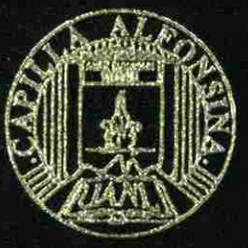

CAPILLA ALFONSINA

APUNTES DE SISTEMAS OPERATIVOS

Sánchez

UNIVERSIDAD AUTONOMA DE NUEVO LEON

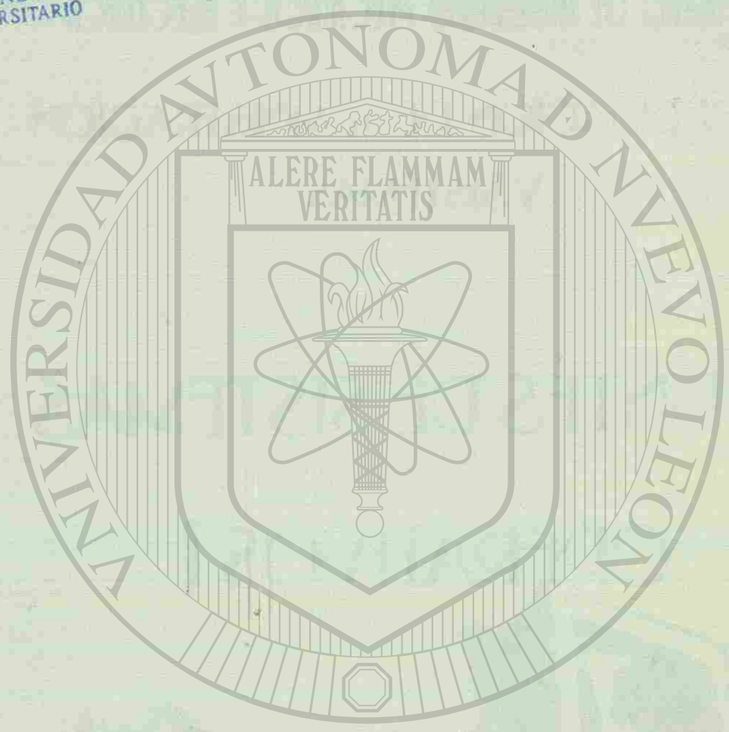
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

COORDINACION DE ADMNISTRACION

Y SISTEMAS

APUNTES DE SISTEMAS OPERATIVOS I

LIC. ARIADNE B. SANCHEZ RUIZ



Índice

INTRODUCCION

El objetivo de esta materia es lograr que el alumno tenga conocimientos del funcionamiento, características principales y la evolución de los sistemas operativos. Es fundamental para el Administrador de Sistemas debido a que son precisamente los sistemas operativos los que proporcionan el medio sobre el cual se desarrollan y se dá mantenimiento a los sistemas computacionales.

Sistemas Operativos I, analiza la clasificación y características de los sistemas operativos, la administración de procesos, la administración de memoria, los sistemas de archivo, entrada/salida y bloqueos.

Unidad I

Panorámica de los Sistemas Operativos

Objetivo de esta unidad.- Durante el desarrollo de esta unidad, se conocerán las bases, la evolución y los diferentes tipos de sistemas operativos, para así comprender las funciones, estructuras y componentes.

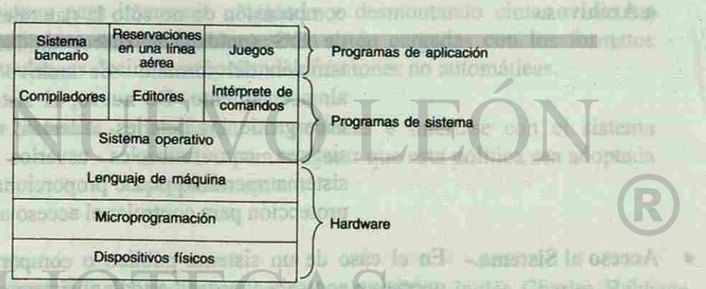
El alumno deberá comprender las funciones, estructuras y componentes

Los sistemas operativos son ante todo administradores de recursos. El principal recurso que administran es el hardware (dispositivos periféricos, recursos del sistema, etc.)

- los procesadores
- los medios de almacenamiento
- los dispositivos de entrada y salida
- los dispositivos de comunicación y datos

Su principal función es:

- proporcionar la interfaz del usuario
- permitir que los usuarios compartan el hardware entre sí
- evitar que los usuarios interfieran entre sí
- facilidad en la entrada/salida
- recuperarse de errores, etc.



El sistema operativo lleva la cuenta del estado de cada recurso y decide quien obtiene un recurso, durante cuanto tiempo y cuando; teniendo por objetivo primario incrementar la productividad de un recurso de proceso tal como el hardware del computador, o los usuarios del sistema de computo.

De igual modo el Sistema Operativo proporciona servicios de manera típica en las siguientes áreas:

- **Creación del Programa.-** El sistema operativo proporciona diversas facilidades y servicios, como editores y depuradores, para ayudar al programador en la creación de programas. En general, estos servicios están en la forma de utilería que en realidad no son parte del sistema operativo pero son accesibles mediante él.

- **Ejecución del Programa.-** Necesitan realizarse varias tareas para ejecutar un programa. Las instrucciones y datos deben cargarse en la memoria principal, los dispositivos de E/S y archivos deben inicializarse, otros recursos deben prepararse. El sistema operativo, entonces, maneja todas estas tareas por el usuario.

- **Acceso a Dispositivos de E/S.-** Cada dispositivo de E/S requiere su propio conjunto particular de instrucciones o señales de control, para la operación. El sistema operativo se encarga de los detalles, de manera que el programador pueda pensar en términos de lecturas y escrituras simples.

- **Acceso Controlado a Archivos.-** En el caso de archivos, el control debe incluir una comprensión de no sólo la naturaleza del dispositivo de E/S (unidad de disco, unidad de cinta) sino también del formato de archivo en el medio de almacenamiento. De nuevo, el sistema operativo se encarga de los detalles. Además, en el caso de un sistema con múltiples usuarios simultáneos, el sistema operativo puede proporcionar mecanismos de protección para controlar el acceso a los archivos.

- **Acceso al Sistema.-** En el caso de un sistema público o compartido, el sistema operativo controla el acceso a todo el sistema y a los recursos específicos del sistema. La función de acceso debe proporcionar protección de recursos y datos desde usuarios no autorizados y debe resolver conflictos en el caso de disputas por recursos.

- **Detección de Errores y Respuesta a Ellos.-** Pueden ocurrir varios errores mientras corre un sistema computacional. Esto incluye errores de hardware, internos, externos (errores de memoria, falla, mal funcionamiento de algún dispositivo) y varios errores de software (sobreflujo aritmético, intento de acceso a posición de memoria prohibida e incapacidad del sistema operativo para conceder la solicitud de una aplicación). El sistema operativo se ve precisado a emitir la respuesta que elimina la condición de error, con el menor impacto sobre las aplicaciones que se corren. La respuesta puede variar desde finalizar el programa que causó el error o volver a intentar la operación, hasta solo comunicar el error a la aplicación.

- **Contabilidad.-** Un buen sistema operativo recopila estadísticas del uso de los diferentes recursos y monitorea los parámetros de ejecución, como tiempo de respuesta. En cualquier sistema, esta información es útil para anticipar la necesidad de mejoras futuras y para ajustar el sistema con el fin de mejorar el rendimiento. En un sistema multiusuario, la información puede usarse para propósitos de facturación

Personal del Ambiente de Computación

Usuarios.- Son clientes del ambiente computacional que utilizan la computadora para organizar un trabajo.

Programadores.- Se encargan del mantenimiento del sistema adaptando las necesidades de instalación.

Operadores.- Se encargan de la vigilancia del sistema operativo respondiendo a peticiones para intervenir montando y desmontando cintas o discos, asegurándose de que las impresoras estén cargadas con los formatos adecuados, es decir, desempeñan las funciones no automáticas.

Administradores de Sistemas.- Establecen las políticas e interfase con el sistema operativo para asegurar que esta política sea adoptada apropiadamente.

Historia

La primera computadora digital fue diseñada por el matemático inglés Charles Babbage (1792-1871) quien gastó parte de su vida y su fortuna en el intento por construir su "máquina analítica". Nunca logró que funcionara adecuadamente ya que era un diseño puramente mecánico y la tecnología de su época no podía producir las ruedas, engranes,

levas y demás partes mecánicas con la precisión que él necesitaba. Resulta obvio que esta máquina no tenía sistema operativo.

Evolución del Hardware

Bulbos → Transistores → Circuitos Integrados → Circuitos Integrados a muy gran escala

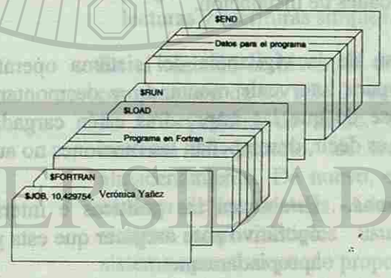
Durante la 2ª Guerra Mundial se vio la necesidad de suministrar alimentos, pertrechos, armamento, etc. de una forma óptima por lo que había que hacer cálculos muy grandes para lo que se necesitó la precisión y velocidad que podría ser proporcionada por una máquina y esto da origen a:

Generación 0 (Década de los 40's).

- No hay sistema operativo
- Se utiliza lenguaje máquina
- Se introducen programas y datos bit por bit mediante switchs mecánicos
- Se introduce la utilización de cintas y tarjetas perforadas
- Se comienza el desarrollo de los Lenguajes Ensambladores para acelerar el proceso de programación.

Generación 1 (Década de los 50's)

- General Motors Research Laboratories implantó el primer sistema operativo en una IBM 701 (a principios de los 50's)
- Ejecución de una sola tarea a la vez
- Se inicio el uso de procesamiento por lotes de secuencia única.
- Se logro cierta fluidez entre los trabajos



Generación 2 (1ª mitad de los 60's)

- Aparece la multiprogramación.
- Inicia el multiprocesamiento.
- Aparece el tiempo compartido (Time-Sharing).
- Se generan los primeros sistemas de tiempo real.
- Independencia de dispositivos

Generación 3 (2ª mitad de los 60's y 1ª mitad de los 70's)

- Mismas problemáticas de la generación dos
- Aparece el sistema 360 de la IBM, también conocido como OS/36
- Computadoras grandes y costosas
- Lentitud y alto porcentaje de error y falla
- Aparece la ingeniería de software.- Debido a que los cambios de personal a menudo hacían desechar sistemas completos
- Así surge un método disciplinado y estandarizado en el que fuera posible construir software *confiable, comprensible y fácil de mantener*.

Generación 4 (2ª mitad de los 70's hasta nuestros días)

- Aparecen las redes computacionales.
- Aparece el computador personal.
- El hardware disminuye en su costo.
- Se realiza la comunicación alejada geográficamente.
- Se presentan los problemas de seguridad en la información de cifrado y descifrado.
- Aparecen los sistemas controlados por menús.
- Aparecen los sistemas de bases de datos.
- Se reafirma la familia de computadoras IBM 360 en el mercado.
- Las reacciones de la industria computacional ante el sistema IBM 360 fueron las siguientes:

a) RCA Y GE : Copiar el sistema 360 e incluso su sistema operativo.

b) UNIVAC y DIGITAL : Se quedaron con los sobrantes de IBM.

C) HITACHI : Decidió mejorar y hacerlas tan poderosas o más que las IBM

- Separación de costos del Software y Hardware
  - Antes se vendía solamente el hardware y el software se daba "gratis" a los clientes.
  - Como el software era regalado no se hacían responsables del mismo.
  - Los vendedores proporcionaban listas de errores a los clientes de ciertas funciones para que no las usaran.

- **Se presentaron los siguientes cambios:**
  - Se eliminaron las cláusulas donde los vendedores no se responsabilizaban del software.
  - Se crean industrias independientes del software (Microsoft, Asthon-Tate, McAffe, etc.)
- Se generaliza la separación de los costos.
- Los usuarios tuvieron mayor posibilidad de seleccionar el software que desearan.
- Se inicio el uso de licencias para software.
- Aparecen las compatibles con IBM.

Tendencias Futuras

1. El hardware bajará aun mas de precio.
2. Las velocidades de procesamiento y capacidades de almacenamiento seguirán aumentando.
3. El tamaño físico del equipo seguirá disminuyendo.
4. El multiprocesamiento se hará mas común.
5. Muchas funciones del Sistema Operativo realizadas por software, migrarán al microcódigo.
6. Se fomentará el concepto de familia de computadoras

Fueron los microprocesadores los que cambiaron la forma de en que se consideraba la computación. El microprocesador ha hecho posible crear ciclos de procesador baratos.

Computación Distribuída

En la actualidad los ciclos de procesador se distribuyen generalmente por toda la organización en las computadoras personales y en las estaciones de trabajo, muchas de las cuales están parados la mayor parte del tiempo. La computación se ha convertido en un fenómeno distribuido mas que centralizado. Esta tendencia es obvia. Para utilizar los ciclos de procesador de un computador central desde una terminal remota, esa terminal necesita estar conectada a la computadora mediante una línea de comunicación. Las líneas de comunicación mas comunes son las telefónicas, que son útiles para lograr transmisiones confiables solo a velocidades relativamente bajas, de hasta unos cuantos miles de bits/segundo. Mas allá de tales velocidades, el ruido de las líneas impide obtener una transmisión rápida y segura. Por muy rápido que procese el computador central la información requerida por los usuarios de las terminales, la información que se intercambia entre el computador central y las terminales remotas no pueden transmitirse en forma confiable a altas velocidades a través de las líneas telefónicas convencionales. La velocidad de las comunicaciones ha sido un serio cuello de botella en el diseño de sistemas de cómputo y comunicación integrados, y es poco probable que tal situación mejore en un futuro próximo.

Solo quedan dos opciones para distribuir la capacidad de computo en los lugares donde se requiera:

1. Tratar de incrementar la velocidad de transmisión para aprovechar mejor la capacidad de un computador central

2. Distribuir computadores dedicados en los lugares de trabajo

Aumentar las velocidades de transmisión es un problema de proporciones enormes; sería necesario modificar por completo la red telefónica mundial

Sistemas Tolerantes a Fallas.- Son sistemas que pueden funcionar a pesar de la existencia de varios problemas.

Sistemas Abiertos.- Son sistemas de computación, de comunicaciones o de ambas cosas, cuyas especificaciones están ampliamente disponibles, aceptadas y estandarizadas.

Los sistemas abiertos tienen varios componentes:

1. Normas de Comunicación Abierta
2. Normas de Sistemas Operativos Abiertos
3. Normas de Interfaz de Usuarios Abiertas
4. Normas de Aplicaciones de Usuarios Abiertas

Hardware, Software y Firmware

Introducción.

Una computadora debe estar formada por:

- Procesador
- Memoria
- Componentes de E/S

Al menos uno o mas de estos elementos de cada tipo. Estos elementos básicos interconectados realizarán la función principal de cada computadora: ***Ejecutar programas.***

Tomaremos cuatro elementos estructurales principales:

1. **Procesador.-** Controla la operación de la computadora y ejecuta sus funciones de procesamiento de datos. Cuando existe un solo procesador, con frecuencia se hace referencia a él como unidad central de procesamiento (CPU, Central Processing Unit).

2. **Memoria Principal.-** Almacena datos y programas. Es típicamente volátil; se le conoce como memoria real o memoria primaria.

3. **Módulos de E/S.-** Mueven datos entre la computadora y su ambiente externo, esto incluye:
   - Dispositivos de memoria secundaria
   - Equipo de comunicaciones
   - Terminales

4. **Interconexión del Sistema**.- Una parte de la estructura y mecanismos que

proporcionan comunicación entre procesadores, memoria principal y módulos de E/S.

HARDWARE

Hardware.- Consiste en los dispositivos del sistema de computo
- su procesador,
- sus dispositivos de almacenamiento,
- sus dispositivos de entrada/salida,
- sus conexiones de comunicación.

Memoria Entrelazada.- Se usa para acelerar el acceso al almacenamiento primario. Normalmente mientras se obtiene acceso a algunas de las localidades de un banco de almacenamiento primario no puede haber otras referencias en proceso.
El entrelazado de memoria coloca las localidades de memoria contigua en diferentes bancos de almacenamiento de manera que pueden existir en proceso muchas referencias al mismo tiempo.

Registro de Reubicación.- Permite reubicar programas de manera dinámica. La dirección base del programa en memoria principal se coloca en el registro de reubicación. El contenido del registro de reubicación se suma a cada dirección generada por un programa en ejecución. El usuario puede programar como si su programa comenzara en la dirección cero. En el momento en que se ejecuta el programa, el registro de reubicación participa en todas las referencias a direcciones, esto permite que el programa resida en localidades diferentes de las que se pretendía que ocuparan.

Escrutinio.- (Polling) Es una técnica con la que una unidad puede verificar el estado de otra unidad de funcionamiento independiente. La primera unidad verifica si la segunda unidad se encuentra en cierto estado, si no es así, entonces la primera unidad continúa con la tarea que estaba realizando. Es una técnica costosa.

Interrupciones.- Por medio de interrupciones una unidad puede obtener de inmediato la atención de otra para informarle un cambio de estado. La interrupción hace que se almacene el estado de la unidad interrumpida antes de procesar la interrupción. Después se procesa la interrupción y se restablece el estado de la unidad interrumpida.

Buffer.- Es un área de memoria principal para retener datos durante transferencias de E/S. Mientras se efectúa una transferencia de entrada/salida su velocidad depende de muchos factores relacionados con el hardware de E/S.

Buffer Único.- El canal deposita datos en el buffer, el procesador los procesa y el canal sigue depositando nuevos datos, mientras el canal esta depositando datos no puede

hacerse ningún procesamiento, mientras se procesa un dato no pueden depositarse mas datos.

Buffer Doble.- Este hace posible empalmar operaciones de entrada/salida con operaciones de procesamiento, mientras el canal esta depositando datos en el buffer el procesador puede procesar los datos del otro buffer. Cuando el procesador acaba de procesar los datos de un buffer puede procesar los datos del otro buffer mientras el canal deposita los otros datos en el primero, por esta razón, el uso de estos buffer dobles se les denomina buffers alternantes.

Dispositivos Periféricos.- Permiten el almacenamiento de cantidades masivas de información fuera de la memoria principal. Los dispositivos de cinta por su naturaleza son dispositivos secuenciales que se escriben o leen sobre una tira larga de cinta magnétiça. Tal vez los dispositivos periféricos mas importantes en lo que se refiere a los sistemas operativos sean los discos magnéticos.
Los discos son dispositivos de acceso directo, permiten hacer referencia a unidades de datos individuales sin necesidad de realizar una búsqueda ordenada entre todos los datos contenidos en el disco.

Cronómetros y Relojes.- (Temporizadores) Un cronómetro de intervalos es útil en un sistema multiusuario para evitar que un usuario monopolice el procesador. Después de un intervalo designado, el cronómetro genera una interrupción para llamar la atención del procesador, este puede asignarse entonces a otro usuario. Con un reloj horario el computador puede controlar la hora de reloj pared con incrementos de millonésima de segundos o menores.

Operación en línea y fuera de línea.- Algunos periféricos estan equipados para la *operación en línea*, en la cual estan conectados al procesador, o para la *operación fuera de línea* en la cual son manejadas por unidades de control que no estan conectadas al sistema central. Las unidades de control fuera de línea son útiles porque hacen posible manejar dispositivos periféricos sin que esto represente una carga directa para el procesador. Ejemplo: Las operaciones para copiar de cinta a impresora se realizan a menudo con unidades fuera de línea.

Canales de E/S.- Un canal es un computador de propósito especial dedicado al control de E/S independientemente del procesador central del sistema. Un canal puede obtener acceso directo al almacenamiento primario para guardar o recuperar información. La importancia real de los canales es que aumenta considerablemente la cantidad de actividad concurrente del hardware del computador y libera al procesador de gran parte del control de E/S.

Canal Selector o Simple.- Son utilizados para la transferencia de datos a alta velocidad entre los dispositivos y la memoria principal. Los canales selectores tienen un solo subcanal y solamente pueden atender a un dispositivo a la vez.

Canales Multiplexores.- Los canales multiplexores tienen muchos subcanales; pueden intercalar muchos flujos de datos a la vez.

Canal Multiplexor de Bytes.- Alterna las transmisiones de dispositivos lentos como terminales, impresoras y líneas de comunicación de baja velocidad.

Canal Multiplexor de Bloques.- Alterna la transmisión de varios dispositivos de alta velocidad como impresoras láser y unidades de disco.

Robo de Ciclos.- Un punto de conflicto entre los canales y el procesador es el acceso a memoria principal. Como no puede realizarse mas que un acceso (a un determinado banco de memoria principal) a la vez, y como es posible que los canales y el procesador deseen obtener acceso simultáneo a memoria principal, los canales suelen tener prioridad y a esto se denomina *robo de ciclos*. Cuando se inicia una operación de E/S, los caracteres se transfieren hacia la memoria principal mediante el robo de ciclos; el canal usurpa temporalmente la ruta del procesador hacia el almacenamiento mientras se transfiere un caracter; el procesador después con su operación.

