PROGRAMA EDUCATIVO:
INGENIERÍA EN COMPUTACIÓN

UNIDAD DE APRENDIZAJE:
SISTEMAS OPERATIVOS

ÍNDICE

UNIDAD I:

COMPONENTES BÁSICOS Y ESTRUCTURA DE LOS SISTEMAS OPERATIVOS

Concepto de un sistema operativo

Es el conjunto de programas que nos permiten utilizar la computadora y que nos sirven para:

- Interfaz con la computadora:
    - Desarrollo de programas
    - Ejecución de programas
    - Acceso a dispositivos de E/S
    - Acceso al sistemas de archivos
    - Protección y seguridad
    - Detección y respuesta de errores
    - Contabilidad
- Gestor de recursos

Componentes de un sistema operativo

1. Núcleo o Kernel: Ejecuta tareas, trabaja con los procesos y las IRQ´s.
2. Interprete de comandos: Interpreta instrucciones.
3. Sistema de archivos

Historia de los sistemas operativos

- Primera generación (1945-55)
- Segunda generación (1955-65)
- Tercera generación (1965-80)
- Cuarta generación (1980-hoy)

Primera generación (1945-55)

- Utilidad: máquinas de cálculo.
- Tecnología: Tubos de vacío y paneles.
- Métodos de programación: cables, interruptores y tarjetas perforadas.
- Diseño, construcción, operación, programación y mantenimiento: genios como Aiken, von Newman o Mauchley.

Segunda generación (1955-65)

- Utilidad: calculo científico e ingeniería.
- Tecnología: transistores y sistemas por lotes, que redujo su tamaño y precio.
- Método de programación: ensamblador y lenguajes de alto nivel (FORTRAN) sobre tarjetas perforadas.
- Fue el paso de procesamiento secuencial a procesamiento por lotes, por ejemplo FMS Y IBSYS.

Tercera generación (1965-80)

- Utilidad: calculo científico e ingeniería y procesamiento de caracteres.
- Tecnología: circuitos integrados y multiprogramación
- Ejemplos: OS/360, CTSS, MULTICS, UNIX.

Cuarta generación (1980-hoy)

- Tecnología: computadora personal.
- Microprocesadores: 8080, z80, 8086, 286, 386,486, Pentium, Core 2, Athlon, Alpha, Ultrasparc.
- Logros destacables: GUI, SO de red, SMP, SO distribuidos.

Variedad de sistemas operativos

Sistemas operativos de mainframe

Son las computadoras del tamaño de un cuarto completo que aún se encuentran en los principales centros de datos corporativos, es decir, las mainframes también están volviendo a figurar en el ámbito computacional como servidores web de alto rendimiento, servidores para sitios de comercio electrónico a gran escala y servidores para transacciones de negocio a negocio.

Los sistemas operativos para las mainframes están profundamente orientados hacia el procesamiento de muchos trabajos a la vez, de los cuales la mayor parte requiere muchas operaciones de E/S. Por lo general ofrecen tres tipos de servicios: procesamiento por lotes, procesamiento de transacciones y tiempo compartido.

Un ejemplo de sistema operativo de mainframe es el OS/390, un descendiente del OS/360. Sin embargo, los sistemas operativos de mainframes están siendo reemplazados gradualmente por variantes de UNIX, como Linux, sistemas operativos para servidores.

Se ejecutan en servidores, que son computadoras personales muy grandes, estaciones de trabajo o incluso mainframes. Dan servicio a varios usuarios a la vez a través de una red y les permiten compartir los recursos de hardware y de software. Los servidores pueden proporcionar servicio de impresión, de archivos o web. Los proveedores de Internet operan muchos equipos servidores para dar soporte a sus clientes y los sitios Web utilizan servidores para almacenar las páginas Web y hacerse cargo de las peticiones entrantes. Algunos sistemas operativos de servidores comunes son Solaris, Free BSD, Linux y Windows Server 200x.

Sistemas operativos de multiprocesadores

Una manera cada vez más común de obtener poder de cómputo de las grandes ligas es conectar varias CPU en un solo sistema. Dependiendo de la exactitud con la que se conecten y de lo que se comparta, estos sistemas se conocen como computadoras en paralelo, multicomputadoras o multiprocesadores. Muchos sistemas operativos populares (incluyendo Windows y Linux) se ejecutan en multiprocesadores.

Sistemas operativos de computadoras personales

Su trabajo es proporcionar buen soporte para un solo usuario. Se utilizan ampliamente para el procesamiento de texto, las hojas de cálculo y el acceso a Internet. Algunos ejemplos comunes son Linux, FreeBSD, Windows Vista y el sistema operativo Macintosh.

Sistemas operativos de computadoras de bolsillo

Una computadora de bolsillo o PDA (*Personal Digital Assitant*, Asistente personal digital) es una computadora que cabe en los bolsillos y realiza una pequeña variedad de funciones, como libreta de direcciones electrónica y bloc de notas. Además, hay muchos teléfonos celulares muy similares a los PDA's, con la excepción de su teclado y pantalla.

Una de las principales diferencias entre los dispositivos de bolsillo y las PC's es que los primeros no tienen discos duros de varios cientos de gigabytes, lo cual cambia rápidamente. Dos de los sistemas operativos más populares para los dispositivos de bolsillo son Symbian OS y Palm OS.

Sistemas operativos integrados

Los sistemas integrados (*embedded*), que también se conocen como incrustados o embebidos, operan en las computadoras que controlan dispositivos que no se consideran generalmente como computadoras, ya que no aceptan software instalado por el usuario. Algunos ejemplos comunes son los hornos de microondas, las televisiones, los autos, los grabadores de DVD's, los teléfonos celulares y los reproductores de MP3. Los sistemas como QNX y VxWorks son populares en este dominio.

Sistemas operativos de nodos sensores

Cada nodo sensor es una verdadera computadora, con una CPU, RAM, ROM y uno o más sensores ambientales. Ejecuta un sistema operativo pequeño pero real, por lo general manejador de eventos, que responde a los eventos externos o realiza mediciones en forma periódica con base en un reloj interno. El sistema operativo tiene que ser pequeño y simple debido a que los nodos tienen poca RAM y el tiempo de vida de las baterías es una cuestión importante.

Estas redes de sensores se utilizan para proteger los perímetros de los edificios, resguardar las fronteras nacionales, detectar incendios en bosques, medir la temperatura y la precipitación para el pronóstico del tiempo, deducir información acerca del movimiento de los enemigos en los campos de batalla y mucho más. TinyOS es un sistema operativo bien conocido para un nodo sensor.

Sistemas operativos en tiempo real

Estos sistemas se caracterizan por tener el tiempo como un parámetro clave. Muchos de estos sistemas se encuentran en el control de procesos industriales, en aeronáutica, en la milicia y en áreas de aplicación similares. Estos sistemas deben proveer garantías absolutas de que cierta acción ocurrirá en un instante determinado.

Como en los sistemas en tiempo real es crucial cumplir con tiempos predeterminados para realizar una acción, algunas veces el sistema operativo es simplemente una biblioteca enlazada con los programas de aplicación, en donde todo está acoplado en forma estrecha y no hay protección entre cada una de las partes del sistema. Un ejemplo de este tipo de sistema en tiempo real es e-Cos.

Sistemas operativos de tarjetas inteligentes

Los sistemas operativos más pequeños operan en las tarjetas inteligentes, que son dispositivos del tamaño de una tarjeta de crédito que contienen un chip de CPU. Tienen varias severas restricciones de poder de procesamiento y memoria. Algunos sistemas de este tipo pueden realizar una sola función, como pagos electrónicos; otros pueden llevar a cabo varias funciones en la misma tarjeta inteligente. A menudo éstos son sistemas propietarios.

Conceptos básicos de sistemas operativos

Procesos

Un proceso es en esencia un programa en ejecución. Cada proceso tiene asociado un espacio de direcciones, una lista de ubicaciones de memoria que va desde algún mínimo (generalmente 0) hasta cierto valor máximo, donde el proceso puede leer y escribir información. Cuando un proceso se suspende en forma temporal, todos estos apuntadores deben guardarse de manera que una llamada a read que se ejecute después de reiniciar el proceso lea los datos apropiados. En muchos sistemas operativos, toda la información acerca de cada proceso (además del contenido de su propio espacio de direcciones) se almacena en una tabla del sistema operativo, conocida como la tabla de procesos, la cual es un arreglo (o lista enlazada) de estructuras, una para cada proceso que se encuentre actualmente en existencia.

Así, un proceso (suspendido) consiste en su espacio de direcciones, que se conoce comúnmente como imagen de núcleo (en honor de las memorias de núcleo magnético utilizadas antaño) y su entrada en la tabla de procesos, que guarda el contenido de sus registros y muchos otros elementos necesarios para reiniciar el proceso más adelante.

Las llamadas al sistema de administración de procesos clave son las que se encargan de la creación y la terminación de los procesos. Los procesos relacionados que cooperan para realizar un cierto trabajo a menudo necesitan comunicarse entre sí y sincronizar sus actividades. A esta comunicación se le conoce como comunicación entre procesos.

Cada persona autorizada para utilizar un sistema recibe una UID (*User Identification*, Identificación de usuario) que el administrador del sistema le asigna. Cada proceso iniciado tiene el UID de la persona que lo inició. Un proceso hijo tiene el mismo UID que su padre. Los usuarios pueden ser miembros de grupos, cada uno de los cuales tiene una GID (*Group Identification*, Identificación de grupo).

Espacios de direcciones

Dado que la administración del espacio de direcciones de los procesos está relacionada con la memoria, es una actividad de igual importancia. Por lo general, cada proceso tiene cierto conjunto de direcciones que puede utilizar, que generalmente van desde 0 hasta cierto valor máximo. En el caso más simple, la máxima cantidad de espacio de direcciones que tiene un proceso es menor que la memoria principal. De esta forma, un proceso puede llenar su espacio de direcciones y aun así habrá suficiente espacio en la memoria principal para contener todo lo necesario.

En muchas computadoras las direcciones son de 32 o 64 bits, con lo cual se obtiene un espacio de direcciones de 232 o 264 bytes, respectivamente. En esencia, el sistema operativo crea la abstracción de un espacio de direcciones como el conjunto de direcciones al que puede hacer referencia un proceso.

El espacio de direcciones se desacopla de la memoria física de la máquina, pudiendo ser mayor o menor que la memoria física. La administración de los espacios de direcciones y la memoria física forman una parte importante de lo que hace un sistema operativo.

Archivos

Un archivo es una agrupación de información que se guarda en algún dispositivo no volátil. Desde la perspectiva del usuario, es la unidad mínima de almacenamiento que el sistema le provee.

Se requieren las llamadas al sistema para crear los archivos, eliminarlos, leer y escribir en ellos. Antes de poder leer un archivo, debe localizarse en el disco para abrirse y una vez que se ha leído información del archivo debe cerrarse, por lo que se proporcionan llamadas. Los archivos constan de las siguientes propiedades:

- Atributos:
    - Nombre: permite identificar el archivo a los usuarios.
    - Identificador: el SO le asigna este símbolo que lo identifica (número) que no hace único.
    - Tipo: puede ser un programa ejecutable, archivo de datos, etc.
    - Ubicación: puntero al dispositivo y lugar donde reside el archivo.
    - Tamaño: cantidad de información que contiene.

    - Protección: información de control para el acceso al archivo.
    - Información de conteo: contiene la fecha de creación, ultimo acceso, etc.
- Operaciones:
    - Crear y abrir: se crea o se abre un archivo.
    - Escribir: permite escribir es el archivo creado.
    - Leer: permite leer el contenido del archivo.
    - Reposicionar dentro de un archivo: logra acceder a cualquier parte del archivo.
    - Eliminar: se destruye el archivo al nivel del sistema d archivos.
    - Truncar: elimina la información que está dentro del archivo, pero sin eliminar el archivo como tal.

Tener un archivo abierto para el sistema implica mantener una estructura que tengan por o menos:

- Puntero(file pointer)
- Ubicación

En el FCB (File Control Block) se encuentra toda la información de los archivos y los punteros. Así mismo el LOCK es el sistema que provee acceso único a un archivo por parte de los procesos.

Entrada/salida

Cada sistema operativo tiene un subsistema de E/S para administrar sus dispositivos de E/S. Parte del software de E/S es independiente de los dispositivos, es decir, se aplica a muchos o a todos los dispositivos de E/S por igual. Otras partes del software, como los drivers de dispositivos, son específicas para ciertos dispositivos de E/S. Existen muchos tipos de dispositivos de entrada y de salida, incluyendo teclados, monitores, impresoras, etcétera. Es responsabilidad del sistema operativo administrar estos dispositivos.

Protección

Las computadoras contienen grandes cantidades de información que los usuarios comúnmente desean proteger y mantener de manera confidencial. Esta información puede incluir mensajes de correo electrónico, planes de negocios, declaraciones fiscales y mucho más. Es responsabilidad del sistema operativo administrar la seguridad del sistema de manera que los archivos, por ejemplo, sólo sean accesibles para los usuarios autorizados. Además de la protección de archivos, existen muchas otras cuestiones de seguridad. Una de ellas es proteger el sistema de los intrusos no deseados, tanto humanos como no humanos (por ejemplo, virus).

El Shell

Sirve como un buen ejemplo de la forma en que se pueden utilizar las llamadas al sistema. También es la interfaz principal entre un usuario sentado en su terminal y el sistema operativo, a menos que el usuario esté usando una interfaz gráfica de usuario. Existen muchos Shell, incluyendo *sh*, *csh*, *ksh* y *bash*. Todos ellos soportan la funcionalidad antes descrita, que se deriva del Shell original (*sh*).Cuando cualquier usuario inicia sesión, se inicia un Shell. El Shell tiene la terminal como entrada estándar y salida estándar. Empieza por escribir el indicador de comandos (prompt), un carácter tal como un signo de dólar, que indica al usuario que el Shell está esperando aceptar un comando.

Actualmente, muchas computadoras personales utilizan una GUI. De hecho, la GUI es sólo un programa que se ejecuta encima del sistema operativo, como un Shell. En los sistemas Linux, este hecho se hace obvio debido a que el usuario tiene una selección de (por lo menos) dos GUI’s: Gnome y KDE o ninguna (se utiliza una ventana de terminal en X11).

En Windows también es posible reemplazar el escritorio estándar de la GUI (*Windows Explorer*) con un programa distinto, para lo cual se modifican ciertos valores en el registro, aunque pocas personas hacen esto.

Estructura del sistema operativo

Un sistema operativo se puede estructurar respecto a:

- Capas
    - Capa 0: proporciona la multiprogramación básica de la CPU, es decir que los procesos cuando ocurren las interrupciones o expiran los cronómetros. Dichos sistemas constan de procesos secuenciales, estos se pueden programar sin importar que varios procesos se estén ejecutando en el mismo procesador.
    - Capa 1: aquí se administra la memoria, al mismo tiempo se asignaba el espacio de memoria principal para los diversos procesos y depósitos de palabras de 512k en el cual se utilizaba para almacenar partes de los procesos, en este caso las páginas.
    - Capa 2: comunicación entre procesos y la consola del usuario.
    - Capa 3: flujo de información a través de hardware, E/S.
    - Capa 4: soporta aplicaciones de los usuarios.
    - Capa 5: proceso operador del sistema.

- Niveles
    - Nivel 1:
        - sincronización entre procesos.
        - conmutación de la CPU.
        - gestión de interrupciones.
        - arranque inicial.
    - Nivel 2:
        - Gestión de memoria: asigna la memoria entre procesos.

        - Asignación y liberación de memoria.
        - Control, violación de acceso.
    - Nivel 3:
        - Nivel superior de gestión de procesos.
    - Nivel 4:
        - Administra hardware: creación de procesos E/S, asigna y libera, planifica E/S.
    - Nivel 5:
        - Control de archivos.

- Clasificación
    - Por estructura
        - Monolíticos: solo una unidad.
        - Estructurada: agregamos componente adicional.
    - Por usuario
        - Monousuario: solo un usuario.
        - Multiusuario: existen dos accesos por un solo canal.
    - Por tareas
        - Monotareas: solo soporta una acción.
        - Multitareas: realiza muchas acciones al mismo tiempo.
    - Por procesos
        - Uniprocesos: solo se ejecuta un proceso.
        - Multiprocesos:
            - ✓ Simétricos: llegan y asignan a quien esté disponible.
            - ✓ Asimétrico: el SO selecciona a uno de los procesos el cual jugara el papel de procesador maestro.

UNIDAD II

MÉTODOS DE COMUNICACIÓN ENTRE PROCESOS, Y ALGORITMOS CLÁSICOS PARA LA CALENDARIZACIÓN

MÉTODOS DE COMUNICACIÓN ENTRE PROCESOS

Los procesos pueden ser cooperantes o independientes, en el primer caso se entiende que los procesos interactúan entre sí y pertenecen a una misma aplicación. En el caso de procesos independientes en general se debe a que no interactúan y un proceso no requiere información de otros o bien porque son procesos que pertenecen a distintos usuarios.

Regiones críticas

Para solucionar las condiciones de competencia se implementó un modelo para prohibir que dos procesos accedan al mismo recurso. El modelo en cuestión se denomina exclusión mutua. La parte del programa en la que se tiene acceso a la memoria compartida se denomina región crítica o sección crítica. Necesitamos que se cumplan cuatro condiciones para tener una buena solución:

1. Dos procesos no pueden estar al mismo tiempo dentro de sus regiones críticas.
2. No pueden hacerse suposiciones sobre las velocidades ni el número de las CPU’s.
3. Ningún proceso que esté ejecutando afuera de su región crítica puede bloquear a otros procesos.
4. Ningún proceso deberá de tener que esperar de manera indefinida para entrar en su región crítica.

Exclusión mutua con espera ocupada

Las soluciones con espera ocupada funcionan de la siguiente manera, cuando un proceso intenta ingresar a su región crítica, verifica si está permitida la entrada. Si no, el proceso se queda esperando hasta obtener el permiso.

Inhabilitación de interrupciones

El método más simple para evitar las condiciones de competencia es hacer que cada proceso desactive todas sus interrupciones antes de entrar a su sección crítica y las active una vez que salió de la misma. Este modelo como se puede observar, éste modelo tiene una gran problema y es que si se produce una falla mientras que el proceso está en la región crítica no se puede salir de la misma y el sistema operativo no recuperaría el control.

Variables de bloqueo

En éste caso se genera una variable la cual puede tener dos valores o bien 0 (no hay ningún proceso en su sección crítica) o bien 1 (indicando que la sección crítica está ocupada) entonces cada proceso antes de ingresar a la sección crítica verifica el estado de la variable de cerradura y en caso de que la misma este en 0, le cambia el valor e ingresa a la misma y en caso de que la misma sea 1 el proceso se queda verificando el estado de la misma hasta que el mismo sea 0. El problema aquí se presenta si dos procesos verifican al mismo tiempo que la variable cerradura está en 0 e ingresan a la región crítica.

Alternancia estricta

El algoritmo de alternancia estricta no bloquea el ingreso a la región crítica cuando otro proceso se está ejecutando.

El problema de ésta solución es que cuando un proceso no está en la sección crítica igualmente tiene bloqueado el acceso a la misma y por lo tanto no permite que otro proceso que requiera ingresar a la misma logre hacerlo.

Instrucción TSL

Esta solución requiere ayuda del hardware y es debido a que en general las computadoras diseñadas para tener más de un procesador tienen una instrucción TLS RX, bloqueo que funciona: lee el contenido de la palabra de memoria bloqueo, lo coloca en el registro RX y luego guarda un valor distinto de cero en la dirección de memoria bloqueo. La CPU que ejecuta la instrucción TSL cierra el bus de memoria para impedir que otras CPU's tengan acceso a la memoria antes de que termine.

Activar y desactivar

El modelo de espera acotada tienen el inconveniente que se desperdicia tiempo de procesador. Este enfoque no sólo desperdicia tiempo de CPU, si no que puede tener efectos inesperados. Consideramos una computadora con dos procesos; A que es prioritario, y B, que no lo es. Las reglas de calendarización son de tal forma que A se ejecuta siempre que está en estado listo. En cierto momento, cuando B está en región crítica, A queda listo para ejecutarse. A inicia una espera activa, pero dado que B nunca se calendariza mientras A se está ejecutando, B nunca tendrá oportunidad de salir de su región crítica, y A seguirá dando vueltas en forma indefinida.

El problema del productor y el consumidor

El problema del productor y el consumidor describe el hecho de que cuando hay dos o más procesos interactuando a través de un buffer común habiendo procesos que ponen información o datos y otros que los sacan se pueden llegar a dar condiciones en las cuales los procesos que ingresan los datos no puedan hacerlo

debido a que el buffer ya se encuentra lleno y para el caso de los que sacan los datos del buffer intenten sacar datos cuando ya no hay nada que sacar. Para evitar estas condiciones se desarrollaron métodos de comunicación/sincronización entre procesos en los cuales se impide que esto suceda haciendo que el proceso productor "duerma" si el buffer está lleno y una vez que exista espacio el proceso "consumidor" despierte al productor para que siga generando o viceversa.

Semáforos

Un semáforo es una variable especial que constituye el método clásico para restringir o permitir el acceso a recursos compartidos en un entorno de multiprocesamiento (en el que se ejecutarán varios procesos concurrentemente). Los semáforos se emplean para permitir el acceso a diferentes partes de programas (llamados secciones críticas) donde se manipulan variables o recursos que deben ser accedidos de forma especial. Según el valor con que son inicializados se permiten a más o menos procesos utilizar el recurso de forma simultánea.

Un tipo simple de semáforo es el binario, que puede tomar solamente los valores 0 y 1. Se inicializan en 1 y son usados cuando sólo un proceso puede acceder a un recurso a la vez. Son esencialmente lo mismo que los mutex. Cuando el recurso está disponible, un proceso accede y decrementa el valor del semáforo con la operación P. El valor queda entonces en 0, lo que hace que si otro proceso intenta decrementarlo tenga que esperar. Cuando el proceso que decremento el semáforo realiza una operación *V*, algún proceso que estaba esperando comienza a utilizar el recurso. Los semáforos resuelven el problema del despertar perdido. Es indispensable que se implementen de manera indivisible. El procedimiento normal es implementar Down y up como llamadas al sistema y que el sistema operativo inhabilite en forma breve todas las interrupciones, mientras está probando y actualizando el semáforo, así como poniendo el proceso a dormir, si es necesario.

