar a que la cuenta de vínculos en el nodo-i sea igual al número actual de entradas del directorio.

La técnica más común utilizada para reducir los accesos al disco es **la caché de bloques o caché de búfer**. En este contexto, una caché es una colección de bloques que pertenecen lógicamente al disco, pero se mantienen en memoria por cuestiones de rendimiento.

Un algoritmo común es verificar si el bloque se encuentra en cache, ya que de ser así se evita el acceso al disco (más lento).

Una segunda técnica para mejorar el rendimiento percibido por el sistema de archivos es tratar de **colocar bloques en la caché antes de que se necesiten**,

para incrementar la proporción de aciertos (esta estrategia de lectura adelantada sólo funciona para los archivos que se leen en forma secuencial)

Otra técnica importante es reducir la **cantidad de movimiento del brazo del disco**, al colocar los bloques que tengan una buena probabilidad de utilizarse en secuencia cerca unos de otros, de preferencia en el mismo cilindro.

Una mejora fácil en el rendimiento es colocar los nodos-i en medio del disco, en vez de hacerlo al principio, con lo cual se reduce la búsqueda promedio entre el nodo-i y el primer bloque por un factor de dos.

Sistemas de archivos:

CD-Roms: Diseñados para medios en los que solo se pude escribir una vez. Estándar Internacional **ISO 9660**. Los CD-ROMs consisten en una sola espiral continua que contiene los bits en una secuencia lineal (aunque son posibles las búsquedas a través de la espiral). Los bits a lo largo de la espiral se dividen en **bloques lógicos de 2352 bytes.**

Extensiones Rock Rigde: UNIX empezaron a trabajar en una extensión para que fuera posible representar los sistemas de archivos de UNIX en un CD-ROM. Estas extensiones se llamaron **Rock Ridge**.

Extensiones Joliet: Microsoft inventó ciertas extensiones, a las cuales llamó Joliet. Estas extensiones estaban diseñadas para permitir que los sistemas de archivos de Windows se copiaran a CD-ROM y después se restauraran, precisamente en la misma forma que se diseñó Rock Ridge para UNIX.

5. ENTRADA Y SALIDA

Los **dispositivos de E/S** se pueden dividir básicamente en dos categorías: dispositivos de bloque y dispositivos de carácter. Un dispositivo de bloque almacena información en bloques de tamaño fijo, cada uno con su propia dirección. Todas las transferencias se realizan en unidades de uno o más bloques completos (consecutivos). es posible leer o escribir cada bloque de manera independiente de los demás.

Un **dispositivo de carácter** envía o acepta un flujo de caracteres, sin importar la estructura del bloque. **No es direccionable** y no tiene ninguna operación de búsqueda.

Disco = **Preámbulo** (formato al disco y contiene el cilindro y número de sector, el tamaño del sector y datos similar) + Datos (Bits) + ECC (Código de Corrección de Errores).

Cada controlador tiene unos cuantos registros que se utilizan para comunicarse con la CPU. Muchos dispositivos tienen un búfer de datos que el sistema operativo puede leer y escribir.

En el primer método, a cada registro de control se le asigna un número de puerto de E/S. El conjunto de todos los puertos de E/S forma el espacio de puertos de E/S.

El segundo método es **asignar todos los registros de control al espacio de memoria**. A cada registro de control se le asigna una dirección de memoria única a la cual no hay memoria asignada. Este sistema se conoce como E/S con asignación de memoria.

Ventajas:

- Direccionales en C
- No requieren mecanismo de protección,
- Reduce el tamaño del kernel.

Desventajas:

- Se suelen colocar las palabras de la memoria en la cache (sería erróneo colocar ahí registros de control de dispositivos)
- Todos los dispositivos deben examinar toda la memoria.

En todos los casos, cuando la CPU desea leer una palabra, ya sea de la memoria o de un puerto de E/S, coloca la dirección que necesita en las líneas de dirección del bus y después impone una señal

READ en una línea de control del bus. **DMA** (Acceso Directo a Memoria): El controlador de DMA tiene acceso al bus del sistema de manera independiente de la CPU. Contiene varios registros en los que la CPU puede escribir y leer; éstos incluyen un registro de dirección de memoria, un registro contador de bytes y uno o más registros de control. Los registros de control especifican el puerto de E/S a utilizar, la dirección de la transferencia, la unidad de transferencia, y el número de bytes a transferir en una ráfaga.

El **DMA** interrumpe al sistema cuando la cuenta de bytes llega a 0. Mientras tanto repite en ciclo los pasos 2 a 4.

Muchos buses pueden operar en dos modos: El controlador de DMA solicita la transferencia de una palabra y la obtiene; si la CPU también desea el bus, tiene que esperar. El mecanismo se llama robo de ciclo. El controlador de DMA indica al dispositivo que debe adquirir el bus, emitir una serie de transferencias y después liberar el bus. Esta forma de operación se conoce como modo de ráfaga.

La mayoría de los controladores de DMA **utilizan direcciones físicas de memoria** para sus transferencias.

Un esquema alternativo es escribir direcciones virtuales en el controlador DMA y utilizar así la **MMU**.

Búfer interno: Hay dos razones: en primer lugar, al utilizar el búfer interno, el controlador de disco puede verificar la suma de comprobación antes de iniciar una transferencia y si la suma de

comprobación es incorrecta se señala un error y no se realiza la transferencia; la segunda es que, una vez que se inicia una transferencia de disco, los bits siguen llegando a una velocidad constante, esté listo o no el controlador para ellos.

Interrupciones de E/S: Cuando un dispositivo de E/S ha terminado el trabajo que se le asignó, produce una interrupción. Para ello, impone una señal en una línea de bus que se le haya asignado.

Esta señal es detectada por **el chip controlador** de interrupciones en la tarjeta principal, que después decide lo que debe hacer.

Para manejar la **interrupción**, el controlador coloca un número en las líneas de dirección que especifican cuál dispositivo desea atención.