Estado problema.- Para los programas de usuarios, el subconjunto de instrucciones que se puede ejecutar en el estado problema impide, por ejemplo, la ejecución directa de instrucciones de entrada/salida; un programa usuario a l que se le permitiera realizar entrada y salida en forma arbitraria podría vaciar en pantalla la lista maestra de contraseñas del sistema, imprimir información de cualquier otro usuario o destruir el sistema operativo.

Estado Supervisor.- El sistema operativo se suele ejecutar con una categoría de usuario de máxima confianza en un estado supervisor, en el cual se tiene acceso a todo el conjunto de instrucciones de la máquina.

Multiprocesamiento.- Varios procesadores comparten un almacenamiento primario común y un solo sistema operativo. En el multiprocesamiento es necesario secuencializar el acceso a una localidad de almacenamiento compartido, de manera que dos procesadores no intenten modificar al mismo tiempo, con la posibilidad de alterar inadecuadamente su contenido.

Acceso Directo a

Memoria (DMA) .- es reducir al mínimo el número de interrupciones. El DMA no requiere mas que una sola interrupción por cada bloque de caracteres transferidos en una operación de E/S. Esto es bastante mas rápido que el método en el cual el procesador es interrumpido por cada carácter que se transfiere. Cuando un dispositivo esta listo para transmitir un carácter de un bloque, "interrumpe" al procesador. Pero con el DMA no es necesario guardar el estado del procesador; este sufre un retraso, más que una interrupción. Bajo el control de hardware especial, el carácter se transfiere a la memoria principal. Una vez terminada la transferencia, el procesador prosigue su operación.
El DMA es una característica de rendimiento particularmente útil en sistemas que manejan gran volumen de transferencias de E/S.
El hardware encargado del robo de ciclos y de operar los dispositivos de E/S en el modo DMA se conoce como *canal de DMA*.

Canalización.- Es una técnica de hardware utilizada en computadoras de alto rendimiento para aprovechar ciertos tipos de paralelismo en el procesamiento de instrucciones.

SOFTWARE

Software.- Se compone de los programas con instrucciones en lenguaje maquina y los datos que son interpretados por el hardware. Algunos tipos comunes de software son los compiladores, ensambladores, cargadores, editores de enlace, programas de aplicación, sistemas de manejo de base de datos, sistemas de comunicación y sistemas operativos.

Lenguaje de Máquina.- Es el lenguaje de programación que el computador puede entender directamente.

Lenguajes Ensambladores .- La programación en lenguaje máquina consume mucho tiempo y es propensa al error, los lenguajes ensambladores se desarrollaron para aumentar la velocidad de proceso, de programación y reducir los errores de codificación; se utilizan abreviaturas significativas para reemplazar grandes hileras de 1's y 0's, pero los programas en lenguaje ensamblador deben ser traducidos por la máquina. Los programas ensambladores son dependientes de la máquina.

MAQUINA	LENGUAJE ENSAMBLADOR
CYBER	COMPAS
PDP	CARDIAC

Lenguajes de Alto Nivel.- Permitieron a los usuarios, escribir programas de un modo

independiente de la máquina. Con ellos se aumenta en gran medida la velocidad de programación, se hicieron transportables entre diferentes sistemas y permite escribir programas de aplicaciones sin necesidad de ser un experto en la estructura interna de la máquina.

Lenguajes Orientados Hacia el Procedimiento.- Los lenguajes de alto nivel orientados al procedimiento, son los de propósito general y puede utilizarse para resolver gran variedad de problemas, ejemplo: Pascal, Basic, Cobol, Fortran.

Lenguajes Orientados Hacia el Problema.- Los lenguajes orientados hacia el problema se usan para un solo fin, ejemplos: SPSS, GPSS, ZAZ.

Programación Orientada a Objetos.- Las aplicaciones se han vuelto grandes y complejas y ya no es posible pensar en instrucciones individuales en lenguaje de máquina, ni siquiera en subrutinas pequeñas, como unidades de desarrollo de programas y de cálculo, dando importancia a la programación orientada a objetos. Los objetos son entidades abstractas que encapsulan todos los procedimientos y datos que tienen alguna relación. Tales entidades pueden manejarse como un paquete que se puede manipular de diferentes formas. Los objetos pueden representar hardware, como el CPU, la memoria y los dispositivos; entidades de software, como programas o archivos y varias otras identidades. La programación orientada a objetos se ha popularizado bastante en los proyectos de desarrollo de sistemas operativos.

Spool.- Se encarga de supervisar la transmisión de salida del almacenamiento principal al canal, del canal al disco duro, y del disco duro a la impresora. Este proceso de SPOOL rompe la asociación entre un programa en ejecución y la operación de dispositivos lentos como una impresora.

Sistemas de Control de E/S .- (IOCS) Es un programa supervisor desarrollado para manejar las complejidades de E/S. Evita al usuario el trabajo de controlar los detalles de la E/S. Estos paquetes son componente importante en los sistemas operativos actuales.

Compiladores Rápidos y Sucios.- Producen rápidamente programa objeto, pero el código generado puede ser bastante ineficiente, en cuanto al consumo de memoria y su velocidad de ejecución.

Compiladores Optimizadores.- Se ejecutan más lentamente, pero producen código de

calidad comparable o superior a la del código escrito por programadores expertos en lenguaje ensamblador.

Intérpretes.- No producen programas objeto, sino que ejecutan directamente los programas fuente. Se ejecutan lentamente si comparamos con un código compilado ya que debe traducir cada instrucción siempre que esta se ejecuta.

Cargador.- Es un programa que coloca las instrucciones de un programa y sus datos en las localidades de memoria principal.

Cargador absoluto.- Coloca las instrucciones y datos en localidades precisas indicadas por el programa en lenguaje máquina.

Cargador con reubicación.- Puede cargar un programa en diversos lugares de memoria dependiendo de la disponibilidad del espacio en el momento de la carga.

Cargador de enlace.- Combina todos los programas necesarios y los carga directamente en memoria principal.

Editor de enlace.- También combina programas pero además crea una imagen de carga que se conserva en almacenamiento secundario para referencia futura. Son útiles sobre todo en ambientes de producción; cuando se va a ejecutar un programa, la imágen de carga producida por el editor de enlace puede cargarse de inmediato sin la sobrecarga de recombinar segmentos de programa.

FIRMWARE

Firmware.- (Memoria fija) Consiste en los programas en microcódigo almacenados en una memoria de control de alta velocidad. Los programas objeto de uso común colocados en memoria de solo lectura, ROM y PROM también se les conoce como Firmware.

El concepto de microprogramación se atribuye por lo general al Prof. Maurice Wilkes, su artículo publicado en 1951 presentó los conceptos que forman la base de las técnicas actuales de microprogramación. En los años 60's se generalizó la microprogramación.

Microprogramación.- Introduce una capa de programación debajo del lenguaje de máquina de la computadora, como tal, hace posible la definición de instrucciones en lenguaje máquina. Es parte integral de las arquitecturas de computador modernas y es importante para los aspectos de rendimiento y seguridad de los sistemas operativos.

Microprogramación Dinámica.- Permite cargar fácilmente nuevos microprogramas en la memoria de control desde donde se ejecutan

los microprogramas. De este modo, los conjuntos de instrucciones de máquina se pueden variar dinámica y frecuentemente.

Microprogramas.- Se ejecutan en una memoria de control de alta velocidad y están formados por microinstrucciones individuales de naturaleza mucho más elemental y de función menos densa que las instrucciones convencionales de lenguaje máquina.

Microcódigo.- Una importante decisión de diseño es cuales funciones deben realizarse en microcódigo. El microcódigo representa una oportunidad real para mejorar el rendimiento de la ejecución de un sistema. Mediante la codificación cuidadosa de secuencias de instrucciones de ejecución frecuente en firmware, en lugar de software, los diseñadores han logrado extraordinarias mejoras en el rendimiento.

Microdiagnóstico.- Los microprogramas tienen acceso a mas hardware que los programas en lenguaje de máquina, por lo que es posible efectuar una detección y una corrección de errores más amplia y con mayor grado de precisión.
Algunos sistemas alternan el microdiagnóstico con lasinstrucciones en lenguaje de máquina. Esto permite evitar errores potenciales y lograr una operación más confiable, por lo que la microprogramación puede ser efectiva en el diseño de sistemas tolerantes a fallas.

Emulación.- Es una técnica para lograr que un computador se comporta como si fuera otro. El conjunto de instrucciones en lenguaje de máquina del computador que se va a emular se microprograma en la máquina anfitriona. Así, los programas en lenguaje de maquina de la maquina emulada pueden ejecutarse directamente en la anfitriona.
Los fabricantes de computadores utilizan muy a menudo la emulación cuando presentan nuevos sistemas. Los usuarios habituados a los computadores anteriores pueden ejecutar directamente sus antiguos programas en los sistemas nuevos sin alteración alguna, esto modela el proceso de conversión.

Algunas funciones del Sistema Operativo que suelen realizarse en Microcódigo son:

- Manejo de Interrupciones
- Mantenimiento de varias estructuras de datos
- primitivas de sincronización que controlan el acceso a datos compartidos y a otros recursos
- "Conmutación contextual"; esto es, la conmutación rápida de un procesador entre varios usuarios en un sistema multiusuario.
- Secuencias de llamada a procedimientos y retorno.

Realizar en microcódigo las funciones del sistema operativo puede mejorar el desempeño, reducir los costos de desarrollo de programas y mejorar la seguridad de un sistema.

Unidad II
CONCEPTOS DE PROCESOS

Objetivo de esta unidad.- Durante el desarrollo de esta unidad, se analizarán las principales técnicas para la admiinistración del CPU y de los procesos en su secuencia de ejecución.

El alumno deberá comprender qué es un proceso, cuáles son las transiciones de estado de un proceso, cómo se logra la comunicación entre estos conjuntamente con los problemas que enfrenta el Sistema Operativo para lograr la comunicación; cómo decide el sistema operativo que procesos debe efectuar cuando dos o más procesos son ejecutables lógicamente.

Definiciones de "proceso"

El término "proceso" ha sido utilizado a veces como sinónimo de tarea, ha tenido muchas definiciones:

- Es una actividad asíncrona.
- Un programa en ejecución.
- El "espíritu animado" de un procedimiento.
- El "centro de control" de un procedimiento en ejecución.
- Lo que se manifiesta por la existencia de un "bloque de control de proceso" (PCB) en el sistema operativo.
- La entidad a la que se asignan los procesadores
- La unidad despachable.

Un programa es una entidad inanimada, solo cuando un procesador le infunde vida, se convierte en la entidad activa que se denomina proceso.

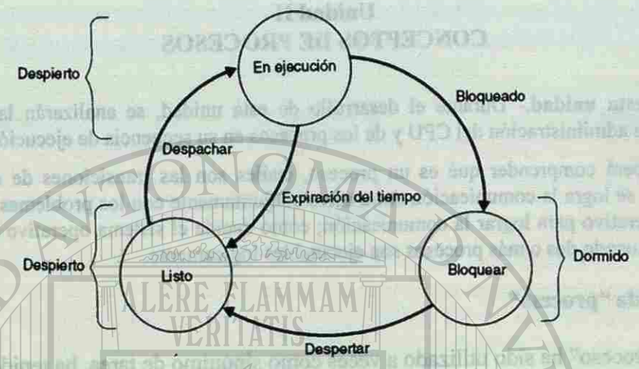
Estados de un Proceso

Se dice que un proceso esta en estado de ejecución si tiene asignado el CPU. Se dice que un proceso esta en estado listo si pudiera utilizar un CPU en caso de haber uno disponible. Un proceso esta en estado bloqueado si esta esperando algún evento antes de proseguir su ejecución.

Transiciones de Estado de los Procesos

Cuando se admite una tarea en el sistema, se crea el proceso correspondiente y se inserta normalmente al final de la lista de procesos listos. El proceso se desplaza poco poco hacia el frente de la lista de procesos listos, a medida que los procesos que se encuentran antes que él completan su turno de uso de el CPU.

Cuando el proceso llega al principio de la lista, se le asigna el CPU cuando este queda disponible y entonces se dice que hay una *transición de estado* del estado listo al estado de ejecución. A la asignación del procesador al primer proceso de la lista de procesos listosse le llama despacho.



Despachar(x_proceso):Listo ⟶ Ejecución

Para evitar que un proceso monopolice el sistema, en forma accidental o intencional, el sistema operativo utiliza un reloj de interrupción por hardware (también llamado cronómetro de intervalos) para que las tareas de ese usuario se ejecuten durante un intervalo específico de tiempo o cuanto (quantum). Si el proceso no libera voluntariamente el CPU antes de que expire el cuanto, el reloj genera una interrupción, haciendo que el sistema operativo retome el control. El sistema operativo envia al proceso que se estaba ejecutando a la lista de procesos listo y procede a ejecutar el primero de la lista de procesos listos.

Tiempo Expirado(x_proceso):Ejecución ⟶ Listo

Si el proceso que se esta ejecutando inicia una operación de entrada/salida antes de que expire su cuanto, libera voluntariamente al CPU, es decir el proceso se bloquea a sí mismo esperando que se complete la operación de entrada/salida.

Bloquear(x_proceso):Ejecución⟶ Bloqueado

Cuando se completa una operación de entrada/salida o algún otro evento que espere un proceso. El proceso realiza la transición del estado bloqueado al estado listo.

Despertar(x_proceso):Bloqueado⟶ Listo

Como se observará la única transición de estado iniciada por el proceso de usuario es el bloqueo; las otras tres transiciones son iniciadas por entidades externas al proceso.

Bloque de control del proceso (PCB)

La forma en que se manifiesta un proceso en un sistema operativo es mediante el PCB (Bloque de Control de Proceso). EL PCB es una estructura de datos que contiene información importante acerca de un proceso incluyendo:

- El estado actual del proceso.
- Un identificador único del proceso.
- Un apuntador hacia el padre del proceso (hacia el proceso que lo creó).
- Apuntadores a los hijos del proceso en el caso de tener procesos creados por él.
- La prioridad del proceso.
- Apuntadores hacia las zonas de memoria del proceso.
- Apuntadores a los recursos asignados del proceso.
- Un área para salvaguarda de los registros.
- El procesador en que se esta ejecutando el proceso (en el caso de múltiples procesadores).

El PCB es un almacén central de información que permite al sistema operativo localizar toda la información importante acerca de un proceso. PCB es la entidad que define un proceso al sistema operativo.

Cuando el sistema operativo conmuta el CPU entre varios procesos activos utiliza las áreas de salvaguarda de los PCB para guardar la información que necesita para reiniciar un proceso cuando obtenga el CPU.

Operaciones sobre procesos

Los sistemas que administran procesos deben ser capaces de realizar ciertas operaciones sobre procesos:

- Crear un proceso que implica:

1. Darle un nombre al proceso.
2. Dar de alta en la lista de procesos conocidos del sistema o de la tabla de procesos (insertar)
3. Determinar la prioridad inicial del proceso.
4. Crear el PCB de ese proceso.
5. Asignar recursos al proceso.

Un proceso puede crear un nuevo proceso y al proceso creador se le va a llamar proceso padre y al proceso creado se le va a llamar proceso hijo, lo cual nos va a originar la estructura jerárquica de procesos.

- Destruir un proceso que implica:

1. Eliminarlo de la lista de procesos, no solamente del sistema.
2. Los recursos que estaba utilizando se devuelven al sistema.
3. El PCB se borra.

La destrucción de un proceso es más difícil cuando este ha creado a otros procesos.

- Suspender un proceso

Un proceso suspendido no puede proseguir hasta que lo reanuda otro proceso. La suspensión es una operación importante y ha sido puesta en práctica en diferentes formas en diversos sistemas. La suspensión dura por lo regular breves momentos o breves periodos de tiempo. El sistema efectúa suspensiones para eliminar temporalmente ciertos procesos y así reducir la carga del sistema durante situaciones de carga máxima. Cuando hay suspensiones largas se deben liberar los recursos del proceso.

- Reanudar un proceso

Reanudar o activar un proceso implica reiniciarlo a partir del punto en el que se suspendió.

- Bloquear un proceso
- Despertar un proceso
- Despachar un proceso
- Permitir que un proceso se comunique con otros (comunicación entre procesos)
- Cambiarle la prioridad a un proceso.

Razones por las que se puede suspender un proceso:

- Si un sistema esta funcionando mal y es probable que falle se puede suspender los procesos activos para reanudarlos cuando se haya corregido el problema.
- Un usuario que desconfie de los resultados parciales de un proceso puede suspenderlo en lugar de abortarlo hasta que verifique si el proceso funciona correctamente o no.
- Algunos procesos se pueden suspender como respuesta a las fluctuaciones a corto plazo de la carga del sistema y reanudarse cuando las cargas regresen a niveles normales.

Estados de Suspención

Se ha añadido dos nuevos estados denominados: suspendido_listo y suspendido_bloqueado, por lo que ahora contaremos con estados activos que vimos con anterioridad, y estados de suspención.

Tenemos que, una suspensión puede ser iniciada por el propio proceso o por otro. En un sistema con un solo procesador, el proceso en ejecución puede suspenderse a sí mismo, ningún otro proceso podría estar en ejecución para realizar la suspención.

Solamente otro proceso puede suspender un proceso listo. La transición sería:

Suspender(x_proceso):Listo ⟶ Suspendido_listo

Un proceso puede hacer que otro proceso que se encuentre en estado suspendido_listo pase al estado listo. La transición sería:

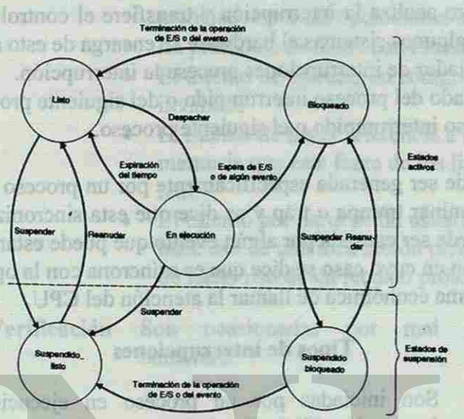
Reanudar(x_proceso):Suspendido_Listo ⟶ Listo

Un proceso puede suspender a otro que este bloqueado. La transisción sería:

Suspender(x_proceso):Bloqueado ⟶ Suspendido_Bloqueado

Un proceso puede reanudar otro proceso que este suspendido_bloqueado.La transición sería:

Reanudar(x_proceso):Suspendido_Bloqueado ⟶ Bloqueado



Transacciones

Suspender(x_proceso):Listo ⟶ Suspendido_Listo

Despachar(x_proceso):Listo ⟶ Ejecución

Suspender(x_proceso):Ejecución ⟶ Suspendido_Listo

Reanudar(x_proceso):Suspendido_Listo ⟶ Listo

Tiempo excedido(x_proceso):Ejecución ⟶ Listo

Suspender(x_proceso):Bloqueado ⟶ Suspendido_bloqueado

Completar(x_proceso):Bloqueado ⟶ Listo

Reanudar(x_proceso):Suspendido_Bloqueado ⟶ Bloqueado

Completar(x_proceso):Suspendido_Bloqueado ⟶ Suspendido_Listo

Esperar(x_proceso):Ejecución ⟶ Bloqueado

Procesamiento de interrupciones

Interrupción.- Es un evento que altera la secuencia en que el procesador ejecuta las instrucciones. La interrupción es generada por el hardware del sistema.

Cuando se genera una interrupción:

1. El sistema operativo toma el control, es decir, el hardware pasa el control al sistema operativo.
2. El sistema operativo guarda el estado del proceso interrumpido, en muchos sistemas esta información se guarda en el PCB del proceso interrumpido.
3. El sistema operativo analiza la interrupción y transfiere el control a la rutina apropiada para atenderla. En algunos sistemas el hardware se encarga de esto automáticamente.
4. La rutina del manejador de interrupciones procesa la interrupción.
5. Se restablece el estado del proceso interrumpido o del siguiente proceso.
6. Se ejecuta el proceso interrumpido o el siguiente proceso.

Una interrupción puede ser generada específicamente por un proceso en ejecución en cuyo caso se le suele denominar trampa o trap y se dice que esta sincronizada con la operación del proceso o bien puede ser causada por algún evento que puede estar relacionado o no con el proceso en ejecución en cuyo caso se dice que es asíncrona con la operación en proceso. Interrumpir es una forma económica de llamar la atención del CPU.

Tipos de interrupciones

Interrupción SVC.- (Supervisor Call) Son iniciadas por un proceso en ejecución que ejecute una instrucción SVC. Una interrupción SVC es un petición generada por el usuario de un servicio en particular del sistema. Puede ser como ejecutar una operación de E/S, obtener más memoria, o comunicarse con el operador del sistema.
Un usuario no puede entrar arbitrariamente al sistema operativo, debe solicitar permiso por medio de una SVC. El sistema operativo esta enterado de los usuarios que intentan rebasar sus límites y puede rechazar ciertas peticiones si el usuario no cuenta con los privilegios necesarios.