Dado que estas acciones sólo requieren unas cuantas instrucciones, la inhabilitación de las interrupciones no es perjudicial. Si se están utilizando múltiples CPU's, cada semáforo deberá protegerse con una variable de bloqueo, utilizándose la instrucción TSL para garantizar que sólo una CPU a la vez examine el semáforo. Debe entender que el uso de TSL para evitar que varias CPU's tengan acceso simultáneo al semáforo es muy distinto de la espera activa del productor o el consumidor para que el otro vacíe o llene el búfer.

Mutexes

Los algoritmos de exclusión mutua (comúnmente abreviad como mutex por mutual exclusión) se usan en programación concurrente para evitar el ingreso a sus secciones críticas por más de un proceso a la vez. La sección crítica es el fragmento de código donde puede modificarse un recurso compartido.

La mayor parte de estos recursos son las señales, contadores, colas y otros datos que se emplean en la comunicación entre el código que se ejecuta cuando se da servicio a una interrupción y el código que se ejecuta el resto del tiempo. Se trata de un problema de vital importancia porque, si no se toman las precauciones debidas, una interrupción puede ocurrir entre dos instrucciones cualesquiera del código normal y esto puede provocar graves fallos.

La técnica que se emplea por lo común para conseguir la exclusión mutua es inhabilitar las interrupciones durante el conjunto de instrucciones más pequeño que impedirá la corrupción de la estructura compartida (la sección crítica). Esto impide que el código de la interrupción se ejecute en mitad de la sección crítica. En un sistema multiprocesador de memoria compartida, se usa la operación indivisible test-and-set sobre una bandera, para esperar hasta que el otro procesador la despeje. La operación test-and-set realiza ambas operaciones sin liberar el bus de memoria a otro procesador. Así, cuando el código deja la sección crítica, se despeja la bandera. Esto se conoce como spin lock o espera activa.

Algunos sistemas tienen instrucciones multioperación indivisibles similares a las anteriormente descritas para manipular las listas enlazadas que se utilizan para las colas de eventos y otras estructuras de datos que los sistemas operativos usan comúnmente.

Monitores

Un monitor es una colección de procedimientos, variables y estructuras de datos que se agrupan en un tipo especial de módulo o paquete. Los procesos pueden invocar a los procedimientos de un monitor cuando lo deseen, pero no pueden acceder en forma directa a sus estructuras de datos internas desde procedimientos declarados fuera de dicho monitor.

Los monitores tiene una prioridad importante que los hace útiles para lograr exclusión mutua: sólo un proceso puede estar activo en un monitor a la vez. Los monitores son una construcción de lenguaje de programación, así que el compilador sabe que son especiales y puede manejar las llamadas a los procedimientos de monitor, de manera distinta como maneja otras llamadas a procedimientos. Por lo regular, cuando un proceso llama a un proceso de monitor, las primeras instrucciones de éste verifican si algún otro proceso está activo dentro del monitor. Si es así, el proceso invocador se suspenderá hasta que el otro proceso haya salido del monitor. Si ningún otro proceso lo está usando, el proceso invocador puede entrar.

El compilador debe implementar la exclusión mutua en los ingresos a un monitor, pero es posible utilizar un mutex o un semáforo binario. Puesto que el compilador, no el programador, quien “tramita” la exclusión mutua, es mucho menos probable que algo salga mal.

En todo caso, quien escribe el monitor no tiene que saber la manera en que el compilador maneje la exclusión mutua. Basta saber que convirtiendo todas las regiones críticas en procedimiento de monitor, dos procesos nunca podrán ejecutar sus regiones críticas al mismo tiempo.

Barreras

Algunas aplicaciones se dividen en fases y tienen la regla que ningún proceso puede pasar a la siguiente fase antes de que todos los procesos esté listos para hacerlo. Este comportamiento puede lograrse colocando una barrera al final de cada fase. Cuando un proceso llega a la barrera. Se bloque has que todos los procesos han llegado a ella.

DIFERENCIA ENTRE PROCESOS Y SUBPROCESOS ASÍ COMO SUS CARACTERÍSTICAS

Procesos

Un proceso es un programa en ejecución. Un proceso simple tiene un hilo de ejecución, por el momento dejemos esta última definición como un concepto, luego se verá en más detalle el concepto de hilo. Una vez definido que es un proceso nos podríamos preguntar cuál es la diferencia entre un programa y un proceso, y básicamente la diferencia es que un proceso es una actividad de cierto tipo que contiene un programa, entradas salidas y estados.

Estados de los procesos

Un proceso puede estar en cualquiera de los siguientes tres estados: listo, en ejecución y bloqueado.

Los procesos en el estado listo son los que pueden pasar a estado de ejecución si el planificador los selecciona. Los procesos en el estado ejecución son los que se están ejecutando en el procesador en ese momento dado. Los procesos que se encuentran en estado bloqueado están esperando la respuesta de algún otro proceso para poder continuar con su ejecución. Por ejemplo operación de E/S.

El modelo de procesos

En este modelo. Todo el software ejecutable de una computadora, que a veces incluye al sistema operativo, se organiza en varios procesos secuenciales, o simplemente procesos. Un proceso no es más que un programa en ejecución, e incluye los valores que tienen el contador del programa, los riesgos y las variables. Cada proceso tiene su CPU virtual, claro que en realidad la verdadera CPU cambia en forma continua de un proceso a otro, que trata de comprender en que la CPU cambia de un programa a otro.

Con la CPU conmutando entre los procesos, la rapidez con la que un proceso efectúa sus operaciones no será uniforme y es probable que ni siquiera sea reproducible si los mismos procesos se ejecutan a la vez. Los procesos no deben programarse con base en supuestos acerca de los tiempos.

Creación de procesos

Los sistemas operativos requieren alguna forma de comprobar que existan todos los procesos necesarios. En los sistemas de propósito general hace falta algún mecanismo para crear y terminar procesos según donde se necesite durante la operación. Hay cuatro sucesos principales que causan la creación de procesos:

1. Inicialización del sistema.
2. Ejecución de una llamada al sistema para crear procesos por parte de un proceso en ejecución.

3. Solicitud de un usuario para crear un proceso.
4. Inicio de un trabajo por lotes.

Cuando se arranca un sistema operativo, por lo regular se crean varios procesos. Algunos son de primer plano, procesos que interactúan con usuarios y que trabajan para ellos. Otros son procesos de segundo plano que no están asociados con un usuario en particular, sino que tiene una función específica. Podría diseñarse un proceso de segundo plano que acepte el correo electrónico entrante; este proceso quedaría inactivo casi todo el día pero entraría en acción repentinamente si llega algún mensaje de correo electrónico. Podría diseñarse otro proceso para aceptar las solicitudes de página web alojadas en esa máquina, y se activaría cuando llegar una solicitud para atenderla. Los procesos que permanecen en segundo plano para encargarse de alguna actividad, como correo electrónico, páginas Web, noticias, impresiones etcétera, se llaman demonios (deamons).

Terminación de procesos

Los procesos se ejecutan ya creados y realizan la labor que se les encomendó. Nada es eterno y los procesos no son la excepción. Tarde o temprano el proceso nuevo terminará, por lo regular debido a una de las siguientes condiciones:

1. Terminación normal (voluntaria).
2. Terminación por error (voluntaria).
3. Error fatal (involuntaria).
4. Terminado por otro proceso (involuntaria).

La mayoría de los procesos terminan porque ya realizó su trabajo. Una vez que un compilador ha compilado el programa que se le alimentó, ejecuta una llamada para iniciar al sistema operativo que ya terminó.

Jerarquía de procesos

Un proceso termina cuando ejecuta su última instrucción, sin embargo existen circunstancias en que se requiere terminarlo en forma forzada. Un proceso termina a otro mediante una instrucción del tipo kill Id. La operación kill es usualmente invocada solamente por un proceso padre para culminar la ejecución de un hijo. Como la instrucción requiere la identificación del proceso a ser terminado, la instrucción fork da como retorno esta información (Id = fork L). Existen numerosas razones por las cuales un padre puede detener la ejecución de un hijo.

En muchos sistemas operativos se establece la condición de que los procesos hijos no pueden continuar la ejecución si el padre ha sido finalizado. Un proceso que no termina su ejecución durante todo el tiempo en que el sistema operativo está funcionando se dice que es estático. Un proceso que finaliza se dice que es dinámico.

Si un sistema operativo consiste de solo un número limitado de procesos estáticos entonces su correspondiente grafo de procesos será también estático. En caso diferente será dinámico. La estructura estática solo está presente en sistemas operativos muy simples.

Estados de procesos

En el modelo de procesos todo el Software ejecutable, a menudo incluyendo el propio sistema de operación, se organiza como procesos secuenciales. Aparentemente cada proceso tiene su propio procesador central, pero en realidad este cambia de uno a otro de acuerdo con el concepto de multiprogramación (seudoparalelismo).

Los procesos son totalmente aleatorios en el tiempo y el comportamiento de un conjunto de ellos dependerá de las condiciones en un instante dado. Esto implica que los programas no pueden ser elaborados asumiendo lo que pasará en el futuro cuando se están procesando. Un proceso puede tener diferentes estados durante su existencia. El número de estados dependerá del diseño del sistema operativo, pero al menos hay tres que siempre estarán presentes:

1. En ejecución: El proceso está en posesión del CPU en ese instante.
2. Listo: El proceso está en condiciones de ejecutar, pero está detenido temporalmente para permitir a otro proceso la ejecución.
3. Bloqueado: El proceso está esperando hasta que ocurra un evento externo (por ejemplo, una E/S).

Desde el punto de vista lógico, los primeros dos estados son similares. En ambos casos, el proceso está dispuesto a ejecutarse, sólo que en el segundo por el momento no hay CPU disponible para él. El tercer estado es diferente, el proceso no puede ejecutarse, aunque la CPU no tenga nada más que hacer.

Implementación de procesos

La implementación del modelo de procesos se logra debido a que el sistema operativo almacena en una tabla denominada tabla de control de procesos información relativa a cada proceso que se está ejecutando en el procesador. Cada línea de esta tabla representa a un proceso. La información que se almacena es la siguiente:

1) Identificación del proceso.
2) Identificación del proceso padre.
3) Información sobre el usuario y grupo.
4) Estado del procesador.
5) Información de control de proceso

5.1) Información del planificador.
5.2) Segmentos de memoria asignados.
5.3) Recursos asignados.

Subprocesos

En los sistemas operativos tradicionales cada proceso tiene un espacio de direcciones y un solo subproceso de control. Abundan las simulaciones en las que es deseable tener varios subprocesos de control en el mismo espacio de direcciones, operando de forma seudoparalela, como si fueran procesos individuales.

Modelo de subprocesos

Un proceso tiene un espacio de direcciones que contiene los datos y el texto de programa, así como otros recursos, que podrían incluir archivos abiertos, procesos hijos, alarmas pendientes, manejadores de señales, información contable, etc. Al juntar todas estas cosas en forma de un proceso, se le puede administrar con más facilidad.

El otro concepto que tiene un proceso es un subproceso de ejecución, o simplemente subproceso. Éste tiene un contador de programa que indica cuál instrucción se ejecutará a continuación; tiene riesgos, que contiene sus variables de trabajo actuales, y tiene una pila, que contiene el historial de ejecución, con un marco por cada procedimiento invocado del cual todavía no se haya regresado. Aunque un subproceso debe ejecutarse en algún proceso, el subproceso y su proceso son conceptos distintos que pueden tratarse aparte. Los procesos sirven para agrupar recursos; los subprocesos son las entidades que se calendarizan para ejecutarse en la CPU.

Los subprocesos aportan al modelo de procesos la posibilidad de que haya varias ejecuciones en el mismo entorno de un proceso, en gran medida independientes una de otra. Tener múltiples subprocesos ejecutándose en paralelo en un proceso es análogo a tener múltiples procesos ejecutándose en paralelo en una computadora. Primero, los subprocesos comparten un espacio de direcciones, archivos abiertos y otros recursos. Segundo, los procesos comparten una memoria física, discos, impresoras y otros recursos. Debido a que los subprocesos tienen algunas de las propiedades de los procesos, a veces se les llama procesos ligeros. También se emplea el término múltiples procesos para describir la situación en la que se permiten varios subprocesos en el mismo proceso.

Los distintos subprocesos de un proceso no son tan independientes como son los procesos distintos. Todos los subprocesos tienen exactamente el mismo espacio de direcciones, lo que implica que comparten las mismas variables globales. Puesto que cada subproceso puede tener acceso a todas las direcciones de memoria del espacio de direcciones del proceso, un subproceso podría leer, modificar o incluso borrar por completo la pila de otro subproceso. No existe protección entre los procesos porque es imposible y no debería ser necesaria. A diferencia de procesos distintos, que podrían pertenecer a usuarios distintos y ser hostiles entre sí, un proceso siempre pertenece a un solo usuario, y es de suponer que el usuario creó múltiples subprocesos con el fin de que cooperen, no de que peleen. Además de compartir un espacio de direcciones, todos los subprocesos comparten el mismo conjunto de archivos abiertos, procesos hijos, alarmas, señales, etc.

Si hay múltiples subprocesos, los procesos generalmente empiezan con un solo subproceso. Éste puede crear subprocesos nuevos invocando a un procedimiento de biblioteca, thread_create y cuando termina thread_exit. Una vez hecho esto, desaparecerá y ya no podrá calendarizarse.

Uso de subprocesos

Es precisamente el que se da en favor de tener procesos. Sólo que ahora con los subprocesos añadimos un elemento nuevo: la posibilidad de que las entidades paralelas compartan un espacio de direcciones y todos sus datos. Esta capacidad es indispensable en ciertas aplicaciones, y es por ello que el esquema de múltiples procesos no es la solución.

Un segundo argumento en favor de los subprocesos es que, al no estar enlazados con recursos, son más facies de crear y destruir que los procesos. En muchos sistemas, la creación de un subproceso es 100 veces más rápida que la creación de un proceso. Si el número de subprocesos necesarios cambia en forma dinámica y con rapidez esta propiedad es útil.

Un tercer motivo para tener subprocesos también se relaciona con el desempeño. Los subprocesos no mejoran el desempeño cuando todos usan intensivamente la CPU, pero si se realiza una cantidad considerable tanto de cómputo cono de E/S, los subprocesos permiten traslapar estas actividades y así acelerar la aplicación.
Son útiles en sistemas con múltiples CPU's, en los que es posible un verdadero paralelismo.

Si el programa fuera de un solo subproceso, cada vez que se iniciara un respaldo en disco se ignorarían los comandos provenientes del teclado y el ratón, hasta que dicho respaldo terminara. El usuario percibiría esto como lentitud del sistema. Los sucesos de teclado y ratón interrumpirían el respaldo en disco para que el desempeño mejorara, pero eso daría pie a un modelo de programación complejo, controlado por interrupciones. Con tres subprocesos, el modelo de programación es mucho más sencillo. El primero se limita a interactuar con el usuario. El segundo reformatea el documento cuando se le solicita. El tercero escribe el contenido de la RAM en disco en forma periódica.

Implementación de subprocesos en espacio de usuario

Hay dos formas principales de implementar subprocesos: en espacio de usuario y en el Kernel. La decisión ha dado pie a cierta controversia, y también existe una implementación híbrida. El primer método consiste en colocar por completo el sistema de subprocesos en espacio de usuario. El Kernel no sabe nada de ellos. En lo que a él respecta, está administrando procesos ordinarios, de un solo subproceso. La primera ventaja, es que puede implementarse un sistema de subprocesos en el nivel de usuario en un sistema operativo que no maneje subprocesos. Todos los sistemas operativos solían pertenecer a esta categoría, y toda vía subsisten algunos.

Cuando los subprocesos se administran en espacio de usuario, cada proceso necesita su propia tabla de subprocesos privada para dar seguimiento a sus subprocesos. Esta tabla es análoga a la de procesos del Kernel, salvo que sólo guarda las propiedades de subprocesos individuales, como el contador de programa, el programa, apuntador de pila, registros, estado, etcétera, de cada uno. La tabla de subprocesos es administrada por el sistema de tiempo de ejecución. Cuando un proceso pasa al estado listo o al bloqueo, en la tabla de subprocesos se guarda la información necesaria para reiniciarlo, exactamente en la misma forma en la que el Kernel guarda información acerca de los procesos en la tabla de procesos.

Cuando un subproceso hace algo que podría hacer que se bloquee localmente, como esperar que otro subproceso del proceso permite cierto trabajo, invoca un procedimiento del sistema de tiempo de ejecución. Este procedimiento verifica si el subproceso debe colocarse en estado bloqueado. Si es así, en la tabla de subprocesos se guardan los registros del subproceso, se busca un subproceso que esté listo para ejecutarse y se cargan en los registros de la máquina los valores guardados de ese nuevo proceso.

Tan pronto como se ha conmutado el apuntador de pila y el contador de programa, el nuevo subproceso se activa en forma automática. Si la máquina cuenta con una instrucción para almacenar todos los registros y otra para recuperarlos, toda la conmutación de subprocesos puede efectuarse con unas cuantas instrucciones. Este tipo de conmutación de subprocesos es al menos un orden de magnitud más rápido que un salto al Kernel, y es un argumento de peso de manejar los subprocesos en el nivel de usuario.

Tanto el procedimiento que guarda el estado del subproceso como el calendarizado son procedimientos locales, así que invocarlos es mucho más eficiente que llamar al Kernel. No se necesita una interrupción de sistema, no es preciso conmutar el contexto, no hay que guardar en disco el caché de memoria, etc. Esto agiliza mucho la calendarización de subprocesos.

Los subprocesos en el nivel de usuario tienen otras ventajas, como permitir que cada proceso tenga su propio algoritmo de calendarización personalizado. En algunas aplicaciones, como las que tienen un subproceso recolector de basura, es una ventaja no tener que preocuparse porque un subproceso vaya a detenerse en un momento poco conveniente. Es fácil aumentar su escala, pues los subprocesos de Kernel siempre requieren espacio de tabla y de pila en el Kernel, y esto puede ser problemático si hay un gran número de ellos.

Los sistemas de subprocesos en el nivel de usuario tienen problemas importantes. El principal es la forma en la que se implementen las llamadas bloqueadoras al sistema. Otro problema, hasta cierto punto análogo al de las llamadas bloqueadoras al sistema, es el de los fallos de página. Si el programa salta a una instrucción que no está en la memoria, ocurre un fallo de página y el sistema operativo trae la instrucción faltante del disco. El proceso se bloquea mientras se localiza y lee la instrucción necesaria.

Si un subproceso causa un fallo de página, el Kernel, que ni siquiera sabe de la existencia de los subprocesos, bloqueará todo el proceso hasta que termine la E/S de disco, aunque otros subprocesos puedan seguir ejecutándose. Otro problema de los sistemas de subprocesos en el nivel de usuario es que, si un proceso comienza a ejecutarse, ningún otro subproceso de ese proceso se ejecutará si el primero no cede de manera voluntaria la CPU. Dentro de un proceso dado no hay interrupciones de reloj, así que es imposible la calendarización de subprocesos por turno circular (roud-robin). A menos que un subproceso ingrese en el sistema de tiempo de ejecución por voluntad propia, el calendarizador no tendrá oportunidad de trabajar.

Una posible solución al problema de la ejecución indefinida de subprocesos es que el sistema de tiempo de ejecución solicite una señal de reloj una vez por segundo para asumir el control, pero esto también es burdo y molesto de programar. No siempre es posible emitir interrupciones de reloj periódicas con mayor frecuencia y, aunque se pudiera, implicaría un procesamiento adicional considerable. Un proceso también podría necesitar una interrupción de reloj, lo cual interferiría con el uso que hace del reloj el sistema de tiempo de ejecución.

Implementación de subprocesos en el Kernel

La tabla de subprocesos del Kernel contiene los riesgos, el estado y demás información de cada subprocesos. Estos datos son los mismos que se usan con subprocesos en el nivel de usuario, pero ahora están en el Kernel, no en el espacio de usuario. Esta información es un subconjunto de la que los kernels tradicionales mantienen acerca de cada uno de los procesos de un solo subprocesos, el estado del proceso. Además, el Kernel también mantiene la tabla de procesos tradicional con la que da seguimiento a los procesos.

Todas las llamadas que podrían bloquear un subproceso que implementan como llamadas al sistema y tienen un costo mucho mayor que las llamadas a procedimientos de un sistema de tiempo de ejecución. Cuando un subproceso se bloquea, el Kernel, puede ejecutar otro subproceso del mismo proceso o alguno de otro proceso. Con subprocesos en el nivel de usuario, el sistema de tiempo de ejecución sigue ejecutando subprocesos de su propio proceso hasta que el Kernel le quita la CPU.

Debido al costo relativamente mayor de crear y destruir subprocesos en el Kernel, algunos sistemas adoptan un enfoque ecológico correcto y reciclan sus procesos. Cuando un subproceso se destruye, se marca como no ejecutable, pero sus estructuras de datos en el Kernel no sufren alteración. Cuando es necesario crear un subproceso, se reactiva un antiguo, ahorrando algo de procesamiento extra. Los subprocesos en el nivel de usuario también pueden reciclarse, pero debido a que el procedimiento extra que implica su administración es mucho menor, hay menos incentivo para hacerlo.

Los subprocesos de Kernel no necesitan nuevas llamadas al sistema no bloqueadoras. Si un subproceso de un proceso causa un fallo de página, el Kernel puede verificar con facilidad si el proceso tiene algún otro subproceso ejecutable, y en su caso, ejecutar uno de ellos mientras espera que la página necesaria llegue del disco. Su propia desventaja es el costo elevado de una llamada al sistema; si las operaciones de subprocesos son usuales, se requerirá mucho procesamiento extra.