El número en las líneas de dirección se utiliza como índice en una tabla llamada vector de interrupciones. Esta tabla puede estar de manera estática en la maquina o en la memoria.

El hardware siempre guarda cierta información antes de iniciar el procedimiento de servicio. La mayoría de las CPUs guardan la información en la pila, aunque también se pude colocar en los registros internos (ambos tienen problemas).

Una interrupción que deja al equipo en un estado bien definido se conoce como **interrupción precisa**. Dicha interrupción tiene cuatro propiedades:

1. El contador del programa (PC) se guarda en un lugar conocido.

2. Todas las instrucciones antes de la instrucción a la que apunta el PC se han ejecutado por completo.

3. Ninguna instrucción más allá de la instrucción a la que apunta el PC se ha ejecutado.

4. Se conoce el estado de ejecución de la instrucción a la que apunta el PC.

Una interrupción que no cumple con estos requerimientos se conoce como interrupción imprecisa,

en donde hay distintas instrucciones cerca del contador del programa en distintos estados de avance, y las más antiguas no necesariamente están más completas que las más recientes. Las máquinas con **interrupciones imprecisas** por lo general vuelcan una gran cantidad de estado interno en la pila, para dar al sistema operativo la posibilidad de averiguar qué está pasando

Un concepto clave en el diseño del software de E/S se conoce **como independencia de dispositivos**, significa es que debe ser posible escribir programas que puedan acceder a cualquier dispositivo de E/S sin tener que especificar el dispositivo por adelantado. Para ello debe tener en cuenta

Denominación uniforme: El nombre de un archivo o dispositivo simplemente debe ser una cadena o un entero sin depender del dispositivo de ninguna forma

Transferencias Sincronicas (de bloqueo) y **Asicronicas** (la CPU inicia la transferencia y se va a hacer algo más hasta que llega la interrupción)

El uso de búfer: involucra una cantidad considerable de copiado

Dispositivos compartidos (discos) y dedicados (**USB**)

E/S Programada: La forma más simple de E/S es cuando la CPU hace todo el trabajo. A este método se le conoce como E/S programada. Primero se copian los datos en el kernel. Después el sistema operativo entra a un ciclo estrecho, imprimiendo los caracteres uno a la vez. El aspecto esencial de la E/S programada es que después de imprimir un carácter, la CPU sondea en forma continua el dispositivo para ver si está listo para aceptar otro. Este comportamiento se conoce comúnmente como sondeo u ocupado en espera. Tiene la desventaja de ocupar la CPU tiempo completo hasta que se completen todas las operaciones de E/S.

E/S controlada por interrupciones: La forma de permitir que la CPU haga algo más mientras espera a que la impresora esté lista es utilizar interrupciones. Cuando se realiza la llamada al sistema para imprimir la cadena, el búfer se copia en espacio de kernel (como vimos antes) y el

primer carácter se copia a la impresora, tan pronto como esté dispuesta para aceptar un carácter. En ese momento, la CPU llama al planificador y se ejecuta algún otro proceso. El proceso que pidió imprimir la cadena se bloquea hasta que se haya impreso toda la cadena.

Cuando la impresora ha impreso el carácter, y está preparada para aceptar el siguiente, genera una interrupción.

E/S mediante DMA: Aquí la idea es permitir que el controlador de DMA alimente los caracteres a la impresora uno a la vez, sin que la CPU se moleste (reducir el número de interrupciones).

Capas del software de E/S:

Manejo de Interrupciones:

1. Guardar los registros (incluyendo el **PSW**) que no han sido guardados por el hardware de la interrupción.

2. Establecer un contexto para el procedimiento de servicio de interrupciones. Para ello tal vez sea necesario establecer el **TLB**, la **MMU** y una tabla de páginas.

3. Establecer una pila para el procedimiento de servicio de interrupciones.

4. Reconocer el **controlador de interrupciones**. Si no hay un controlador de interrupciones centralizado, rehabilitar las interrupciones.

5. Copiar los registros desde donde se guardaron (posiblemente en alguna pila) a la tabla de procesos.

6. Ejecutar el procedimiento de servicio de interrupciones. Éste extraerá información de los registros del controlador de dispositivos que provocó la interrupción.

7. Elegir cuál proceso ejecutar a continuación. Si la interrupción ha ocasionado que cierto proceso de alta prioridad que estaba bloqueado cambie al estado listo, puede elegirse para ejecutarlo en ese momento.

8. Establecer el contexto de la MMU para el proceso que se va a ejecutar a continuación.

También puede ser necesario establecer un TLB.

9. Cargar los registros del nuevo proceso, incluyendo su PSW.

10. Empezar a ejecutar el nuevo proceso.

Drivers: Cada dispositivo de E/S conectado a una computadora necesita cierto código específico para controlarlo. Este código, conocido como driver, es escrito por el fabricante del dispositivo yse incluye junto con el mismo. Por lo general forman parte del kernel

Las principales tareas son: aceptar peticiones abstractas de lectura y escritura del software independiente del dispositivo, e inicializar el dispositivo (si es necesario), administrar sus propios requerimientos y eventos del registro.

Un controlador tiene que ser reentrante, lo cual significa que un controlador en ejecución tiene que esperar a ser llamado una segunda vez antes de que se haya completado la primera llamada.

Softwate E/S independiente:

1. **Interfaz Uniforme**:
    - Los controladores tienen interfaces similares para facilitar la conexión de nuevos dispositivos.
    - El nodo-i único para archivos especiales contiene el número mayor de dispositivo (para localizar el controlador) y el número menor de dispositivo (para especificar la unidad).
2. **Uso de Búferes**:
    - **Doble búfer**: Dos búferes alternan entre copiarse al espacio de usuario y acumular nuevos datos de entrada.
    - **Búfer circular**: Con una región de memoria y dos apuntadores, se almacenan y eliminan datos de manera eficiente.
    - El uso excesivo de búferes puede afectar el rendimiento.