Interrupciones de E/S.- Son iniciadas por hardware de E/S. Este tipo de interrupciones indican al CPU el cambio de estado de algún canal o un dispositivo. Se producen cuando finaliza una operación de entrada/salida o cuando un dispositivo pasa al estado listo.

Interrupciones Externas.- Son causadas por diversos eventos, incluyendo la expiración de un cuanto (Quantum del reloj) que interrumpe, cuando pulsamos la tecla de interrupción de la consola o bien cuando recibimos la señal de otro procesador, esto en el caso de un sistema de múltiples procesadores.

Interrupciones de Reinicio.- Cuando se oprime el botón de reinicio de la consola o si llegara desde otro procesador una señal de reinicio.

Interrupciones de Verificación del Programa.- Son causadas por una amplia gama de problemas que pueden ocurrir cuando se ejecutan las instrucciones en lenguaje de máquina de un programa, dichos problemas incluyen:

- Una división entre cero.
- El exceso o defecto de los números que pueden ser manejados por las operaciones aritméticas.
- La presencia de datos con formato erróneo.
- El intento de ejecutar un código de operación inválido.
- El intento de hacer referencia a una localidad de memoria que este fuera de los límites de la memoria real.
- El intento por parte de un usuario no privilegiado de hace uso de una instrucción privilegiada o el intento de hacer uso de un recurso protegido

Interrupciones por Verificación de Máquina.- Son ocasionadas por mal funcionamiento del hardware.

Cambio de Contexto
En el esquema de las interrupciones de los procesadores a alta escala se maneja mediante el sistema operativo rutinas denominadas manejadores de interrupciones de primer nivel o FLIH (First Level Interrupt Handlers) para procesar las diferentes clases de interrupciones. Existe un manejador de interrupciones de primer nivel por cada tipo de interrupción.
Cuando ocurre una interrupción, el sistema operativo guarda el estado del proceso interrumpido y transfiere el control al manejador de interrupciones del primer nivel apropiado. Esto se logra mediante una técnica llamada Cambio de Contexto.
Los manejadores de primer nivel deben distinguir entre interrupciones de la misma clase; el procesamiento de estas interrupciones es realizado entonces por alguno de los manejadores de interrupciones de segundo nivel.

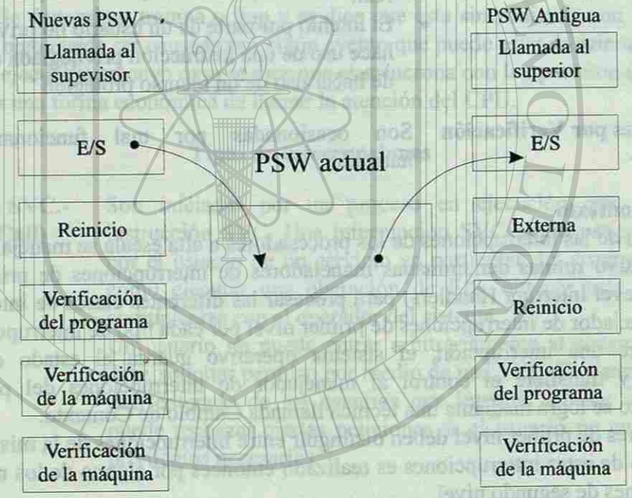
Palabras de Estado o PSW (Program Status Word)
Las palabras de estado del programa controlan el orden de ejecución de las instrucciones y contienen información acerca del estado de un proceso. Hay tres tipos de PSW: nuevas, actuales y antiguas.
La dirección de la siguiente instrucción que se debe ejecutar se almacena en la PSW actual que indica también los tipos de interrupciones habilitadas e inhabilitadas en ese momento. El CPU permite que ocurran las interrupciones habilitadas; las interrupciones inhabilitadas permanecerán pendientes y solo en algunos casos se pasaran por alto.
El procesador nunca puede estar inhabilitado para una interrupción SVC o para una de reinicio o para ciertas interrupciones de verificación del programa.

En un sistema de un procesador solo hay una PSW actual, seis nuevas y seis antiguas, una por cada tipo de interrupción.

La PSW nueva para algún tipo de interrupción contiene la dirección permanente de la memoria principal en la que reside el manejador de interrupciones correspondiente.
Cuando ocurre una interrupción, si el procesador no esta inhabilitado para este tipo de interrupción, el hardware automáticamente cambia la PSW como sigue:

1. Se guarda la PSW actual en la PSW antigua de ese tipo de interrupción.
2. Se guarda la PSW nueva de ese tipo de interrupción en la PSW actual.

Después de este intercambio de PSW, la PSW actual contendrá la dirección del manejador de interrupciones apropiada y este ejecutara y procesara la interrupción.



Núcleo del Sistema Operativo (Nucleus/Kernel/Core).
Todas las operaciones en las que participan procesos son controladas por la parte del sistema operativo llamada núcleo. El núcleo normalmente representa solo una pequeña parte de lo que por lo general se piensa que es todo el sistema operativo pero también es el código que más se utiliza. Reside por lo regular en la memoria principal mientras que otras partes del sistema operativo son cargadas solo cuando se necesitan.

Una de las funciones mas importantes incluidas en ese núcleo es el procesamiento de interrupciones. El núcleo inhabilita las interrupciones cuando atiende una interrupción y las habilita nuevamente una vez que ha completado el procesamiento de la interrupción. Cuando se da un flujo continuo de interrupciones, es posible que el núcleo deje inhabilitadas las interrupciones por un periodo largo; esto puede ocasionar un elevado

tiempo de respuesta a las interrupciones. Por esta razón los núcleos son diseñados para realizar el "mínimo" posible de procesamiento en cada interrupción y dejar que el resto lo realice el proceso apropiado del sistema, que puede operar mientras el núcleo se habilita para atender otras operaciones. Lo que se traduce en que las interrupciones permanecen habilitadas un porcentaje mucho mayor del tiempo y que el sistema responde mejor.

Resumen de las funciones del núcleo del Sistema Operativo:

1. Manejo de Interrupciones.
2. Crear y Destruir procesos.
3. El cambio de los estados de los procesos.
4. El despacho de los procesos.
5. Suspender y reanudar un proceso.
6. Sincronización de los procesos.
7. La comunicación entre procesos.
8. El manejo de los bloques de control de procesos.
9. Apoyo a las actividades de entrada/salida.
10. Asignación y liberación de memoria.
11. Es apoyo para el sistema de archivos.
12. Apoyo para el mecanismo de llamada y retorno de un procedimiento.
13. Apoyo para ciertas funciones de contabilidad del sistema

Habilitación e Inhabilitación de Interrupciones

Al núcleo del sistema operativo solo puede llegarse mediante una interrupción. El núcleo inhabilita las interrupciones mientras responde a la interrupción que esta procesando. Una vez que se determina la causa de la interrupción, el núcleo pasa el tratamiento de la interrupción a un proceso específico del sistema que se ha diseñado para manejar ese tipo de interrupciones.

En algunos sistemas el procesamiento de cada interrupción es realizado por un sistema operativo de una sola pieza.

Procesos Concurrentes Asíncronos

Proceso Concurrente.- Los procesos son concurrentes si existen simultáneamente. Los procesos concurrentes pueden funcionar en forma totalmente independiente unos de otros o pueden ser asíncronos, lo cual significa que en ocasiones requieren cierta sincronización y cooperación.

Procesamiento en Paralelo

A medida que el hardware siga disminuyendo en tamaño y costo irán apareciendo tendencias definidas hacia el multiprocesamiento, el procesamiento distribuido y el paralelismo a gran escala. Si algunas operaciones pueden ejecutarse de manera lógica en paralelo, los computadores también podrán realizarlas físicamente en paralelo, aun sí el nivel de

paralelismo es de miles o tal vez de millones de actividades concurrentes. Ello puede significar mejoras en el rendimiento muy superiores a las que son posibles en los computadores secuenciales.

El procesamiento paralelo resulta interesante y complejo por diversas razones. Resulta más fácil para la gente concentrar su atención en una sola actividad a la vez que pensar en paralelo. Ejemplo.- Intente leer dos libros al mismo tiempo, leyendo una del primero y una del segundo y así sucesivamente.

Determinar qué actividades pueden realizarse en paralelo es difícil y lleva mucho tiempo. Los programas en paralelo son mucho mas difíciles de depurar que los secuenciales. Ejemplo.- Después de corregir un error, puede resultar imposible recinstruir la secuencia de eventos que provocaron su aparición original, por lo que sería impropio asegurar que ya se ha corregido.

Una Estructura de Control para Indicar Paralelismo: Parbegin/Parend

- Existen muchas construcciones de lenguajes de programación para indicar paralelismo. Normanlmente se trata de parejas de enunciados como a continuación se describen:
- Un enunciado para indicar que la ejecución secuencial se dividirá en varias secuencias de ejecución paralelas (hilos de control).
- Un enunciado para indicar la fusión de ciertas secuencias de ejecución paralelas y la reanudación de la ejecución secuencial.

Estos enunciados aparecen en pareja y se conocen como parbegin/parend (para la ejecución en paralelo) o cobegin/coend (para la ejecución concurrente) para indicar el principio y fin.

```
parbegin
        enunciado1;
        enunciado2;
          .
          .
          .
        enunciadon
parend
```

Ejemplo.- Supongamos que un programa se esta ejecutando una sola secuencia de instrucciones encuentra la contrucción parbegin. Eso hace que el hilo de control único se divida en *n* hilos separados; uno para cada enunciado de la construcción parbegin/parend. Puede tratarse de enunciados simples, llamadas a procedimientos, bloques de enunciados secuenciales delimitados por un par *begin/end*, o combinaciones de ellos. Cada uno de los hilos de control terminará en algún momento y llegará a *parend*. Cuando terminen todos los hilos de control, reanudará un hilo de control único y el sistema ejecutará el enunciado que sigue al *parend*.

Ejemplo.- Considérese el cálculo de una raíz de una ecuación cuadrática:

```
x:= (-b + ( b **  2  -  4 * a * c) ** .5)/(2 * a)
```


Procesamiento Secuencial	Procesamiento Paralelo
1 b ** 2	1 parbegin
2 4 * a	temp1: = -b;
3 (4 * a) * c	temp2: = b ** 2;
4 (b ** 2) - (4 * a * c)	temp3: = 4 * a;
5 (b ** 2 - 4 * a * c) ** .5	temp4: = 2 * a
6 - b	parend;
7 (-b) + ((b ** 2 - 4 * a * c) **.5)	2 temp5: = temp3 * c;
8 2 * a	3 temp5: = temp2 - temp5;
9 (-b + (b **2 - 4 * a * c) **.5) / (2 * a)	4 temp5: = temp5 ** .5;
	5 temp5: = temp1 + temp5;
	6 x: = temp5 / temp4

Las cuatro operaciones contenidas dentro del parbegin/parend se evalúan en paralelo y las cinco restantes en secuencia, reduciendo considerablemente el tiempo.

Exclusión Mutua.- Se llama exclusión mutua a la situación en la cual los procesos cooperan de tal forma que, mientras un proceso obtiene acceso a datos compartidos modificables, los demas procesos no pueden hacer lo mismo.

Secciones Críticas.- Cuando un proceso obtiene acceso a datos compartidos modificables se dice que se encuentra en una sección critica o región critica.

Las secciones críticas deben ejecutarse lo más rápido posible; un proceso no debe bloquearse en su sección crítica y las secciones críticas deben ser codificadas cuidadosamente para evitar la posibilidad de ciclos infinitos por ejemplo.

Algoritmo de Dekker.- Es una realización en software de las primitivas de exclusión mutua y tiene las siguientes propiedades:

- No requiere ninguna instrucción especial de hardware.
- Un proceso que se encuentra fuera de su sección crítica no puede evitar que otro proceso entre en su propia sección crítica.

Un proceso que desea entrar en su sección crítica puede hacerlo sin que haya posibilidad de un aplazamiento indefinido.

```
program algoritmo_dekker;
var proceso_favorecido: (primero, segundo);
    p1deseaentrar, p2deseaentrar: boolean;
procedure proceso_uno;
    begin
      while true do
        begin
          p1deseaentrar := true;
          while p2deseaentrar do
            if proceso_favorecido = segundo then
              begin
                p1deseaentrar:= false;
                while proceso_favorecido = segundo do;
                p1deseaentrar := true
              end;
          sección_crítica_uno;
```

```
                proceso_favorecido:= segundo;
                p1deseaentrar:= false;
                otras_tareas_uno
            end
        end;
procedure proceso_dos;
    begin
        while true do
            begin
                p2deseaentrar := true;
                while p1deseaentrar do
                    if proceso_favorecido = primero then
                        begin
                            p2deseaentrar:= false;
                            while proceso_favorecido = primero do;
                            p2deseaentrar := true
                        end;
                sección_crítica_dos;
                proceso_favorecido:= primero;
                p2deseaentrar:= false;
                otras_tareas_dos
            end
    end;
begin
    p1deseaentrar := false;
    p2deseaentrar := false;
    proceso_favorecido := primero;
    parbegin
        proceso_uno;
        proceso_dos
    parend
end.
```

El algoritmo Dekker se aplica para la exclusión mutua de n pasos; estas suelen ser soluciones muy complejas. Sin embargo durante muchos años representó el último adelanto en cuestión de algoritmos de espera activa para manejar la exclusión mutua.

En 1981, G.L. Peterson publicó un algoritmo mucho más sencillo para manejar la exclusión mutua de dos procesos con espera activa.

```
program algoritmo_peterson;
var proceso_favorecido: (primero, segundo);
    p1deseaentrar, p2deseaentrar: boolean;
procedure proceso_uno;
    begin
        while true do
            begin
                p1deseaentrar := true;
                proceso_favorecido := segundo;
                while p2deseaentrar
                    and proceso_favorecido = segundo do;
                sección_crítica_uno;
                p1deseaentrar:= false;
                otras_tareas_uno
            end
    end;
procedure proceso_dos;
    begin
        while true do
```



```
            begin
                p2deseaentrar := true;
                proceso_favorecido := primero;
                while p1deseaentrar
                    and proceso_favorecido = primero do;
                sección_crítica_dos:= false;
                otras_tareas_dos
            end
    end;
begin
    p1deseaentrar := false;
    p2deseaentrar := false;
    proceso_favorecido := primero;
    parbegin
        proceso_uno;
        proceso_dos
    parend
end.
```

Semáforos

Dijkstra extrajo los conceptos fundamentales de la exclusión mutua en su concepto de semáforos. Un semáforo es una variable protegida cuyo valor solo puede ser leído y alterado mediante las operaciones P y V y una operación de asiganción de valores iniciales.

Los *semáforos binarios* pueden tener solamente los valores 0 o 1. Los semáforos contadores o semáforos generales pueden tener tan solo valores enteros no negativos.

La operación P sobre el semáforo S, escrita P(S), opera como sigue:

```
si S > 0
    entonces S := S - 1
    si no (esperar  S)
```

La operación V sobre el semáforo S, escrita V(S), opera como sigue:

```
si (uno o mas procesos esperan S )
entonces (dejar que prosiga uno de esos procesos)
si no S:=S + 1
```

Se adoptará una disciplina de colas de primeras entradas-primeras salidas para los procesoso que esperan terminar una operación P(S).

P y V son indivisibles. La exclusión mutua sobre el semáforo S se implanta dentro de P(S) y V(S). Si varios procesos desean ejecutar una operación P(S) de manera simultánea, solamente se permitirá la ejecución de uno de ellos.. Los otros quedaran en espera, pero la manera como se realizan P y V garantiza que los procesos no se aplazarán en forma indefinida.
Los semáforos pueden llevarse a la práctica en software o hardware. Se suelen incluir en el núcleo del sistema operativo donde se controla la comunicación de los estados de los procesos.

Los semáforos pueden servir para llevar a la práctica un mecanismo de sincronización bloquear/despertar: un proceso se bloquea a sí mismo mediante P(S), con S=0 inicialmente, para esperar que ocurra un evento; otro proceso detecta el evento y despierta al proceso bloqueado mediante V(S).
En una relación productor-consumidor, un proceso productor, genera información que utiliza un segundo proceso consumidor (comunicación entre procesos). Si estos procesos se comunican por medio de un buffer compartido, el productor no debe producir cuando el buffer esta lleno y el consumidor no debe consumir cuando el buffer esta vacío (sincronización de procesos).
Los semáforos contadores son particularmente útiles cuando un recurso debe asignarse a partir de un banco de recursos idénticos. Cada operación P indica que se ha asignado un recurso, en tanto que una operación V indica que se ha devuelto un recurso al banco.
Las operaciones de los semáforos pueden llevarse a la práctica con espera activa, pero esto puede ocasionar desperdicio (pueden incluirse en el núcleo para evitar la espera activa).

Programación Concurrente

Los métodos anteriores poseen en su estructura varios defectos. Son lo bastante primitivos que es difícil expresar las soluciones a los problemas de concurrencia más complejos aumentando el ya de por sí difícil problema de probar la corrección de los programas. De hecho el mal uso de estas primitivas podría corromper la operación de un sistema concurrente.
En el método de los semáforos, si omitimos P, no obtenemos la exclusión mutua; si omitimos V, las tareas que están esperando por causa de operaciones P podrían quedar en bloqueo permanente. Una vez que comienza una operación P, el usuario no puede arrepentirse y seguir otro curso de acción mientras el semáforo esta en uso. Una tarea puede esperar solo un semáforo a la vez, lo cual pudiera originar un bloqueo mutuo en situaciones de asiganción de recursos.
Ha surgido gran interés en los lenguajes de programación concurrente porque permiten expresar de una forma natural las soluciones a ciertos problemas de carácter inherentemente paralelo, y también gracias al verdadero paralelismo de hardware que es posible lograr con los multiprocesadores y sistemas distribuidos. Los programas concurrentes son más difíciles de escribir, depurar, modificar y probar que son correctos, sin embargo, tienen muchas aplicaciones posibles. Los sistemas operativos en sí son ejemplos importantes de sistemas concurrentes, como también los son los sistemas de control de tráfico aéreo, los sistemas de control de tiempo real (los que controlan las refinerías, plantas de productos químicos, procesadoras de alimentos, etc.). Los robots son sistemas altamente concurrentes, cuando un robot camina debe ser capaz de ver, oír, tocar, saborear y oler al mismo tiempo. De hecho, el proceso de caminar, es preciso coordinar con cuidado la operación de cada motor y cada parte de los miembros para lograr incluso los movimientos más sencillos.
ADA es el primer lenguaje concurrente de uso generalizado, pero pasará algún tiempo antes de que haya un número significativo de sistemas realizados por completo en ADA.

Regiones Críticas y Regiones Críticas Condicionales

La noción de *región crítica* se desarrollo para expresar de forma sencilla la exclusión mutua. Por ejemplo para indicar que alguna acción se realizará con acceso exclusivo a ciertos datos_ compartidos se escribe la siguiente instrucción en Pascal:

región datos_compartidos **do** acción (Región Crítica)

La noción de *región crítica condicional* permite especificar la sincronización, además de la exclusión mutua cuyo ejemplo es:

región datos_compartidos **do begin await** condición; acción **end**

Es posible realizar acción con acceso exclusivo a datos_compartidos cuando se cumple condición.

MONITORES

Monitor.- Es una construcción para concurrencia que contienen tanto los datos como los procedimientos necesarios para asignar un recurso compartido específico y reutilizable en serie o en grupos de estos recursos.

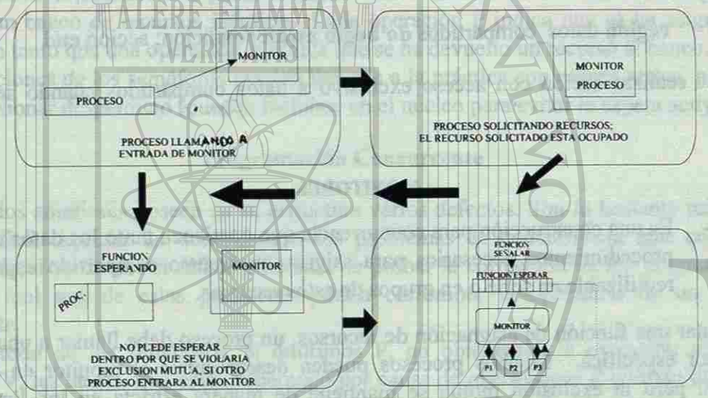
Para ejecutar una función de asignación de recursos, un proceso debe llamar a una entrada del monitor especifica. Muchos procesos pueden desear entrar al monitor en diversos momentos, pero la exclusión mutua se mantiene de manera estricta en los límites del monitor ya que solo se permite la entrada de un proceso a la vez. Los procesos que desean entrar en el monitor cuando ya esta en uso deben esperar; el monitor controla automáticamente esta espera, de esta forma la exclusión mutua esta garantizada.

Los datos que se encuentran dentro del monitor pueden ser *globales* con respecto de todos los procedimientos del monitor o *locales* con respecto de un procedimiento específico, Todos esos datos son solo accesibles dentro del monitor; un proceso fuera del monitor no puede tener acceso a esos datos, a esto le llamamos *ocultación de información*, que es una técnica de estructuración de sistemas que facilita mucho el desarrollo de sistemas de software confiables.