Implementaciones híbridas

En este diseño, el Kernel sólo tiene conocimiento de los subprocesos en el nivel de Kernel y únicamente los calendariza a ellos. Algunos de ellos podrían tener multiplexados múltiples subprocesos de nivel de usuario. Estos se crean, destruyen y calendarizan al igual que los de nivel de usuario en un proceso que se

ejecuta en un sistema operativo sin capacidad de múltiples subprocesos, en este modelo, cada subproceso en el nivel del Kernel tiene algún conjunto de subprocesos en el nivel de usuario que se turnan para usarlo.

Activaciones de calendarizador

Lo que se busca en las activaciones del calendarizador es imitar la funcionalidad de los subprocesos del Kernel pero con el desempeño y con la mayor flexibilidad que suelen tener los sistemas de subprocesos implementados en el espacio de usuario. En particular los subprocesos de usuario no deberían tener que emitir llamadas al sistema no bloqueadoras especiales ni verificar con antelación es si posible emitir con seguridad ciertas llamadas al sistema. Cuan un subproceso sea Bloqueado por una llamada al sistema o un fallo de página, debería ser posible ejecutar otro subproceso dentro del mismo proceso, si haya alguno listo.

La eficiencia se logra evitando transiciones innecesarias entre el espacio de usuario y el de Kernel. Si un subproceso se bloquea en espera de que otro haga algo, por otro, el Kernel no tiene por qué intervenir, y es innecesario el procedimiento adicional de la transición Kernel usuario. El sistema de tiempo de ejecución en espacio de usuario puede bloquear el subprocesos sincronizador y calendarizar otro por su cuenta.

Cuando se usan activaciones del calendarizador, el Kernel asigna ciertas numerosas de procesadores virtuales a cada proceso y permite al sistema de tiempo de ejecución asignar subprocesos a procesadores. Este mecanismo también puede usarse en un multiprocesador en el que los procesadores virtuales podrían ser CPU’s reales. Al principio el números de procesadores virtuales asignados a un proceso es uno, pero el proceso puede pedir más pero también devolver procesadores que ya necesite. El Kernel también puede recuperar procesadores virtuales ya asignados para asignarlos a otros procesos necesitados.

La idea fundamental que hace este esquema funcione es que cuando el Kernel sabe que un subproceso sea bloqueado, lo notifica al sistema de tiempo de ejecución del proceso, pasándole como parámetro de la pila el número de subprocesos en cuestión y una descripción del suceso. La notificación consiste en el que el Kernel activa el sistema del tiempo de ejecución en una dirección de inicio conocida, algo parecido al uso de señale. Este mecanismo se denomina llamada directa.

Una vez activado, el sistema de tiempo de ejecución recalendarizar sus subprocesos, casi siempre marcando el actual como bloqueado, tomando otros subprocesos listos preparando sus registros y reiniciándolo. Más adelante cuando el Kernel se entere que el subproceso original puede continuar su ejecución, en parar otra llamada al sistema de tiempo de ejecución para informarle del suceso. El sistema de tiempo de ejecución, puede reiniciar de inmediato el subproceso bloqueado o bien marcado como listo para que se ejecute después.

Si se presenta una interrupción de software mientras se está ejecutando un subproceso de usuario, la CPU interrumpida cambia a modo de Kernel. Si la interrupción se debió a un suceso que no interesa el proceso interrumpido, digamos que termino la E/S de otro proceso cuando el manejador de interrupciones termine, colocará el subproceso interrumpido otra vez en el estado que estaba antes de interrupciones termine, colorará el subproceso interrumpido otra vez en el estado en el que estaba antes de la interrupción. En cambio si la interrupción le interesa al proceso, no se iniciará el subproceso interrumpido; en vez de ello se suspenderá el sistema de tiempo de ejecución en esa CPU virtual, con el estado del subproceso interrumpido en la pila. Entonces corresponderá al sistema de tiempo de ejecución decidir cuál subproceso calendarizará en esa CPU: el interrumpido el que reinicien está listo o algún otro.

Subprocesos emergentes

Los subprocesos pueden ser útiles en los sistemas distribuidos. El enfoque tradicional consiste en tener un proceso o subproceso que se bloquea después de emitir una llamada, en espera de un mensaje. Cuando llega un mensaje este se acepta y procesa.

Puede adaptarse un enfoque completamente distinto en el que la llega de un mensaje hace que el sistema crea un subproceso para manejar dicho mensaje. Este tipo de subprocesos se denomina subproceso emergente. Una ventaja clave de los subprocesos emergentes es que debió a que son totalmente nuevos no tienen historial-registros, pila, etcétera- que debe restaurarse; cada uno inicia desde cero y todos son idénticos, y esto agiliza su creación. El mensaje que llegó se entrega al subproceso para que lo procese; el resultado de usar subprocesos emergentes es un importan reducción en la tardanza entre la llegada del mensaje y el inicio del subproceso.

Se necesita planear un poco por adelanto cuando se usan subprocesos emergentes. Si el sistema maneja subproceso que se ejecutan en el contexto del Kernel el subproceso podría ejecutarse allí. Hacer que el subproceso emergente se ejecute el espacio del Kernel suele ser más fácil y rápido que colocarlo en el espacio de usuario. Además un subproceso emergente en espacio de kernel puede tener acceso con facilidad a todas las tablas del kernel y a los dispositivos E/S lo cual podría ser necesario para procesar interrupciones. Por otra parte un subproceso de kernel con errores podría causar más daño que uno de usuario que también tuviera errores.

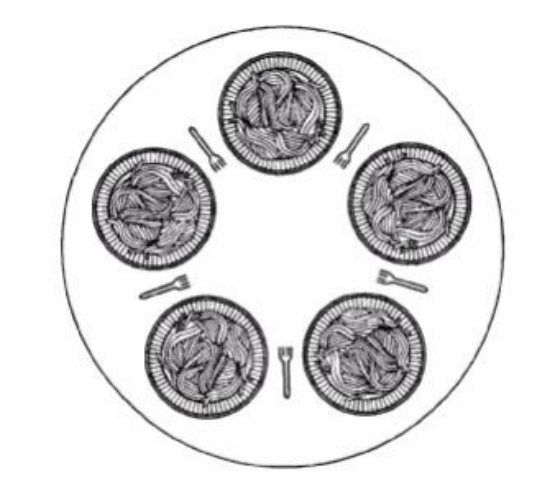
Problemas clásicos para entender la utilidad de los métodos de comunicación

Problemas clásicos de IPC

El problema general de la Calendarización ha sido descrito de diferentes maneras en la literatura. Usualmente es una redefinición en la noción clásica del problema de secuencia, considerando la calendarización como un proceso para la administración de recursos. Esta administración de recursos es básicamente un mecanismo o política usada para el manejo eficiente y efectivo del acceso a los recursos y el uso de los mismos recursos por sus varios consumidores (procesos).

Filósofos comelones

El problema de la cena de filósofos en 1965, Dijkstra planteó y resolvió un problema de sincronización al que llamó problema de la cena de filósofos. El problema tiene un planteamiento muy sencillo. Cinco filósofos están sentados alrededor de una mesa circular. Cada filósofo tiene ante sí un plato de espagueti. El espagueti es tan resbaloso que un filósofo necesita dos tenedores para comerlo. Entre cada par de platos hay un tenedor



La vida de un filósofo consiste en periodos alternantes de comer y pensar. Cuando un filósofo siente hambre, trata de adquirir sus tenedores izquierdo y derecho, uno a la vez, en cualquier orden. Si logra adquirir dos tenedores, comerá durante un rato, luego pondrá los tenedores en la mesa y seguirá pensando. La pregunta clave es: ¿podemos escribir un programa para cada filósofo que haga lo que se supone que debe hacer y nunca se trabe?

```
#define N 5 //N = número de comensales, palillos, filósofos, etc…
semáforo palillo[N]={1}; //arreglo de N semáforos que representan los palillos.
void filosofo (int i) //proceso de cada filosofo
{
while (true) //bucle infinito, se puede colocar condición de parada con variable
{
pensar ( ); //tiempo de espera aleatorio de c/filosofo antes de 'comer'
wait (palillo [i], palillo[i +1] mod 5); //pido los 2 palillos simultáneamente
comer ( ); //tiempo de espera mientras come (puede ser aleatorio)
signal (palillo [i]); //libero palillo izquierdo
signal (palillo [(i + 1) mod 5]); //libero palillo derecho
/* El orden en que libero los palillos no es de especial relevancia. */
}
}
void main()
{
parbegin (filosofo (0), filosofo (1), filosofo (2), filosofo (3), …, filosofo (N));
//inicio de los procesos (filósofos)
}
```

El procedimiento "tomar tenedor" espera hasta que el tenedor especificado está disponible y luego se apodera de él. Desafortunadamente, la solución obvia está equivocada. Supongamos que todos los filósofos toman su tenedor izquierdo simultáneamente. Ninguno podrá tomar su tenedor derecho, y tendremos un

bloqueo mutuo. Podríamos modificar el programa de modo que, después de tomar el tenedor izquierdo, el programa verifique si el tenedor derecho está disponible. Si no es así, el filósofo soltará su tenedor izquierdo, esperará cierto tiempo, y repetirá el proceso. Esta propuesta también fracasa, aunque por una razón distinta. Con un poco de mala suerte, todos los filósofos podrían iniciar el algoritmo simultáneamente, tomar su tenedor izquierdo, ver que su tenedor derecho no está disponible, dejar su tenedor izquierdo, esperar, tomar su tenedor izquierdo otra vez de manera simultánea, y así eternamente. Una situación así, en la que todos los programas continúan ejecutándose de manera indefinida pero no logran avanzar se denomina inanición.

*Inanición: Ocurre cuando a un proceso o un hilo de ejecución se le deniega siempre el acceso a un recurso compartido. Sin este recurso, la tarea a ejecutar no puede ser nunca finalizada.

Ahora podríamos pensar: “si los filósofos esperan un tiempo aleatorio en lugar del mismo tiempo después de fracasar en su intento por .disponer del tenedor derecho, la posibilidad de que sus acciones continuaran coordinadas durante siquiera una hora es excesivamente pequeña”. Esto es cierto, pero en algunas aplicaciones preferiríamos una solución que siempre funcione y que no tenga posibilidad de fallar debido a una serie improbable de números aleatorios. Una mejora que no está sujeta a bloqueo ni inanición consiste en proteger las cinco instrucciones que siguen a la llamada a “pensar” con un semáforo binario. Antes de comenzar a conseguir tenedores, un filósofo ejecutaría DOWN con mutex. Después de dejar los tenedores en la mesa, ejecutaría up con mutex. Desde un punto de vista teórico, esta solución es adecuada.

En la práctica, tiene un problema de rendimiento: sólo un filósofo puede estar comiendo en un instante dado. Si hay cinco tenedores disponibles, deberíamos estar en condiciones de permitir que dos filósofos comieran al mismo tiempo.

La solución correcta admite un paralelismo máximo con un número arbitrario de filósofos. Se utiliza un arreglo “estado” para mantenerse al tanto de si un filósofo está comiendo, pensando o hambriento tratando de disponer de tenedores. Un filósofo sólo puede pasar a la situación de “comiendo” si ninguno de sus vecinos está comiendo. Los vecinos del filósofo i están definidos por las macros LEFT y RIGHT.

En otras palabras, si “i” es 2, LEFT es 1 y RIGHT es 3.

El programa utiliza un arreglo de semáforos, uno por filósofo, de modo que los filósofos hambrientos pueden bloquearse si los tenedores que necesitan están ocupados. Cada proceso ejecuta el procedimiento “filósofo” como código principal, pero los demás procedimientos, “tomar tenedores”, “poner tenedores” y “probar” son procedimientos ordinarios y no procesos aparte. El problema de la cena de filósofos es útil para modelar procesos que compiten por tener acceso exclusivo a un número limitado de recursos, como dispositivos de E/S.

El problema de lectores y escritores

Otro problema famoso es el de los lectores y escritores (Courtois et al., 1971), que modela el acceso a una base de datos. Supóngase una base de datos por ejemplo un sistema de reservaciones de una línea aérea, con muchos procesos que compiten por leer y escribir en ella. Se puede permitir que varios procesos lean de la base de datos al mismo tiempo, pero si uno de los procesos está escribiendo (es decir, modificando) la base de datos, ninguno de los demás debería tener acceso a ésta, ni siquiera los lectores La pregunta es, ¿cómo programamos a los lectores y escritores?

```
typedef int semaphore;                  /* use su imaginación */
semaphore mutex = 1;                    /* controla el acceso a 'rc' */
semaphore db = 1;                       /* controla el acceso a la base de datos */
int rc = 0;                             /* núm. de procesos que leen o quieren leer */

void reader(void)
{
 while (TRUE) {                         /* repetir indefinidamente */
        down(&mutex);                   /* obtener acceso exclusivo a 'rc' */
        rc = rc + 1;                    /* ahora un lector más */
        if (rc == 1) down(&db);         /* si éste es el primer lector ... */
        up(&mutex);                     /* liberar el acceso exclusivo a 'rc' */
        read_data_base();               /* acceder a los datos */
        down(&mutex);                   /* obtener acceso exclusivo a 'rc' */
        rc = rc - 1;                    /* ahora un lector menos */
        if (rc == 0) up(&db);           /* si éste es el último lector ... */
        up(&mutex);                     /* liberar el acceso exclusivo a 'rc' */
        use_data_read();                /* región no crítica */
 }
}

void writer(void)
{
 while (TRUE) {                         /* repetir indefinidamente */
        think_up_data();                /* región no crítica */
        down(&db);                      /* obtener acceso exclusivo */
        write_data_base();              /* actualizar los datos */
        up(&db);                        /* liberar el acceso exclusivo */
 }
```

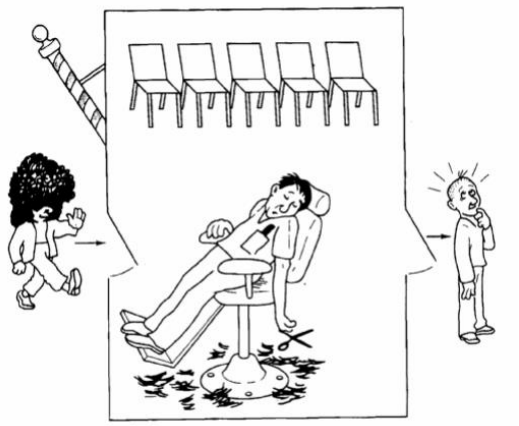
En esta solución, el primer lector que obtiene acceso a la base de datos ejecuta DOWN con el semáforo “db”. Los lectores subsecuentes se limitan a incrementar un contador, “rc”. Conforme los lectores salen, se decrementa el contador, y el último en salir ejecuta UP con el semáforo para permitir que un escritor bloqueado, si lo existe, entre. Supongamos que mientras un lector está usando la base de datos, llega otro lector. Puesto que tener dos lectores al mismo tiempo no está prohibido, se admite al segundo lector. También pueden admitirse un tercer lector y lectores subsecuentes si llegan.

Supongamos ahora que llega un escritor. El escritor no puede ser admitido en la base de datos, pues requiere acceso exclusivo, de modo que el escritor queda suspendido. Más adelante, llegan lectores adicionales. En tanto haya al menos un lector activo, se admitirán lectores subsecuentes. A consecuencia de esta estrategia, en tanto haya un suministro constante de lectores, entrarán tan pronto como lleguen. El escritor se mantendrá suspendido hasta que no haya ningún lector presente esto implica que el escritos nunca entrara.

Para evitar esta situación, el programa podría incluir una pequeña modificación: cuando llega un lector y un escritor está esperando, el lector queda suspendido detrás del escritor en lugar de ser admitido inmediatamente. Así, un escritor tiene que esperar hasta que terminan los lectores que estaban activos cuando llegó, pero no a que terminen los lectores que llegaron después de él. La desventaja de esta solución es que logra menor concurrencia y por tanto un menor rendimiento.

El problema del peluquero dormido

Otro problema de IPC clásico ocurre en una peluquería. Esta peluquería tiene un peluquero, una silla de peluquero y n sillas donde pueden sentarse los clientes que esperan. Si no hay clientes presentes, el peluquero se sienta en la silla de peluquero y se duerme.



Cuando llega un cliente, tiene que despertar al peluquero dormido. Si llegan clientes adicionales mientras el peluquero está cortándole el pelo a un cliente, se sientan (si hay sillas vacías) o bien salen del establecimiento (si todas las sillas están ocupadas). El problema consiste en programar al peluquero y sus clientes sin entrar en condiciones de competencia. La solución hace uso de semáforos: “customers” que cuenta a los clientes en espera (excluyendo al que está siendo atendido), “barbers” el peluquero que está ocioso, esperando clientes (0 o 1), y mutex (exclusión mutua).

También necesitamos una variable, “esperando” que también cuenta los clientes que están esperando, y que en esencia es una copia de customers. Necesitamos esta variable porque no es posible leer el valor actual de un semáforo. En esta solución, un cliente que entra en la peluquería debe contar el número de clientes que esperan. Si este número es menor que el número de sillas, se queda; si no, se va.

Cuando el peluquero llega a trabajar en la mañana, ejecuta el procedimiento barber (peluquero) que lo obliga a bloquearse en espera de customers (clientes) hasta que llegue alguien. Luego se duerme, cuando un cliente llega, ejecuta customer (cliente), cuya primera instrucción es adquirir mutex para entrar en una región crítica. Si otro cliente llega poco tiempo después, no podrá hacer nada hasta que el primero haya liberado mutex.

A continuación, el cliente verifica si el número de clientes en espera si es menor que el número de sillas. Si no es así, el cliente libera mutex y se sale sin su corte de pelo. Si hay una silla disponible, el cliente incrementa la variable “espera” y luego ejecuta UP con el semáforo customers, lo que despierta al peluquero. En este punto, tanto el peluquero como el cliente están despiertos. Cuando el cliente libera mutex, el peluquero lo toma, realiza algo de aseo e inicia el corte de pelo. Una vez terminado el corte de pelo, el cliente sale del procedimiento y de la peluquería. A diferencia de los ejemplos anteriores, no hay un ciclo para el cliente porque cada uno sólo recibe un corte de pelo. El peluquero sí opera en un ciclo, tratando de atender al siguiente cliente. Si hay uno presente, el peluquero realiza otro corte de pelo si no, se duerme.

El problema productor/consumidor

El problema Productor/Consumidor consiste en el acceso concurrente por parte de procesos productores y procesos consumidores sobre un recurso común que resulta ser un buffer de elementos.

Los productores tratan de introducir elementos en el buffer de uno en uno, y los consumidores tratan de extraer elementos de uno en uno. Para asegurar la consistencia de la información almacenada en el buffer, el acceso de los productores y consumidores debe hacerse en exclusión mutua. Adicionalmente, el buffer es de capacidad limitada, de modo que el acceso por parte de un productor para introducir un elemento en el buffer lleno debe provocar la detención del proceso productor. Lo mismo sucede para un consumidor que intente extraer un elemento del buffer vacío.

ALGORITMOS DE CALENDARIZACIÓN ENTRE PROCESOS

Componente del sistema operativo que decide cuál de los procesos es el que entrara a la CPU. Su decisión es basada según el sistema que este administrando y es resuelta por los Algoritmos de Calendarización. En la época de los sistemas por lote con entradas en forma de imágenes de tarjetas en una cinta magnética, el algoritmo de planificación era sencillo: simplemente se ejecutaba el siguiente trabajo de la cinta: Cuando aparecieron los sistemas de tiempo compartido, el algoritmo de calendarización se volvió más complejo porque casi siempre había varios usuarios en espera de ser atendidos Incluso en las computadoras personales, puede haber varios procesos iniciados por el usuario compitiendo por la CPU, sin mencionar los trabajos de segundo plano, como los demonios de red o de correo electrónico que envían o reciben mensajes.

Antes de examinar algoritmos de planificación específicos, debemos pensar en qué está tratando de lograr el planificador. Después de todo, éste se ocupa de decidir una política, no de proveer un mecanismo.

1. Equitatividad —asegurarse de que cada proceso reciba una parte justa del tiempo de CPU.
2. Eficiencia —mantener la CPU ocupada todo el tiempo.

3. Tiempo de respuesta —minimizar el tiempo de respuesta para usuarios interactivos.
4. Retorno —minimizar el tiempo que los usuarios por lotes tienen que esperar sus salidas.
5. Volumen de producción —maximizar el número de trabajos procesados por hora.

Algunos de estos objetivos son contradictorios. Si queremos minimizar el tiempo de respuesta para los usuarios interactivos, el planificador no deberá ejecutar trabajos por lotes a los usuarios por lotes seguramente no les gustaría este algoritmo, pues viola el criterio 4. Después de todo, la cantidad de tiempo de CPU disponible es finita. Para darle más a un usuario tenemos que darle menos a otro.
Una complicación que deben enfrentar los planificadores es que cada proceso es único e impredecible.

Algunos dedican una buena parte del tiempo a esperar E/S de archivos, mientras otros usarían la CPU durante horas si se les permitiera hacerlo. Cuando el planificador comienza a ejecutar un proceso, nunca sabe con certeza cuánto tiempo pasará antes de que dicho proceso se bloquee, sea para E/S, en espera de un semáforo o por alguna otra razón. Para asegurarse de que ningún proceso se ejecute durante demasiado tiempo, casi todas las computadoras tienen incorporado un cronómetro o reloj electrónico que genera interrupciones periódicamente.

Cuando calendarizamos:

1.-Al crear un proceso (decidir si entra el padre o el hijo)
2.-Al terminar un proceso (Antes de agotar su tiempo se debe elegir a otro listo o inactivo)
3.-Al bloquear un proceso

4.-Al recibir una interrupción (debe decidirse cual sacar/meter una vez terminada la interrupción)

La estrategia de permitir que procesos lógicamente ejecutables se suspendan temporalmente se denomina planificación expropiativa. La ejecución hasta terminar se denomina planificación no expropiativa. Un proceso puede ser suspendido en un instante arbitrario, sin advertencia, para que otro proceso pueda ejecutarse. Esto da pie a condiciones de competencia y requiere semáforos, monitores, mensajes o algún otro método avanzado para prevenirlas.