3. **Reporte de Errores**:
    - Los errores de programación generan códigos de error al hacer llamadas imposibles (como escribir en un dispositivo de entrada).
    - Los errores de E/S reales incluyen situaciones como escribir en un bloque de disco dañado o leer de una cámara apagada.
4. **Asignación y Liberación de Dispositivos**:
    - Algunos dispositivos solo pueden ser usados por un proceso a la vez.
    - El sistema operativo debe evaluar las solicitudes de uso y aceptarlas o rechazarlas.
5. **Tamaño de Bloque Independiente del Dispositivo**:
    - El software debe ocultar las diferencias en los tamaños de sectores de distintos discos.
    - Proporcionar un tamaño de bloque uniforme a niveles superiores.

Software de E/S en espacio de usuario: Las llamadas al sistema, incluyendo las llamadas al sistema de E/S, se realizan comúnmente mediante procedimientos de biblioteca.

Otra categoría importante es el sistema de colas. El uso de colas (spooling) es una manera de lidiar con los dispositivos de E/S dedicados en un sistema de multiprogramación. Además, se utiliza un proceso especial, conocido como demonio, y un directorio especial llamado directorio de cola de impresión. Para imprimir un archivo, un proceso genera primero todo el archivo que va a imprimir y lo coloca en el directorio de la cola de impresión. Es responsabilidad del demonio, que es el único proceso que tiene permiso para usar el archivo especial de la impresora, imprimir los archivos en el directorio.

Discos magnéticos: Los discos magnéticos se organizan en cilindros, cada uno de los cuales

contiene tantas pistas como cabezas apiladas en forma vertical. Las pistas se dividen en sectores.

En otros discos, en especial los discos IDE (Electrónica de Unidad Integrada) y SATA (ATA Serial), la unidad de disco contiene un microcontrolador que realiza un trabajo considerable y permite al controlador real emitir un conjunto de comandos de nivel superior. A menudo el controlador coloca las pistas en caché, reasigna los bloques defectuosos y mucho más. Muchos controladores también pueden leer o escribir en una unidad mientras buscan en otra u otras unidades. Sin embargo, sólo es posible una transferencia entre el controlador y la memoria principal.

La mayoría de los discos modernos tienen una geometría virtual que se presenta al sistema operativo. Se instruye al software para que actúe como si hubiera x cilindros, y cabezas y z sectores por pista, a lo cual se conoce como direccionamiento de bloques lógicos en el que los sectores de disco sólo se enumeran en forma consecutiva empezando en 0, sin importar la geometría del disco.

RAID: Es una nueva clase de dispositivo basado en 6 organizaciones de disco. La idea básica detrás de un RAID es instalar una caja llena de discos a un lado de la computadora (que por lo general es un servidor grande), reemplazar la tarjeta controladora de discos con un controlador RAID, copiar los datos al RAID y después continuar la operación normal.

1. **RAID 0**:
    - Distribuye datos en bandas de sectores en múltiples unidades (striping).
    - Permite E/S en paralelo para peticiones grandes.
2. **RAID 1**:
    - Duplica todos los discos para mayor seguridad.
    - Rendimiento de lectura puede ser hasta el doble.

3. **RAID 2**:
    - Divide datos en palabras y asigna códigos Hamming a cada unidad.
    - Requiere sincronización rotacional y es complejo.
4. **RAID 3**:
    - Simplificación del nivel 2 de RAID.
    - Requiere sincronización exacta entre unidades.
5. **RAID 4**:
    - Protege contra pérdida de una unidad.
    - Paridad se almacena en una unidad adicional.
6. **RAID 5**:
    - Distribuye bits de paridad uniformemente en todas las unidades.
    - No utiliza una unidad separada para la paridad.

Tipos de CDs: Los CDs son dispositivos ópticos que tienen densidades de grabación mucho más altas que los discos magnéticos convencionales.

Estándar Internacional oficial (IS 10149): Los CDs de distintas compañías disqueras y los reproductores de distintos fabricantes electrónicos puedan funcionar en conjunto. Todos los CDs tienen 120 mm de diámetro y 1.2 mm de grosor, con un hoyo de 15 mm en medio.

Las depresiones en el sustrato de policarbonato se llaman hoyos (pits); las áreas no quemadas entre los hoyos se llaman áreas lisas (lands). Una transición de hoyo a área lisa o de área lisa a un hoyo para un 1 y su ausencia como un 0, por lo que se utiliza este esquema. Los hoyos y las áreas lisas se escriben en una sola espiral continua, que empieza cerca del hoyo y recorre una distancia de 32 mm hacia el borde.

La velocidad de rotación del CD se debe reducir en forma continua a medida que la cabeza de lectura se desplaza desde la parte interna del CD hacia la parte externa.

CD-ROMs: tenían que ser del mismo tamaño físico que los CDs de audio, ser compatibles en sentido mecánico y óptico con ellos, y se debían producir utilizando las mismas máquinas de moldeado por inyección de policarbonato.

El formato básico de un CD-ROM consiste en codificar cada byte en un símbolo de 14 bits, que es suficiente como para usar el código de Hamming en un byte de 8 bits con 2 bits de sobra.

Libro amarillo (estándar): Agrupamiento de 98 estructuras en un sector de CD-ROM en dos modos: El modo 1 utiliza un preámbulo de 16 bytes, 2048 bytes de datos y un código de corrección de errores de 288 bytes. El modo 2 combina los campos de datos y ECC en un campo de datos de 2336 bytes para aquellas aplicaciones que no necesitan la corrección de errores, como el audio y el video.

Los errores de un solo bit se corrigen en el nivel más bajo, los errores de ráfagas cortas se corrigen en el nivel de estructura y los errores residuales se atrapan en el nivel de sectores.