Si un proceso que esta llamando a la entrada del monitor encuentra que el recurso ya ha sido asignado, el procedimiento monitor llama a la función *esperar*. El proceso podría permanecer dentro del monitor pero ello violaría la *exclusión mutua* si otro proceso entrara en el monitor. Por lo tanto el proceso que llamó a *esperar* debe aguardar fuera del monitor hasta que se libere el recurso.

En algún momento, el proceso que tiene el recurso llamará a una entrada al monitor para devolver el recurso al sistema. Esta entrada podría limitarse a aceptar el recurso devuelto y esperar a que llegue otro proceso solicitante. Pero puede haber procesos esperando el

recurso, así que la entrada del monitor invoca a la función *señalar* para permitir que uno de los procesos en espera adquiera el recurso y salga del monitor.
Si un proceso señala la liberación del recurso y ningún proceso esta esperando, la señal no tendrá efecto, pero el monitor habrá recuperado un recurso. Es claro que un proceso que esta en espera de un recurso deberá hacerlo afuera para permitir que otro proceso entre al monitor a devolver el recurso.
Se da mayor prioridad a los procesos en espera para asegurar que un proceso que esta en espera de un recurso termine por obtenerlo, de otra forma un proceso nuevo podría obtener ese recurso y ocasionar un problema de aplazamiento indefinido.



Variables Condicionales.- Las variables condicionales son diferentes de las variables convencionales con las que estamos familiarizadas. Se asocia una variable condicional diferente a cada una de las razones por las que un proceso puede verse obligado a esperar. Las operaciones de *esperar* y *señalar* se modifican para incluir el nombre de la variable condicional que esta esperando o señalando.

ESPERAR (nombre de variable condicional)
SEÑALAR (nombre de variable condicional)

Cuando se define una variable condicional se crea una cola. Un proceso que llama a la función *esperar* se agrega a la cola; un proceso que llama a la función *señalar* hace que un proceso en espera sea sacado de la cola y entre en el monitor.

A continuación se muestra un monitor sencillo para el manejo de la asignación de recursos. El atractivo de la función asignador_de_recursos redica en que funciona exactamente como un semáforo binario; la función obtener_recurso trabaja como la operación P y la función devolver_recurso trabaja como la operación V.


```
monitor asignador_de_recursos;
var recurso_en_uso:boolean;
recurso_libre:condition;

procedure obtener_recurso;
begin
if recurso_en_uso then                    <----------->   Función P
esperar(recurso_libre);
recurso_en_uso := true
end;
procedure devolver_recurso;
begin
recurso_en_uso := false;                  <----------->   Función V
señalar (recurso_libre);
end;
begin
recurso_en_uso := false
end;
```

Buffer circular .-
(Ring Buffer, Bounded Buffer o Buffer Acotado)

Un ejemplo de Monitor. Se utiliza en situaciones en las que un proceso productor pasa datos a un proceso consumidor. Por lo que debemos considerar de vital importancia la sincronización entre los procesos productores y consumidores.
Los sistemas operativos asignan a menudo una cantidad fija razonable de memoria para las comunicaciones mediante buffer entre los procesos productores y consumidores. Esto se puede simular mediante una tabla de tamaño indicado. El productor deposita los datos en elementos sucesivos de la tabla y el consumidor los saca en el orden en que se depositaron. El productor puede haber producido varios elementos que aun no ha utilizado el consumidor. En algún momento el productor llenara el ultimo elemento de la tabla, cuando produzca mas datos deberá dar la vuelta y comenzar a depositar los datos en el primer elemento de la tabla dando por asentado que el consumidor halla sacado los datos previamente colocados ahí por el productor, la tabla se cierra efectivamente en circulo, de ahí el termino de buffer circular.

Debido al tamaño fijo del buffer circular el productor puede encontrar a veces ocupados todos los elementos de la tabla, en este caso, el productor debe esperar hasta que el consumidor vacíe algunos elementos de la tabla, del mismo modo en que habrá ocasiones en que el consumidor deberá esperar hasta que el productor deposite algunos datos en algunos elementos de la tabla.

Un mecanismo de buffer circular es adecuado para implantar un mecanismo de *spool* en los sistemas operativos.
Cuando un proceso genera líneas que se deben imprimir en un dispositivo de salida relativamente lento, como una impresora. Como el proceso puede producir las líneas mucho

más rápidamente de lo que la impresora puede imprimirlas y como también se desea que el proceso se ejecute lo mas rápidamente posible, las líneas de salida del proceso se envían a un mecanismo de buffer circular. El buffer circular puede estar en el almacenamiento primario, pero es más probable que este en disco. A este proceso se le llama *spooler*.
Otro proceso lee las líneas del buffer circular y las escribe en la impresora trabajando a la velocidad de la impresora a este proceso a menudo se le llama *despooler*.
El buffer circular tiene suficiente espacio para absorber el desajuste resultante de la diferencia de velocidad entre *spooler y despooler*. Desde luego suponemos que el sistema no genera indefinidamente, líneas más rápido de lo que la impresora pueda imprimirlas; si así fuera el buffer siempre estaría lleno y sería de poco valor mantener la continuidad de la operación de impresión.

Llamadas anidadas a monitores.- Es posible que un sistema que utilice monitores permita a los procedimientos de un monitor llamar a procedimientos que estén en otro, con lo que se originan las llamadas anidadas a monitores.

Expresiones de ruta.- La noción de expresiones de ruta se introdujo para poder especificar el orden en que se ejecutarán los procedimientos al tiempo que se mantienen separados de esta especificación los procedimientos.
Ejemplos:
path leer **end**
path comenzar_lectura, leer, lectura_terminada **end**
path C:\DOS\WINDOW\F_PROT96

Cuando una expresión de ruta especifica una secuencia de llamadas, no implica que el mismo invocador tenga que hacer cada una de las llamadas.

Paso de mensajes.- Las dos formas existentes de comunicación entre procesos son:
- La memoria compartida
- El paso de mensajes.

Los procesos envían y reciben mensajes mediante llamadas como las siguientes:

```
enviar (proceso_receptor, mensaje);
recibir(proceso_transmisor, mensaje);
```

Las llamadas enviar y recibir se realizan normalmente como llamadas al sistema operativo accesibles desde muchos ambientes de lenguajes de programación.

La llamada a enviar puede señalar explícitamente el nombre del proceso receptor, o puede omitir el nombre, indicando así que el mensaje será difundido a todos los procesos.

Envío con bloqueo.- Debe esperar a que el receptor reciba el mensaje (comunicación sincrónica). Una llamada a recibir con bloqueo obligara al receptor a esperar.

Envío sin bloqueo.- Permite al transmisor continuar con otro procesamiento aunque el receptor no haya recibido aún el mensaje (comunicación asincrona). Este tipo de envíos requiere un mecanismo de buffer para almacenar el mensaje hasta que lo reciba el receptor.
La comunicación asíncrona sin bloqueo aumenta el nivel de concurrencia.
Una llamada a recibir sin bloqueo permite que el receptor continúe con otro procesamiento antes de volver a intentar la recepción.

Cuando se habló de comunicación entre procesos en el mismo computador siempre damos por asentado una transmisión sin errores, sin embargo, en los sistemas distribuidos la transmisión puede tener errores, incluso puede perderse, por ello los transmisores y receptores cooperan muchas veces utilizando un *protocolo de reconocimiento* para confirmar la recepción correcta de cada transmisión.
El transmisor que espera un reconocimiento del receptor puede utilizar un mecanismo de tiempo, cuando expira el tiempo si es que no ha llegado el reconocimiento el transmisor puede retransmitir el mensaje.
Una complicación de los sistemas distribuidos con paso de mensajes es la necesidad de dar nombres en forma no ambigua a los procesos, de forma que las llamadas explícitas a enviar y recibir hagan referencia a los procesos correctos.
La creación y destrucción de procesos se puede coordinar mediante algún mecanismo centralizado de asignación de nombres pero tal cosa puede introducir una sobrecarga considerable de transmisión ya que cada maquina deberá solicitar permiso para utilizar nuevos nombres.
Una alternativa es asegurar que cada computador asigne nombres únicos a sus propios procesos; de esta manera puede hacerse referencia a los procesos combinando el nombre del computador con el nombre del proceso. Lograrlo, requiere un control centralizado para determinar un nombre único para cada computador de un sistema distribuido, pero esto, no representa una sobrecarga importante, pues los computadores se añaden o eliminan de las redes con muy poca frecuencia.
Las comunicaciones a base de mensajes en los sistemas distribuidos presentan serios problemas de seguridad, uno de ellos es el problemas de comprobar la autenticidad ya que siendo receptor o transmisor en realidad no se sabe si son impostores los que transmiten o los que reciben los mensajes.

Buzones y puertos
La comunicación a base de mensajes se puede establecer directamente entre receptor y transmisor, o pueden utilizar colas intermedias. En el caso de las colas intermedias, el transmisor y receptor especifican la cola, en lugar del proceso, con que se van a comunicar.

Buzón.- Es una cola de mensajes que puede ser utilizada por múltiples transmisores y receptores.
En un sistema distribuido, los receptores pueden estar situados en muchos computadores, por lo cual requieren dos transmisiones para la mayor parte de los mensajes, una del transmisor al buzón y otra del buzón al receptor. El problema se resuelve dando a cada receptor un puerto.

Puerto.- Se define como un buzón utilizado por múltiples transmisores y un solo receptor.
En los sistemas distribuidos, cada despachador tiene normalmente un puerto en el cual recibe solicitudes de muchos clientes.

Conductos.- Unix introdujo la noción de conductos o pipes se como un esquema para manejar comunicaciones entre procesos. Conducto es en esencia un buzón que permite extraer un número especificado de caracteres a la vez. No tiene la noción de mensaje, de modo que,si el usuario desea simular un buzón de mensajes con un conducto, deberá leer cada vez el número apropiado de caracteres (el tamaño del mensaje). Los conductos se utilizan con frecuencia para manejar las comunicaciones entre un proceso productor único y un solo consumidor.

Llamadas a procedimientos remotos.- La noción de llamadas a procedimientos remotos o RPC (remote procedure calls) se introdujo con el objeto de ofrecer un mecanismo estructurado de alto nivel para realizar comunicaciones entre procesos en sistemas distribuidos.
Con una llamada a un procedimiento remoto, un proceso de un sistema puede llamar a un procedimiento de un proceso de otro sistema. El proceso que llama se bloquea esperando el retorno desde el procedimiento llamado en el sistema remoto y después continua su ejecución desde el punto que sigue inmediatamente a la llamada.
El procedimiento llamado y el que llama residen en procesos distintos, con espacios de direcciones distintos, por lo cual no existe la noción de variables globales compartidas, como en los procedimientos normales dentro de un solo proceso. Por lo tanto, las RPC transfieren la información estrictamente a través de parámetros de la llamada.
Una RPC puede realizarse aprovechando un mecanismo existente del tipo enviar/recibir. Las llamadas al procedimiento remoto sería:

```
enviar ( proceso_remoto,  parámetros_de_entrada);
recibir (proceso_remoto,  parámetros_de_salida);
```


PLANIFICACION DE TRABAJOS Y DEL PROCESADOR

En la planificación del procesador se estudian los problemas de cuándo asignar procesadores y a cuáles procesos asignarlos, también se considera la admisión de nuevas tareas en el sistema, la suspención y reactivación de procesos para ajustar la carga del sistema.

Niveles de planificación:

Planificación de Alto Nivel .- **(Planificación de trabajo)** Determina cuáles trabajos podrán competir activamente por los recursos del sistema o bien cuales trabajos deben admitirse por lo que también la llamamos planificación de admisión.

Planificación de Nivel Intermedio.- Determina que procesos pueden competir por el CPU. Responde a las fluctuaciones temporales en la carga del sistema mediante la suspensión temporal y la activación (reanudación) de procesos para lograr una operación mas fluída del sistema y ayuda a alcanzar ciertas metas globales del rendimiento del sistema. Actúa como amortiguador entre la admisión de los trabajos y la asignación del CPU a esos trabajos.

Planificación de Bajo Nivel.- Determina a cuál proceso listo se le asignará el CPU cuando éste se encuentre disponible y se encargará de asignar el CPU a ése proceso. Se lleva a cabo mediante el despachador y este debe residir en la memoria principal.

Objetivos de la planificación

En el diseño de una disciplina de planificación deben ser considerados varios objetivos, mismos que suelen caer en conflicto haciendo esto un tanto más complejo. Sin embargo a continuación se listan algunos objetivos:

Ser justa.- Una disciplina de planificación es justa si todos los procesos se tratan de la misma forma y ningún proceso se queda en aplazamiento indefinido.

Elevar al máximo la producción y el rendimiento.- Una disciplina de planificación debe de tratar de atender el mayor número posible de procesos por unidad de tiempo.

Aumentar al máximo el número de usuarios interactivos que reciben respuesta en tiempos aceptables (segundos).

Ser predecible.- Una tarea debe ejecutarse aproximadamente en el mismo tiempo y casi al mismo costo sea cual sea la carga del sistema.

Reducir al mínimo el gasto extra.- El gasto extra se considera por lo común un desperdicio de recursos, pero la inversión de cierta porción de los recursos del sistema como gasto extra puede mejorar en gran medida el rendimiento total del sistema.

Equilibrar el aprovechamiento de los recursos.- Los mecanismos de planificación deben mantener ocupados los recursos del sistema. Deben favorecerse los procesos que requieren los recursos poco utilizados.

Lograr un equilibrio entre la respuesta y el aprovechamiento.- La mejor manera de garantizar tiempos de respuesta adecuados es tener suficientes recursos disponibles en el momento en que son necesarios. El precio que debe pagarse por esta estratégia es que el aprovechamiento global de los recursos será pobre. En los sistemas de tiempo real, las respuestas rápidas son esenciales y el aprovechamiento de los recursos es menos importante. En otros tipos de sistemas, la economía exige un aprovachamiento efectivo de recursos.

Evitar el Aplazamiento Indefinido.- La mejor manera de evitarlo es el empleo del *envejecimiento*, es decir, mientras un proceso espera un recurso, su prioridad debe crecer. En algún momento, la prioridad será tan alta que el recurso se asignará al proceso.

Imponer prioridades.- En los ambientes en que se asignan prioridades a los procesos, los mecanismos de planificación deben favorecer a los procesos de alta prioridad.

Dar preferencia a los procesos que ocupan recursos decisivos.- Aunque un proceso tenga baja prioridad, podría estar ocupando un recurso decisivo, y el recurso puede ser requerido por un proceso de alta prioridad. Si el recurso no es apropiable, el mecanismo de planificación debe dar al proceso un trato mejor del que se le daría comunmente, de tal modo que libere el recurso con más rapidez.

Dar un mejor trato a los procesos, muestra un comportamiento deseable. Ejemplo: bajas tasas de paginación.

Degradarse paulatinamente con las cargas pesadas.- Un mecanismo de planificación no debe desplomarse bajo el peso de una carga fuerte en el sistema. Debe evitar la carga excesiva impidiendo la creación de procesos nuevos cuando la carga es pesada, o bien debe dar servicio a la carga mayor con una reducción moderada del nivel de

atención a todos los procesos.

Criterios de planificación

La limitación de un proceso por E/S.- Cuando un proceso obtiene el CPU ¿lo usará en forma breve antes de generar una petición de E/S?

La limitación de un proceso por el CPU.- Cuando un proceso obtiene el CPU, ¿tiende a usarlo hasta que expira su cuanto de tiempo?

Si un proceso es por lotes o es interactivo.- Los usuarios interactivos suelen hacer peticiones "triviales" que deben atenderse de inmediato para garantizar tiempos de respuesta adecuados. Los usuarios por lotes, por lo general pueden tolerar retrasos razonables.

Qué tan urgente es la respuesta.- Un proceso por lotes que tarda toda la noche no requerirá una respuesta inmediata. Un sistema de control de procesos en un tiempo real que supervise a una refinería de petróleo necesita una respuesta rápida, quizá para evitar una explosión.

Las prioridades de los procesos.- Los procesos de alta prioridad deben recibir mejor tratamiento que los de baja prioridad

La frecuencia con la que un proceso está generando fallas de página.- Teoricamente, un proceso que genera pocas fallas de página tiene acumulados sus conjuntos de trabajo en el almacenamiento principal. Pero los procesos que experimentan muchas fallas de página no han establecido aún sus conjuntos de trabajo, y lo convencional es favorecer a los procesos que ya los han establecido. Otro punto de vista es que los procesos con altas tasas de fallas de página deben tener preferencia, ya que usan poco el CPU antes de generar una petición de E/S.

Las frecuencias con las que los recursos de un proceso son apropiados por otro de mayor prioridad.- Los procesos cuyos recursos son apropiados muy a menudo deben recibir un tratamiento menos favorable. La cuestión es que cada vez que el sistema operativo invierte trabajo extra en echar a andar este proceso, el breve lapso de ejecución que se logra antes de la apropiación no justifica el gasto extra necesario para echar a andar el proceso en primer lugar.

Cuanto tiempo real de ejecución ha recibido el proceso.- Algunos diseñadores opinan que deben favorecerse los procesos con poco tiempo de ejecución. Otro punto de vista es que un proceso que ha recibido mucho tiempo

de ejecución debe estar por terminar, debe favorecerse para ayudarlo a que termine y abandone el sistema tan pronto como sea posible

Cuanto tiempo más necesitará el proceso para terminar.- Los tiempos de espera promedio pueden reducirse al mínimo ejecutando primero aquellos procesos que requieran los menores tiempos de ejecución para terminar. Lamentablemente, casi nunca se sabe con exactitud cuanto tiempo durará un proceso.

Planificación no apropiativa y apropiativa

Una disciplina de planificación es *no apropiativa* si una vez que le ha sido asignado el CPU a un proceso, ya no se le puede arrebatar. En los sistemas no apropiativos, los trabajos largos retrasan a los cortos, pero el tratamiento para todos los procesos es más justo. Los tiempos de respuesta son más predecibles porque los trabajos nuevos de alta prioridad no pueden desplazar a los trabajos en espera.

Una disciplina de planificación es *apropiativa* si al proceso se le puede arrebatar el CPU. La planificación apropiativa es importante en aquellos sistemas de planificación en los cuales los procesos de alta prioridad requieren atención rápida. En los sistemas interactivos de tiempo compartido, la planificación apropiativa es importante para garantizar tiempos de respuesta aceptables.
La apropiación tiene un precio. El *cambio de contexto* implica un gasto extra. Para que la técnica de apropiación sea efectiva deben mantenerse varios procesos en el almacenamiento principal de manera que el siguiente proceso se encuentre listo cuando quede disponible el CPU. Conservar en el almacenamiento principal programas que no esten en ejecución también implica gasto extra.
Al hacer el diseño de un mecanismo de planificación apropiativa hay que tomar en cuenta la arbitrariedad de casi todos los sistemas de prioridades. Sin embargo es necesario analizar y evaluar cada mecanismo de planificación antes de ser implantado. La sencillez puede ser atractiva, pero si el mecanismo no se puede hacer sencillo, debe tratarse al menos de hacerlo efectivi y significativo.

Prioridades
Las prioridades pueden ser asignadas en forma automática por el sistema, o bien se pueden asignar externamente. Pueden ganarse o comprarse. Pueden ser estáticas o dinámicas. Pueden asignarse en forma racional, o de manera arbitraria en situaciones en las que un mecanismo del sistema necesita distinguir entre procesos pero no le importa cual de ellos es en verdad más importante.

Prioridades estáticas.- No cambian, son fáciles de llevar a la práctica e implican un gasto extra relativamente bajo. No responden a cambios en el ambiente que podrían hacer necesario un ajuste de prioridades. Es decir se mantienen constantes mientras dura el proceso.

Prioridades dinámicas.- Responden a los cambios. La prioridad inicial asignada a un proceso tiene una duración corta, después de lo cual se ajusta a un valor más apropiado. Este esquema es más complejo e implica más gasto extra que los esquemas estáticos, pero este queda justificado por el aumento de sensibilidad del sistema. Cambian en respuesta a los cambios de las condiciones del sistema.

Prioridades compradas.- Un sistema operativo debe proporcionar un servicio competente y razonable a una gran comunidad de usuarios, pero también debe manejar las situaciones en las cuales un miembro de la comunidad necesite un trato especial. Un usuario con un trabajo urgente puede estar dispuesto a pagar extra, esto es, *comprar prioridad*, por un nivel más alto de servicio. Este pago extra es obligatorio debido a que puede ser necesario arrebatar recursos con otros usuarios que también pagan. Si no hubiera un pago extra, entonces todos los usuarios pedirían un nivel más alto de servicio.