Aunque los algoritmos de planificación no expropiativos son sencillos y fáciles de implementar, por lo regular no son apropiados para sistemas de aplicación general con varios usuarios que compiten entre sí. Por otro lado, en un sistema dedicado como un servidor de base de datos, bien puede ser razonable que el proceso padre inicie un proceso hijo para trabajar con una solicitud y dejarlo que se ejecute hasta terminar o bloquearse.

La diferencia respecto al sistema de aplicación general es que todos los procesos del sistema de bases de datos están bajo el control de uno solo, que sabe lo que cada hijo va a hacer y cuánto va a tardar.

Categorías de algoritmos de Calendarización
1.- POR LOTES
Se recomienda la calendarización no expropiativa.
2.-INTERACTIVOS
Se recomienda la calendarización no expropiativa para que no acaparen recursos
3.- TIEMPO REAL
Casi no requiere la expropiación porque son procesos que actúan sobre una sola aplicación.

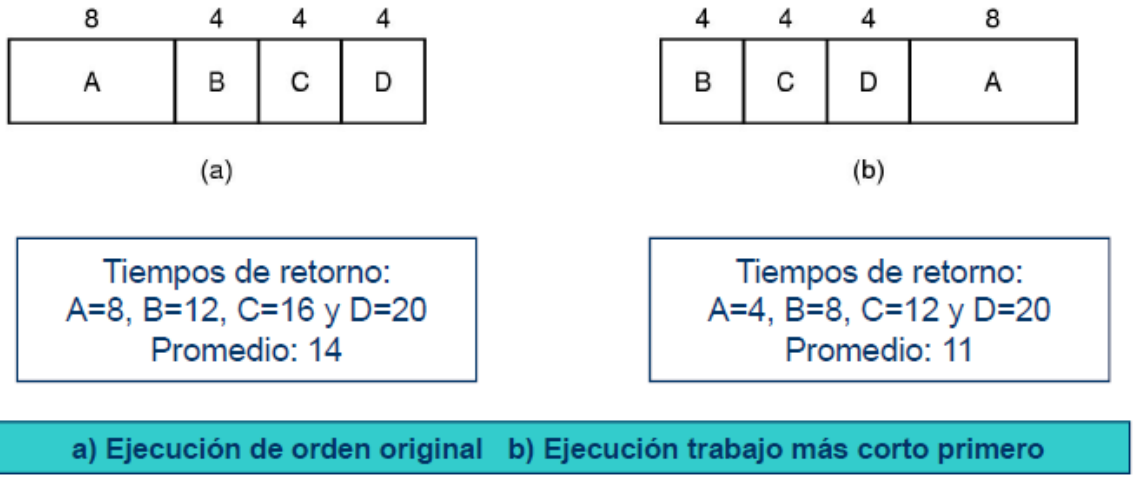
Calendarización en sistemas por lotes

FIFO (primero en llegar primero en ser atendido)

Es uno de los más sencillos que es no expropiativo, aquí la CPU se asigna a los procesos en el orden en que lo solicitan. Hay una cola de procesos listos. Cuando el primer trabajo entra en el sistema en la mañana, se le inicia de inmediato y se le permite ejecutar todo el tiempo que desee. A medida que llegan otros trabajos se les coloca al final de la cola. No se recomienda en procesos con tiempos muy heterogéneos.

Trabajo más corto primero

Este supone un conocimiento anticipado de los tiempos de ejecución. Si hay varios trabajos de la misma importancia en la cola de entrada, el calendarizador escoge el trabajo más corto primero.



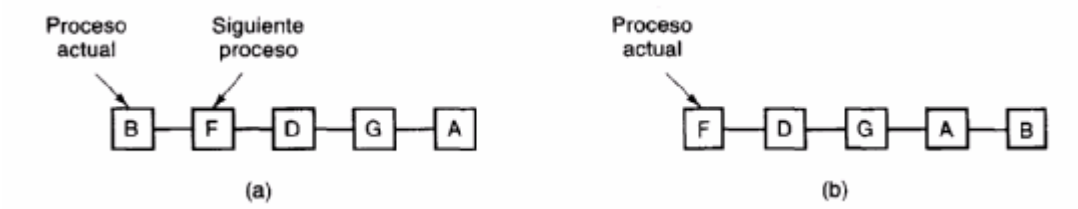
Tiempo restante más corto primero

Una versión expropiativa de la estrategia anterior es la de tiempo restante más corto a primero. En este algoritmo, el calendarizador siempre escoge el proceso con base en el tiempo que falta para que termine de ejecutarse, en este caso también es preciso conocer los tiempos de ejecución. Este esquema permite que trabajos cortos nuevos obtengan buen servicio.

Calendarización en sistemas interactivos

PLANIFICACIÓN ROUND ROBIN

Uno de los más antiguos, sencillos, equitativos y ampliamente utilizados es el de round robin. A cada proceso se le asigna un intervalo de tiempo, llamado cuanto, durante el cual se le permite ejecutarse. Si el proceso todavía se está ejecutando al expirar su cuanto, el sistema operativo se apropia de la CPU y se la da a otro proceso. Si el proceso se bloquea o termina antes de expirar el cuanto, la conmutación de CPU naturalmente se efectúa cuando el proceso se bloquee. Todo lo que el planificador tiene que hacer es mantener una lista de procesos ejecutables Cuando un proceso gasta su cuanto, se le coloca al final de la lista.



La única cuestión interesante cuando se usa el round robin es la duración del cuánto.

La conmutación de un proceso a otro requiere cierto tiempo para llevar a cabo las tareas administrativas: guardar y cargar registros y mapas de memoria, actualizar diversas tablas y listas, etc. Supongamos que el primer proceso se inicia de inmediato, el segundo podría no iniciarse hasta cerca de medio segundo después, y así sucesivamente. El pobre proceso que le haya tocado ser último podría tener que esperar 5 segundos antes de tener su oportunidad, suponiendo que los demás procesos utilizan su cuanto completo. Para casi cualquier usuario, un retardo de 5 segundos en la respuesta a un comando corto sería terrible.

El mismo problema puede presentarse en una computadora personal que maneja multiprogramación. Escoger un cuanto demasiado corto causa demasiadas Conmutaciones de procesos y reduce la eficiencia de la CPU, pero escogerlo demasiado largo puede dar pie a una respuesta deficiente a solicitudes interactivas cortas.

Planificación por prioridad

La planificación en round robin supone implícitamente que todos los procesos son igualmente importantes. Con frecuencia, las personas que poseen y operan sistemas de computadora multiusuario tienen ideas diferentes acerca del tema. La necesidad de tener en cuenta factores externos da pie a la planificación por prioridad. La idea básica es sencilla: a cada proceso se le asigna una prioridad, y se permite que se ejecute el proceso ejecutable que tenga la prioridad más alta. Incluso en una PC con un solo dueño, puede haber múltiples procesos, algunos más importantes que otros.

A fin de evitar que los procesos de alta prioridad se ejecuten indefinidamente, el planificador puede reducir la prioridad de los procesos que actualmente se ejecutan en cada tic del reloj (esto es, en cada interrupción de reloj). Si esta acción hace que la prioridad se vuelva menor que la del siguiente proceso con más alta prioridad, ocurrirá una conmutación de procesos. Como alternativa, se podría asignar a cada proceso un cuanto máximo en el que se le permitiera tener la CPU

continuamente; cuando se agota este cuanto, se da oportunidad al proceso con la siguiente prioridad más alta de ejecutarse. Podemos asignar prioridades a los procesos estática o dinámicamente por hora.

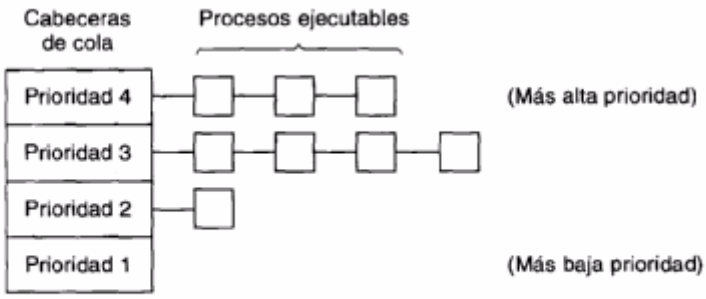


Figura 2-23. Algoritmo de planificación con cuatro clases de prioridad.

Colas múltiples

Uno de los primeros planificadores por prioridad se incluyó en CTSS (Corbato et al., 1962). CTSS tenía el problema de que la conmutación de procesos era muy lenta porque la 7094 sólo podía contener un proceso en la memoria. Cada conmutación implicaba escribir el proceso actual en disco y leer uno nuevo del disco. Los diseñadores de CTSS pronto se dieron cuenta de que resultaba más eficiente dar a los procesos limitados por CPU un cuanto largo de vez en cuando, en lugar de darles cuantos pequeños muy a menudo (porque se reducía el intercambio). Por otro lado, dar a todos los procesos un cuanto largo implicaría un tiempo de respuesta deficiente, como ya hemos visto. Su solución consistió en establecer clases de prioridad. Los procesos de la clase más alta se ejecutaban durante un cuanto.

Los procesos de la siguiente clase más alta se ejecutaban durante dos cuantos. Los procesos de la siguiente clase se ejecutaban durante cuatro cuantos, y así sucesivamente. Cada vez que un proceso agotaba todos los cuantos que tenía asignados, se le degradaba una clase.

Se adoptó la siguiente política para evitar que un proceso que en el momento de iniciarse necesita ejecutarse durante un tiempo largo pero posteriormente se vuelve interactivo fuera castigado indefinidamente. Se han utilizado muchos otros algoritmos para asignar procesos a clases de prioridad.

Por ejemplo, el influyente sistema XDS 940 (Lampson, 1968), construido en Berkeley, tenía cuatro clases de prioridad, llamadas terminal, E/S, cuanto corto y cuanto largo. Cuando un proceso que estaba esperando entradas de la terminal finalmente se despertaba, pasaba a la clase de prioridad más alta (terminal). Cuando un proceso que estaba esperando un bloque de disco quedaba listo, pasaba a la segunda clase. Si un proceso seguía en ejecución en el momento de expirar su cuanto, se le colocaba inicialmente en la tercera clase, pero si agotaba su cuanto demasiadas veces seguidas sin bloquearse para E/S de terminal o de otro tipo, se le bajaba a la cuarta cola. Muchos otros sistemas usan algo similar para dar preferencia a los usuarios y procesos interactivos por encima de los de segundo plano.

El primer trabajo más corto

La mayor parte de los algoritmos anteriores se diseñaron para sistemas interactivos. Examinemos ahora uno que resulta especialmente apropiado para los trabajos por lotes cuyos tiempos de ejecución se conocen por adelantado. En una compañía de seguros, por ejemplo, es posible predecir con gran exactitud cuánto tiempo tomará ejecutar un lote de 1000 reclamaciones, pues se efectúan trabajos similares todos los días. Si hay varios trabajos de igual importancia esperando en la cola de entrada para ser iniciados, el planificador deberá usar el criterio del primer trabajo más corto. Examinemos la figura. Aquí encontramos cuatro trabajos, A, B, C y D, con tiempos de ejecución de 8, 4, 4 y 4 minutos, respectivamente. Si los ejecutamos en ese orden, el tiempo de retomo para A será de 8 minutos, para B, de 12 minutos, para C, de 16 minutos, y para D, de 20 minutos, siendo el promedio de 14 minutos.

Figura 2-24. Ejemplo de planificación del primer trabajo más corto.

Consideremos ahora la ejecución de estos trabajos usando el primer trabajo más corto. Los tiempos de retomo son ahora de 4, 8, 12 y 20 minutos para un promedio de 11 minutos. Se puede demostrar que la política del primer trabajo más corto es óptima. Consideremos el caso de cuatro trabajos, con tiempos de ejecución de a, b, c y d, respectivamente. El primer trabajo termina en un tiempo a, el segundo, en a + b, etc. El tiempo de retomo medio es (4a + 3b + 2c + d)/4. Es evidente que a contribuye más al promedio que los demás tiempos, por lo que debe ser el trabajo más corto, siguiendo b, c y por último d, que es el más largo y sólo afecta su propio tiempo de retomo.

El mismo argumento es aplicable a cualquier cantidad de trabajos. Dado que la política del primer trabajo más corto produce el tiempo de respuesta medio mínimo, sería deseable poderlo usar también para procesos interactivos. Esto es posible hasta cierto punto. Los procesos interactivos generalmente siguen el patrón de esperar un comando, ejecutar el comando, esperar un comando, ejecutar el comando, etc. Si consideramos la ejecución de cada comando como un "trabajo" individual, podremos minimizar el tiempo de respuesta global ejecutando primero el trabajo más corto. El único problema es determinar cuál de los procesos ejecutables es el más corto. Una estrategia consiste en hacer estimaciones basadas en el comportamiento histórico y ejecutar el proceso con el tiempo de ejecución estimado más corto.

La técnica de estimar el siguiente valor de una serie calculando la media ponderada del valor medido actual y el estimado previo también se conoce como maduración, y es aplicable a muchas situaciones en las que debe hacerse una predicción basada en valores previos.

Planificación garantizada

Una estrategia de planificación totalmente distinta consiste en hacer promesas reales al usuario en cuanto al rendimiento y después cumplirlas. Una promesa que es realista y fácil de cumplir es la siguiente: si hay n usuarios en sesión mientras usted está trabajando, usted recibirá aproximadamente 1/n de la capacidad de la CPU. De forma similar, en un sistema monousuario con n procesos en ejecución, si todo lo demás es igual, cada uno deberá recibir un de los ciclos de CPU.

Para poder cumplir esta promesa, el sistema debe llevar la cuenta de cuánto tiempo de CPU ha tenido cada proceso desde su creación. A continuación, el sistema calculará el tiempo de CPU al que tenía derecho cada proceso, es decir, el tiempo desde la creación dividido entre n. Puesto que también se conoce el tiempo de CPU del que cada proceso ha disfrutado realmente, es fácil calcular la relación entre el tiempo de CPU recibido y el tiempo al que se tenía derecho. Una relación de 0.5 implica que el proceso sólo ha disfrutado de la mitad del tiempo al que tenía derecho, y una relación de 2.0 implica que un proceso ha tenido dos veces más tiempo del que debería haber tenido. El algoritmo consiste entonces en ejecutar el trabajo con la relación más baja hasta que su relación haya rebasado la de su competidor más cercano.

Planificación por lotería

Si bien hacer promesas a los usuarios y después cumplirlas es una idea admirable, es difícil de implementar. Podemos usar otro algoritmo para obtener resultados igualmente predecibles con una implementación mucho más sencilla. El algoritmo se llama planificación por lotería (Waldspurger y Weihl, 1994). La idea básica consiste en dar a los procesos boletos de lotería para los diversos recursos del sistema, como el tiempo de CPU. Cada vez que se hace necesario tomar una decisión de planificación, se escoge al azar un boleto de lotería, y el proceso poseedor de ese boleto obtiene el recurso.

Cuando se aplica a la planificación del tiempo de CPU, el sistema podría realizar una lotería 50 veces por segundo, concediendo a cada ganador 20 ms de tiempo de CPU como premio. Parafraseando a George Orwell, “todos los procesos son iguales, pero algunos son más iguales que otros”.

Podemos dar más boletos a los procesos más importantes, a fin de aumentar sus posibilidades de ganar. Si hay 100 boletos pendientes, y un proceso tiene 20 de ellos, tendrá una probabilidad del 20% de ganar cada lotería. A largo plazo, obtendrá cerca del 20% del tiempo de CPU. En contraste con los planificadores por prioridad, donde es muy difícil establecer qué significa realmente tener una prioridad de 40, aquí la regla es clara: un proceso que posee una fracción de los boletos obtendrá aproximadamente una fracción f del recurso en cuestión.

La planificación por lotería tiene varias propiedades interesantes. Por ejemplo, si aparece un proceso nuevo y se le conceden algunos boletos, en la siguiente lotería ya tendrá una probabilidad de ganar que será proporcional al número de boletos que recibió. En otras palabras, la planificación por lotería es de respuesta muy rápida

Planificación en tiempo real

Un sistema de tiempo real es uno en el que el tiempo desempeña un papel esencial. Por lo regular, uno o más dispositivos físicos externos a la computadora generan estímulos, y la computadora debe reaccionar a ellos de la forma apropiada dentro de un plazo fijo. Por ejemplo, la computadora de un reproductor de discos compactos recibe los bits conforme salen de la unidad de disco y los debe convertir en música dentro de un intervalo de tiempo muy estricto. Si el cálculo toma demasiado tiempo, la música sonará raro.

Otros sistemas de tiempo real son los de monitoreo de pacientes en las unidades de cuidado intensivo de los hospitales, el piloto automático de un avión y los controles de seguridad de un reactor nuclear. En todos estos casos, obtener la respuesta correcta pero demasiado tarde suele ser tan malo como no obtenerla.

Los sistemas de tiempo real generalmente se clasifican como de tiempo real estricto, lo que implica que hay plazos absolutos que deben cumplirse a como dé lugar, y tiempo real flexible, lo que implica que es tolerable no cumplir ocasionalmente con un plazo. En ambos casos, el comportamiento de tiempo real se logra dividiendo el programa en varios procesos, cada uno de los cuales tiene un comportamiento predecible y conocido por adelantado. Estos procesos generalmente son de corta duración y pueden ejecutarse hasta terminar en menos de un segundo. Cuando se detecta un suceso externo, el planificador debe programar los procesos de modo tal que se cumplan todos los plazos.

Los sucesos a los que puede tener que responder un sistema de tiempo real pueden clasificarse también como periódicos (que ocurren a intervalos regulares) o aperiódicos (que ocurren de forma impredecible). Es posible que un sistema tenga que responder a múltiples corriente de eventos periódicos. Dependiendo de cuánto tiempo requiere cada suceso para ser procesado, tal vez ni siquiera sea posible manejarlos todos.

Por ejemplo, si hay m eventos periódicos y el evento i ocurre con el periodo P. y requiere C segundos de tiempo de CPU para ser manejado, la carga sólo podrá manejarse si Un sistema de tiempo real que satisface este criterio es panificable.
Consideremos brevemente unos cuantos algoritmos de planificación de tiempo real dinámicos. El algoritmo clásico es el algoritmo de tasa monotónica (Liu y Layland, 1973), que asigna por adelantado a cada proceso una prioridad proporcional a la frecuencia de ocurrencia de su evento disparador. Por ejemplo, un proceso que se debe ejecutar cada 20 ms recibe una prioridad de 50, y uno que debe ejecutarse cada 100 ms recibe una prioridad de 10.

En el momento de la ejecución, el planificador siempre ejecuta el proceso listo que tiene la más alta prioridad, desalojando al proceso en ejecución si es necesario. Liu y Layland demostraron que este algoritmo es óptimo. Otro algoritmo de planificación en tiempo real muy utilizado es el del primer plazo más próximo. Cada vez que se detecta un evento, su proceso se agrega a la lista de procesos listos, la cual se mantiene ordenada por plazo, que en el caso de un evento periódico es la siguiente ocurrencia del evento. El algoritmo ejecuta el primer proceso de la lista, que es el que tiene el plazo más próximo.

Un tercer algoritmo calcula primero para cada proceso la cantidad de tiempo que tiene de sobra, es decir, su holgura. Si un proceso requiere 200 ms y debe terminar en un plazo de 250 ms, tiene una holgura de 50 ms. El algoritmo, llamado de menor holgura, escoge el proceso que tiene menos tiempo de sobra. Si bien en teoría es posible convertir un sistema operativo de aplicación general en uno de tiempo real usando uno de estos algoritmos de planificación, en la práctica el gasto extra de la conmutación de contexto de los sistemas de aplicación general es tan grande que el desempeño de tiempo real sólo puede lograrse en aplicaciones con restricciones de tiempo muy holgadas. En consecuencia, en la mayor parte de los trabajos en tiempo real se usan sistemas operativos de tiempo real especiales que tienen ciertas propiedades importantes.

Por lo regular, éstas incluyen un tamaño pequeño, un tiempo de interrupción rápido, una conmutación de contexto rápida, un intervalo corto durante el cual se inhabilitan las interrupciones, y la capacidad para controlar múltiples cronómetros con precisión de milisegundos o microsegundos.

UNIDAD III:

ESQUEMAS DE ADMINISTRACIÓN DE MEMORIA: PAGINACIÓN Y SEGMENTACIÓN.

Memoria

La memoria es un bloque fundamental del computador, cuya misión consiste en almacenar los datos y las instrucciones. La memoria principal, es el órgano que almacena los datos e instrucciones de los programas en ejecución.

La memoria solo puede realizar dos operaciones básicas: lectura y escritura. En la lectura, el dispositivo de memoria debe recibir una dirección de la posición de la que se quiere extraer la información depositada previamente. En la escritura, además de la dirección, se debe suministrar la información que se desea grabar.

Existen muchos tipos de memorias:

- Memoria ROM (Read Only Memory).

Esta memoria es de solo lectura, es decir, no se puede escribir en ella. Su información fue grabada por el fabricante al construir el equipo y no desaparece al apagar el computador. Esta memoria es imprescindible para el funcionamiento de la computadora y contiene instrucciones y datos técnicos de los distintos componentes del mismo.

- Memoria RAM (Random Access Memory).

Esta memoria permite almacenar y leer la información que la CPU necesita mientras está ejecutando un programa. Además, almacena los resultados de las operaciones efectuadas por ella.

Este almacenamiento es temporal, ya que la información se borra al apagar la computadora. La memoria RAM se instala en los zócalos que para ello posee la placa base.

- Memoria cache o RAM cache.