El sistema de archivos es conocido como High Sierra. Tiene tres niveles. El nivel 1 utiliza nombres de archivo de hasta 8 caracteres, seguidos opcionalmente de una extensión de hasta 3 caracteres, y requiere que todos los archivos sean contiguos. El nivel 2 permite nombres de hasta32 caracteres, y el nivel 3 permite archivos no contiguos.

CD-Rs: Son CS-ROMs que permiten se grabados. Los CD-Rs tienen una apariencia similar a los CD-ROMs, excepto porque tienen una parte superior dorada, en lugar de una plateada. Para ello hay que agregar una capa de colorante entre el policarbonato y la capa de oro reflectiva.

Para escribir, el láser del CD-R se pone en alto poder. Cuando el haz golpea en un punto del colorante, se calienta y quebranta un lazo químico, el cual crea un punto oscuro, que se interpreta como la diferencia entre los hoyos y las áreas lisas.

Los CD-ROM XA, que permite escribir CD-Rs en forma incremental; unos cuantos sectores hoy, otros pocos mañana, y unos cuantos el siguiente mes.

Las pistas se pueden agrupar en sesiones, lo cual conlleva a los CD-ROMs de multisesión. Para evitar la piratería, uno de los métodos implica grabar todas las longitudes de los archivos en el CD ROM como de varios gigabytes, para frustrar cualquier intento de copiar los archivos en disco duro, y cifrar las verdaderas longitudes.

CD-RW: Son CD-ROMs que permiten re-grabarse. Utiliza medios del mismo tamaño que el CD R. Sin embargo, en vez de colorante, el CD-RW utiliza una aleación de plata, indio, antimonio y telurio para la capa de grabación. Esta aleación tiene dos estados estables: cristalino y amorfo, con distintas reflectividades, las cuales dependiendo la energía de laser se puede fundir para permite grabar, re-grabar o leer (son más costosos).

DVD: utilizan el mismo diseño general que los CDs, con discos de policarbonato moldeado por inyección de 120 mm que contienen hoyos y áreas lisas, que se iluminan mediante un diodo láser y se leen mediante un fotodetector. Lo nuevo es el uso de hoyos más pequeños, una espiral más estrecha y un láser rojo. Esto mejora la capacidad total siete veces, asi como también la velocidad de transferencia.

Existen cuatro formatos:

1. Un solo lado, una sola capa (4.7 GB).

2. Un solo lado, doble capa (8.5 GB).

3. Doble lado, una sola capa (9.4 GB).

4. Doble lado, doble capa (17 GB).

Blu-ray: Utiliza un láser de 0.405 micrones (azul) para empaquetar 25 GB en un disco de una

sola capa y 50 GB en un disco de doble capa. El otro es HD DVD, que utiliza el mismo láser azul, pero tiene una capacidad de sólo 15 GB (una sola capa) y 30 GB (doble capa).

Formato de disco: Antes de poder utilizar el disco, cada plato debe recibir un formato de bajo nivel mediante software. El formato consiste en una serie de pistas concéntricas, cada una de las cuales contiene cierto número de sectores.

El preámbulo empieza con cierto patrón de bits que permite al hardware reconocer el inicio del sector

.El campo ECC contiene información redundante que se puede utilizar para recuperarse de los errores de lectura.

La posición del sector 0 en cada pista está desfasada de la pista anterior cuando se aplica el formato de bajo nivel. Este desplazamiento, conocido como desajuste de cilindros, se realiza para mejorar el rendimiento. La idea es permitir que el disco lea varias pistas en una operación continua sin perder datos.

A menudo la capacidad con formato es 20% menor que la capacidad sin formato (capacidad física != real)

Algoritmos de programación del brazo del disco: Debe cumplir con tres metas:

1. Tiempo de búsqueda (el tiempo para desplazar el brazo al cilindro apropiado).

2. Retraso rotacional (el tiempo para que el sector apropiado se coloque debajo de la cabeza).

3. Tiempo de transferencia de datos actual.

FCFS: Si el software controlador de disco acepta peticiones una a la vez, y las lleva a cabo en

ese orden, es decir, Primero en llegar, primero en ser atendido

1. **SSF (Shortest-Seek-First)**:

- Maneja la petición más cercana primero para reducir el tiempo de búsqueda.
- Recorta el movimiento total del brazo en comparación con FCFS.
- Sin embargo, en discos con mucha carga, el brazo tiende a quedarse en la parte media, lo que puede retrasar las peticiones en los extremos.

2. **SCAN (Algoritmo del Elevador)**:
    - Requiere un bit de dirección (ARRIBA o ABAJO).
    - Después de una petición, el brazo se mueve a la siguiente posición de mayor o menor prioridad según el bit.
    - La caché del controlador de disco es independiente de la del sistema operativo.

Es importante tener en cuenta que todos los algoritmos de planificación de disco anteriores asumen de manera tácita que la geometría de disco real es igual que la geometría virtual.

Manejo de errores: Los defectos de fabricación incluyen sectores defectuosos; es decir, sectores que no leen correctamente el valor que se acaba de escribir en ellos.

Hay dos métodos generales para los bloques defectuosos: lidiar con ellos en el controlador o lidiar con ellos en el sistema operativo.

En el primer método, antes de que el disco salga de la fábrica, se prueba y se escribe una lista de sectores defectuosos en el disco. Cada sector defectuoso se sustituye por uno de los sectores adicionales.

Dos formas de realizar la sustitución: El controlador puede hacer es reasignar uno de los sectores adicionales, o puede desplazar todos los sectores una posición hacia arriba.

El segundo método consiste en que el sistema operativo lo haga desde el software, para lo cual

debe adquirir una lista de sectores defectuosos. Una vez que sepa cuáles sectores están defectuosos, puede construir tablas de reasignación.