TIPOS DE PLANIFICACION

Planificación a Plazo Fijo.- Se programan ciertos trabajos para terminarse en un tiempo específico. Los trabajos pueden tener un gran valor si son entregados a tiempo y carecer de él si son entregados fuera del plazo, por lo que algún usuario puede estar dispuesto a pagar extra para asegurar que sus trabajos sean entregados a tiempo. Este tipo de planificación es compleja por varias razones que debemos considerar:

- El usuario debe informar por adelantado las necesidades precisas de recursos de la tarea. Esta información no suele estar disponible.
- El sistema debe ejecutar la tarea en un plazo determinado sin degradar el servicio a otros usuarios.
- El sistema debe planificar cuidadosamente sus necesidades de recursos dentro del plazo. Esto puede ser difícil por la llegada de nuevos procesos que impongan demandas impredecibles al sistema.
- Si hay muchas tareas a plazo fijo activas al mismo tiempo podría ser necesario la utilización de métodos de optimización para cumplir con los plazos.
- La administración intensiva de recursos requerida por ésta planificación puede ocasionar un gasto extra sustancial. Aunque los usuarios esten dispuestos a pagar una cuota alta por los servicios recibidos, el consumo neto de los recursos del sistema puede ser tan alto que el resto de la comunidad puede sufrir degradación del servicio.

Planificación Primeras Entradas Primeras Salidas (PEPS o First Input First Output FIFO).- Es la disciplina más simple. Los procesos se despachan de acuerdo a su tiempo de llegada a la cola de procesos listos. Es una disciplina no apropiativa. Es justa en el sentido formal pero es injusta con los procesos cortos que tienen que esperar a que los trabajos largos se ejecuten o bien que los trabajos importantes tienen que esperar a que se terminen los de menor importancia. Es la más predecible de las planificaciones. Sin embargo no es

útil en la planificación para usuarios interactivos porque no puede garantizar buenos tiempos de respuesta.

Planificación por turno (Round Robin [RR]).- Es una disciplina apropiativa. Los procesos se despachan en forma PEPS, pero se les asigna una cantidad limitada de tiempo de CPU conocida como *cuanto o quantum*. Si un proceso no termina antes de que expire su tiempo de CPU, se le quitará el CPU y éste se le asignará al siguiente proceso en espera, el proceso desposeído se colocará al final de la cola de procesos listos. Es efectiva en ambientes de tiempo compartido en los que se necesita garantizar tiempos de respuesta razonables para usuarios interactivos.
El gasto extra debido a la apropiación es bajo gracias a eficientes mecanismos de cambio de contexto y a la asignación de suficiente memoria para que los procesos residan en la memoria al mismo tiempo.

Quantum.- La determinación del tamaño de quantum es vital para lograr una buena utilización del sistema y tiempos de respuesta razonable. Un tamaño de quantum muy grande hará que cualquier disciplina apropiativa se aproxime a su contraparte no apropiativa. Un quantum muy pequeño puede desperdiciar tiempo de CPU al obligar a un excesivo cambio de contexto entre procesos. Por lo que se debe de elegir lo bastante grande para que la mayoría de las solicitudes triviales terminen en un quantum. Ejemplo: en un sistema limitado de E/S, el quantum es lo bastante grande para que la mayor parte de los procesos puedan realizar una petición de E/S antes de que expire su quantum.

Planificación por Prioridad del Trabajo más Corto Primero (Shortest-job-first SJF).- Es una disciplina no apropiativa utilizada sobre todo para trabajos por lotes. Según esta disciplina se ejecuta primero el trabajo (o proceso) en espera que tiene el menor tiempo estimado de ejecución hasta terminar. Reduce al mínimo el tiempo promedio de espera pero los trabajos largos pueden verse sometidos a largas esperas.
El problema obvio con SJF es que exige conocer con exactitud el tiempo que tardará en ejecutarse un trabajo o proceso, y esa información no suele estar disponible; lo mejor que se puede hacer es basarse en los tiempos de ejecución estimados por el usuario.

Planificación del Tiempo Restante mas Corto (Shortest-remaining-time-scheduling SRT).- Es la contraparte apropiativa de SJF. En SRT, el proceso con el menor tiempo estimado de ejecución para terminar es el primero en ejecutarse, incluyendo los procesos nuevos. Un proceso en ejecución puede ser despojado por un proceso nuevo con un tiempo estimado de ejecución mas pequeño; implica un gasto extra mayor que SJF, pero proporciona un mejor servicio a los trabajos nuevos cortos. Reduce más los tiempos promedio de espera de todos los trabajos pero los trabajos largos pueden sufrir retrasos mucho mayores que en SJF.

Planificación por Prioridad de la Tasa de Respuesta mas Alta (Highest-response-ratio-next HRN).- Corrige algunos defectos de SJF, particularmente la excesiva predisposición contra los trabajos largos y el favoritismo de trabajos cortos nuevos. Es un disciplina no apropiativa en la cual la prioridad de cada trabajo no sólo es función del tiempo de servicio, sino también del tiempo que ha esperado el trabajo para ser atendido. Cuando un trabajo

obtiene el procesador, se ejecuta hasta terminar. Las prioridades dinámicas se calculan con la siguiente formula:

$$\text{Prioridad} = \frac{\text{Tiempo de espera + Tiempo de servicio}}{\text{Tiempo de servicio}}$$

Como el tiempo de servicio aparece en el denominador, los trabajos cortos tendrán preferencia. Como el tiempo de espera aparece en el numerador, los trabajos largos que han esperado también tendrán un trato favorable.
El tiempo de respuesta del sistema para el trabajo si este se inicia de inmediato se representa por esta suma:

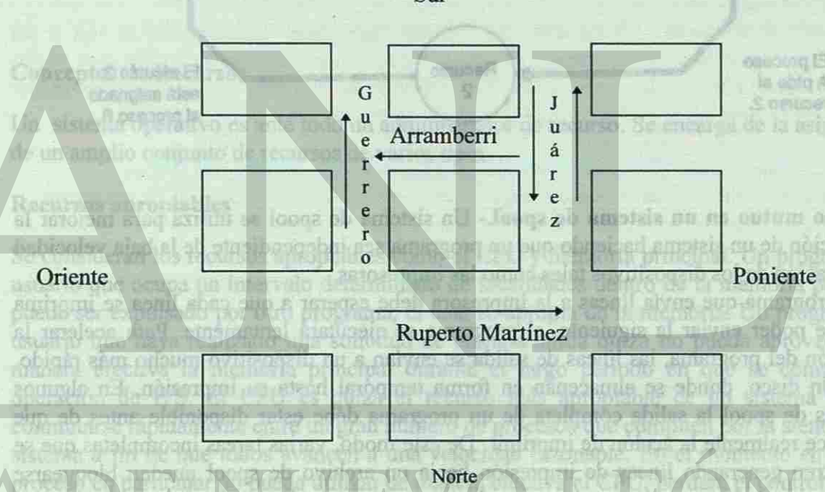
Tiempo de Respuesta = Tiempo de espera + Tiempo de servicio

Unidad III
Bloqueo Mutuo

Objetivo de esta unidad.- Durante el desarrollo de esta unidad, se analizarán los bloqueos de los procesos.
El alumno deberá comprender: ¿Qué es un bloqueo?, las condiciones que lo propician, las áreas de investigación de los bloqueos y temas asociados.

Bloqueos

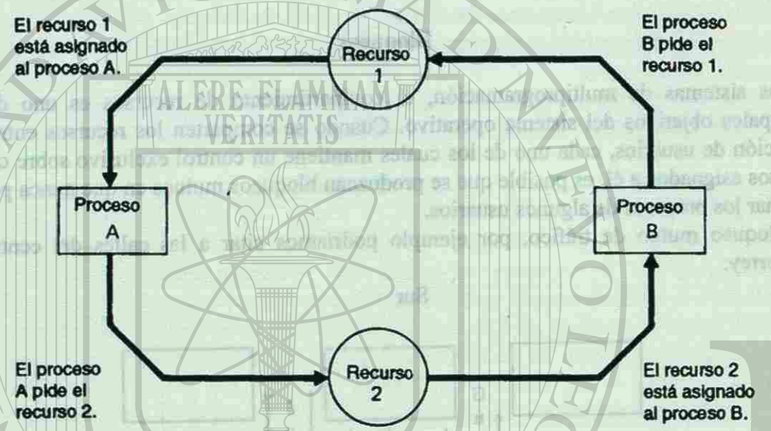
En los sistemas de multiprogramación, el compartimiento de recursos es uno de los principales objetivos del sistema operativo. Cuando se comparten los recursos entre una población de usuarios, cada uno de los cuales mantiene un control exclusivo sobre ciertos recursos asignados a él, es posible que se produzcan bloqueos mutuos en que nunca podrán terminar los procesos de algunos usuarios.
Un bloqueo mutuo de tráfico, por ejemplo podríamos citar a las calles del centro de Monterrey:



El tráfico se detiene por completo, de poco o nada sirven los semáforos controladores del tráfico, siendo necesario la intervención del agente de tránsito para solucionar el embrollo alejando lenta y cuidadosamente los autos y camiones que circulan por el área congestionada. El tráfico comienza a fluir normalmente, no sin antes haber provocado molestias, movilizaciones y una considerable pérdida de tiempo.

La mayor parte de los bloqueos mutuos de los sistemas operativos se presentan a causa de una competencia normal por los recursos dedicados (recursos que sólo pueden ser utilizados por un usuario a la vez, o sea, recursos reutilizables en serie).

Cada proceso espera que el otro libere un recurso que no liberará hasta que el otro libere su recurso, etc. Esta espera circular es característica de los sistemas en bloqueo mutuo. Dado que la tenaz retención de los recursos puede provocar un bloqueo mutuo, este a veces recibe el nombre de abrazo mortal.



Bloqueo mutuo en un sistema de spool.- Un sistema de spool se utiliza para mejorar la producción de un sistema haciendo que un programa sea independiente de la baja velocidad de operación de los dispositivos tales como las impresoras.
Si un programa que envía líneas a la impresora debe esperar a que cada línea se imprima antes de poder enviar la siguiente, el programa se ejecutará lentamente. Para acelerar la ejecución del programa, las líneas de salida se envían a un dispositivo mucho más rápido, como un disco, donde se almacenan en forma temporal hasta su impresión. En algunos sistemas de spool la salida completa de un programa debe estar disponible antes de que comience realmente la acción de imprimir. De este modo, varias tareas incompletas que se encuentren generando líneas de impresión hacia un archivo de spool pueden bloquearse mutuamente si el espacio se termina antes de que acabe cualquiera de las tareas.
La recuperación de un sistema de un bloqueo mutuo de esa naturaleza puede requerir el inicio completo del sistema con la consecuente pérdida de todo trabajo realizado hasta entonces. Si se bloquea de manera que el operador pueda tomar el control, existe la posibilidad de realizar una recuperación menos drástica mediante la desactivación o muerte de una o más tareas hasta disponer del espacio de spool suficiente para que puedan completarse los otros trabajos.
Cuando un programador de sistemas genera un sistema operativo, especifica el espacio para los archivos de spool. Una manera de reducir la posibilidad de un bloqueo mutuo en los sistemas de spool es reservar un espacio mucho mayor para los archivos spool que el considerado indispensable. Si el espacio es insuficiente no siempre es posible esta solución.

Una solución más común es colocar un impedimento en la entrada del spooler para que no acepte más tareas en el momento en que los archivos de spool se acerquen a un *umbral de saturación* que podría ser, un 75% del espacio total por ejemplo. Esto reduciría el rendimiento del sistema que es el precio que hay que pagar por reducir la posibilidad del bloqueo mutuo.
En cualquier sistema que mantenga los procesos en espera mientras se les asigna un recurso o se toman decisiones de planificación, la programación de un proceso puede postergarse indefinidamente mientras otro recibe la atención del sistema. Esta situación se le conoce como Aplazamiento indefinido, Postergación indefinida o Inanición, y es tan peligrosa como un bloqueo mutuo.
Puede ocurrir debido a predisposiciones en las políticas de planificación de recursos del sistema. Cuando los recursos se planifican por prioridad, es posible que un proceso dado espere en forma indefinida un recurso porque siguen llegando otros procesos con mayor prioridad. En algunos sistemas, el aplazamiento indefinido se evita aumentando la prioridad del proceso mientras espera, a esto se le conoce como envejecimiento; en algún momento la prioridad de este superará la prioridad de otros procesos entrantes y el proceso en espera será atendido.

Conceptos de Recursos

Un sistema operativo es ante todo un administrador de recurso. Se encarga de la asignación de un amplio conjunto de recursos de varios tipos.

Recursos apropiables

Se consideran los recursos apropiables como el CPU y memoria principal. Un programa de usuario que ocupa un intervalo determinado de localidades dentro de la memoria principal puede ser expulsado por otro programa, el cual se apropia de la memoria. Un programa de usuario que haya realizado una solicitud de entrada/salida quizá no pueda aprovechar de manera efectiva la memoria principal durante el largo periodo en que se completa la operación de E/S. El CPU es quizá el recurso más apropiable de un sistema y debe conmutarse rápidamente entre un gran número de procesos que compiten por la atención del sistema a fin de que todos avancen a una velocidad razonable. En el momento en que un proceso en particular no pueda utilizar de manera efectiva el CPU, perderá el control de este en favor otro proceso.
La apropiación es de enorme importancia para el éxito de un sistema de computo multiprogramado.

Recursos no apropiables

No pueden arrebatarse al proceso al que han sido asignados.Por ejemplo, las unidades de disco, unidades de cinta que estan asigandas a un proceso en particular por periodos de varios minutos u horas.

Algunos recursos se pueden *compartir* entre varios procesos y otros están dedicados a un solo proceso. Las unidades de disco están dedicadas a un solo proceso, pero a menudo contienen archivos pertenecientes a varios procesos. La memoria principal y el CPU son compartidos por muchos procesos; aunque un solo CPU normalmente sólo puede pertenecer a un proceso a la vez, la conmutación de un CPU entre varios procesos crea la ilusión de un compartimiento simultáneo.

Los datos y programas son sin duda recursos que es necesario controlar y asignar. En los sistemas de multiprogramación, varios usuarios utilizar al mismo tiempo un programa editor, siendo un desperdicio de memoria tener una copia del editor para cada programa, en lugar de ello se lleva a la memoria una sola copia del código y se hacen varias copias de los datos, una para cada usuario. El código no debe cambiar pues muchas personas pueden utilizarlo al mismo tiempo.El código no debe cambiar, pues pueden estarlo usando gran cantidad de personas al mismo tiempo. *El código que no se puede modificar mientras se usa se llama reentrante.* El código reentrante puede ser compartido simultáneamente por varios procesos.

El código que se puede modificar pero que vuelve a su forma original cada vez que se utiliza se llama reutilizable en serie. El reutilizable en serie solo puede ser utilizado por un proceso a la vez.

Cuando se dice que ciertos recursos son compartidos, debe especificarse si van a ser utilizados por varios procesos de manera simultánea o si pueden ser utilizados por varios procesos, pero sólo uno a la vez, siendo estos últimos los recursos que tienden a participar en bloqueos mutuos.

Cuatro condiciones necesarias para que exista un bloqueo mutuo

Exclusión Mutua.- Los procesos exigen un control exclusivo de los recursos que necesitan

Espera.- Los procesos mantienen la posesión de los recursos ya asignados a ellos mientras esperan recursos adicionales.

No apropiación.- Los recursos no pueden arrebatarse a los procesos a los que están asignados hasta que no termine su ejecución o su utilización.

Espera circular.- Existe una cadena circular de procesos en la cual cada proceso tiene uno o más recursos que son requeridos por el siguiente proceso en la cadena.

La existencia de un bloqueo mutuo implica que se han dado todas y cada una de las cuatro condiciones.

Áreas de investigación en los bloqueos mutuos

Prevención.- En la prevención de un bloqueo mutuo vamos a ajustar el sistema para eliminar toda la posibilidad de que ocurra un bloqueo mutuo pero esto puede declinar el aprovechamiento de los recursos o bien empobrecerlos.

Técnicas para evitar.- En las técnicas para evitar el bloqueo mutuo el objetivo es imponer condiciones menos restrictivas que en la prevención para tratar de obtener un mejor aprovechamiento de los recursos. No implica ajustar previamente el sistema para eliminar todas las posibilidades de que se produzca aquel sino que permite la posibilidad de que ocurra el bloqueo mutuo pero se esquiva cuando esta a punto de suceder.

Detección.- Los métodos de detección del bloqueo mutuo se utilizan en sistemas que permiten la ocurrencia de los bloqueos mutuos ya sea de manera voluntaria e involuntaria. El objetivo es determinar si ha ocurrido un bloqueo mutuo y saber exactamente cuales son los procesos y recursos implicados en él, una vez que se determina, el bloqueo mutuo puede eliminarse del sistema.

Recuperación.- Los métodos de recuperación ante un bloqueo mutuo sirven para eliminar los bloqueos mutuos de un sistema para que pueda seguir trabajando y para que los procesos implicados puedan terminar su ejecución y liberen los recursos. Esto viene a ser un problema complejo ya que la recuperación se viene logrando en el mejor de los casos eliminando completamente uno o varios procesos. Después, se inician de nuevo los procesos eliminados, perdiéndose la mayor o todo el trabajop previo realizado por el proceso.

1er Área de Investigación en los Bloqueos Mutuos: Prevención

La técnica más popular más popular empleada por los diseñadores para tratar el bloqueo mutuo es la prevención

J.W. Havender llegó a la conclusión de que si falta una de la cuatro condiciones necesarias no puede haber bloqueo mutuo y para negarlas sugiere lo siguiente:

1. Cada proceso deberá pedir todos sus recursos al mismo tiempo y no podrá continuar su ejecución hasta que no reciba todos.
2. Si a un proceso que tiene ciertos recursos se le niegan los demás, este proceso deberá liberar los recursos y, en caso necesario pedirlos de nuevo junto con los recursos adicionales.
3. Se impondrá un ordenamiento lineal de los tipos de recursos en todos los procesos, esto es, si a un proceso le han asignado un tipo de recursos especifico, en lo sucesivo solo podrá pedir aquellos recursos que siguen en el ordenamiento.

Notese que se presenta tres estrategias y no cuatro debido a que no es deseable impedir la primera de las condiciones (exclusión mutua), porque específicamente queremos permitir la existencia de recursos dedicados.

1[er] Estrategia de Havender.- Negación de la Condición de Espera.

La primera estrategia de Havender requiere que sean pedidos de una sola vez todos los recursos que un proceso necesita. El sistema debe proporcionarlos bajo el principio de todo o nada. Si no esta disponible el conjunto completo de recursos, el proceso debe esperar. Mientras el proceso espera no puede tener ningún recurso, con esto se evita la condición de espera y no puede ocurrir el bloqueo mutuo, pero esto nos va a llevar a un grave desperdicio de recursos.
Ejemplo, considérese el caso en el cual un programa necesita solo una unidad de cinta al comienzo de su ejecución o peor aún ninguna y no necesitará del resto por varias horas. El requisito de que el programa pueda pedir y recibir todos los recursos antes de comenzar la ejecución significa que tendremos recursos importantes ociosos durante horas. Dividir el programa en varios pasos de ejecución de manera relativamente independiente es una técnica empleada con frecuencia para conseguir una mejor utilización de los recursos bajo estas circunstancias. De esta forma, la asignación de recursos puede controlarse por etapas en lugar de hacerlo para todo el proceso, esto reduce efectivamente el desperdicio pero implica un trabajo extra en el diseño del sistema y complica la ejecución, nos puede provocar un aplazamiento indefinido ya que los recursos pueden no estar disponibles al mismo tiempo. El sistema debe permitir que un número suficientes de tareas terminen y liberen sus recursos antes de poder seguir ejecutando el proceso en espera. Mientras se acumulan los recursos no se pueden asignar a otros trabajos, motivo por el cual se desperdician.

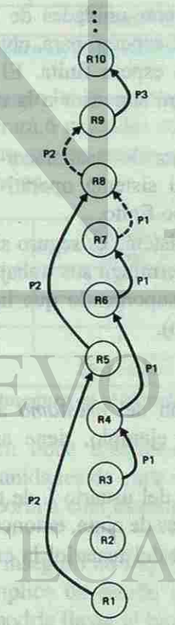
2ª Estrategia de Havender.- Negación de la Condición de No Apropiación.

Supóngase que un sistema permite a un proceso conservar recursos mientras sigue pidiendo otros. En tanto haya suficientes recursos disponibles para satisfacer todas las peticiones, el sistema no puede entrar en bloqueo mutuo. Conmsideremos aquí cuando no se puede atender una petición, en ese momento, el proceso tiene recursos que un segundo proceso puede necesitar para continuar su ejecución, mientras el segundo proceso puede tener recursos que el primero necesita, dando origen a un bloqueo mutuo.
Esta segunda estrategia de Havender requiere que cuando a un proceso que tiene recursos se le niegan recursos adicionales, dicho proceso debe liberar los recursos que tiene y, si es necesario, pedirlos de nuevo junto con los recursos adicionales, anulando efectivamente la condición de no apropiación. Los recursos se le puede quitar al proceso que los tiene antes de la terminación del proceso.
Cuando un proceso libera recursos puede perder el trabajo realizado hasta ese momento lo cual nos puede llevar a pagar un costo demasiado alto. Otra consecuencia seria es el aplazamiento indefinido. Un proceso puede aplazarse indefinidamente mientras pide y libera muchas veces los recursos, si esto llegara a ocurrir el proceso podría eliminarse por el sistema a fin de que otros procesos puedan ejecutarse.