Un caché es un sistema especial de almacenamiento de alta velocidad. Puede ser tanto un área reservada de la memoria principal como un dispositivo de almacenamiento de alta velocidad independiente. Hay dos tipos de caché frecuentemente usados en las computadoras personales: memoria caché y caché de disco. Una memoria caché, llamada también a veces almacenamiento caché o RAM caché, es una parte de memoria RAM estática de alta velocidad (SRAM) más que la lenta y barata RAM dinámica (DRAM) usada como memoria principal. El caché de disco trabaja sobre los mismos principios que la memoria caché, pero en lugar de usar SRAM de alta velocidad, usa la convencional memoria principal. (Serrano, 2012)

Intercambio

El intercambio consiste en llevar cada proceso completo a memoria, ejecutarlo durante cierto tiempo y después regresarlos al disco. Los procesos inactivos mayormente son almacenados en disco, de tal manera que no se ocupan memoria cuando no se están ejecutando. El intercambio de información es una estrategia y comúnmente más simple para lidiar con la sobrecarga de memoria. (Tanenbaum, 2009)

Memoria Virtual

La memoria virtual al igual que el intercambio es una estrategia para lidiar con la sobrecarga de memoria esta estrategia permite que cada proceso se comporte como si tuviéramos memoria ilimitada a su disposición. Para lograr esto el sistema operativo crea por cada proceso un espacio de direcciones virtual, o conocida como memoria virtual, en disco. Parte de la memoria virtual se trae a memoria principal real cuando se necesita. De esta forma, muchos procesos pueden compartir una cantidad relativamente pequeña de memoria principal. (Stalling, 2005)

Administración de memoria

Cuando la memoria se asigna en forma dinámica, el sistema operativo debe administrarla. En términos generales, hay dos formas de llevar el registro del uso de la memoria: mapas de bits y listas libres.

- Administración de memoria con mapas de bits.
  Con los mapas de bits, la memoria se divide en unidades de asignaciones tan pequeñas como unas cuantas palabras y tan grandes como varios kilobytes. 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 solo depende del tamaño de la memoria y el tamaño de la unidad de asignación. El problema principal es, el proceso de buscar en un mapa de bits una serie de cierta longitud es una operación lenta, debido a que el administrador de memoria debe de buscar en el mapa de bits una serie de k bits consecutivos con el valor 0 en el mapa de bits.

- Administración de memoria con listas ligadas.
  Cada entrada de la lista especifica un hueco (H) o un proceso (P), la dirección donde comienza, su longitud y un apuntador a la siguiente entrada. La lista de segmentos está ordenada por direcciones. Este orden tiene la ventaja de que al terminar o intercambiar un proceso, la actualización de la lista es directa. Cuando los procesos y los huecos se mantienen en una lista ordenada por direcciones, se pueden utilizar diversos algoritmos para asignar la memoria para un proceso de reciente creación o intercambio. (Anonimo, 2014)

Gestión de memoria

La gestión de memoria representa un vínculo delicado entre el rendimiento (tiempo de acceso) y la cantidad (espacio disponible). Siempre se busca obtener el mayor espacio disponible en la memoria, pero pocas veces existe la predisposición para comprometer el rendimiento. La gestión de memoria también debe realizar las siguientes funciones:

- Permitir que la memoria se comparta (en sistemas de multiprocesos).
- Asignar bloques de espacio de memoria a distintas tareas.
- Proteger los espacios de memoria utilizados (por ejemplo, evitar que un usuario modifique una tarea realizada por otro usuario).
- Optimizar la cantidad de memoria disponible, específicamente a través de sistemas de expansión de memoria. (Anonimo, Kioskea.net, 2014)

La gestión de memoria debe satisfacer los siguientes requisitos:

Reubicación

- Los programadores no saben dónde estará el programa en memoria cuando se ejecute.
- Mientras el programa se ejecuta, puede ser movido al disco y devuelto a memoria principal en una posición diferente (reubicado).
- Se deben traducir las referencias a memoria del código a las direcciones físicas reales.

Protección

Los procesos no deberían ser capaces de referenciar el espacio de memoria de otros procesos sin permiso.

- Es imposible comprobar las direcciones absolutas de los programas puesto que éstos pueden ser reubicados.
- Deben ser traducidas durante la ejecución.
- El sistema operativo no puede anticipar todas las referencias de memoria que un programa puede generar.

Compartición

- Permitir a varios procesos acceder a la misma zona de memoria.
- Es mejor permitir a cada proceso (persona) acceso a la misma copia del programa que tener cada uno su copia individual.

Organización Lógica

- Los programas son escritos en módulos.
- Los módulos se pueden escribir y compilar por separado.

- A los módulos se les puede dar diferente grado de protección (sólo lectura, sólo ejecución).
- Módulos compartidos.

Organización física

- La memoria disponible para un programa y sus datos puede ser insuficiente.
- El solapamiento permite asignar la misma zona de memoria a diferentes módulos.
- El programador no sabe cuánto espacio habrá disponible. (Stalling, 2005)

Partición

Partición es el nombre genérico que recibe cada división presente en una sola unidad física de almacenamiento de datos. Toda partición tiene su propio sistema de archivos (formato); generalmente, casi cualquier sistema operativo interpreta, utiliza y manipula cada partición como un disco físico independiente, a pesar de que dichas particiones estén en un solo disco físico.

Particionamiento Fijo

- El uso de la memoria principal es ineficiente.
- Un programa, no importa como de pequeño sea, ocupa una partición entera. Esto se conoce como fragmentación interna.

Particiones del mismo tamaño

- Cualquier proceso de tamaño menor o igual al de una partición puede ser cargado en una partición disponible.
- Si todas las particiones están ocupadas, el S.O. puede mover a disco un proceso de una partición.
- Un programa puede no caber en una partición.
- El programador debe diseñar el programa con overlays.

Algoritmo de Ubicación con Particiones

Particiones del mismo tamaño

- Como todas las particiones tienen el mismo tamaño, no importa qué partición asignar.

Particiones de diferente tamaño

- Se puede asignar a cada proceso la partición más pequeña en la que cabe – cola para cada partición.
- Los procesos se asignan de manera que se minimiza la memoria desperdiciada de una partición. (Stalling, 2005)

Particionamiento Dinámico

Las particiones se crean de forma dinámica, de tal forma que cada proceso se cargue en una partición del mismo tamaño que el procesador.

- El tamaño y el número de particiones es variable.

- Al proceso se le asignar exactamente la cantidad de memoria que necesita.
- Aparecen huecos en la memoria. Esto se conoce como fragmentación externa.
- Se debe realizar una compactación para desplazar a los procesos de forma que esté juntos y todo el espacio libre esté en un solo bloque.

Particionamiento Dinámico Algoritmo de Ubicación

El sistema operativo debe decidir qué bloque libre asignar a un proceso.
Algoritmo del mejor ajuste (best fit).

- Elige el bloque que tiene el tamaño más cercano al solicitado.
- Peor rendimiento de todos.
- Como se busca el bloque más pequeño por proceso, se produce el menor volumen de fragmentación, pero hay que compactar más a menudo.

Algoritmo del primer ajuste (first fit)

- Es el más rápido.
- Puede haber muchos procesos cargados en la zona inicial de la memoria, que debe ser examinada cuando se busca un bloque libre.

Algoritmo del siguiente ajuste (Next-fit)

- A menudo se asigna un bloque de memoria en la última parte de la memoria donde está el mayor bloque.
- El mayor bloque de memoria se parte en pequeños bloques.
- Se necesita compactar para obtener un bloque grande en la última zona final memoria.

Buddy System

La memoria disponible completa es tratada como un bloque individual de 2 U. Si una petición de tamaño s es tal que 2 < s <= 2, U-1 U se le asigna el bloque completo

- Si no, el bloque se divide en dos trozos iguales (buddies).
- El proceso continúa hasta generar el bloque más pequeño que es mayor o igual a s.

Direcciones

Lógicas

- Referencias a posiciones de memoria independientes de la asignación vigente de datos en memoria.
- La traducción se realiza a dirección física.

Relativas

- Las direcciones se expresan como posiciones relativas a algún punto conocido.

Físicas

- Es la dirección absoluta o ubicación real en memoria principal.

Paginación

- Partición de la memoria en pequeños pedazos del mismo tamaño (chunks) y dividir cada proceso en trozos del mismo tamaño.
- Los trozos (chunks) de un proceso se llaman páginas y los de la memoria se llaman marcos de página (frames).
- El sistema operativo mantiene una tabla de página para cada proceso.

- Contiene la ubicación del marco de página (frame) de cada página del proceso.
- La dirección de memoria consiste en un número de página y un desplazamiento (offset) dentro de la página. (Stalling, 2005)

Segmentación

- Todos los segmentos de todos los programas no tienen por qué ser del mismo tamaño.
- Hay un tamaño máximo de segmento.
- El direccionamiento consta de dos partes.
- Un número de segmento y un desplazamiento dentro de éste (offset).
- Como los segmentos no son iguales, la segmentación es similar al particionamiento dinámico. (Stalling, 2005)

UNIDAD IV:

MECANISMOS DE BLOQUEOS IRREVERSIBLES ENTRE PROCESOS

Bloqueo irreversible.

Un conjunto de procesos cae en bloqueo irreversibles cada proceso del conjunto está esperando un suceso que otro proceso puede causar, hay bloqueos irreversibles a nivel proceso o a nivel hardware.

Puede ser:

Hardware: un dispositivo de entrada y salida.

Software: información bases de datos archivos registros.

Existen recursos de tipo:

- Expropiativo que consta en se le puede quitar el recurso a quien lo tiene sin hacerle daño ya que cuando un proceso este levantado este no podrá terminarlo hasta que el otro termine su ejecución. Ejemplo la memoria se puede expropiar respaldándose en disco a quien la tenía.
- No expropiativo este recurso se le puede quitar el recurso y no falla el proceso que seria que cuando un proceso está levantado y llega otro proceso este automáticamente lo termina aunque el otro no haya concluido con su ejecución. Un ejemplo de esto puede es una impresora o un escáner se puede quitar el recurso y no pasa nada.

Normalmente los bloqueos irreversibles se dan en los recursos no expropiables.

La secuencia de uso de un recurso.

Es la siguiente.

- Se solicita el recurso.
- Si está disponible.
- Lo usa.
- Lo libera.

Si no está disponible:

- Devuelve un error.
- Se bloquea.
- Lo intenta más tarde.

Características del bloqueo irreversible.

- para que exista un bloqueo irreversible los procesos solo tienen un sub proceso.
- No hay interrupciones que activen a un bloqueo irreversible.

Condiciones para que exista in bloqueo irreversible.

Condiciones para manejar el bloqueo irreversible

1.-Exclusion mutua.

Cada recurso está asignado únicamente a un solo proceso o está
Disponible.

Un recurso está asignado a un proceso libre.

- Retención y espera.

Si ya tienen procesos pueden pedir más.

- De no expropiación.
- No pueden arrebatárseles, antes deben ser liberados.
- De espera circular.

Cadena circular de dos o más procesos.

Exclusión mutua.

No asignar en forma exclusiva todos los recursos, asignar un recurso hasta que sea estrictamente necesario, que la cantidad de procesos que soliciten cierto recurso sea la menor posible.

2.-Retencion y espera.

Una segunda técnica es la detección y recuperación. Cuando se usa esta técnica, el sistema no hace otra cosa que no sea vigilar las peticiones y liberaciones de recursos. Cada vez que un recurso se solicita o libera, se actualiza el grafo de recursos, y se determina si contiene algún ciclo. Si se encuentra uno, se termina uno de los procesos del ciclo. Si esto no rompe el bloqueo mutuo, se termina otro proceso, continuando así hasta romper el ciclo, Un método un tanto más burdo consiste en no mantener siquiera el grafo de recursos, y en vez de ello verificar periódicamente si hay procesos que hayan estado bloqueados continuamente durante más de, digamos, una hora. A continuación se terminan esos procesos.

La detección y recuperación es la estrategia que a menudo se usa en las macro computadoras, sobre todo los sistemas por lotes en los que terminar y luego reiniciar un proceso suele ser aceptable. Sin embargo, se debe tener cuidado de restaurar todos los archivos modificados a su estado original, y revertir todos los demás efectos secundarios que pudieran haber ocurrido.

3.-De no expropiación.

No es posible quitarle por la fuerza a un proceso los recursos que le fueron otorgados previamente. El proceso que los tiene debe liberarlos explícitamente.

4.-Espera circular.

Debe haber una cadena circular de dos o más procesos, cada uno de los cuales están esperando un recurso retenido por el siguiente miembro de la cadena. Deben estar presentes estas cuatro condiciones para que ocurra un bloqueo mutuo. Si una o más de estas Condición de espera circular. Es que un proceso solo pueda tener un recurso a la vez, al solicitar uno, que libere otro (imposible), numerar los recursos, y que los procesos al solicitarlos los vayan tomando en orden ascendente. Otros aspectos a tomar en cuenta:

- Bloqueos de dos fases sobre todo en bases de datos.
- Primera fase bloqueos de registros hasta que pueda bloquearlos todos.
- Segunda fase actualización.
- Bloqueos irreversibles que no son por recursos.
- Puede ser solo entre procesos.
- Que uno espere a que otro lo despierte.
- Caso productor consumidor.
- Inanición.

Puede haber proceso que no estén en un bloqueo irreversible y aun así casi nunca son atendidos por lo que mueren de inanición una solución es usar un algoritmo d calendarización que lo evite.

Estrategias para el bloqueo irreversibles

El bloqueo mutuo se evitaba no imponiendo reglas arbitrarias a los procesos sino analizando con detenimiento cada petición de recurso para ver si se puede satisfacer sin peligro. Surge la pregunta: ¿hay algún algoritmo que siempre pueda evitar el bloqueo mutuo tomando la decisión correcta en todos los casos? La respuesta es que sí se puede evitar el bloqueo mutuo, pero sólo si se cuenta con cierta información por adelantado. En esta sección examinaremos formas de evitar los bloqueos mutuos mediante una asignación cuidadosa de los recursos y una de las estrategias serían las siguientes:

1.- Ignorar el problema.

2.- Denar que sucedan, detectarlos y recuperarse de ellos.

3.- Evitar que sucedan, con una asignación cuidadosa del recurso.

4.- Prevención, anulado una de la cuatro condiciones para evitar un bloqueo irreversible.

Algoritmo de avestruz

Es la estrategia más sencilla es el algoritmo del avestruz: meter la cabeza en la arena y pretender que el problema no existe. La gente reacciona a esta estrategia de diversas maneras. Los matemáticos la encuentran totalmente inaceptable y dicen que los bloqueos mutuos deben prevenirse a toda costa.

Los ingenieros preguntan con qué frecuencia se espera que se presente el problema, qué tan seguido se cae el sistema por otras razones, y qué tan grave es un bloqueo mutuo. Si ocurren bloqueos mutuos una vez cada 50 años en promedio, pero las caídas del sistema debido a fallas de hardware, errores del compilador y defectos del sistema operativo ocurren una vez al mes, la mayoría de

los ingenieros no estarían dispuestos a pagar un precio sustancial en términos de reducción del rendimiento o de la comodidad a fin de evitar los bloqueos mutuos.
En un sistema real los bloqueos irreversibles no son tan comunes, en la mayoría de los sistemas operativos pueden tener bloqueos irreversibles que ni siquiera se detectan. Se tiene limitante que pueden originar un bloqueo irreversible. Cantidad de procesos y sub procesos que pueden estar activos al mismo tiempo. Número máximo de archivos abiertos.

Algoritmo Excepción Dinámica

Detección de bloqueos irreversibles con un recurso de cada tipo solo hay un recurso de cada tipo, un sistema con estas condiciones podría tener un escáner o un DVD ROM o impresora. Detección de bloqueos irreversibles con múltiples recursos si hay varias copas los recursos se requiere un enfoque para detectar se basa en una comparación de vectores.

Prevención

Evitar los bloqueos irreversibles son procesos irreales ya que consisten en evitar que al menos uno de los 4 básicas no se cumpla.

El espacio de intercambio en disco.

Windows y Linux los ignoran, prefieren un bloqueo irreversible de vez en cuando a que haya limitantes con los recursos.

Detección de bloqueos irreversibles y recuperación posterior.

No se intenta prevenir, solo esperar a que suceda, detectarlo y recuperarse del bloqueo.

Detección de bloqueo irreversible con un recurso de cada tipo.

-Solo hay un recurso de cada tipo.

-Si hay uno o más ciclos, existe un bloqueo irreversible si no, no lo hay

Como recuperarse de un bloqueo irreversible.

1.- Mediante la expropiación.

Consiste en quitarle el recurso un proceso y después reanudarlo.

Depende del recurso, de preferencia un recurso expropiable.

2.- Por eliminación de procesos.

Elimina un proceso, sino lo rompe, se elimina otro y así sucesivamente.

Podría ser uno que no esté en el ciclo, uno de baja prioridad, uno que pueda reiniciarse.

3.-Mediante reversión.

- Estableciendo puntos de verificación.
- Deben ser archivos nuevos constantemente creados.
- El punto de verificación contiene, la memoria, el estado de los recursos y procesos.

Al detectar un bloqueo irreversible se busca un punto de verificación antes de haber solicitado el recurso para reanudar a partir de ahí.

Como evitar los bloqueos irreversibles.

- Objetivo evitar los bloqueos irreversibles, mediante la asignación cuidadosa de recursos.
- Sin tener que darles todos los recursos que requiere, como el método anterior.

Estados seguros e inseguros.

Estado seguro.

Cuando no ha caído un bloqueo irreversible, existe algún orden de calendarización en el cual todos los procesos pueden ejecutarse hasta terminar. En un estado seguro no se ofrece garantía que los procesos terminen, un estado inseguro no necesariamente es un bloqueo irreversible.

Algoritmo del banquero para un solo recurso:

- Se examina cada solicitud al momento que se hace y se analiza si esto llevara a un estado seguro, si es así lo concede si no lo manda a espera.

Para ver si su estado es seguro.

- Checa si hay suficientes recursos para satisfacer el proceso.
- Verifica que proceso está más cercano al límite.

Al igual que el caso de un solo recurso, se requiere conocer las necesidades totales de los procesos antes de ejecutarse, requieren que la cantidad de procesos sea fija, los recursos pueden aparecer y desparecer en cualquier momento, no se usa en la práctica.

Prevención de bloqueos irreversibles.

Como los mecanismos de bloqueos irreversibles son poco irreales y es difícil caer en un bloqueo irreversible, se opta por la prevención.

Consiste en evitar que al menos una de las cuatro condiciones básicas no se cumpla.

UNIDAD V:

ADMINISTRACIÓN DE LOS ARCHIVOS EN LOS DIFERENTES SISTEMAS OPERATIVOS

Se tienen 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.

Archivos

Desde el punto de vista que tiene el usuario acerca del sistema de archivos.

Nomenclatura de archivos

Los archivos son un mecanismo de abstracción. Proporcionan una manera de almacenar información en el disco y leerla después. Pero para ello el sistema debe de poder identificarlos, todos los sistemas operativos actuales permiten cadenas de una a ocho letras, dígitos y caracteres especiales como nombres de archivos legales.

Después del nombre del archivo lleva un punto y algunas letras a las cuales se les conocen como extensión del archivo y por lo general indica algo acerca de su naturaleza.

Algunas de las extensiones de archivos más comunes y sus significados son las siguientes.

Extensión	**Significado**
archivo.bak	Archivo de respaldo
archivo.c	Programa fuente en C
archivo.gif	Imagen en Formato de Intercambio de Gráficos de CompuServe
archivo.hlp	Archivo de ayuda
archivo.html	Documento en el lenguaje de Marcación de Hipertexto de www
archivo.jpg	Imagen fija codificada con el estándar JPEG
archivo.mp3	Música codificada en formato de audio MPEG capa 3
archivo.mpg	Película codificada con el estándar MPEG
archivo.o	Archivo objeto
archivo.pdf	Archivo en formato de documento portable
archivo.ps	Archivo de PostScript
archivo.tex	Entrada para el programa formateador TEX
archivo.txt	Archivo de texto general
archivo.zip	Archivo comprimido

En algunos sistemas (como UNIX) las extensiones de archivo son solo convenciones y no son impuestas por los sistemas operativos. Por lo contrario Windows está consciente de las extensiones y les asigna significado.

Estructura de archivos

Tres posibilidades comunes de describir la estructura de los archivos es:

1. El sistema operativo no sabe, ni le importa, qué hay en el archivo. Todo lo que ve son bytes. Tanto UNIX, MS-DOS y Windows utilizan esta metodología. El sistema operativo no ayuda pero no estorba.
2. En otro modelo un archivo es una secuencia de registros de longitud fija, cada uno con uno con cierta estructura interna. La idea de esto es que la operación de lectura devuelva un registro y la operación de escritura sobrescriba o agregue un registro.
3. En otra organización, un archivo consiste de un árbol de registros, donde no todos son de la misma longitud; cada uno de ellos contiene un campo llave en una posición fija dentro del registro. El árbol se organiza con base en el campo llave para permitir una búsqueda rápida por una llave específica.

Tipos de archivos

Archivos regulares los que contienen información del usuario. Los directorios son sistemas de archivos para mantener la estructura del sistema de archivos. Por lo general estos archivos son ASCII (consisten en líneas de texto) o binarios.

Archivos especiales de caracteres se relacionan con la entrada/salida y se utilizan para modelar dispositivos de entrada/salida en serie, tales como terminales, impresoras y redes.

Archivos especiales de bloques se utilizan para modelar discos.

Acceso a archivos

Existen dos formas de acceso a los archivos.

1. **Archivos de acceso secuencial:** en estos sistemas, un proceso puede leer todos los bytes o registros en un archivo en orden, empezando desde el principio, pero no puede saltar alguno y leerlos fuera de orden.

2. **Archivos de acceso aleatorio:** son los archivos cuyos bytes o registros se pueden leer en cualquier orden. Estos archivos son esenciales para muchas aplicaciones, como los sistemas de base de datos.

Atributos de archivos

Todo archivo tiene un nombre y sus datos, a estos datos se les llamaran atributos del archivo o metadatos. La lista de atributos varía considerablemente de un sistema a otro. A continuación se muestran algunas posibilidades de atributos.