Si el disco se respalda archivo por archivo, es importante que la herramienta de respaldo no trate de copiar el archivo con el bloque defectuoso. Para evitar esto, el sistema operativo tiene que ocultar el archivo con el bloque defectuoso tan bien que ni siquiera una herramienta de respaldo pueda encontrarlo

También ocurren errores de búsqueda producidos por problemas mecánicos en el brazo, los cuales se corrigen de forma automática por el controlador al recalibrar el brazo al cilindro 0.

Almacenamiento estable: Para algunas aplicaciones es esencial que los datos nunca se pierdan o corrompan. Lo que se puede lograr es un subsistema de disco que tenga la siguiente propiedad: cuando se emita una escritura, el disco debe escribir correctamente los datos o no hacer nada, dejando los datos existentes intactos. A dicho sistema se le conoce como almacenamiento estable y se implementa en software.

El almacenamiento estable utiliza un par de discos idénticos, en donde los bloques correspondientes funcionan en conjunto para formar un bloque libre de errores.

Una enorme mejora es llevar la cuenta de cuál bloque se estaba escribiendo durante una escritura estable, de manera que se tenga que comprobar sólo un bloque durante la recuperación. Esto se hace mediante una RAM no volátil la cual es una CMOS especial.

Relojes: Los relojes (también conocidos como temporizadores) son esenciales para la operación de cualquier sistema de multiprogramación

Hay dos tipos de relojes: Los relojes más simples están enlazados a la línea de energía de 110 o 220 voltios y producen una interrupción en cada ciclo de voltaje, a 50 o 60 Hz.

Otro tipo de reloj se construye a partir de tres componentes: un oscilador de cristal, un contador y un registro contenedor. Cuando una pieza de cristal de cuarzo se

corta en forma apropiada y se monta bajo tensión, puede generar una señal periódica con una precisión muy grande, de varios cientos de megahertz.

Esta señal se alimenta al contador para hacer que cuente en forma descendente hasta cero. Cuando el contador llega a cero, produce una interrupción de la CPU.

Modos: En el modo de un solo disparo, cuando se inicia el reloj copia el valor del registro contenedor en el contador y después decrementa. Cuando el contador llega a cero, produce una interrupción y se detiene hasta que vuelve a ser iniciado en forma explícita mediante el software

En el modo de onda cuadrada, después de llegar a cero y producir la interrupción, el registro contenedor se copia automáticamente en el contador y todo el proceso se repite de nuevo en forma indefinida. Estas interrupciones periódicas se conocen como pulsos de reloj.

Funciones del reloj:

1. **Función de Reloj - Hora del Día**:
    - La tarea principal es actualizar continuamente un contador con cada tic del reloj para mantener la hora actual.
2. **Limitación de Tiempo de Proceso**:
    - Al iniciar un proceso, el planificador establece un contador con un límite de tiempo (quantum) para prevenir que un proceso acapare el CPU.
3. **Conteo de Uso de CPU**:
    - Se utiliza un segundo contador, independiente del principal, para registrar el tiempo de CPU utilizado por un proceso. Este contador se pausa y reanuda con las interrupciones para mantener una cuenta precisa.
4. **Temporizadores Guardianes**:

- Son temporizadores especiales del sistema operativo que, al alcanzar cero, emiten una señal de reinicio para el sistema, actuando como un mecanismo de seguridad.

5. **Perfilamiento de Programas**:
    - Algunos sistemas permiten a los programas de usuario solicitar la creación de un histograma del contador de programa, lo que ayuda a entender en qué partes del código se invierte más tiempo.

En general hay dos formas de manejar las interrupciones de E/S y el sondeo. Las interrupciones tienen baja latencia; es decir, ocurren justo después del evento en sí, con muy poco o nada de retraso. Por otra parte, con las CPUs modernas las interrupciones tienen una sobrecarga considerable, debido a la necesidad del cambio de contexto y su influencia en la canalización, el TLB y la caché.

Los temporizadores de software evitan las interrupciones. En vez de ello, cada vez que el kernel se ejecuta por alguna otra razón, justo antes de regresar al modo de usuario comprueba el reloj de tiempo real para ver si ha expirado un temporizador de software. Si el temporizador ha expirado, se realiza el evento programado sin necesidad de cambiar al modo de kernel debido a que el sistema ya se encuentra ahí. Una vez realizado el trabajo, el temporizador de software se restablece para empezar de nuevo.

Software de entrada: La entrada del usuario proviene principalmente del teclado y del ratón. Se genera una interrupción cada vez que se oprime una tecla, y se genera una segunda interrupción cada vez que se suelta. En cada una de estas interrupciones de teclado, el software controlador del mismo extrae la información acerca de lo que ocurre desde el puerto de E/S asociado con el teclado.

Todo lo demás ocurre en el software y es muy independiente del hardware

Software de teclado: El número en el puerto de E/S es el número de tecla, conocido como código de exploración.

Se pueden adoptar dos filosofías posibles para el controlador. En la primera, el trabajo del controlador es sólo aceptar la entrada y pasarla hacia arriba sin modificarla. Un programa que lee del teclado obtiene una secuencia pura de códigos ASCII.

En la segunda filosofía el controlador maneja toda la edición entre líneas, y envía sólo las líneas corregidas a los programas de usuario. La primera filosofía está orientada a caracteres (modo no canónico); la segunda está orientada a líneas (modo canónico). Si el teclado está en modo canónico (cocido), los caracteres se deben almacenar hasta que se haya acumulado una línea completa, debido a que tal vez el usuario decida posteriormente borrar parte de ella. Para esto se puede utilizar un búfer dedicado o se pueden asignar búferes de una reserva.

Producir eco: Ver los caracteres que acaban de escribir aparecer en la pantalla. Para ello el controlador del teclado tiene que averiguar dónde colocar la nueva entrada sin que sea sobrescrita por la salida del programa.

Software de ratón: Dos tipos de ratones: Un tipo común de ratón tiene una bola de goma en su interior que se asoma por un hoyo en la parte inferior y gira, a medida que el ratón se desplaza por una superficie dura, frotándose contra unos rodillos posicionados en ejes ortogonales. El movimiento en la dirección este-oeste hace que gire el eje paralelo al eje y; el movimiento en la dirección norte sur hace que gire el eje paralelo al eje x.