3ª Estrategia de Havender.- Negación de la Condición de Espera Circular

Como todos los recursos tienen una numeración única y como los procesos deben pedir en orden lineal ascendente es imposible que se presente una espera circular.

- Los recursos deben pedirse en orden ascendente por número de recurso. El número de recurso es asignado por la instalación y debe tener un tiempo de vida largo (meses o años); si se agregan nuevos tipos de recursos, puede ser necesario reescribir los programas y sistemas.
- Por lógica, cuando se asigna el número de recurso, estos deben reflejar el orden normal en que los usan la mayoría de las tareas; si espera que las tareas que siguen este orden tengan una operación eficiente. Pero las tareas que necesitan los recursos en forma diferente que el previsto en la instalación, los deberán adquirir y conservar, quizá durante mucho tiempo antes de utilizarlo realmente, lo cual se puede traducir en desperdicio.
- Uno de los objetivos de más importancia en los sistemas operativos actuales es crear ambientes amables con el usuario, los usuarios deben ser capaces de desarrollar sus aplicaciones con un mínimo de interferencia es sus ambientes por las molestas restricciones de hardware y software. El ordenamiento lineal de Havender elimina la posibilidad de una espera circular, pero afecta la capacidad del usuario para escribir libre y fácilmente el código de sus aplicaciones.



2ª ÁREA DE INVESTIGACIÓN EN LOS BLOQUEOS MUTUOS: Técnicas para Evitar el Bloqueo Mutuo.

Si se presentan las condiciones necesarias para que ocurra un bloqueo mutuo, es posible aún evitarlo mediante una cuidadosa asignación de recursos. El algoritmo del banquero de Dijkstra cuyo nombre atañe precisamente las funciones de un banquero que otorga préstamos y recibe pagos a partir de una determinada fuente de capital. Aquí se implementa en el contexto de la asignación de recursos.

El algoritmo del banquero.

El algoritmo del banquero puede generalizarse a lotes de recursos del mismo tipo.

Ejemplo. Consideremos la asignación de una cantidad, *t*, de unidades de cinta idénticas.

- Un sistema operativo comparte un número fijo, *t*, de unidades de cinta equivalente entre un número fijo, *u*, de usuarios. Cada usuario especifica por adelantado el número máximo de unidades de cinta que necesitará durante la ejecución de su trabajo en el sistema.
- El sistema operativo aceptará la petición de un usuario si la *necesidad máxima* de ese usuario no es mayor que *t*.
- Un usuario puede obtener o liberar unidades de cinta una a una. Algunas veces un usuario puede verse obligado a esperar para obtener una unidad adicional, pero el sistema operativo garantiza una espera finita. El número real de unidades de cinta asignadas a un usuario nunca será superrior a la necesidad máxima declarada por ese usuario.
- Si el sistema operativo es capaz de satisfacer la necesidad máxima del usuario, entonces éste debe garantizar al sistema operativo que las unidades de cinta serán utilizadas y liberadas en un tiempo finito.
- Se dice que el estado actual del sistema es seguro si el sistema operativo puede permitir que todos sus usuarios actuales terminen sus trabajos en un tiempo finito, de no ser así el estado es inseguro (estamos suponiendo que las unidades de cinta son los únicos recursos requeridos por el usuario).

Supóngase que existen *n* usuarios.

- Sea *préstamo(i)* la representación del *préstamo* actual de unidades de cinta para el usuario *i*; si el usuario 5, por ejemplo, tiene asignadas cuatro unidades de cinta, entonces *préstamo(5)* = 4.
- Sea *máx(i)* la necesidad máxima del usuario *i*, de manera que si el usuario 3 tiene una necesidad máxima de dos unidades de cinta, entonces *máx(3)* = 2.
- Sea petición(i) la petición actual del usuario, la cual es igual a su máxima necesidad menos su préstamo actual.

Si el usuario 7, por ejemplo tiene una necesidad máxima de 6 unidades y un préstamo actual de 4, entonces

petición(7) = máx(7) - préstamo(7) = 6 - 4 = 2

El sistema operativo controla *t* unidades de cinta.

- Sea *a* el número de unidades de cinta todavía disponibles para asignar. Entonces *a* es igual a *t* menos la suma de todos los préstamos a todos los usuarios.

El algoritmo del banquero permite la asignación de recursos cuando esta conduzca a estados seguros. Un estado seguro es aquel en el que la situación total de recursos es tal que todos los usuarios serán capaces de terminar en algún momento.

Ejemplo de estado seguro.

Supongamos un sistema con doce unidades de cinta y tres usuarios que las comparten.

	Préstamo Actual	Necesidad Máxima
Usuario1	1	4
Usuario2	4	6
Usuario3	5	8
Disponibles = 2		

Estado seguro en el algoritmo del banquero Dijkstra

Es seguro porque es posible que terminen los tres usuarios. Si las unidades disponibles se le asignan al usuario 2, al terminar liberará 6 unidades con lo que es posible que terminen los demás usuarios. Es decir, la clave para que un estado sea seguro es que exista al menos una forma de que terminen todos los usuarios.

	Préstamo Actual	Necesidad Máxima
Usuario1	8	10
Usuario2	2	5
Usuario3	1	3
Disponibles = 1		

Estado inseguro en el algoritmo del banquero Dijkstra

Supongamos el mismo sistema con doce unidades de cinta y tres usuarios que las comparten. En este caso once de las unidades de cinta estan en uso y solo queda una unidad disponible para ser asignada. No importa a cual usuario se le asigne, no se puede garantizar que terminen los tres usuarios.
Es importante señalar que un estado inseguro no implica la existencia, ni siquiera eventual, de un bloqueo mutuo. Lo que sí implica un estado inseguro es simplemente que alguna secuencia desafortunada de eventos podría llevar al bloqueo mutuo.

	Préstamo Actual	Necesidad Máxima
Usuario1	1	4
Usuario2	4	6
Usuario3	5	8
2		

SEGURO

La clave para que un estado sea seguro es que exista al menos una forma de que terminen todos los usuarios.

	Préstamo Actual	Necesidad Máxima
Usuario1	8	10
Usuario2	2	5
Usuario3	1	3
1		

INSEGURO

Un estado inseguro no implica la existencia ni siquiera eventual de un bloqueo mutuo. Lo que si implica un estado inseguro es simplemente que alguna secuencia desafortunada de eventos podría llevarnos a un bloqueo mutuo.

Saber que un estado es seguro no implica que serán seguros todos los estados futuros. La política de asignación de recursos debe considerar todas las peticiones antes de satisfacerlas. Por ejemplo, si en el cuadro del estado seguro en lugar de satisfacer la necesidad del usuario 2 se le otorgaran al usuario 3 o al usuario 1, ya no se podría garantizar la terminación de todos los procesos.

Asignación de recursos con el algoritmo del banquero

Están permitidas las condiciones de espera circular, espera y no apropiación, pero los procesos si exigen el uso exclusivo de los recursos que requieren. Los procesos pueden conservar recursos mientras piden y esperan recursos adicionales. Los recursos no pueden arrebatarse a los procesos que los tienen.
El sistema puede satisfacer o rechazar cada petición. Si una petición es rechazada, el usuario conserva los recursos ya asignados y espera un tiempo finito a que se satisfaga su petición. El sistema solo satisface peticiones que lleven a estados seguros ya que una petición que condujese a un estado inseguro se rechazaría repetidamente hasta que pudiera quedar satisfecha.Como el sistema siempre se mantiene en estado seguro, tarde o temprano todas las peticiones podrán ser atendidas y los usuarios terminaran.

Defectos del algoritmo del banquero

Permite la ejecución de tareas que tendrían que esperar en una situación de prevención del bloqueo mutuo como las siguientes:

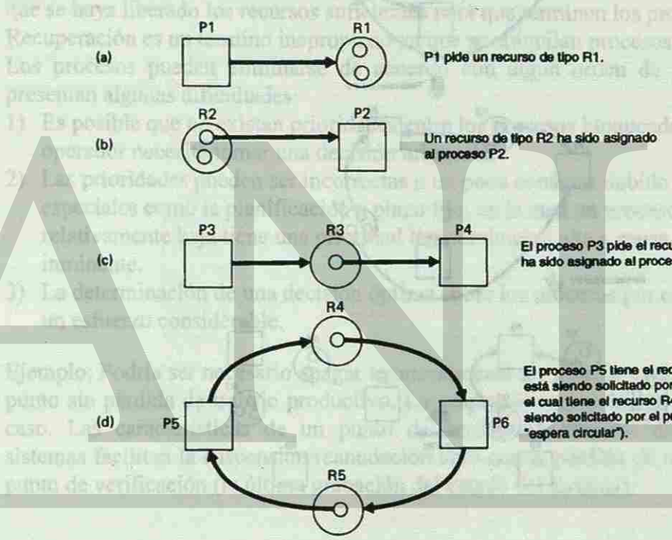
1. El algoritmo requiere un número fijo de recursos asignables.
2. Requiere de una población constante de usuarios.
3. El algoritmo requiere que el banquero satisfaga todas las peticiones en un tiempo finito. Es evidente que los sistemas reales se necesitan mayores garantías.
4. El algoritmo requiere que los clientes (trabajos) salden sus préstamos en un tiempo finito. Se necesitan mayores garantías en los sistemas de tiempo reales.

5. El algoritmo requiere que los usuarios declaren por anticipado sus necesidades máximas.

3ª ÁREA DE INVESTIGACIÓN: DETECCIÓN

La detección del bloqueo mutuo es el proceso de determinar si realmente existe un bloqueo mutuo e identificar los procesos y recursos implicados en él. Los algoritmos de detección del bloqueo mutuo determinan por lo general si existe una espera circular y son demasiado costosos.

Para facilitar la detección de bloqueos mutuos, se utilizará una notación en la cual una gráfica dirigida indica las asignaciones y peticiones de recursos. Los cuadrados representan procesos, los círculos grandes, clases de dispositivos idénticos; los círculos pequeños en el interior de los grandes indican el número de dispositivos de cada clase.



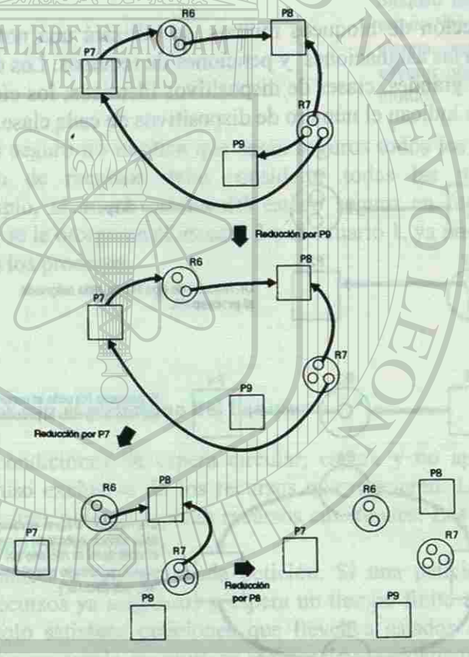
Las gráficas de asignación y solicitud de recursos cambian cuando los procesos piden recursos, los adquieren y luego los devuelven al sistema operativo.

Reducción de las Gráficas de Asignación de Recursos

Una técnica útil para detectar los bloqueos mutuos implica reducciones de gráficas en las que se determinan los procesos que pueden terminar su ejecución y los que se encuentran en bloqueo mutuo.

Si pueden atenderse las peticiones de recursos de un proceso, se dice que la gráfica puede ser reducida por ese proceso. Es equivalente a mostrar la gráfica como si el proceso hubiera terminado su ejecución y devueltos los recursos al sistema. La reducción se muestra eliminando las flechas de los recursos hacia el proceso eliminando las flechas de ese proceso hacia los recursos.

Si una gráfica puede ser reducida por todos sus procesos, entonces no hay bloqueo mutuo.
Si una gráfica no puede ser reducida por todos sus procesos, los procesos irreducibles constituyen el conjunto de procesos en bloqueo mutuo de la gráfica.



Es importante señalar aquí que no importa el orden en el cual se realizan las reducciones el resultado final es siempre el mismo.

3ª ÁREA DE INVESTIGACIÓN: Recuperación

El bloqueo mutuo debe romperse mediante la eliminación de uno o más de las condiciones necesarias. Por lo general, varios procesos perderán una parte o la totalidad del trabajo que se ha efectuado pero es el precio pagado que puede ser pequeño comparado con las consecuencias de permitir que nuestro proceso siga bloqueado. La recuperación después de un bloqueo mutuo puede ser complicada debido a:

1. Puede no estar claro que el sistema se haya bloqueado.
2. La mayor parte de los sistemas tienen medios muy deficientes para suspender indefinidamente un proceso, eliminarlo del sistema y reanudarlo más tarde. De hecho, algunos procesos como los de tiempo real, que deben funcionar en forma continua, sencillamente no se pueden suspender y reanudar.
3. Aún cuando existieran medios efectivos de suspensión/reanudación, con toda seguridad implicarían un gasto extra considerable y requerirían la atención de un operador altamente calificado. No siempre es posible encontrar un operador así.
4. La recuperación después de un bloqueo mutuo de dimensiones modestas puede significar una cantidad razonable de trabajo, en tanto que un bloqueo mutuo a gran escala puede requerir una cantidad enorme de trabajo.

En los sistemas actuales, la recuperación se suele realizar eliminando un proceso y arrebatándole sus recursos. Generalmente, el proceso eliminado se pierde, pero es posible concluir los procesos restantes. Algunas veces es necesario eliminar varios procesos hasta que se haya liberado los recursos suficientes para que terminen los procesos restantes.
Recuperación es un termino inapropiado ya que se aniquilan procesos en beneficio de otros.
Los procesos pueden eliminarse de acuerdo con algún orden de prioridad, también se presentan algunas dificultades:

1) Es posible que no existan prioridades entre los procesos bloqueados, de modo que el operador necesita tomar una decisión arbitraria.
2) Las prioridades pueden ser incorrectas o un poco confusas debido a consideraciones especiales como la planificación a plazo fijo, en la cual un proceso de prioridad relativamente baja tiene una prioridad temporalmente alta a causa de un fin de plazo inminente.
3) La determinación de una decisión óptima sobre los procesos por eliminar puede requerir un esfuerzo considerable.

Ejemplo: Podría ser necesario apagar temporalmente un sistema y reiniciarlo a partir de ese punto sin pérdida de trabajo productivo. La suspención/reanudación seria muy útil en este caso. Las características de un punto de verificación/reinicio disponibles en muchos sistemas facilitan la suspensión/reanudación sólo con la pérdida de trabajo desde el último punto de verificación (la última grabación del estado del sistema).

Unidad IV

ALMACENAMIENTO REAL

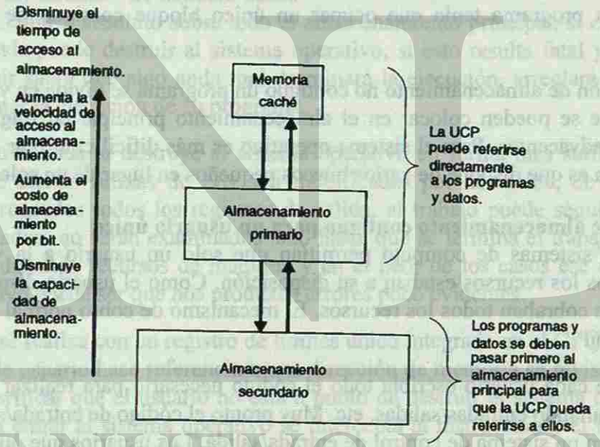
Objetivo de esta unidad.- Durante el desarrollo de esta unidad, se analizarán las técnicas más comunes en las asignaciones y relocalizaciones de programas y datos en memoria principal.

El alumno deberá entender las técnicas para el manejo de memoria real y su aplicación para sistemas operativos de un solo usuario o para sistemas de memoria virtual; deberá comprender los conceptos de páginación, segmentación y memoria asociativa, para entender el funcionamiento de la memoria virtual en un sistema operativo.

Por sencillez consideremos: Memoria = Almacenamiento

Memoria Principal	=	Memoria Real	=	Memoria Primaria

Organización de almacenamiento se entiende que es la forma en que se considera el almacenamiento principal.



Estrategias de Administración de Almacenamiento

1. Estrategias de Obtención
2. Estrategias de Colocación
3. Estrategias de Reemplazo

Estrategias de Obtención.- Determinan cuando debe obtenerse la siguiente parte del programa o los datos que se van a transferir del almacenamiento secundario al principal.

La estrategia convencional es la *obtención por demanda* en la cual la siguiente parte del programa o de los datos se

transfiere al almacenamiento principal cuando un programa en ejecución hace referencia a ella.
Se tiene la creencia de que como generalmente no se puede predecir hacia donde pasara el control de un programa, el trabajo extra para hacer superposiciones y anticipar el futuro superaría por mucho los beneficios esperados aunque se dice que mejorará el rendimiento de los sistemas la *obtención anticipada.*

Estrategias de Colocación.- Tienen que ver con la determinación de la parte del almacenamiento principal donde se colocara un programa entrante (Primer ajuste, Mejor ajuste y Peor ajuste).

Estrategias de Reemplazo.- Están relacionadas con la determinación de qué parte del programa o de los datos se debe desalojar para dejar espacio a los programas entrantes.

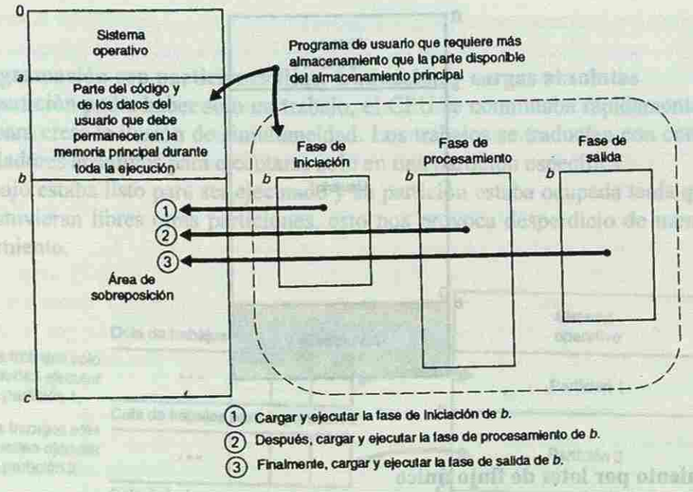
Asignación de almacenamiento contiguo y no contiguo
Los primeros sistemas de computo requerían una asignación de almacenamiento contiguo, es decir, cada programa tenia que ocupar un único bloque contiguo de localidades de memoria.

En la asignación de almacenamiento no contiguo un programa se divide en varios bloques o segmentos que se pueden colocar en el almacenamiento principal en fragmentos que no necesitan ser adyacentes. Para el sistema operativo es más difícil controlar esta asignación pero la ventaja es que dispone de varios huecos pequeños en lugar de un solo hueco grande.

Asignación de almacenamiento contiguo para un usuario único
Los primeros sistemas de computo permitían que solo un usuario a la vez utilizara la máquina, todos los recursos estaban a su disposición. Como el usuario disponía de toda la máquina, se le cobraban todos los recursos. El mecanismo de cobro normal se basaba en el tiempo de reloj.

Originalmente cada usuario escribía todo el código necesario para realizar una aplicación especifica detallando entradas/salidas, etc. Muy pronto el código de entrada/salida requerido se consolidó en un sistema de control de entrada/salida. Los usuarios que querían hacer E/S ya no tenían la necesidad de codificar directamente las instrucciones, sino que llamaban a la rutina de IOCS para que ejecutara el trabajo.

El tamaño de los programas está limitado por la cantidad de memoria principal pero es posible ejecutar programas más grandes utilizando el concepto de superposiciones. Si una sección de un programa no se necesita durante el resto de la ejecución del programa se puede traer del almacenamiento secundario otra sección para ocupar el espacio utilizado por la que ya no se necesite.

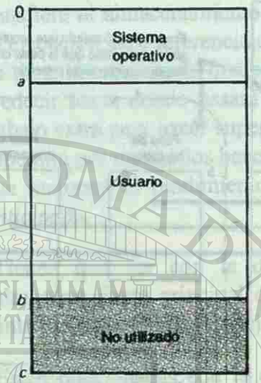


Protección de los sistemas de usuario único
Como se tiene control absoluto sobre todo el almacenamiento principal, si el programa del usuario se desvía puede destruir al sistema operativo, si esto resulta fatal y el usuario no puede proseguir sabrá que algo anda mal, terminará la ejecución, arreglará el problema y volverá a intentar la ejecución de su programa.

Supóngase que el usuario destruye al sistema operativo en forma más sutil, por ejemplo, supóngase que ciertas rutinas de E/S son modificadas por accidente, de tal modo que pueden estar truncados todos los registros de salida, el trabajo puede seguir su ejecución pero los resultados no serán examinados sino hasta que se termina el trabajo, entonces se estarán desperdiciando recursos de máquina y en el peor de los casos ese daño a nuestro sistema operativo puede ser que nos produzca errores poco evidentes.

La protección se realiza con un registro de límites único integrado en el CPU. Cada vez que un programa de usuario hace referencia a una dirección de memoria se revisa el registro de límites para verificar que el usuario no esta a punto de destruir el sistema operativo. Si el usuario intenta entrar al sistema operativo se intercepta la instrucción y el trabajo termina con un mensaje de error apropiado.