Atributo	**Significado**
Protección	Quién tiene acceso al archivo y en qué forma
Contraseña	Contraseña necesaria para acceder al archivo
Creador	ID de la persona que creó el archivo
Propietario	El propietario actual
Bandera de sólo lectura	0 para lectura/escritura; 1 para sólo lectura
Bandera oculto	0 para normal; 1 para que no aparezca en los listados
Bandera del sistema	0 para archivos normales; 1 para archivo del sistema
Bandera de archivo	0 si ha sido respaldado; 1 si necesita respaldarse
Bandera ASCII/binario	0 para archivo ACII; 1 para archivo binario
Bandera de acceso aleatorio	0 para solo acceso secuencial; 1 para acceso aleatorio
Bandera temporal	0 para normal; 1 para eliminar archivos al salir del proceso
Banderas de bloqueo	0 para desbloqueado; distinto de 0 para bloqueado
Longitud de registro	Número de bytes en un registro

Posición de la llave	Desplazamiento de la llave dentro de cada registro
Hora de creación	Fecha y hora en la que se creó el archivo
Hora de último acceso	Fecha y hora en que se accedió al archivo por última vez
Hora de la última modificación	Fecha y hora en que se modificó por última vez el archivo
Tamaño actual	Número de bytes en el archivo
Tamaño máximo	Número de bytes hasta donde puede crear el archivo
Longitud de llave	Número de bytes en el campo llave

Operaciones de archivos

Distintos sistemas proveen diferentes operaciones para permitir el almacenamiento y la recuperación. A continuación se muestra las llamadas al sistema más comunes.

1. Crear. El archivo se crea sin datos. El propósito de la llamada es anunciar la llegada del archivo y establecer algunos de sus atributos.
2. Borrar. Cuando el archivo ya no se necesita, se tiene que eliminar para liberar espacio en el disco.
3. Abrir. El propósito de la llamada a open es permitir que el sistema lleve los atributos y la lista de direcciones de disco a memoria principal para tener un acceso rápido a estos datos en llamadas posteriores.
4. Cerrar. Cuando terminan todos los accesos, los atributos y las direcciones de disco ya no son necesarias, por lo que el archivo se debe cerrar para liberar espacio en la tabla interna.
5. Leer. El llamador debe especificar cuántos datos se necesitan y también debe proporcionar un buffer para colocarlos.

6. Escribir. Los datos se escriben en el archivo otra vez por lo general en la posición actual. Si la posición actual es al final del archivo, aumenta su tamaño.
7. Buscar. Reposiciona el apuntador del archivo en una posición especifica del archivo. Una vez que se completa esta llamada se pueden leer o escribir datos en esa posición.
8. Renombrar. Con frecuencia ocurre que un usuario necesita cambiar el nombre de un archivo existente.

Cada archivo abierto tiene asociados varios elementos de información.

1. Puntero al archivo: el sistema debe seguir la pista de la última posición de lectura/escritura con un puntero a la posición actual en el archivo. Este puntero es exclusivo para cada proceso que está trabajando con el archivo, por lo que debe mantenerse a parte de los atributos del archivo en disco.
2. Contador de aperturas del archivo: este contador sigue la pista al número de aperturas y cierres, y llega a cero después del último cierre.
3. Ubicación del archivo en disco: la información necesaria para localizar el archivo en disco se mantiene en la memoria para no tener que leerla del disco en cada operación.

Directorios

Algunos sistemas almacenan miles de archivos de cientos de gigabytes de disco. Para administrar todos estos datos se necesitan organizar. Esta organización por lo regular se efectúa en dos partes:

1. El sistema de archivos se divide en particiones. Típicamente cada disco de un sistema contiene al menos una partición. A veces se usan particiones para contar con varias áreas independientes dentro de un disco.

2. Cada partición contiene información acerca de los archivos que hay en ella. Esta información se mantiene como entradas de un directorio de dispositivo o tabla de contenido del volumen. El directorio del dispositivo registra información como nombre, ubicación, tamaño y tipo de todos los archivos de esa partición.

Operaciones que se realizan con un directorio

- Buscar un archivo.
- Crear un archivo.
- Eliminar un archivo.
- Cambiar el nombre de un archivo.
- Recorrer el sistema de archivos.
- Listar un directorio.

Sistemas de directorios de un solo nivel

La forma más simple de un sistema de directorios es tener un directorio que contenga todos los archivos, algunas veces se le llama directorio raíz. A menudo se utilizan en dispositivos incrustados simples como teléfonos, cámaras digitales y algunos reproductores de música portátil.

Sistemas de directorios jerárquicos

Con este esquema, puede haber tantos directorios como se necesite para agrupar los archivos en formas naturales. La capacidad de los usuarios para crear un número arbitrario de subdirectorios provee una poderosa herramienta de estructuración para que los usuarios organicen su trabajo. Por esta razón, casi todos los sistemas de archivos modernos se organizan de esta manera.

Nombre de rutas

Cuando el sistema de archivos está organizado como un árbol de directorios, se necesita cierta forma de especificar los nombres de los archivos. Por lo general se utilizan dos métodos distintos. En el primer método, cada archivo recibe un nombre de ruta absoluto que consiste en la ruta desde el directorio raíz al archivo. Los nombres de ruta absolutos siempre empiezan en el directorio raíz y son únicos. Sin importar cuál carácter se utilice, si el primer carácter del nombre de la ruta es el separador, entonces la ruta es absoluta.

El otro tipo de nombre es el nombre de ruta relativa. Éste se utiliza en conjunto con el concepto del directorio de trabajo (también llamado directorio actual). Todos los nombres de las rutas que no empiecen en el directorio raíz se toman en forma relativa al directorio de trabajo.

Cada proceso tiene su propio directorio de trabajo, por lo que cuando éste cambia y después termina ningún otro proceso se ve afectado y no quedan rastros del cambio en el sistema de archivos.

1. Create. Se crea un directorio. Está vacío, excepto por punto y punto, que el sistema coloca ahí de manera automática.
2. Delete. Se elimina un directorio. Se puede eliminar sólo un directorio vacío.
3. Opendir. Los directorios se pueden leer. Antes de poder leer un directorio se debe abrir, en forma análoga al proceso de abrir y leer un archivo.
4. Closedir. Cuando se ha leído un directorio, se debe serrar para liberar espacio en la tabla interna.
5. Readdir. En contraste, readdir siempre devuelve una entrada en formato estándar, sin importar cuál de las posibles estructuras de directorio se utilice.
6. Rename. En muchos aspectos, los directorios son sólo como archivos y se les puede cambiar el nombre de la misma forma que a los archivos.
7. Link. La vinculación (ligado) es una técnica que permite a un archivo aparecer en más de un directorio.

8. Unlink. se elimina una entrada de directorio. Si el archivo que se va a desvincular sólo está presente en un directorio, se quita del sistema de archivos. Si está presente en varios directorios, se elimina sólo el nombre de ruta especificado.

Implementación de sistemas de archivos

Ahora se cambiara del punto de vista que tiene el usuario acerca del sistema de archivos, al punto de vista del que lo implementa.

Distribución del sistema de archivos

A menudo el sistema de archivos contiene algunos de los siguientes elementos.

El primero es el superbloque. Contiene todos los parámetros clave acerca del sistema de archivos y se lee en la memoria cuando se arranca la computadora o se entra en contacto con el sistema de archivos por primera vez. La información típica en el superbloque incluye un número mágico para identificar el tipo de sistema de archivos, el número de bloque que contiene el sistema de archivos y otra información administrativa clave.

A continuación podría venir información acerca de los bloque libres en el sistema de archivos, por ejemplo en la forma de un mapa de bits o una lista de apuntadores. Éste podría ir seguida de los nodos i, un arreglo de estructuras de datos, uno por archivo, que indica todo acerca del archivo. Después de eso podría venir el directorio raíz, que contiene la parte superior del árbol del sistema de archivos. Por último, el resto del disco contiene todos los otros directorios y archivos.

Implementación de archivos

Probablemente la cuestión más importante al implementar el almacenamiento de archivos sea mantener un registro acerca de qué bloques de discos van con cual archivo. Se utilizan varios métodos en distintos sistemas operativos.

Asignación contigua

El esquema de asignación más simple es almacenar cada archivo como una seria contigua de bloques de disco. La asignación de espacio en disco contiguo tiene dos ventajas significativas. En primer lugar es simple de implementar, ya que llevar un registro de la ubicación de los bloques de un archivo se reduce a recordar dos números: la dirección de disco del primer bloque y el número de bloque en el archivo.

En segundo lugar, el rendimiento de la lectura es excelente debido a que el archivo completo se puede leer del disco en una sola operación. Por desgracia la asignación contigua también tiene una desventaja: con el transcurso del tiempo, los discos se fragmentan. Cuando se quita un archivo los bloques se liberan naturalmente, dejando una serie de bloques libres en el disco. Hay una situación en la que es factible la asignación contigua y de hecho, se utiliza ampliamente: en los CD-ROMs.

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 actualiza como apuntador al siguiente. El resto del bloque es para los datos. A diferencia de la asignación contigua, este método se puede utilizar cada bloque del disco.

No se pierde espacio debido a la fragmentación del disco. Además, para la entrada del directorio sólo le basta con almacenar la dirección de disco del primer bloque. Por otro lado, aunque la lectura secuencial a un archivo es directa, el acceso aleatorio es en extremo lento.

Asignación de lista enlazada utilizando una tabla de memoria

Ambas desventajas de la asignación de lista enlazada se puede eliminar si tomamos la palabra del apuntador de cada bloque de disco y la colocamos en una tabla en memoria. Dicha tabla en memoria principal se conoce como FAT (File Allocation Table, tabla de asignación de archivos).

Utilizando esta organización, el bloque completo está disponible para los datos. Además, el acceso aleatorio es mucho más sencillo. La principal desventaja de este método es que toda la tabla debe estar en memoria todo el tiempo para que funcione.

Nodos-i

Nuestro último método para llevar un registro de qué bloques pertenecen a cuál archivo es 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 bloque del archivo. La gran desventaja de este esquema en comparación con los archivos vinculados que utilizan una tabla en memoria, es que el nodo-i necesita estar en memoria sólo cuando está abierto el archivo correspondiente.

En contraste, el esquema del nodo-i requiere un arreglo en memoria cuyo tamaño sea proporcional al número máximo de archivos que pueden estar abiertos a la vez.

Implementación de directorios

Antes de poder leer un archivo éste debe abrirse. Cuando se abre un archivo, el sistema operativo utiliza el nombre de la ruta suministrado por el usuario para localizar la entrada de directorio. Esta entrada provee la información necesaria para encontrar los bloques de disco.

Cada sistema de archivo mantiene atributos de archivo, como el propietario y la hora de creación de cada archivo, debiendo almacenarse en alguna parte. Una posibilidad obvia es almacenarlos directamente en la entrada de directorio.

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

Casi todos los sistemas operativos modernos aceptan nombres de archivos largos, con longitud variable, el esquema más simple para almacenarlos es establecer un límite en la longitud del nombre de archivo que por lo general es de 255 caracteres. Este esquema es simple pero desperdicia mucho espacio de directorio, ya que pocos archivos tienen nombres tan largos.

Una alternativa es renunciar a la idea de que todas las entradas de directorio sean del mismo tamaño. Con este método, cada entrada de directorio contiene una porción fija, que por lo general empieza con la longitud de la entrada y después va seguida de datos con un formato fijo, que comúnmente incluyen el propietario, la hora de creación, información de protección y demás atributos.

Una desventaja de este método es que cuando se elimina un archivo, en su lugar queda un hueco de tamaño variable dentro del directorio, dentro del cual el siguiente archivo a introducir puede que no quepa.

Otra manera de manejar los nombres de longitud variable es hacer que las mismas entradas de directorio sean de longitud fija y mantener los nombres de los archivos juntos en un heap al final del directorio. Este método tiene la ventaja de que cuando se remueva una entrada, el siguiente archivo a introducir siempre cabrá ahí.

En todos los diseños mostrados hasta ahora se realizan búsquedas lineales en los directorios de principio a fin cuando hay que buscar el nombre un archivo. Para los directorios en extremo largos, la búsqueda lineal puede ser lenta. Una manera de acelerar la búsqueda es utilizar una tabla de hash en cada directorio. El uso de una tabla de hash tiene la ventaja de que la búsqueda es mucho más rápida, pero la desventaja de una administración más compleja.

Una manera distinta de acelerar la búsqueda en directorios extensos es colocar en caché los resultados de las búsquedas. Antes de iniciar una búsqueda, primero se realiza una verificación para ver si el nombre del archivo está en la caché. De ser así se puede localizar de inmediato. Desde luego el uso de la caché sólo funciona si un número relativamente pequeño de archivos abarcan la mayoría de las búsquedas.

Archivos compartidos

Cuando hay varios usuarios trabajando en conjunto en un proyecto, a menudo necesitan compartir archivos. Como resultado, con frecuencia es conveniente que aparezca un archivo compartido en forma simultánea en distintos directorios que pertenezcan a distintos usuarios. La conexión entre el directorio x y el archivo compartido se conoce como un vínculo (liga). El sistema de archivos en sí es ahora un gráfico acíclico dirigido (Directed Acyclic Graph, DAG) en vez de un árbol.

Pero también introduce ciertos problemas. Para empezar, si los directorios en realidad contienen direcciones de disco, entonces habrá que realizar una copia de las direcciones de disco en el directorio x cuando se ligue el archivo.

Este problema se puede resolver de dos formas. En la primera solución, los bloques de disco no se listan en los directorios, sino que en una pequeña estructura de datos asociada con el archivo en sí. Entonces, los directorios apuntarían sólo a la pequeña estructura de datos.

En la segunda solución, x se vincula a uno de los archivos de y haciendo que el sistema cree una archivo, de tipo link introduciendo ese archivo en el directorio de x. El nuevo archivo contiene sólo el nombre de la ruta del archivo el cual está vinculado. Cuando x lee del archivo vinculado, el sistema operativo ve que el archivo del que se están leyendo datos es de tipo link, busca el nombre del archivo y lee el archivo. A este esquema se le conoce como vínculo simbólico (liga simbólica).

Al crear un vínculo no se cambia la propiedad, sino incrementa la cuenta de vínculos en el nodo-i, por lo que el sistema sabe cuántas entradas de directorio actualmente apuntan al archivo. Al eliminar un vínculo simbólico, el archivo no se ve afectado.

El problema con los vínculos simbólicos es el gasto adicional de procesamiento requerido. Se debe leer el archivo que contiene la ruta, después ésta se debe analizar sintácticamente y seguir, componente por componente, hasta llegar al nodo-i. Toda esta actividad puede requerir una cantidad considerable de acceso adicional al disco.

Sistemas de archivos estructurados por registro

LFS (Log-structured File System, Sistema de archivos estructurado por registro). La idea básica es estructurar todo el disco como un registro. De manera periódica, y cuando haya una necesidad especial para ello, todas las escrituras pendientes que se colocaron en el búfer en memoria se recolectan en un solo segmento y se

escriben en el disco como un solo segmento continuo al final del registro. Por lo tanto, un solo segmento puede contener nos-i, bloques de directorio y bloques de datos, todos mezclados entre sí.

Para que sea posible encontrar nodos-i, se mantiene un mapa de nodos-i, indexados por número-i. La entrada i en este mapa apunta al nodo-i i en el disco. El mapa se mantiene en el disco, pero también se coloca en la caché, de manera que las partes más utilizadas estén en memoria la mayor parte del tiempo. (Tanenbaum, 2009)

Protección

Cuando se guarda información en un sistema de computación una preocupación importante es su protección tanto de daños físicos como de un acceso indebido. La confiabilidad se logra duplicando los archivos.

Tipos de acceso

Lo que se necesita en un sistema multiusuario es un acceso controlado. Los mecanismos de protección proporcionan un acceso controlado limitando las formas en que se puede acceder a los archivos. Pueden controlarse varios tipos de operaciones distintas como:

- Leer.
- Escribir.
- Ejecutar.
- Anexar.
- Eliminar.
- Listar.

La protección sólo puede proporcionarse en el nivel más bajo.

Listas y grupos de acceso

La forma más común de abordar el problema de la protección es hacer que el acceso dependa de la identidad del usuario. El esquema más general para implementar el acceso independiente de la identidad es asociar a cada archivo y directorio a una lista de acceso que especifique el nombre del usuario y los tipos de acceso que se permiten a cada usuario.

Para condensar la lista de acceso, muchos sistemas reconocen tres categorías de usuarios en relación con cada archivo:

1. Propietario: el usuario que creó el archivo.
2. Grupo: un conjunto de usuarios que están compartiendo el archivo.
3. Universo: todos los demás usuarios del sistema.

El sistema Unix define tres campos de tres bits cada uno: r w x, donde r controla el acceso de lectura, w el acceso de escritura y x controla la ejecución.

Implementación del sistema de archivos

El sistema de archivos reside de manera permanente en almacenamiento secundario, cuyo requisito principal es que debe poder contener una gran cantidad de datos permanentemente.

Estructura del sistema de archivos

Para mejorar la eficiencia de E/S, las transferencias entre la memoria y el disco se efectúan en unidades de bloques. Cada bloque ocupa uno o más sectores. Dependiendo de la unidad de disco, el tamaño de los sectores varía entre 32 bytes y 4096 bytes, aunque por lo regular es de 512 bytes. Los discos tienen dos características importantes que los convierten en un medio cómodo para almacenar muchos archivos.

1. Se puede reescribir en el mismo lugar.
2. Se puede acceder directamente a cualquier bloque de información del disco.

Organización del sistema de archivos

Para ofrecer un acceso eficiente y cómodo al disco, el sistema operativo impone en él un sistema de archivos que permita almacenar, encontrar y recuperar con facilidad los datos.

Un sistema de archivos presenta dos problemas de diseño.

1. Definir qué aspecto debe presentar el sistema de archivos a los usuarios.
2. Hay que crear algoritmos y estructuras de datos que establezcan una correspondencia entre el sistema de archivos lógico y los dispositivos de almacenamiento secundario físico.

El sistema de archivos en si generalmente se compone de muchos niveles diferentes, ejemplo:

Programas de aplicación.

Sistema de archivos lógicos.

Módulo de organización de archivos.

Sistema de archivos básicos.

Control de E/S.

Dispositivos.

El nivel más bajo, el control E/S, consta de drivers o controladores de dispositivos y manejadores de interrupciones para transferir información entre la memoria y el sistema de disco. Se puede considerar un driver de dispositivo como un traductor.

El driver de dispositivo por lo regular escribe patrones de bits específicos en posiciones especiales de la memoria del controlador de E/S para decirle a éste sobre qué posición del dispositivo debe actuar y qué acciones debe emprender. El sistema de archivos básico sólo necesita emitir órdenes genéricas al driver de dispositivo apropiado para leer y escribir bloque físicos en el disco.

El módulo de organización de archivos conoce los archivos y sus bloques lógicos, además de los bloque físicos. Sabiendo el tipo de asignación de archivos empleada y la ubicación del archivo, el módulo de organización de archivos puede traducir las direcciones de bloque lógicos en direcciones de bloque físicos para que el sistema de archivos básico realice la transferencia. El módulo de organización de archivos también incluye el administrador de espacio libre, que sigue la pista a los bloques no asignados y los proporciona al módulo de organización de archivos cuando se le solicita.

El sistema de archivos lógico emplea la estructura de directorios para proporcionar al módulo de organización de archivos la información que éste necesita, a partir de un nombre de archivo simbólico. El sistema de archivos lógico también se encarga de la protección y la seguridad. (Silberschatz, 1999, págs. 337-372)

Tipos de sistemas de archivos

BTRFS

Btrfs (B-tree FS o normalmente pronunciado "Butter FS") es un sistema de archivos copy-on-write anunciado por Oracle Corporation para GNU/Linux. Es un nuevo sistema de archivos con potentes funciones, similares al excelente ZFS de Sun/Oracle. Estas incluyen la creación de instantáneas, striping y mirroring multi-disco (RAID software sin mdadm), sumas de comprobación, copias de seguridad incrementales, y compresión sobre la marcha integrada, que pueden dar un significativo aumento de las prestaciones, así como ahorrar espacio.

Ext2

Second Extended Filesystem es un consolidado y maduro sistema de archivos para GNU/Linux muy estable. Uno de sus inconvenientes es que no tiene apoyo para el registro (journaling). La falta de *registro por diario* («journaling») puede traducirse en la pérdida de datos en caso de un corte de corriente o fallo del sistema. También puede no ser conveniente para las particiones root (/) y /home, porque las comprobaciones del sistema de archivos pueden tomar mucho tiempo. Un sistema de archivos ext2 puede ser convertido a ext3.

Ext3

Ext3 (third extended filesystem o tercer sistema de archivos extendido) se diferencia de ext2 en que trabaja con registro por diario (journaling) y porque utiliza un árbol binario balanceado (árbol AVL, creado por los matemáticos rusos Georgii Adelson-Velskii y Yevgeniy Landis) y también por incorporar el método Orlov de asignación para bloques de disco. Además ext3 permite ser montado y utilizado como si fuera ext2 y actualizar desde ext2 hacia ext3 sin necesidad de formatear la partición y sin perder los datos almacenados en ésta.

Ext4

Ext4 (fourth extended filesystem o cuarto sistema de archivos extendido) es un sistema de archivos con registro por diario, publicado por Andrew Morton como una mejora compatible con el formato Ext3. Las mejoras respecto de Ext3 incluyen, entre otras cosas, el soporte de volúmenes de hasta 1024 PiB, soporte añadido de extends (conjunto de bloques físicos contiguos), menor uso de recursos de sistema, mejoras sustanciales en la velocidad de lectura y escritura y verificación más rápida con fsck. Es el sistema de archivos predeterminado en CentOS 6 y Red Hat™ Enterprise Linux 6.

Acerca del registro por diario (journaling).

El registro por diario (journaling) es un mecanismo por el cual un sistema de archivos implementa transacciones. Consiste en un registro en el que se almacena la información necesaria para restablecer los datos dañados por una transacción en caso de que ésta falle, como puede ocurrir durante una interrupción de energía.

FAT

FAT es con diferencia el sistema de archivos más simple de aquellos compatibles con Windows NT. El sistema de archivos FAT se caracteriza por la tabla de asignación de archivos (FAT), que es realmente una tabla que reside en la parte más "superior" del volumen. Para proteger el volumen, se guardan dos copias de la FAT por si una resultara dañada. Además, las tablas FAT y el directorio raíz deben almacenarse en una ubicación fija para que los archivos de arranque del sistema se puedan ubicar correctamente.