Otro tipo popular de ratón es el óptico, que está equipado con uno o más diodos emisores de luz y fotodetectores en su parte inferior. Los ratones ópticos modernos tienen un chip de procesamiento de imágenes en ellos y sacan fotos continuas de baja resolución de la superficie debajo de ellos, buscando cambios de imagen en imagen.

El mensaje para la computadora contiene tres elementos: Δx, Δy, botones. La mayoría de los ratones se reportan con la computadora un máximo de 40 veces/seg.

Software de salida: En su mayor parte, el programa envía caracteres a la ventana en uso y se muestran ahí. Por lo general, un bloque de caracteres (por ejemplo, una línea) se escribe en una llamada al Sistema.

La mayoría de los controladores de software de salida proporcionan una serie de comandos para desplazar el cursor, insertar y eliminar caracteres o líneas en el cursor, entre otras tareas. A menudoestos comandos se conocen como secuencias de escape.

Sistema X Window: Es la interfaz de usuario utilizada por los sistemas UNIX. Es muy portátil y se ejecuta por completo en espacio de usuario.

Cuando X se ejecuta en un equipo, el software que recolecta la entrada del teclado y el ratón, y que escribe la salida en la pantalla, se llama servidor X. Este software tiene que llevar el registro de cuál ventana está seleccionada en un momento dado. Se comunica con los programas en ejecución (posiblemente a través de una red), llamados clientes X. Les envía la entrada del ratóny del teclado, y acepta los comandos de pantalla de ellos.

Administración de Energía: La invención del transistor, el uso de la energía disminuyó en forma considerable. Sin embargo, debido al poco desarrollo que hay en el mundo de las baterías (en comparación con la CPU) es que es importante administrar la batería de forma correcta, más que nada en las denominadas notebooks.

Hay dos métodos generales para reducir el consumo de energía. El primero es que el sistema operativo apague partes de la computadora (en su mayoría, dispositivos de E/S) que no estén en uso.

El segundo método es que el programa de aplicación utilice menos energía, lo que posiblemente degradaría la calidad de la experiencia del usuario, para poder alargar el tiempo de la batería.

El método general que la mayoría de los distribuidores de computadoras utilizan para conservar las

baterías es diseñar la CPU, la memoria y los dispositivos de E/S para que tenga múltiples estados: encendido, inactivo, hibernando y apagado.

Los principales tres consumidores de energía eran la pantalla, el disco duro y la CPU.

1. **Pantalla**:
    - Ahorro de energía al apagar la pantalla cuando está inactiva.
    - La regeneración desde la RAM de video es rápida al reactivarla.
2. **Disco Duro**:
    - Detener el disco después de inactividad para ahorrar energía.
    - Caché en RAM reduce la necesidad de reiniciar el disco.
3. **CPU**:
    - Modo inactivo reduce el consumo de energía.
    - Relación entre voltaje, ciclo de reloj y velocidad de CPU.
4. **Memoria**:
    - Opciones para ahorrar energía: vaciar y apagar la caché o hibernar la memoria principal.
5. **Conexiones Inalámbricas**:
    - Mensajes a la estación base antes de desconectar el radio.
    - Acumulación de mensajes en un búfer para enviar después.
6. **Administración Térmica**:
    - Uso de ventiladores para disipar calor.
    - Reducción de consumo de energía como alternativa.
7. **Administrador de Batería**:
    - Cambio a la siguiente batería sin interrupciones.

- Advertencia al usuario antes de apagar por completo.

8. **Programas y Ahorro de Energía**:
    - Indicar a los programas usar menos energía, incluso si afecta la experiencia del usuario.
    - Información pasada cuando la carga de la batería es baja.

Recursos que pueden ser utilizados por sólo un proceso a la vez. Cuando dos procesos escriben de manera simultánea en la impresora se producen incoherencias. En consecuencia, todos los sistemas operativos tienen la habilidad de otorgar (en forma temporal) a un proceso el acceso exclusivo a ciertos recursos.

Los interbloqueos también pueden ocurrir entre máquinas. Por ende, los interbloqueos pueden ocurrir en los recursos de hardware o de software.

RECURSOS Interbloqueos involucra a los recursos. Los interbloqueos pueden ocurrir cuando a los procesos se les otorga acceso exclusivo a los dispositivos, registros de datos, archivos, etcétera.

Un recurso puede ser un dispositivo de hardware, o una pieza de información. Computadora tendrá muchos recursos.

Recursos apropiativos y no apropiativos:

Los recursos son de dos tipos: apropiativos y no apropiativos. Un recurso apropiativo es uno que se puede quitar al proceso que lo posee sin efectos dañinos. La memoria es un ejemplo.

Un recurso no apropiativo es uno que no se puede quitar a su propietario actual sin hacer que el cómputo falle.

En general, los interbloqueos involucran a los recursos no apropiativos.

La secuencia de eventos requerida para utilizar un recurso: 1. Solicitar un recurso, 2. Utilizar el recurso, 3. Liberar el recurso

Si el recurso no está disponible cuando se le solicita, el proceso solicitante se ve obligado a espera. Un proceso al que se le ha negado la petición de un recurso por lo general permanece en un ciclo estrecho solicitando el recurso, sin poder realizar ningún trabajo útil.

Adquisición de recursos:

Una manera de permitir que los usuarios administren los recursos es asociar un semáforo con cada recurso.

Se pueden utilizar mutexes de igual forma.

Algunas veces los procesos necesitan dos o más recursos. Se pueden adquirir de manera secuencial: El proceso A adquiera el recurso 1 y el proceso B adquiera el recurso 2. Ahora cada uno se bloqueará al tratar de adquirir el otro. Ninguno de los procesos se volverá a ejecutar. Esa situación es un interbloqueo.