El usuario necesita entrar de vez en cuando al sistema operativo para utilizar ciertos servicios, como por ejemplo la E/S, el problema se resuelve dando al usuario una instrucción específica con la cual puede solicitar los servicios del sistema operativo, es decir, una llamada al supervisor.



Procesamiento por lotes de flujo único

Los trabajos en general requerían un tiempo de preparación se cargaba el sistema operativo, se montaban las cintas y los discos, se preparaban impresoras, etc y también requerían de largos tiempos de descarga. El computador estaba ocioso todo ese tiempo. Los diseñadores se dieron cuenta que podían optimizar esos tiempos de transición entre procesos o trabajos pudiendo reducir la cantidad de tiempo que se desperdiciaba en los trabajos, esto condujo al desarrollo de los sistemas de procesamiento por lotes (BATCH).

En el procesamiento por lotes de flujo único, los trabajos se agrupan en lotes cargándolos consecutivamente en cinta o en disco. Un procesador de flujo de trabajos lee las instrucciones en JCL (Job Control Languaje) y facilita la preparación para el siguiente trabajo. Cuando termina un trabajo, el lector de flujo de trabajos lee automáticamente las instrucciones del lenguaje de control para el siguiente trabajo y realiza las acciones de mantenimiento apropiadas para facilitar la transición del siguiente trabajo.

Multiprogramación de particiones fijas

Los diseñadores ven una vez más que se puede aprovechar el CPU notablemente mediante una administración intensiva. Esta vez decidieron administrar sistemas de multiprogramación en los cuales los usuarios compiten al mismo tiempo con los recursos del sistema.

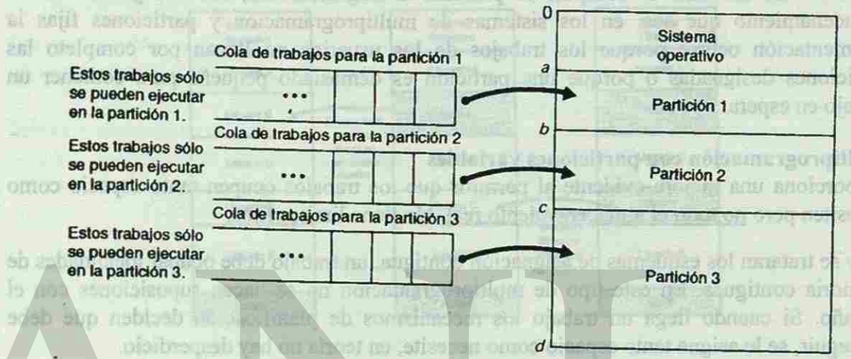
El trabajo que esta esperando la terminación de un proceso de E/S cederá el CPU a otro trabajo que este listo para realizar cálculos si es que hay uno en espera, de este modo pueden efectuarse simultáneamente las operaciones de E/S en los cálculos del CPU.

Para aprovechar al máximo la multiprogramación es necesario que varios trabajos residan al mismo tiempo en el almacenamiento principal del computador, de este modo cuando un trabajo solicita E/S el CPU puede conmutarse a otro y realizar cálculos sin retraso.

Multiprogramación con particiones fijas: traducción y cargas absolutas

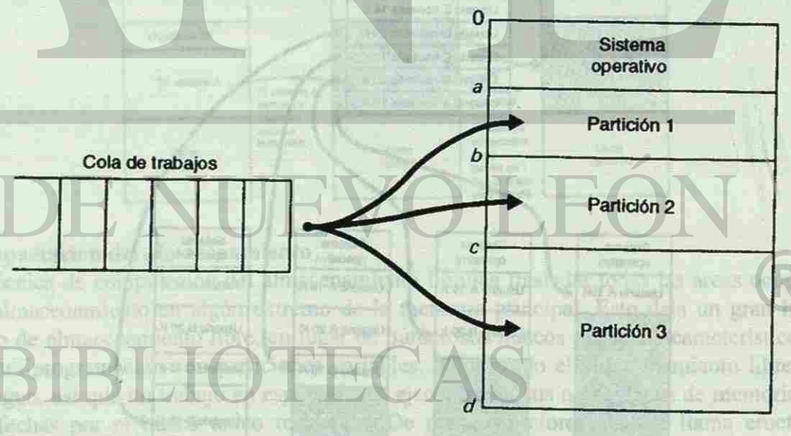
En cada partición podía haber solo un trabajo, el CPU se conmutaba rápidamente entre los usuarios para crear la ilusión de simultaneidad. Los trabajos se traducían con compiladores y ensambladores absolutos para ejecutarse solo en una partición específica.

Si un trabajo estaba listo para ser ejecutado y su partición estaba ocupada tenía que esperar aunque estuvieran libres otras particiones, esto nos provoca desperdicio de memoria o de almacenamiento.



Multiprogramación con particiones fijas: traducción y carga con reubicación

Los compiladores, ensambladores y cargadores con reubicación sirven para producir programas que se puedan ejecutar en cualquier partición disponible que sea lo suficientemente grande para contenerlos.



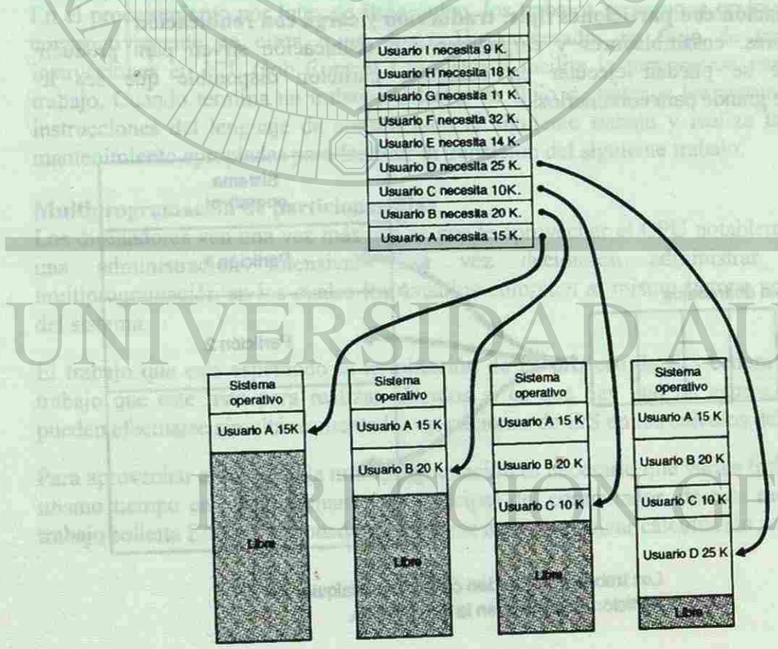
Los trabajos se pueden colocar en cualquier partición disponible en la que quepan.

Protección en los sistemas con multiprogramación
Esto se logra a menudo con varios registros límites. Con dos registros se pueden establecer los límites superior e inferior de la partición de un usuario o bien el límite superior o el inferior y la longitud de la región. El usuario que necesita llamar al sistema operativo utiliza una instrucción de llamada al supervisor para hacerlo, esto permite al usuario cruzar el limite del sistema operativo y solicitar sus servicios sin poner en peligro la seguridad del sistema operativo.

Fragmentación en la multiprogramación con particiones fijas
En todos los sistemas de computo se presenta la fragmentación, sea la organización de almacenamiento que sea, en los sistemas de multiprogramación y particiones fijas la fragmentación ocurre porque los trabajos de los usuarios no llenan por completo las particiones designadas o porque una partición es demasiado pequeña para contener un trabajo en espera.

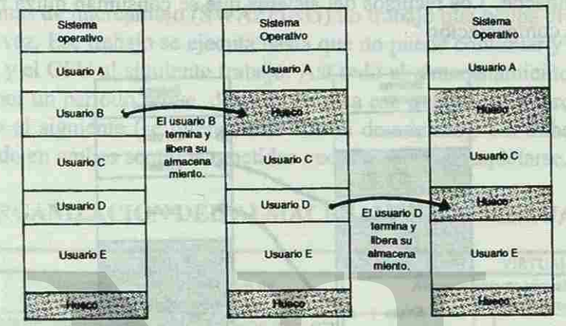
Multiprogramación con particiones variables
Proporciona una mejora evidente al permitir que los trabajos ocupen tanto espacio como necesiten pero no todo el almacenamiento real. No tiene limites fijos.

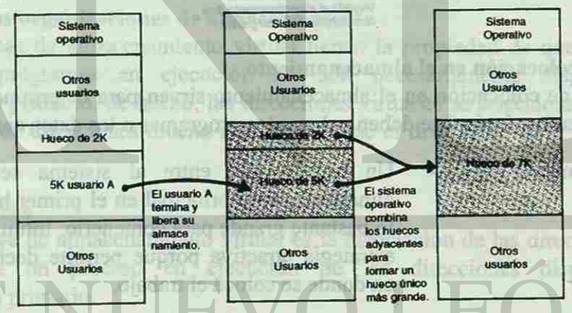
Solo se trataran los esquemas de asignación contigua, un trabajo debe ocupar localidades de memoria contiguas. En este tipo de multiprogramación no se hacen suposiciones con el tamaño. Si cuando llega un trabajo los mecanismos de planificación deciden que debe proseguir, se le asigna tanto espacio como necesite, en teoría no hay desperdicio.



Condensación de huecos
Cuando termina un trabajo en un sistema de multiprogramación con particiones variables se puede comprobar si el almacenamiento liberado colinda con otras áreas libres de almacenamiento, comúnmente denominados huecos.

El proceso de fusionar huecos adyacentes para formar un solo hueco más grande se le denomina condensación. Mediante la condensación de huecos se puede recuperar los bloques contiguos de almacenamiento más grande que sea posible.

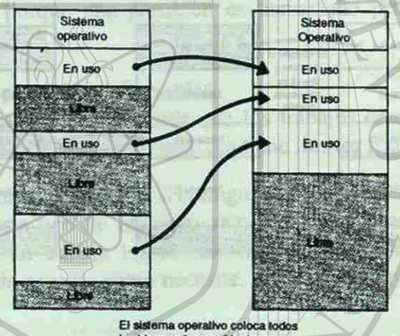




Compactación del almacenamiento
La técnica de computación del almacenamiento implica trasladar todas las áreas ocupadas del almacenamiento en algún extremo de la memoria principal. Esto deja un gran hueco único de almacenamiento libre, en lugar de números huecos pequeños característicos de la multiprogramación con particiones variables. Ahora todo el almacenamiento libre esta contiguo, así que un trabajo en espera puede ejecutarse si sus necesidades de memoria son satisfechas por el hueco único resultante. De manera pintoresca se le llama eructo de almacenamiento (Burping Storage) o recolección de basura (Garbage Collection).

Inconvenientes:

- Consume recursos del sistema que podrían utilizarse en forma más productiva.
- El sistema debe detener todas sus actividades mientras realiza la compactación provocando tiempos de respuesta irregulares para usuarios interactivos y podría ser devastador en sistemas de tiempo real.
- La compactación implica reubicar los trabajos que estén en el almacenamiento. Esto significa que la información requerida para realizar la reubicación que se pierde por lo regular cuando se carga un programa, debe mantenerse accesible ahora.
- Con una combinación normal de trabajos que cambia rápidamente, a menudo es necesario compactar. Los recursos del sistema que se consuman quizá no justifiquen las ventajas de la compactación



El sistema operativo coloca todos los bloques "en uso" juntos, dejando el almacenamiento libre como un hueco único grande.

Estrategias de colocación en el almacenamiento

Las estrategias de colocación en el almacenamiento sirven para determinar en qué lugar del almacenamiento principal se deben colocar los programas o los datos entrantes.

Estrategia del primer ajuste.- Un trabajo que entre al sistema se coloca en el almacenamiento principal en el primer hueco disponible lo bastante grande para contenerlo. Intuitivamente es una estrategia atractiva porque permite decidir con rapidez en donde se coloca el trabajo.

Estrategia del mejor ajuste.- Un trabajo que entre al sistema se colocara en el almacenamiento principal en el hueco que quepa mejor y quede la menor cantidad posible de espacio sin utilizar. Intuitivamente es la estrategia más atractiva.

Estrategia del peor ajuste.- A primera vista es una elección extravagante, pero tras un examen cuidadoso también posee su atractivo. El peor ajuste consiste en colocar un programa en el almacenamiento primario en el hueco donde peor se ajusta, es decir, en el hueco más grande. Su atractivo de manera sencilla consiste

en que después de colocar el programa en ese gran hueco, el hueco restante también será grande y por consiguiente también podrá contener un nuevo programa relativamente grande.

Una variante de la estrategia del primer ajuste es la llamada estrategia del siguiente ajuste. Comienza cada búsqueda de un hueco disponible en el lugar donde termino la búsqueda anterior.

Multiprogramación con intercambio de almacenamiento

En algunos sistemas de intercambio (SWAPPING) un trabajo que ocupa el almacenamiento principal de una vez. Ese trabajo se ejecuta hasta que no puede continuar y entonces cede el almacenamiento y el CPU al siguiente trabajo. Así todo el almacenamiento esta dedicado a un solo trabajo por un periodo breve, después se saca ese trabajo (se intercambia con el de fuera) y se carga el siguiente (se intercambia con el de adentro). Un trabajo normalmente será intercambiado en ambos sentidos repetidas veces antes de completarse.

ORGANIZACION DEL ALMACENAMIENTO VIRTUAL

<table>
<tr><td>REAL</td><td colspan="3">REAL</td><td colspan="3">VIRTUAL</td></tr>
<tr><td>Sistemas dedicados a un solo usuario</td><td colspan="3">Almcenamiento real en sistemas de multiprogramación</td><td colspan="3">Almacenamiento virtual en sistemas de multiprogramación</td></tr>
<tr><td></td><td colspan="2">Multiprogramación con particiones fijas</td><td>Multiprogramación con particiones variables</td><td>Páginación pura</td><td>Segmentación pura</td><td>Páginación/ Segmentación combinada</td></tr>
<tr><td></td><td>Absolutas</td><td>Reubicables</td><td></td><td></td><td></td><td></td></tr>
</table>

Evolución de las organizaciones de almacenamiento

Todos lo sistemas de almacenamiento virtual tienen la propiedad de que las direcciones calculadas o programas en ejecución no son necesariamente disponibles en el almacenamiento primario, de hecho, las direcciones virtuales se seleccionan por lo regular de un conjunto de direcciones mucho más grande que el disponible en el almacenamiento primario.

Conceptos básicos

El concepto clave de almacenamiento virtual es la disociación de las direcciones a las que hace referencia un proceso en ejecución de las direcciones disponibles en el almacenamiento primario.

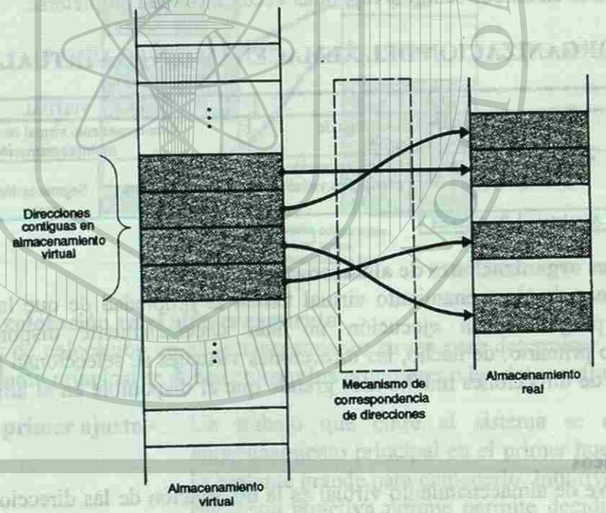
Las direcciones a las que hace referencia un proceso en ejecución se conocen como direcciones virtuales. Las direcciones disponibles en el almacenamiento primario se conocen como direcciones reales.

El intervalo de las direcciones virtuales a las que se puede hacer referencia en un proceso en ejecución se le conoce como espacio de direcciones virtuales V. El intervalo de direcciones reales disponibles en un sistema de computo se conoce como espacio de direcciones reales R del computador.

El número de direcciones en V se denota como | V |. El número de direcciones en R se denota como | R |.

Aunque los procesos solo hacen referencia a direcciones virtuales deben ejecutarse en almacenamiento real. Por lo tanto, es preciso establecer la correspondencia entre direcciones reales y direcciones virtuales durante la ejecución de un proceso. Se han desarrollado varios métodos para asociar las direcciones virtuales con las reales. Los mecanismos de traducción dinámica de direcciones (Dynamic Address Translation) convierten las direcciones virtuales en direcciones reales mientras se ejecuta un proceso.

Todos estos sistemas tienen la propiedad de que las direcciones contiguas en el espacio de direcciones virtuales de un proceso no son necesariamente contiguas en el almacenamiento real, a esto se le conoce como **contigüidad artificial**.



Organización en almacenamiento con múltiples niveles

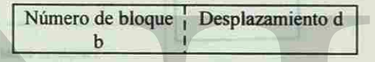
Si se va a permitir que el espacio de direcciones virtual del usuario que sea más grande que el espacio de direcciones reales y sobre todo si se va a multiprogramar un sistema en el cual muchos usuarios comparten el recurso del almacenamiento real hay que proporcionar lo necesario para proporcionar lo necesario para obtener los programas y datos en un almacenamiento auxiliar grande.

El primer nivel es el almacenamiento real en el cual se ejecutan los procesos y en el cual deben de encontrarse los datos para que un proceso en ejecución pueda hacer referencia a ellos. El segundo nivel consiste en medios de almacenamiento de gran capacidad como discos y tambores capaces de almacenar programas que no quepan en el almacenamiento real en un momento dado, se conoce también como almacenamiento auxiliar o secundario.

Correspondencia de bloques

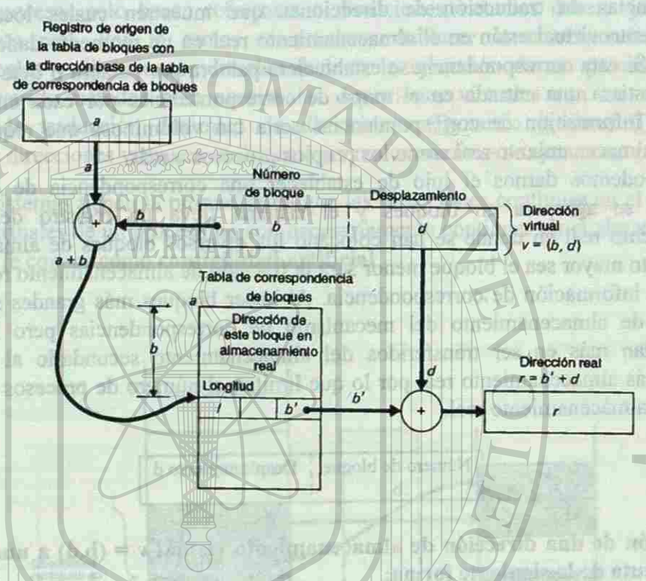
Los mecanismos de traducción dinámica de direcciones deben tener mapas de correspondencias de traducción de direcciones que muestren cuales localidades del almacenamiento virtual están en el almacenamiento real en un momento dado y donde se encuentran. Si esta correspondencia se estableciera palabra por palabra o byte por byte, es decir, si existiera una entrada en el mapa de correspondencias por cada entrada en V, entonces la información de correspondencias seria tan voluminosa que requeriría tanto espacio del almacenamiento real como los propios procesos o más.

Como no podemos darnos el lujo de establecer una correspondencia de este tipo la información se agrupara en bloques y el sistema lleva un registro del lugar del almacenamiento real en donde se han colocado los diversos bloques de almacenamiento virtual, cuanto mayor sea el bloque menor será la fracción de almacenamiento real dedicada a guardar la información de correspondencia. Al hacer bloques más grandes se reduce el gasto extra de almacenamiento del mecanismo de correspondencias pero los bloques grandes tardan más en ser transferidos del almacenamiento secundario al primario y consumen más almacenamiento real por lo que limitan el número de procesos que pueden compartir el almacenamiento real.



La traducción de una dirección de almacenamiento virtual v = (b,d) a una dirección real r se ejecuta de la siguiente forma:

1. Cada proceso tiene su propia tabla de correspondencia de bloques mantenida por el sistema dentro del almacenamiento real.
2. Un registro especial dentro de la unidad de procesamiento llamado registro de origen de la tabla de correspondencia de bloques se carga con la dirección real, *a*, de la tabla de correspondencia de bloques del proceso durante la conmutación de contexto (cambio de contexto).
3. La tabla contiene una entrada por cada bloque del proceso y las entradas siguen un orden secuencial (b_0, b_1, b2,...,etc.)
4. Ahora se suma el número de bloque "b" a la dirección base "*a*" de la tabla de bloques para formar la dirección real de la entrada del bloque "b" en la tabla de correspondencia de bloques. Esta entrada contiene la dirección "b" del inicio del bloque "b".
5. El desplazamiento "d" se suma a la dirección de inicio del bloque "b" para formar la dirección real deseada r = (b' + d).