Ventajas de FAT

No es posible realizar una recuperación de archivos eliminados en Windows NT en ninguno de los sistemas de archivos compatibles. Las utilidades de recuperación de archivos eliminados intentan tener acceso directamente al hardware, lo que no se puede hacer en Windows NT. Sin embargo, si el archivo estuviera en una partición FAT y se reiniciara el sistema en MS-DOS, se podría recuperar el archivo. El sistema de archivos FAT es el más adecuado para las unidades y/o particiones de menos de 200 MB aproximadamente, ya que FAT se inicia con muy poca sobrecarga.

Desventajas de FAT

Cuando se utilicen unidades o particiones de más de 200 MB, es preferible no utilizar el sistema de archivos FAT. El motivo es que a medida que aumente el tamaño del volumen, el rendimiento con FAT disminuirá rápidamente. No es posible establecer permisos en archivos que estén en particiones FAT.

HFS

Sistema de Archivos Jerárquico o Hierarchical File System (HFS), es un sistema de archivos desarrollado por Apple Inc. para su uso en computadores que corren Mac OS. Originalmente diseñado para ser usado en disquetes y discos duros, también es posible encontrarlo en dispositivos de solo-lectura como los CD-ROMs.

HFS+

HFS Plus o HFS+ es un sistema de archivos desarrollado por Apple Inc. para reemplazar al HFS (Sistema jerárquico de archivos). También es el formato usado por el iPod al ser formateado desde un Mac. HFS Plus también es conocido como HFS Extended y Mac OS Extended. Durante el desarrollo, Apple se refirió a él con el nombre clave Sequoia.

HFS Plus es una versión mejorada de HFS, soportando archivos mucho más grandes (Bloques direccionables de 32 bits en vez de 16) y usando Unicode (En vez de Mac OS Roman) para el nombre de los archivos, lo que además permitió nombres de archivo de hasta 255 letras.

NTFS

Desde el punto de vista de un usuario, NTFS sigue organizando los archivos en directorios que, al igual que ocurre en HPFS, se ordenan.

Sin embargo, a diferencia de FAT o de HPFS, no hay ningún objeto "especial" en el disco y no hay ninguna dependencia del hardware subyacente, como los sectores de 512 bytes. Además, no hay ninguna ubicación especial en el disco, como las tablas de FAT o los superbloques de HPFS.

Los objetivos de NTFS son proporcionar lo siguiente:

- Confiabilidad, que es especialmente deseable para los sistemas avanzados y los servidores de archivos
- Una plataforma para tener mayor funcionalidad
- Compatibilidad con los requisitos de POSIX
- Eliminación de las limitaciones de los sistemas de archivos FAT y HPFS

Ventajas de NTFS

NTFS es la mejor opción para volúmenes de unos 400 MB o más. El motivo es que el rendimiento no se degrada en NTFS, como ocurre en FAT, con tamaños de volumen mayores.

La posibilidad de recuperación está diseñada en NTFS de manera que un usuario nunca tenga que ejecutar ningún tipo de utilidad de reparación de disco en una partición NTFS.
Desventajas de NTFS

No se recomienda utilizar NTFS en un volumen de menos de unos 400 MB, debido a la sobrecarga de espacio que implica. Esta sobrecarga de espacio se refiere a los archivos de sistema de NTFS que normalmente utilizan al menos 4 MB de espacio de unidad en una partición de 100 MB.

NTFS no integra actualmente ningún cifrado de archivos. Por tanto, alguien puede arrancar en MS-DOS u otro sistema operativo y emplear una utilidad de edición de disco de bajo nivel para ver los datos almacenados en un volumen NTFS.

No es posible formatear un disco con el sistema de archivos NTFS; Windows NT formatea todos los discos con el sistema de archivos FAT porque la sobrecarga de espacio que implica NTFS no cabe en un disco. (MICROSOFT, s.f.)

UNIDAD VI:

ENTRADA/SALIDA EN LOS DIFERENTES SISTEMAS OPERATIVOS

El papel del sistema operativo en el sistema E/S de una computadora es administrar y controlar las operaciones y dispositivos de E/S.

Vista general

Los elementos básicos del hardware E/S – puertos, buses y controladores de dispositivos – acomodan una amplia variedad de dispositivos de E/S. para encapsular los detalles y peculiaridades de los diferentes dispositivos, se estructura el kernel del sistema operativo para usar módulos de manejadores de dispositivo. Los manejadores de dispositivos presentan una interfaz uniforme de acceso a dispositivos con el subsistema de E/S, de manera muy similar a como las llamadas al sistema proporcionan una interfaz estándar entre la aplicación y el sistema operativo.

Hardware de E/S

Las computadoras operan muchas clases de dispositivos. En los tipos generales incluyen los dispositivos de almacenamiento (discos, cintas), dispositivos de trasmisión (tarjetas de red, módem) y dispositivos para la interfaz con el ser humano (pantalla, teclado, ratón). Un dispositivo se comunica con un sistema de cómputo enviando señales a través de un cable o incluso a través del aire. El dispositivo se comunica con la máquina mediante un punto de conexión denominado puerto (por ejemplo, un puerto serial).

Si uno o más dispositivos utilizan un conjunto común de cables, la conexión se denomina bus. Un bus es un conjunto de cables y un protocolo definido rígidamente que especifica un conjunto de mensajes que pueden enviarse por los cables. En términos de la electrónica, los mensajes se trasmiten mediante patrones de voltajes eléctricos que se aplican a los cables con tiempos (timings) definidos. Cuando el dispositivo A tiene un cable que se conecta al dispositivo B, y el dispositivo B tiene un cable que se conecta al dispositivo C y el dispositivo C tiene un cable que se conecta a un puerto en la computadora, este arreglo se denomina cadena de margarita. Generalmente opera como un bus.

Un controlador es un conjunto de componentes electrónicos que pueden operar un puerto, un bus o un dispositivo. Un controlador de puerto serial es un ejemplo de un controlador de dispositivos sencillo. Es una sola pastilla (chip) en la computadora que controla las señales en los cables de un puerto serial.

¿Cómo puede el procesador entregar comandos y datos a un controlador para realizar una transferencia de E/S? La respuesta sencilla es que el controlador tiene uno o más registros para datos y señales de control. El procesador se comunica con el controlador leyendo y escribiendo patrones de bits en estos registros. Una forma en que puede darse esta comunicación es mediante el uso de instrucciones especiales de E/S que especifican la transferencia de un byte o palabra a la dirección de un puerto de E/S. La instrucción de E/S activa líneas del bus para seleccionar el dispositivo apropiado y mover bits dentro o fuera de un registro del dispositivo. De manera alterna, el controlador de dispositivo puede soportar E/S con mapeo en memoria. En este caso, los registros de control del dispositivo se mapean en el espacio de direcciones del procesador. La CPU ejecuta solicitudes de E/S utilizando las instrucciones estándar de transferencia de datos para leer y escribir los registros de control del dispositivo.

El controlador de graficación tiene puertos de E/S para las operaciones de control básicas, pero el controlador tiene una gran región con mapeo en memoria para alojar los contenidos de la pantalla. El controlador envía salida a la pantalla escribiendo datos en dicha región.

El controlador genera la imagen en la pantalla con base en el contenido de esta memoria. La facilidad de escritura en un controlador de E/S con mapeo en memoria tiene como contrapeso una desventaja: debido a que un tipo común de fallo de software en la operación de escritura a través de un apuntador incorrecto a una región no deseada de la memoria, un registro de dispositivo con mapeo en memoria es vulnerable a una modificación accidental. Por supuesto, la memoria protegida reduce este riesgo.

Un puerto de E/S típicamente consta de cuatro registros, denominados registros de status, control, data-in y data-out. El registro status contiene bits que el anfitrión puede leer. Estos bits indican diversos estados, tales como: si el comando actual ya se completó, si está disponible un byte para leer del registro data-in, o si ha habido un error de dispositivo. El anfitrión puede escribir el registro control para iniciar un comando o cambiar el modo de un dispositivo. El anfitrión lee el registro data-in para obtener entradas y escribe el registro data-out para enviar salidas. Los registros de datos tienen comúnmente un tamaño de uno a cuatro bytes. Algunos controladores cuentan con pastillas (chips) FIFO que pueden contener varios bytes de datos de E/S para ampliar la capacidad del controlador más allá del tamaño del registro de datos. Una pastilla FIFO puede contener una pequeña ráfaga de datos hasta que el dispositivo o anfitrión sea capaz de recibir dichos datos.

Interrupciones

El mecanismo básico de interrupción funciona de la siguiente manera. El hardware de la CPU tiene un cable llamado línea de solicitud de interrupciones que la CPU revisa luego de ejecutar cada instrucción. Cuando la CPU detecta que un controlador ha colocado una señal en la línea de solicitud de interrupciones, la CPU guarda una pequeña cantidad del estado (como el valor actual del apuntador de instrucciones) y salta a la rutina del manejador de interrupciones en una dirección fija en memoria.

El manejador de interrupciones determina la causa de la interrupción, realiza el procesamiento necesario y ejecuta la instrucción return from interrupt para regresar la CPU al estado de ejecución antes de la interrupción. El controlador de dispositivo genera una interrupción y despacha al manejador de interrupciones, y éste apaga (clear) la interrupción dando servicio al dispositivo. El mecanismo básico de interrupción habilita a la CPU para responder a un evento asíncrono, el cual puede ser que el controlador de dispositivo quede listo para dar servicio.

La mayoría de la CPU tiene dos líneas de solicitud de interrupción. Una es la interrupción no mascarable, que se reserva para eventos como errores de memoria no recuperables. La segunda línea de interrupción es mascarable: puede ser apagada por la CPU antes de la ejecución de secuencias de instrucciones críticas que no deben ser interrumpidas. La interrupción mascarable es utilizada por los controladores de dispositivos para solicitar servicio.

El mecanismo de interrupción acepta una dirección – un número que selecciona una rutina específica para manejo de interrupciones de entre un pequeño conjunto. En la mayoría de las arquitecturas, esta dirección es un desplazamiento en una tabla denominada vector de interrupciones. Este vector contiene las direcciones de memoria de los manejadores de interrupción especializados. El propósito de un mecanismo de interrupción con base en el vector es reducir la necesidad de que un manejador único busque todas las posibles fuentes de interrupción para determinar cuál necesita servicio. En la práctica, sin embargo, las computadoras tienen más dispositivos que elementos de direcciones en el vector de interrupciones.

El mecanismo de interrupción también implementa un sistema de niveles de prioridad de interrupción. Este mecanismo habilita a la CPU para diferir el manejo de interrupciones de baja prioridad sin enmascarar todas las interrupciones y hace posible que una interrupción de alta prioridad tenga precedencia sobre la ejecución de una interrupción de baja prioridad.

Un sistema operativo tiene otros buenos usos para un mecanismo eficiente de hardware que guarda una pequeña cantidad del estado del procesador, y luego llama una rutina privilegiada en el kernel. Otro uso se encuentra en la implementación de llamadas al sistema. También se pueden emplear interrupciones para administrar el flujo de control dentro del kernel.

Las interrupciones se emplean en todos los sistemas operativos modernos para manejar los eventos asíncronos y para generar trampas que se comunican a rutinas en modo supervisor en el kernel. Para lograr que primero haga el trabajo más urgente, las computadoras modernas utilizan un sistema de prioridades de interrupción. Los controladores de dispositivos, los fallos de hardware y las llamadas al sistema, todos ellos generan interrupciones para activar rutinas del kernel. Debido a que las interrupciones se emplean intensamente para el procesamiento sensible al tiempo, se requiere un eficiente manejo de interrupciones para un buen desempeño del sistema.

Acceso Directo a Memoria

Para iniciar una transferencia del controlador de acceso directo a memoria (Direct Memory Access, DMA), el anfitrión escribe un bloque de comandos DMA en la memoria. Este bloque contiene un apuntador a la fuente de una transferencia, un apuntador al destino de la transferencia y una cuenta del número de bytes a transferir. El CPU escribe la dirección de este bloque de comandos en el controlador de DMA, luego sigue con otro trabajo. El controlador de DMA procede entonces a operar directamente el bus de memoria, colocando direcciones en dicho bus para realizar transferencias sin la ayuda de la CPU principal. Un controlador de DMA sencillo es un componente estándar en las PC, y unas tarjetas de E/S de control de bus para la PC generalmente contienen su propio hardware DMA de alta velocidad.

La secuencia de reconocimiento entre el controlador de DMA y el controlador de dispositivo se efectúa mediante un par de cables denominados DMA-request (petición) y DMA-acknowledge (reconocimiento). El controlador de dispositivo coloca una señal en el cable DMA-request cuando está disponible para su transferencia una palabra de datos. Esta señal hace que el controlador de DMA se apropie del bus de memoria, que coloque la dirección deseada en los cables de dirección de memoria, y que coloque una señal en el cable DMA-acknowledge.

Cuando el controlador de dispositivo recibe esta señal de reconocimiento, transfiere a memoria la palabra de datos y remueve la señal DMA-request. Cuando termina toda la transferencia, el controlador de DMA interrumpe a la CPU. Algunas arquitecturas de computadoras utilizan direcciones de memoria física para DMA, pero otras efectúan un acceso directo a memoria virtual (direct virtual memory access, DVMA), utilizando direcciones virtuales que pasan por un proceso de traducción de dirección de memoria virtual a dirección de memoria física. El DVMA puede realizar una transferencia entre dos dispositivos con mapeo en memoria sin la intervención de la CPU o el uso de la memoria principal.

En los kernels de modo protegido, el sistema operativo generalmente impide que los procesos emitan directamente comandos a dispositivos. Esta disciplina protege a los datos de violaciones al control de acceso y también protege al sistema de un empleo erróneo de controladores de dispositivo que podrían provocar una caída del sistema. En los kernels sin protección de memoria, los procesos pueden acceder directamente a los controladores de dispositivos. Se puede emplear este acceso directo para obtener un alto desempeño, ya que con él se puede evitar comunicación con el kernel, conmutación de contexto y capas de software de kernel. Desafortunadamente, el acceso directo interfiere con la seguridad y estabilidad del sistema.

Principales conceptos de los aspectos de hardware del sistema de E/S son:

- Un bus.
- Un controlador.

- Un puerto de E/S y sus registros.
- La relación de secuencia de reconocimiento entre el anfitrión y un controlador de dispositivo.
- La ejecución de esta secuencia de reconocimiento en un ciclo de escrutinio o mediante interrupciones.
- La descarga de este trabajo a un controlador de DMA para transferencias grandes. (Silberchatz, 2002, págs. 401-412)

Fundamentos del software de E/S

Objetivos del software de E/S

Un concepto clave en el diseño del software de E/S se conoce como independencia de dispositivo. Lo que 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.

Un objetivo muy relacionado con la independencia de los dispositivos es la denominación uniforme. El nombre de un archivo o dispositivo simplemente debe ser una cadena o un entero sin depender del dispositivo de ninguna forma.

Otra cuestión importante relacionada con el software de E/S es el manejo de errores. En general los errores se deben manejar lo más cerca del hardware que sea posible. Si el controlador descubre un error de lectura, debe tratar de corregir el error por sí mismo. Si no puede entonces el software controlador de dispositivo debe manejarlo, tal vez con sólo tratar de leer el bloque de nuevo.

Otra cuestión clave es la de las transferencias síncronas (de bloqueo) contra las asíncronas (controladas por interrupciones). La mayoría de las operaciones de E/S son asíncronas: la CPU inicia la transferencia y se va a hacer algo más hasta que llega a la interrupción.

Otra cuestión relacionada con el software de E/S es el uso de búfer. A menudo los datos que provienen de un dispositivo no se pueden almacenar directamente en su destino final.

E/S programada

Hay tres maneras fundamentales distintas en que se puede llevar a cabo la E/S.

1. E/S programada.
2. E/S controlada por interrupciones.
3. E/S mediante el uso de DMA.

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.

La E/S programada es simple, pero tiene la desventaja de ocupar la CPU tiempo completo hasta que se completen todas las operaciones de E/S. Si el tiempo para “imprimir” un carácter es muy corto (debido a que todo lo que hace la impresora es copiar el nuevo carácter en un búfer interno), entonces está bien usar ocupado en espera. Además, en un sistema incrustado o embebido, donde la CPU no tiene más que hacer, ocupado en espera es razonable. Sin embargo, en sistemas más complejos en donde la CPU tiene otros trabajos que realizar, ocupado en espera es ineficiente.

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.

E/S mediante el uso de DMA

Una obvia desventaja de la E/S controlada por interrupciones es que ocurre una interrupción en cada carácter. Las interrupciones requieren tiempo, por lo que este esquema desperdicia cierta cantidad de tiempo de la CPU. Una solución es utilizar 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. Es esencia, el DMA es E/S programada, sólo que el controlador de DMA realiza todo el trabajo en vez de la CPU principal. Esta estrategia requiere hardware especial (el controlador de DMA) pero libera la CPU durante la E/S para realizar otro trabajo.

La gran ganancia con DMA es reducir el número de interrupciones de una por cada carácter a una por cada búfer impreso. Si hay muchos caracteres y las interrupciones son lentas, esto puede ser una gran mejora. Por otra parte, el controlador de DMA es comúnmente más lento que la CPU principal.

Capas del software de E/S

Por lo general el software de E/S se organiza en cuatro capas.

NIVEL	FUNCIONES DE E/S
Proceso de usuario.	Hacer la llamada de E/S; aplicar formato a la E/S; poner en cola.
Software independiente del dispositivo.	Nombramiento, protección, bloqueo, uso de búfer, asignación.
Controladores de dispositivo.	Establecer los registros de dispositivo; verificar el estado.
Manejadores de interrupciones.	Despertar el controlador cuando se completa la E/S.
Hardware.	Realizar operaciones de E/S.

Manejadores de interrupciones

Una vez se haya completado la interrupción de hardware, se realizarán una serie de pasos que se deben llevar a cabo en el software.

1. Guardar los registros que no han sido guardados por el hardware de la interrupción.
2. Establecer un contexto para el procedimiento de servicio de interrupción.
3. Establecer una pila para el procedimiento de servicio de interrupciones.
4. Reconocer el controlador de interrupciones.
5. Copiar los registros desde donde se guardaron a la tabla de procesos.
6. Ejecutar el procedimiento de servicio de interrupciones.
7. Elegir cuál cual proceso ejecutar a continuación.
8. Establecer el contexto de la MMU para el proceso que se va a ejecutar a continuación.
9. Cargar los registros del nuevo proceso.
10. Empezar a ejecutar el nuevo proceso.

Drivers de dispositivos

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 y se incluye junto con el mismo.

Cada driver maneja un tipo de dispositivo o, a lo más, una clase de dispositivos estrechamente relacionados.

Generalmente los sistemas operativos clasifican los controladores en una de un pequeño número de categorías. Las categorías más comunes son los dispositivos de bloque como los discos, que contienen varios bloques de datos que se pueden direccionar de manera independiente, y los dispositivos de carácter como los tecleados y las impresoras, que generan o aceptan un flujo de caracteres.

Un controlador de dispositivo tiene varias funciones. La más obvia es aceptar peticiones abstractas de lectura y escritura del software independiente del dispositivo que está por encima de él, y ver que se lleven a cabo. Pero también hay otras tantas funciones que deben realizar. Por ejemplo, el controlador debe inicializar el dispositivo, si es necesario. También puede tener que administrar sus propios requerimientos y eventos del registro.

Software de E/S independiente del dispositivo

Las funciones que se realizan comúnmente en el software independiente del dispositivo son:

- Interfaz uniforme para controladores de dispositivos.
- Uso de búfer.
- Reporte de errores.
- Asignar y liberar dispositivos dedicados.
- Proporcionar un tamaño de bloque independiente del dispositivo.

La función básica del software independiente del dispositivo es realizar las funciones de E/S que son comunes para tos los dispositivos y proveer una interfaz uniforme para el software a nivel de usuario.

Software de E/S en espacio de usuario

Aunque la mayor parte del software de E/S está dentro del sistema operativo, una pequeña porción de éste consiste en bibliotecas vinculadas entre sí con programas de usuario, e incluso programas enteros que se ejecutan desde el exterior del kernel. Las llamadas al sistema, incluyendo las llamadas al sistema de E/S, se realizan comúnmente mediante procedimientos de biblioteca. El software de E/S de bajo nivel consiste en 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 a un sistema de multiprogramación

Discos

Hardware de disco

Los discos son de varios tipos. Los más comunes son los discos magnéticos (discos duros y discos flexibles). Se caracterizan por el hecho de que las operaciones de lectura y escritura son igual de rápidas, lo que los hace ideales como memoria secundaria. Algunas veces se utilizan arreglos de estos discos para ofrecer un almacenamiento altamente confiable. Para la distribución de programas, datos y películas son también importantes varios tipos de discos ópticos (CD-ROM's, CD-grabable y DVD).

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. El número de sectores alrededor de la circunferencia es por lo general de 8 a 32 en los discos flexibles, y hasta vario cientos en los discos duros. El número de cabezas varía entre 1 y 16.

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.

Los discos modernos se dividen en zonas, con más sectores en las zonas exteriores que en las interiores.

RAID

La idea básica 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.

Además de aparecer como un solo disco para el software, todos los RAID's tiene la propiedad de que los datos se distribuyen entre las unidades, para permitir la operación en paralelo. Patterson y sus colaboradores definieron varios esquemas distintos para hacer esto, y ahora se conocen como RAID nivel 0 hasta RAID nivel

CD-ROM's

En años recientes se han empezado a utilizar los discos ópticos (en contraste a los magnéticos). Estos discos tienen densidades de grabación mucho más altas que los discos magnéticos convencionales. Los discos ópticos se desarrollaron en un principio para grabar programas de televisión, pero se les puede dar un uso más estético como dispositivos de almacenamiento de computadora.