INTERBLOQUEOS

Un conjunto de procesos se encuentra en un interbloqueo si cada proceso en el conjunto está esperando un evento que sólo puede ser ocasionado por otro proceso en el conjunto.

Debido a que todos los procesos están en espera, ninguno de ellos producirá alguno de los eventos que podrían despertar a cualquiera de los otros miembros del conjunto, y todos los procesos seguirán esperando para siempre.

En otras palabras, cada miembro del conjunto de procesos en interbloqueo está esperando un recurso queposee un proceso en interbloqueo. A este tipo de interbloqueo se le conoce como interbloqueo de recursos.

Condiciones de interbloqueos:

Se deben cumplir cuatro condiciones para un interbloqueo (de recursos):

1. Condición de exclusión mutua. Cada recurso se asigna en un momento dado a sólo un proceso, o está disponible.

2. Condición de contención y espera. Los procesos que actualmente contienen recursos que se les otorgaron antes pueden solicitar nuevos recursos.

3. Condición no apropiativa. Los recursos otorgados previamente no se pueden quitar a un proceso por la fuerza. Deben ser liberados de manera explícita por el proceso que los contiene.

4. Condición de espera circular. Debe haber una cadena circular de dos o más procesos, cada uno de los cuales espera un recurso contenido por el siguiente miembro de la cadena.

Si una de ellas está ausente, no es posible el interbloqueo de recursos.

Modelado de interbloqueos:

Los interbloqueos se pueden moldear mediante gráficos dirigidos. Se utilizan dos tipos de nodos: procesos, que se muestran como círculos, y recursos, que se muestran como cuadro.

Un ciclo en el gráfico indica que hay un interbloqueo que involucra a los procesos y recursos en el ciclo.

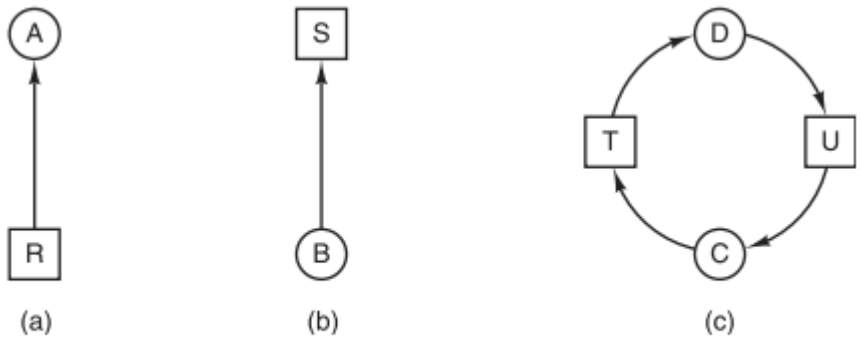

Figura 6-3. Gráficos de asignación de recursos. (a) Contención de un recurso. (b) Petición de un recurso. (c) Interbloqueo.

Por ahora, el punto a comprender es que los gráficos de recursos son una herramienta que nos permite ver si

una secuencia de petición/liberación dada conduce al interbloqueo.

En general, se utilizan cuatro estrategias para lidiar con los interbloqueos:

1. Sólo ignorar el problema. Tal vez si usted lo ignora, él lo ignorará a usted.

2. Detección y recuperación. Dejar que ocurran los interbloqueos, detectarlos y tomar acción.

3. Evitarlos en forma dinámica mediante la asignación cuidadosa de los recursos.

4. Prevención, al evitar estructuralmente una de las cuatro condiciones requeridas.

1. ALGORITMO AVEZTRUZ: El método más simple es el algoritmo del avestruz: meta su cabeza en la arena y pretenda que no hay ningún problema. Esta estrategia consiste en ver con qué frecuencia falla el

sistema y qué tan grave es un interbloqueo. Si ocurren interbloqueos en promedio de una vez cada cinco años, los interbloqueos no representan un problema inmediato el cual se tenga que resolver.

2. DETENCION Y RECUPERACIÓN: El sistema no trata de evitar los interbloqueos. En vez de ello, intenta detectarlos cuando ocurran y luego realiza cierta acción para recuperarse después del hecho.

Detención de un interbloqueo: Si sólo existe un recurso de cada tipo, podemos construir un gráfico de recursos. Si este gráfico contiene uno o más ciclos, existe un interbloqueo. Cualquier proceso que forme parte de un ciclo está en interbloqueo. Si no existen ciclos, el sistema no está en interbloqueo.

Lo que hace este algoritmo es tomar cada nodo en turno, como la raíz de lo que espera sea un árbol, y realiza una búsqueda de un nivel de profundidad en él. Si alguna vez regresa a un nodo que ya se haya encontrado, entonces ha encontrado un ciclo. En cambio, si regresa hasta la raíz y no puede avanzar más, no contiene ciclos y, por ende, no hay interbloqueo.

En cambio, si existe varios recursos de cada tipo, se necesita un método distinto para detectar interbloqueo.

Ahora presentaremos un algoritmo basado en dos matrices y dos vectores:

E es el vector de recursos existentes. Este vector proporciona el número total de instancias de cada recurso en existencia.

A sea el vector de recursos disponibles. Este vector proporciona el número de instancias del recurso i que están disponibles en un momento dado (es decir, sin asignar).

C, la matriz de asignaciones actuales y R, la matriz de peticiones. La i-ésima fila de C nos indica cuántas instancias de cada clase de recurso contiene Pi. Cij es el número de instancias del recurso j que están contenidas por el proceso i. De manera similar, Rij es el número de instancias del recurso j que desea P

Recuperación de un interbloqueo: varias formas de recuperarse de un interbloqueo

Recuperación por medio de apropiación: En algunos casos puede ser posible quitar temporalmente un recurso a su propietario actual y otorgarlo a otro proceso. En muchos casos se puede requerir intervención manual.