Un CD se prepara en varios pasos. El primero consiste en utilizar un láser infrarrojo de alto poder para quemar hoyos de 0.8 micrones de diámetro en un disco maestro con cubierta de vidrio. A partir de este disco maestro se fabrica un molde, con protuberancias en lugar de los hoyos del láser. En este molde se inyecta resina de policarbonato fundido para formar un CD con el mismo patrón de hoyos que el disco maestro de vidrio. Después se deposita una capa muy delgada de aluminio reflectivo en el policarbonato, cubierta por una laca protectora y

finalmente una etiqueta. Las depresiones en el sustrato de policarbonato se llaman hoyos (pits); las áreas no quemadas entre los hoyos se llaman áreas lisas (lands).

CD-regrabables

Las unidades de CD-RW utilizan láseres con tres potencias: en la posición de alta energía el láser funde la aleación y la convierte del estado cristalino de alta reflectividad al estado amorfo de baja reflectividad para representar un hoyo; en la posición de energía media la aleación se funde y se vuelve a formar en su estado cristalino natural para convertirse en un área lisa nuevamente; en baja energía se detecta el estado del material (para lectura), pero no ocurre una transición de estado.

DVD

Ahora se conoce oficial como disco versátil digital (digital versatile disk). Los DVD's utilizan el mismo diseño general que los CD's, 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:

1. Hoyos más pequeños (0.4 micrones).
2. Una espiral más estrecha.
3. Un láser rojo.

En conjunto, estas mejoras elevan la capacidad siete veces, hasta 4.7 GB. Una unidad de DVD 1X opera a 1.4 MB/seg (en comparación con los de 150 KB/seg de los CDs).

Formato de disco

Un disco duro consiste en una pila de platos de aluminio, aleación de acero o vidrio, de 5.25 o 3.5 pulgadas de diámetro. En cada plato se deposita un oxido de metal delgado magnetizable.

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 con huecos cortos entre los sectores.

El preámbulo empieza con cierto patrón de bits que permite al hardware reconocer el inicio del sector. También contiene los números de cilindro y sector, junto con cierta información adicional. El tamaño de la porción de datos se determina con base en el programa de formato de bajo nivel. La mayoría de los discos utilizan sectores de 512 bytes.

Una vez que se completa el formato de bajo nivel, el disco se particiona. En sentido lógico, cada partición es como un disco separado. Las particiones son necesarias para permitir que coexistan varios sistemas operativos. En la mayoría de las computadoras, el sector 0 contiene el registro de inicio maestro (MBR), el cual contiene cierto código de inicio además de la tabla de particiones al final. La tabla de particiones proporciona el sector inicial y el tamaño de cada partición.

Relojes

Los relojes (también conocidos como temporizadores) son esenciales para la operación de cualquier sistema de multiprogramación, por una variedad de razones. Mantienen la hora dl día y evitan que un proceso monopolice la CPU. El software de reloj puede tomar la forma de un software controlado de dispositivo, aun y cuando un reloj no es un dispositivo de bloque (como un disco) ni un dispositivo de carácter (como un ratón).

Hardware de reloj

Hay dos tipos de relojes de uso común en las computadoras, y ambos son bastante distintos de los relojes que utilizan las personas. 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 a 60 Hz. Estos relojes solían dominar el mercado, pero ahora son raros.

El otro tipo de reloj se construye a partir de 3 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 tención, puede generar una señal periódica con una precisión muy grande, por lo general en el rango de varios cientos de megahertz, dependiendo del cristal elegido. Mediante el uso de componentes electrónicos, esta señal base puede multiplicarse por un pequeño entero para obtener frecuencias de hasta mil MHz o incluso más. Por lo menos uno de esos circuitos se encuentra comúnmente en cualquier computadora, el cual proporciona una señal de sincronización para los diversos circuitos de la misma. 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.

Si se utiliza un cristal de 500 MHz, entonces de aplica un pulso al contador cada dos nseg. Con registros de 32 bits (sin signo), se pueden programar interrupciones para que ocurran a intervalos de 2 nseg hasta 8.6 seg. Los chips de reloj programables por lo general contienen dos o tres relojes que pueden programarse de manera independiente.

Para evitar que se pierda la hora actual cuando se apaga la computadora, la mayoría cuentan con un reloj de respaldo energizado por batería, implementando con el tipo de circuitos de baja energía que se utilizan en los relojes digitales. El reloj de batería puede leerse al iniciar el sistema. Si no está presente, el software puede pedir al usuario la fecha y hora actuales.

Software de reloj

Todo lo que hace el hardware de reloj es generar interrupciones a intervalos conocidos. Todo lo demás que se relacione con el tiempo debe ser realizado por el software controlador del reloj. Las tareas exactas del controlador de reloj varían de un sistema operativo a otro, pero por lo general incluyen la mayoría de las siguientes tareas:

1. Mantener la hora del día.
2. Evitar que los procesos se ejecuten por más tiempo del que tienen permitido.
3. Contabilizar el uso de la CPU.
4. Manejar la llamada al sistema alarm que realizan los procesos de usuario.
5. Proveer temporizadores guardianes (watchdosgs) para ciertas partes del mismo sistema.
6. Realizar perfilamiento, supervisión y recopilación de estadísticas.

Temporizadores de software

La mayoría de las computadoras tienen un segundo reloj programable que se puede establecer para producir interrupciones del temporizador, a cualquier velocidad que quiera un programa. Mientras que la frecuencia de interrupción sea baja, no habrá problema al usar este segundo temporizador para fines específicos de la aplicación. El problema surge cuando la frecuencia del temporizador específico de la aplicación es muy alta.

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 (por ejemplo, transmitir paquetes o comprobar si llegó un paquete), sin necesidad de cambiar al modo 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. Todo lo que hay que hacer es copiar el valor actual del reloj en el temporizador y sumarle el intervalo de tiempo de inactividad.

Interfaces de usuario: teclado, ratón, monitor

Software de entrada

La entrada de usuario proviene principalmente del teclado y del ratón. En una computadora personal, el teclado contiene un microprocesador integrado que por lo general se comunica, a través de un puerto serial especializado, con un chip controlador en la tarjeta principal (aunque cada vez con más frecuencia, los teclados se conectan a un puerto USB). 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, no el código ASCII. Los teclados tienen menos de 128 teclas, por lo que solo se necesitan 7 bits para representar el número de tecla. El octavo bit se establece en 0 cuando se oprime una tecla, y el 1 cuando se suelta.

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.

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; la segunda está orientada a líneas. En un principio se conocieron como modo crudo y modo cocido, respectivamente.

Software de ratón

La mayoría de las PC's tienen un ratón, o algunas veces un trackball, que sencillamente es un ratón boca arriba. 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. 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.

Los ratones inalámbricos son iguales a los alámbricos, excepto que en vez de devolver sus datos a la computadora a través de un cable, utilizan radios de baja energía, por ejemplo mediante el uso del estándar bluetooth.

Software de salida

Los editores de pantalla y muchos otros programas sofisticados necesitan la capacidad de actualizar la pantalla en formas complejas, como sustituir una línea a mitad de la pantalla.

Para satisfacer está necesidad 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 menudo estos comandos se conocen como secuencias de escape. En cierto momento, la industria vio la necesidad de estandarizar la secuencia de escape por lo que3 se desarrolló un estándar ANSI.

El sistema X Window

El sistema X window es muy portátil y se ejecuta por completo en espacio de usuario. En un principio tenía como propósito principal conectar un gran número de terminales de usuario remotas con un servidor de cómputo central, por lo que está dividido lógicamente en software cliente y software servidor, que puede ejecutarse potencialmente en distintas computadoras.

Cuando se inicia un programa de X, abre una conexión a uno o más servidores X, que se van a llamar estaciones de trabajo, aun cuando se podrían colocar en el mismo equipo que el programa X en sí. X considera que esta conexión es confiable en cuanto a que los mensajes perdidos y duplicados se manejan mediante el software de red y no tienen que preocuparse por errores de comunicación. Por lo general se utiliza TCP/IP entre el cliente y el servidor. Pasan cuatro tipos de mensajes a través de la conexión:

1. Comandos de dibujo del programa a la estación de trabajo.
2. Respuestas de la estación de trabajo a las solicitudes del programa.
3. Mensajes del teclado, del ratón y de otros eventos.
4. Mensajes de error.

Un concepto clave en X es el recurso. Un recurso es una estructura de datos que contiene cierta información. Los programas de aplicación crean recursos en las estaciones de trabajo.

Los recursos se pueden compartir entre varios procesos en la estación de trabajo. Los recursos tienen un tiempo de vida corto y no sobreviven a los reinicios de la estación de trabajo. Algunos recursos típicos son las ventanas, los tipos de letra, los mapas de colore (paletas de colores), mapas de píxeles (mapas de bits), los cursores y los contextos gráficos. Estos últimos se utilizan para asociar las propiedades con las ventanas y son similares en concepto a los contextos de dispositivos en Windows.

Interfaces gráficas de usuario

Una GUI tiene cuatro elementos esenciales, denotados por los caracteres WIMP. Las letras representan ventanas (windows), iconos (Icons), menús (Menus) y dispositivo señalador (pointing device), respectivamente. Las ventanas son áreas rectangulares en la pantalla que se utilizan para ejecutar programas. Los iconos son pequeños símbolos en los que se puede hacer clic para que ocurra una acción. Los menús son listas de acciones, de las que se puede elegir una. Por último, un dispositivo señalador es un ratón, trackball u otro dispositivo de hardware utilizado para desplazar un cursor alrededor de la pantalla para seleccionar elementos.

El elemento básico de la pantalla es un área rectangular llamada ventana. La posición y el tamaño de una ventana se determinan en forma única al proporcionar las coordenadas (en píxeles) de dos esquinas diagonalmente opuestas. Una ventana puede contener una barra de título, una barra de menús, una barra de desplazamiento vertical y una barra de desplazamiento horizontal.

Mapas de bits

Los procedimientos de la GDI son ejemplos de gráficos vectoriales. Se utilizan para colocar figuras geométricas y texto en la pantalla.

Se pueden escalar con facilidad a pantallas más grandes o pequeñas (siempre y cuando el número de píxeles en la pantalla sea el mismo). También son relativamente independientes del dispositivo. Una colección de llamadas a procedimientos de la GDI se puede ensamblar en un archivo que describa un dibujo completo. A dicho archivo se le conoce como metarchivo de Windows y es ampliamente utilizado para transmitir dibujos de un programa de Windows a otro.

No todas las imágenes se pueden manipular las computadoras se pueden generar mediante gráficos vectoriales. Por ejemplo, las fotografías y los videos no utilizan gráficos vectoriales en vez de ello, estos elementos se exploran al sobreponer una rejilla en la imagen. Los valores rojo, verde y azul promedio de cada cuadro de la rejilla se muestran y se guardan como el valor de un píxel. A dicho archivo se le conoce como mapa de bits. Hay muchas herramientas en Windows para manipular mapas de bits.

Otro uso para los mapas de bits es el texto. Una forma de representar un carácter específico en cierto tipo de letra es mediante un pequeño mapa de bits. Al agregar texto a la pantalla se convierte entonces en cuestión de mover mapas de bits. Un problema con los mapas de bits es que no se escalan. (Tanenbaum A. , 2009, págs. 343-412)

UNIDAD VII:

CONCEPTOS DE SEGURIDAD Y PROTECCION EN LOS SISTEMAS OPERATIVOS

Un sistema operativo puede dar soporte de ejecución a múltiples procesos de múltiples usuarios, que ejecutan de manera concurrente. Por ello, una de las funciones principales del sistema operativo es proteger los recursos de cada usuario para que pueda ejecutar en un entorno seguro. Donde los mecanismos permiten controlar el acceso a los objetos del sistema permitiéndolo o denegándolo sobre la base de información tal como la identificación del usuario, el tipo de recurso, la pertenencia del usuario a cierto grupo de personas, las operaciones que puede ,hacer el usuario o el grupo con cada recurso.

La protección consiste en controlar y, en su caso, impedir el acceso de los programas, procesos o usuarios a los recursos del sistema (archivos, memoria, CPU), las cuales son la perdida de datos y los intrusos.

Causas de pérdida de datos**:**

• Actos divinos: Incendios, inundaciones, terremotos, guerras, revoluciones o ratas que roen las cintas o discos flexibles.

• Errores de Hardware o Software: Mal funcionamiento de la CPU, discos o cintas ilegibles, errores de telecomunicación o errores en el programa.

• Errores Humanos: Entrada incorrecta de datos, mal montaje de las cintas o el disco, ejecución incorrecta del programa, perdida de cintas o discos.

La seguridad estudia cómo proteger la información almacenada en el sistema (datos o código) contra accesos indebidos o no autorizados (intrusos, fallos de la privacidad, etc).

Las razones para proveer protección a un sistema operativo son:

La necesidad de prevenirse de violaciones intencionales de acceso por un usuario.

La necesidad de asegurar que cada componente de un programa, use solo los recursos del sistema de acuerdo con las políticas fijadas para el uso de esos recursos.

Para proteger un sistema, debemos optar las necesarias medidas de seguridad en cuatro niveles distintos:

Físico: El nodo o nodos que contengan los sistemas informáticos deben dotarse de medidas de seguridad físicas frente a posibles intrusiones armadas o subrepticias por parte de potenciales intrusos. Hay que dotar de seguridad tanto a las habitaciones donde las maquinas residan como a los terminales o estaciones de trabajo que tengan acceso a dichas maquinas.

Humano: La autorización de los usuarios debe llevarse a cabo con cuidado, para garantizar que solo los usuarios apropiados tengan acceso al sistema. Sin embargo, incluso los usuarios autorizados pueden verse “motivados” para permitir que otros usen su acceso (por ejemplo, a cambio de un soborno). También pueden ser engañados para permitir el acceso de otros, mediante técnicas de ingeniería social.

Uno de los tipos de ataque basado en las técnicas de ingeniería social es el denominado phishing; con este tipo de ataque, un correo electrónico o página web de aspecto autentico llevan a engaño a un usuario para que introduzca información confidencial. Otra técnica comúnmente utilizada es el análisis de desperdicios, un término autorizado a la computadora (por ejemplo, examinando el contenido de las papeleras, localizando listines de teléfonos encontrando notas con contraseñas). Estos problemas de seguridad son cuestiones relacionadas con la gestión y con el personal, más que problemas relativos a los sistemas operativos.

Sistema operativo: El sistema debe auto protegerse frente a los diversos fallos de seguridad accidentales o premeditados. Un problema que este fuera de control puede llegar a constituir un ataque accidental de denegación de servicio. Asimismo, una cierta consulta a un servicio podría conducir a la revelación de contraseñas o un desbordamiento de la pila podría permitir que se iniciara un proceso no autorizado. La lista de posibles fallos es casi infinita.

Red: Son muchos los datos en los modernos sistemas informáticos que viajen a través de líneas arrendadas privadas, de líneas compartidas como Internet, de conexiones inalámbricas o de líneas de acceso telefónico. La interceptación de estos datos podría ser tan dañina como el acceso a un computador, y la interrupción en la comunicación podría constituir un ataque remoto de denegación de servicio, disminuyendo la capacidad de uso del sistema y la confianza en el mismo por parte de los usuarios.

Problemas de seguridad

- Usuarios inexpertos o descuidados.
- Usuarios no autorizados.
- Ataques por programa.
- Caballo de Troya.
- Puerta secreta.
- Amenazas al sistema.
- Gusanos.
- Virus.

Posibles efectos de las amenazas

- ❖ Revelación de información no autorizada.
- ❖ Destrucción de información.
- ❖ Utilización indebida de servicios del sistema.

- ❖ Daños físicos al sistema.
- ❖ Degradación en el funcionamiento del sistema.
- ❖ Denegación de acceso a usuarios autorizados.

Vigilancia de amenazas

- ➢ Buscar patrones de actividad sospechosos.
- ➢ Contar las veces que se proporcionan contraseñas incorrectas.
- ➢ Explorar el sistema en busca de agujeros.
- ➢ Contraseñas cortas o fáciles de adivinar.
- ➢ Programas no autorizados en directorios del sistema.
- ➢ Procesos con duración inusitada.
- ➢ Protecciones inapropiadas en directorios y archivos del sistema.
- ➢ Cambios en los programas del sistema.
- ➢ Cortafuegos (firewall) en computadores conectados en red.
- ➢ Separa los sistemas confiables de los no confiables.
- ➢ Limita el acceso por red a determinados dominios.

Diferencias entre riesgo y seguridad

La seguridad: es la ausencia de un riesgo. Aplicando esta definición a al tema correspondiente, se hace referencia al riesgo de accesos no autorizados, de manipulación de información, manipulación de las configuraciones, entre otros.

La protección: son los diferentes mecanismo utilizados por el SO para cuidar la información, los procesos, los usuarios, etc.

Un sistema de seguridad debe cumplir con unos requisitos:

- Confidencialidad: Acceso solo a usuarios autorizados.
- Integridad: Modificación solo por usuarios autorizados.
- Disponibilidad: Recursos solamente disponibles para usuario autorizado.

La seguridad se clasifica en:

- Externa: protección contra desastres y contra intrusos.
- Operacional: básicamente nos determina que acceso se permite a quien.

Una de las obligaciones de un sistema seguro es permanecer en constante vigilancia, verificando y validando las posibles amenazas, esto lo hacen con uso de contraseñas, controles de acceso

Se plantea que es más fácil haces un sistema seguro si esto se ha incorporado desde los inicios del diseño, porque no se puede hablar de un SO seguro si su núcleo no lo es; de igual manera es posible hacer seguridad por hardware donde se obtiene como ventaja la velocidad de operación permitiendo controles más frecuentes y mejora el performance.

Con respecto a los SO más seguros es difícil listarlos ya que todos tienen sus seguidores y contractares los cuales por instinto suelen defender lo que usan, pero es sin duda alguna lo que responden las encuestas hay una delas distribuciones de Linux denominada open BSD que es conocido como el SO más seguro aparte de que no deja de ser software libre, de igual manera es situado a los SO de Windows encima del Mac OSX donde apenas la última versión empieza a aplicar completamente algoritmos de seguridad que desde antes eran utilizados por la competencia pero sin duda alguna los sistemas libres ganan la batalla con respecto a la seguridad

Para poder garantizar la seguridad es fundamental proteger nuestro sistema, por eso básicamente los mecanismo articulados para la protección son los que nos llevan a un sistema seguro; existen diferentes formas de realizar la protección tal vez la más común y más básica sea definir cuáles son los archivos u objetos a proteger para que posteriormente se delimite que usuarios pueden acceder a que información

Como objetivos de la protección esta:

- Controlar el acceso a los recursos
- Utilización por diferentes usuarios

Generalmente surgen dudas sobre qué es lo que debemos proteger o que debo cuidar más y la respuesta es siempre variable según el tipo de necesidades de cada usuario, pero generalmente los más afectados son la CPU, la memoria, terminales, procesos, archivos y las bases de datos

Un sistema de protección deberá tener la flexibilidad suficiente para poder imponer una diversidad de políticas y mecanismos. La protección se refiere a los mecanismos para controlar el acceso de programas, procesos, o usuarios a los recursos definidos por un sistema de computación. Seguridad es la serie de problemas relativos a asegurar la integridad del sistema y sus datos.

Principales elementos de seguridad en los sistemas operativos

Contraseñas: Idealmente, no quieres que tu sistema operativo (OS) vaya directamente al escritorio cuando se inicia el equipo. Es mejor ir a una pantalla donde el usuario tiene que introducir una contraseña.

Fuerza de la contraseña: No todas las contraseñas son iguales. No deseas utilizar contraseñas relativamente fáciles de adivinar: cosas como segundos nombres, direcciones, cumpleaños, números de teléfono, códigos postales, o cualquier otra información pública vinculada a ti.

Cifrado: El cifrado codifica tus datos para que sólo se puedan leer cuando se proporcione una contraseña. Puedes encriptar archivos individuales y carpetas enteras o "volúmenes" de dispositivos de almacenamiento con software como PGP o TrueCrypt.

Protección contra malware: Los virus, troyanos, gusanos y keyloggers son colectivamente conocidos como "malware". Un escáner de antivirus de Symantec, McAfee o Kaspersky puede ayudarte a proteger tu equipo frente a estas amenazas.

Comportamiento riesgoso: Estos escáneres no pueden detectar el 100 por ciento de todas las amenazas. Ten mucho cuidado cuando navegas por un sitio web dudoso o abres un archivo adjunto de correo electrónico. Ten cuidado con los correos electrónicos fraudulentos que intentan divulgar tu información personal confidencial.

REFERENCIAS

Angelfire. (6 de Noviembre de 2014). Obtenido de http://sistemasoperativos.angelfire.com/html/6.1.html

Anonimo. (5 de Noviemre de 2014). Obtenido de Kioskea: http://es.kioskea.net/contents/648-gestion-de-memoria

Anonimo. (Jueves de Noviembre de 2014). *sistemas operativos*. Obtenido de http://mixteco.utm.mx/~resdi/historial/materias/capitulo5.pdf

Media docencia. (24 de Septiembre de 2014). *Laurel*. Obtenido de http://laurel.datsi.fi.upm.es/_media/docencia/asignaturas/dso/seg_y_prot_07-4pp.pdf

MICROSOFT. (s.f.). *FILE SYSTEM*. Recuperado el 11 de Octubre de 2014, de SOPORTE TECNICO: http://support2.microsoft.com/kb/100108/es

Nebrija. (19 de Septiembre de 2014). Obtenido de http://www.nebrija.es/~jmaestro/AT3148/Seguridad.pdf

Serrano, F. (13 de Septiembre de 2014). Obtenido de Blogspot: http://serranop4030.blogspot.mx/2012/09/tipos-de-memoria-exposicion.html

Silberchatz, G. (2002). *Sistemas operativos.* Mexico: Limusa Wiley.

Silberschatz, G. (1999). *Sistemas operativos.* México.: Adison Wesley.

Stalling, W. (2005). *Sistemas operativos Aspectos internos y principios de diseño. .* Madrid España.: Pearson.

Tanenbaum, A. (2009). *Sistema Operativos Modernos.* México: Pearson, Prentice Hall.