Recuperación a través del retroceso: Si los diseñadores de sistemas y operadores de máquinas saben que es probable que haya interbloqueos, pueden hacer que los procesos realicen puntos de comprobación (checkpoint) en forma periódica. El punto de comprobación no sólo contiene la imagen de la memoria, sino también el estado del recurso. Los nuevos puntos de comprobación no deben sobrescribir a los anteriores.

Cuando se detecta un interbloqueo, es fácil ver cuáles recursos se necesitan. Para realizar la recuperación, un proceso que posee un recurso necesario se revierte a un punto en el tiempo antes de que haya adquirido ese recurso, para lo cual se inicia uno de sus puntos de comprobación anteriores.

Recuperación a través de la eliminación de procesos: La forma más cruda y simple de romper un interbloqueo es eliminar uno o más procesos. Si esto no ayuda, se puede repetir hasta que se rompa el ciclo.

De manera alternativa, se puede elegir víctima a un proceso que no esté en el ciclo, para poder liberar sus recursos.

Si bien es un algoritmo muy agresivo, es mejor eliminar un proceso que se pueda volver a ejecutar desde el principio sin efectos dañinos.

6.6 CÓMO PREVENIR INTERBLOQUEOS

El sistema debe ser capaz de decidir si es seguro otorgar un recurso o si no lo es, y realizar la asignación sólo cuando sea seguro. Podemos evitar los interbloqueos, pero sólo si hay cierta información disponible de antemano.

Trayectorias de los recursos: Los principales algoritmos para evitar interbloqueos se basan en el concepto de los estados seguros.

El eje horizontal representa el número de instrucciones ejecutadas por el proceso A. El eje vertical por el proceso B.

1. **Evitar la Exclusividad de Recursos**:
    - No asignar recursos exclusivamente a un proceso previene interbloqueos.
    - Ejemplo: Usar una cola de impresión permite que varios procesos soliciten impresión simultáneamente, con un único proceso gestionando la impresora física.
2. **Prevenir la Contención y Espera**:
    - Impedir que procesos con recursos esperen por más recursos elimina interbloqueos.
    - Requerir que los procesos soliciten todos sus recursos de antemano puede ser un desafío, ya que no siempre se conoce la cantidad necesaria y puede llevar a un uso ineficiente.
    - Una alternativa es que los procesos liberen todos sus recursos antes de solicitar nuevos.
3. **Manejo de Recursos No Apropiativos**:
    - Forzar la retirada de un recurso como una impresora en uso no es ideal.
    - La virtualización de recursos puede ser una solución viable.

4. **Eliminar la Espera Circular**:
    - Establecer una regla de un recurso por proceso a la vez.
    - Asignar un orden numérico global a los recursos y requerir que las solicitudes se hagan siguiendo ese orden para evitar ciclos en el gráfico de asignación de recursos.
    - Permitir que los procesos soliciten recursos sin seguir una secuencia estricta, pero no permitir solicitudes de recursos de menor prioridad a los que ya poseen.

Bloqueo de dos fases (Base de Datos): En la primera fase, el proceso trata de bloquear todos los registros que necesita, uno a la vez. Si tiene éxito pasa a la segunda fase, realizando sus actualizaciones y liberando los bloqueos.

Si durante la primera fase se necesita cierto registro que ya esté bloqueado, el proceso sólo libera todos sus bloqueos e inicia la primera fase desde el principio.

El algoritmo funciona sólo en aquellas situaciones en donde el programador ha ordenado las cosas con mucho cuidado.

Interbloqueos de comunicaciones: Otro tipo de interbloqueo puede ocurrir en los sistemas de comunicaciones (como las redes), en donde dos o más procesos se comunican mediante el envío de mensajes.

Ejemplo: El proceso A envía un mensaje de petición al proceso B. Suponga que el mensaje de petición se pierde. A se bloquea en espera de la respuesta. B se bloquea en espera de una petición. A esta situación se le conoce como interbloqueo de comunicación. No se pueden evitar mediante el ordenamiento de recursos ni mediante una programación cuidados.

Por fortuna hay otra técnica para romper los interbloqueos de comunicación: los tiempos de espera. Cada vez que se envía un mensaje del que se espera una respuesta, también se inicia un temporizador. Si el temporizador termina su conteo antes de que llegue la respuesta, el emisor del mensaje asume que éste se ha perdido y lo envía de nuevo.

El diseño de las reglas de comunicación para que todo funcione bien, que se conocen como protocolo.

Bloqueo activo: En la figura 6-16, si el proceso A se ejecuta primero y adquiere el recurso 1, y después se ejecuta el proceso 2 y adquiere el recurso 2, sin importar quién se ejecute a continuación, no progresará más, pero ninguno de los procesos se bloqueará. Esto se conoce como bloqueo activo.

El bloqueo activo puede ocurrir en formas sorprendente. Si una operación fork falla debido a que la tabla está llena, un método razonable para el programa que realiza la operación fork es esperar un tiempo aleatorio y probar de nuevo. El número máximo de archivos abiertos está restringido de manera similar por el tamaño de la tabla de nodos-i, por lo que ocurre un problema similar cuando se llena.

La mayoría de los sistemas operativo sólo ignoran el problema con base en la suposición de que la mayoría.

Conclusión:

Para concluir esta actividad he de decir que se me hizo algo sin mas una practica bastante tardada, que por el tiempo al menos a mi me costó bastante hacerla, también siguiendo todo lo aprendido en la materia siento que es bastante pesado realizar esta última actividad después de todo lo que hicimos.

Agregando también que me hubiera gustado más la guía que ofrecía este libro en algunos momentos en los que se necesitaba, pero dejando de lado eso si funciona leerlo y comprenderlo aunque bastante tarde para lo que queda de semestre.

En resumen para mi fue una actividad de recuperación 10 firmas es algo excesivo teniendo en cuenta todas las actividades que hicimos con valor de 1 firma para que este valga 10 firmas es excesivo por el tiempo y la cantidad pero bueno.