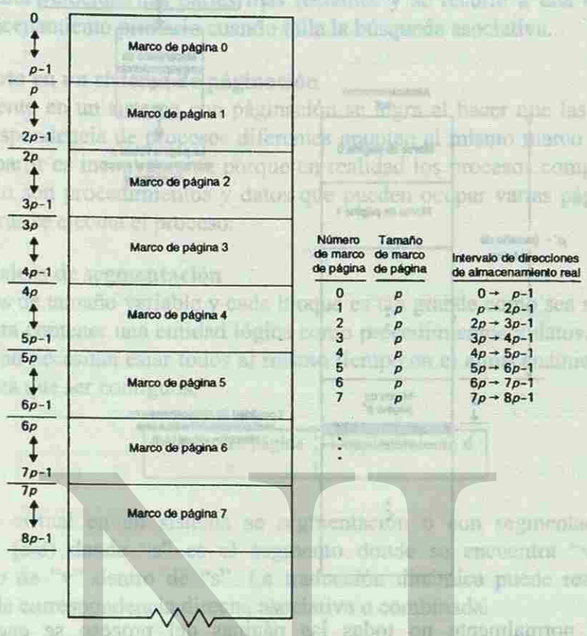
Conceptos básicos de páginación

Los bloques de tamaño fijo se llaman páginas y la organización del almacenamiento vitual correspondiente se le conoce como paginación. Una dirección virtual en un sistema de páginación es un par ordenado (p,d) en el cual "p" es el número de página del almacenamiento virtual en el que reside el elemento al que se hace referencia y "d" es el desplazamiento dentro de la página "p" donde se localiza dicho elemento.

Número de página p	Desplazamiento d

v = (p,d) dirección virtual

Un proceso puede ejecutarse si su página activa se encuentra en el almacenamiento primario. Las páginas se transfieren del almacenamiento secundario al primario y se colocan dentro de bloques de llamadas marcos de página que tienen el mismo tamaño.

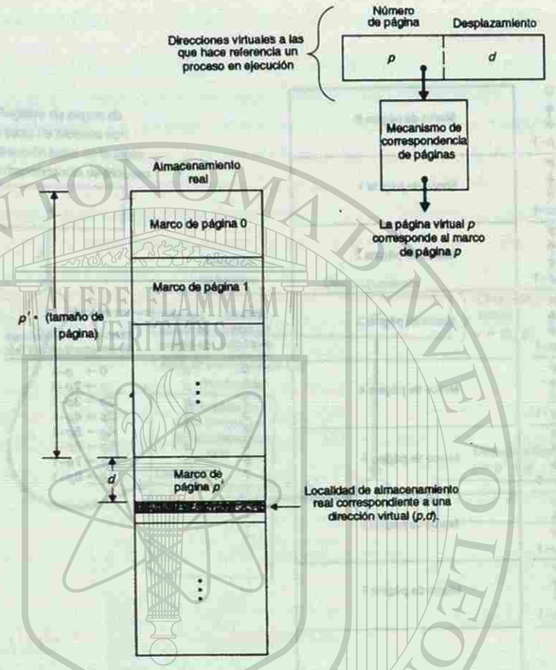


Número de marco de página	Tamaño de marco de página	Intervalo de direcciones de almacenamiento real
0	p	0 → p-1
1	p	p → 2p-1
2	p	2p → 3p-1
3	p	3p → 4p-1
4	p	4p → 5p-1
5	p	5p → 6p-1
6	p	6p → 7p-1
7	p	7p → 8p-1

Los marcos de página comienzan en direcciones de almacenamiento real que son múltiplos enteros de tamaño fijo de la página. Una página entrante puede colocarse en cualquier marco de página disponible.

La traducción dinámica de direcciones en la páginación se realiza de la siguiente manera:

Un proceso en ejecución hace referencia a una dirección de almacenamiento virtual v = (p,d). Un mecanismo de correspondencia de páginas busca la página "p" en una tabla de correspondencias de páginas y determina que la página "p" se encuentra en el marco de página " p' ". La dirección del almacenamiento real se forma concatenando " p' " y "d".



Debido a que normalmente no todas las páginas del proceso se encuentran en el almacenamiento primario al mismo tiempo la tabla de correspondencia de páginas debe indicar si la página a la que se hace referencia se encuentra o no en el almacenamiento primario, si sí esta donde se encuentra y si no donde puede encontrarse en el secundario.

r	s	p'

r = bit de residencia de página
s= dirección en almacenamiento secundario
p'= número de página

Un bit de residencia de página "r" tiene valor cero si la página no se encuentra en el almacenamiento primario y valor 1 en caso contrario, si la página no se encuentra en el almacenamiento primario, entonces "s" es su dirección en el almacenamiento secundario, si la página se encuentra en el almacenamiento secundario, si la página se encuentra en el almacenamiento primario, entonces " p' " es un número de marco de página.

La traducción dinámica de direcciones en la páginación implica la correspondencia entre el número de página "p" y el marco de página " p' ". La correspondencia puede ser directa en cuyo caso se mantiene en una tabla de correspondencia de páginas completa en el almacenamiento primario o en una memoria cache de acceso rápido.

La correspondencia puede ser también puramente asociativa en cuyo caso la tabla de correspondencia se mantiene en un almacenamiento asociativo de acceso rápido.

Debido al alto costo de los almacenamientos asociativos y caché la correspondencia puede ser una combinación asociativa directa donde solo se mantienen en el almacenamiento asociativo de alta velocidad las partes más recientes y se recurre a una correspondencia directa en almacenamiento primario cuando falla la búsqueda asociativa.

Compartimiento en un sistema de páginación

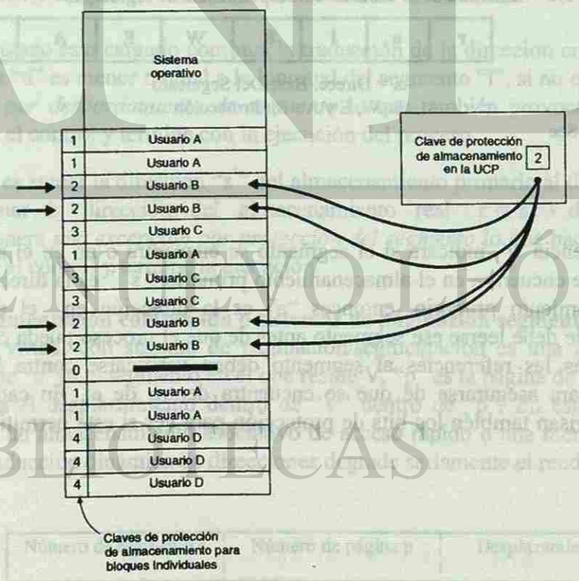
El compartimiento en un sistema con páginación se logra al hacer que las entradas de las tablas de correspondencia de procesos diferentes apunten al mismo marco de página. Esta forma de compartir es inconveniente porque en realidad los procesos comparten entidades lógicas como lo son procedimientos y datos que pueden ocupar varias páginas y crecer o decrecer mientras se ejecuta el proceso.

Conceptos básicos de segmentación

Emplea bloques de tamaño variable y cada bloque es tan grande como sea necesario dentro de su límite para contener una entidad lógica como procedimientos y datos. Los segmentos de un proceso no necesitan estar todos al mismo tiempo en el almacenamiento principal ni tienen por fuerza que ser contiguos.

Número de página s	Desplazamiento d

Una dirección virtual en un sistema se segmentación o con segmentación es un par ordenado v = (s,d) donde "s" es el segmento donde se encuentra "v" y "d" es el desplazamiento de "v" dentro de "s". La traducción dinámica puede realizarse con los mismos tipos de correspondencia directa, asociativa o combinada.



La protección en los sistemas con segmentación es más natural que en la páginación ya que se protegen entidades lógicas no físicas. A los procesos se le otorgan varias combinaciones de acceso de lectura, escritura, ejecución y adición a los distintos segmentos.

Tipos de control de acceso

Tipos de acceso	Abreviatura	Explicación
Lectura	R	Este segmento se puede leer
Escritura	W	Este segmento se puede modificar
Ejecución	E	Este segmento se puede ejecutar
Adición	A	Se puede agregar información al final de este segmento

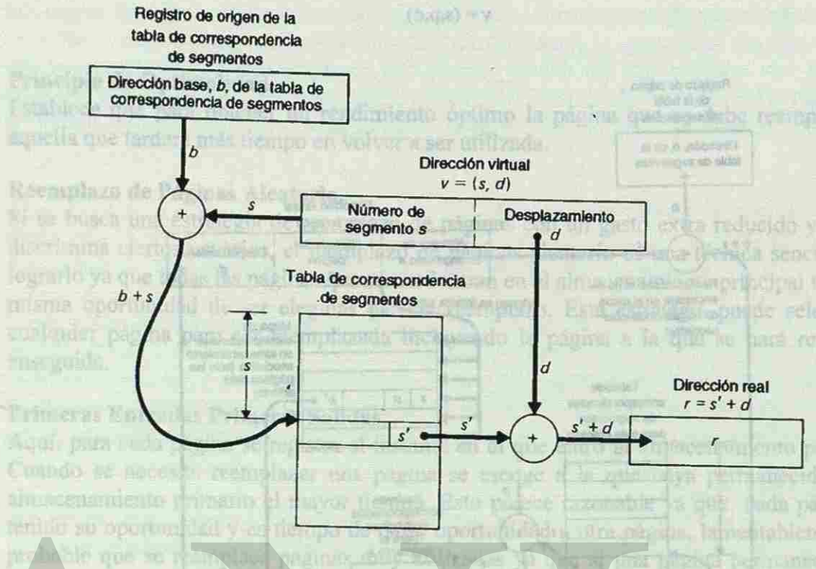
Entrada de la tabla de correspondencia de segmentos

r	a	l	R	W	E	A	s´

r = Bit Residencia
l = Long. Segmento
a = Direcc. Almac. Sec

s´= Direcc. Base Del Segmento
R, W, E y A= Bit Protección

Un bit de residencia "r" indicara si el segmento se encuentra o no en el almacenamiento primario, si sí se encuentra en el almacenamiento primario " s' " es la dirección. Si no esta en el almacenamiento primario, entonces "a" es la dirección en el almacenamiento secundario donde debe leerse ese segmento antes de que el proceso pueda continuar con su ejecución. Todas las referencias al segmento deben verificarse contra la longitud del segmento "l" para asegurarse de que se encuentra dentro de el. En cada referencia al segmento se revisan también los bits de protección para ver si esta permitida la operación que intenta.



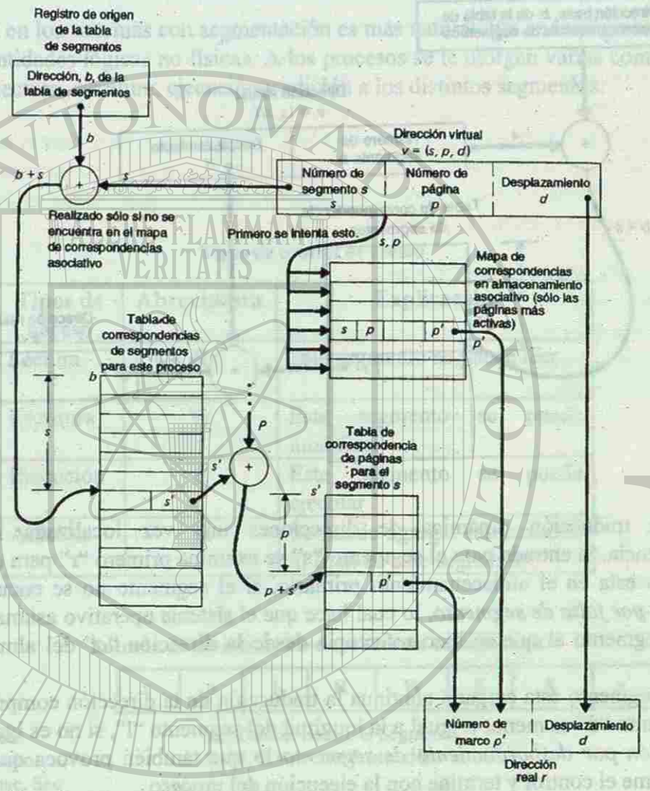
Durante la traducción dinámica de direcciones una vez localizada la tabla de correspondencia, la entrada para el segmento "s" se examina primero "r" para comprobar si el segmento esta en el almacenamiento primario, si el segmento no se encuentra allí se genera *falla por falta de segmento*, lo cual hace que el sistema operativo asuma el control y cargue el segmento al que se hizo referencia desde la dirección "*a*" del almacenamiento secundario.
Cuando el segmento esta cargado continua la traducción de la dirección comprobando si el desplazamiento "d" es menor o igual a la longitud del segmento "l", si no es igual se genera una *excepción por desbordamiento de segmento* lo que también provoca que el sistema operativo tome el control y termine con la ejecución del proceso.

El último paso es sumar la dirección "s'" del almacenamiento primario al desplazamiento " d " para formar la dirección del almacenamiento real r = s' + d, si esto no esta permitido se genera *una excepción por protección del segmento* lo que hace que el sistema operativo tome el control y termine la ejecución.

Páginación/segmentación combinada o sistemas de páginación segmentación
Una dirección virtual con sistema de páginación/segmentación es una tripleta ordenada V=(s,p,d), donde "s" es el segmento en el que reside V, "p" es la página de "s" que contiene a "V" y "d" es el desplazamiento dentro de "V" dentro de "p". En estos sistemas casi siempre se usa un almacenamiento asociativo de acceso rápido o una memoria caché para evitar que la traducción dinámica de direcciones degrade seriamente el rendimiento.

Número de segmento s	Número de página p	Desplazamiento d

v = (s,p,d)



ADMINISTRACION DEL ALMACENAMIENTO VIRTUAL

Estrategias de Reemplazo:

1 Principio de Optimalidad.
2 Reemplazo aleatorio de páginas.
3 Primeras Entradas Primeras Salidas (FIFO).
4 La menos recientemente utilizada (LRU).
5 La menos frecuentemente utilizada (LFU).
6 La no utilizada recientemente (NUR).
7 Segunda Oportunidad
8 Por Reloj
9 Conjuntos de Trabajo
10 Frecuencia de fallas de página (PFF).

Principio de Optimalidad.
Establece que para obtener un rendimiento óptimo la página que se debe reemplazar es aquella que tardará más tiempo en volver a ser utilizada.

Reemplazo de Páginas Aleatorio.
Si se busca una estrategia de reemplazo de páginas con un gasto extra reducido y que no discrimina ciertos usuarios, el reemplazo de páginas aleatorio es una técnica sencilla para lograrlo ya que todas las páginas que se encuentran en el almacenamiento principal tienen la misma oportunidad de ser elegidas para el reemplazo. Esta estrategia puede seleccionar cualquier página para ser reemplazada incluyendo la página a la que se hará referencia enseguida.

Primeras Entradas Primeras Salidas.
Aquí, para cada página se registra el instante en el que entró al almacenamiento primario. Cuando se necesite reemplazar una página se escoge a la que haya permanecido en el almacenamiento primario el mayor tiempo. Esto parece razonable ya que cada página ha tenido su oportunidad y es tiempo de darle oportunidad a otra página, lamentablemente es probable que se reemplace páginas muy utilizadas ya que si una página permanece en el almacenamiento por mucho tiempo, puede deberse a que está en uso constantemente.

Anomalía PEPS
Parece razonable pensar que cuantos más marcos de página se asignen a un proceso menos fallos de página experimentará éste. Sin embargo, se descubrió que al utilizar el reemplazo de páginas PEPS ciertos patrones de referencia a páginas originan más fallas de página cuando aumenta el número de marcos de página asignados a un proceso, a este fenómeno se le llama anomalía PEPS o anomalía Belady.

La Menos Recientemente Utilizada (LRU).
Se selecciona para su reemplazo a aquella página que no ha sido utilizada el mayor tiempo, la estrategia se basa en la heurística de localidad según la cual el pasado reciente es un buen indicador del futuro cercano de modo que LRU exige que se marque cada página con el instante en que se hace referencia a ella lo que origina mucho trabajo adicional por lo que pese a ser atractiva no se usa en los sistemas actuales, lo que se hace es utilizar estrategias que se aproximen a la LRU y que no ocasionen grandes costos.

La página menos recientemente utilizada podría ser la siguiente en ser utilizada, si un programa ocupa un ciclo importante que ocupe varias páginas. Si se reemplazara la página LRU se encontraría a sí mismo transfiriendo ésa página de nuevo al almacenamiento principal casi de inmediato.

La Menos Frecuentemente Utilizada (LFU).
Es una aproximación a la LRU en la que es importante la intensidad con la que se ha utilizado una página, la página por reemplazar aquella que se le ha usado menos frecuentemente ó la que se ha hecho referencia con menos frecuencia.

La página menos frecuentemente utilizada podría ser aquella que se trajo del almacenamiento secundario al principal más recientemente. Esta página ha sido utilizada una sola vez mientras que las demás páginas que se encuentran en el almacenamiento principal pudieron haber sido utilizadas en varias ocasiones. Este mecanismo de reemplazo de página reemplazara esta página aunque tenga una probabilidad de ser usada de inmediato.

La No Utilizada Recientemente (NUR).
Es una aproximación a LRU con un poco de trabajo extra. No es probable que sean utilizadas pronto las páginas que no se han utilizado recientemente por lo que puede reemplazarse con páginas reentrantes.
Es deseable reemplazar una página que no ha sido modificada mientras estaba en el almacenamiento primario, la estrategia NUR se llevará a la práctica con dos bit de hardware, estos son: el bit de referencia y el bit de modificación; en el bit de referencia el valor va a ser 0 si no se ha hecho referencia a la página o 1, si sí se ha hecho referencia, el bit de modificación va a ser 0 si no ha sido modificada y 1, si sí ha sido modificada. Al bit de modificación se le llama "bit sucio".

Segunda Oportunidad.
Es una variante de PEPS. Examina el bit de referencia de la página más antigua, si este vale cero se selecciona de inmediato para ser reemplazado. Si el bit de referencia vale 1 se le asigna el valor cero y la página se pasa al final de la lista y se considera en esencia como una página nueva, ésta página gradualmente irá avanzando hacia el inicio de la lista donde será reemplazada solamente si su bit de referencia vale cero (0).

Por Reloj.
La variación por reloj del algoritmo de la segunda oportunidad dispone las páginas en una lista circular en lugar de una lista lineal, un apuntador a la lista se desplaza alrededor de la lista circular en la misma forma que giran las manecillas de un reloj cuando el bit de referencia de una página toma el valor de cero (0) el apuntador se mueve al siguiente elemento de la lista.

Localidad.- El concepto de localidad indica que los procesos tienden a hacer referencia a la memoriaen patrones no uniformes y altamente localizados. La localidad se manifiesta tanto en el espacio como en el tiempo. Es una propiedad empírica observada más que teórica.

Localidad Temporal.- Es la localidad en el tiempo. Significa que es probable que las localidades de memoria a las que se haya hecho referencia recientemente sean objeto de otra referencia en un futuro cercano, como apoyo de esta observación se puede mencionar:

- Los ciclos.
- Las subrutinas.
- Las pilas.
- Las variables de cuenta y totalización.

Localidad Espacial.- Significa que los elementos cercanos tienden a ser similares. También significa que las referencias a memoria tienden a estar concentradas y una vez que

se hace referencia a una localidad es muy probable que se haga referencia a las localidades cercanas, como apoyo a esta observación:

- Los recorridos de los arreglos.
- La ejecución secuencial de código.
- La tendencia de algunos programadores a colocar las definiciones de variables afines próximas unas de otras.

Conjuntos de trabajo.
Denning desarrolló el conjunto de programas de trabajo para explicar el compartimiento de las páginas de los programas en términos de localidades. Los conjuntos de trabajo intentan mantener el conjunto de trabajo de un proceso (las páginas a las que se hizo referencia recientemente) en el almacenamiento primario de tal manera que el proceso se ejecutará con rapidez. Los procesos nuevos se pueden iniciar solo si hay espacio en el almacenamiento primario para sus conjuntos de trabajo. Los procesos que intentan ejecutarse sin espacio suficiente para sus conjuntos de trabajo a menudo experimentan hiperpáginación, un fenómeno en el cual se reemplaza continuamente páginas que son devueltas de inmediato al almacenamiento primario.

Reemplazo de páginas por frecuencia de fallas de página
El algoritmo de frecuencia de fallas de página (**p**age **f**ault **f**requency) ajusta el conjunto de páginas residentes de un proceso básandose en la frecuencia con que el proceso tiene fallas de página o según el tiempo que hay entre fallas de página.
El PFF registra el tiempoentre la última falla de página y el vigente. Si ese tiempo es mayor que un límite superior, entoces se liberan todas las páginas a las que no se haya hecho referencia en ese intervalo. Si el tiempo es menor que un valor límite inferior, la página entrante se convierte en un miembro del conjunto de páginas residentes del proceso.

Páginación por demanda
Es la más popular por:

1. Los resultados de la teoría de la computabilidad indican que no se pueden lograr esquemas de páginación anticipada totalmente preciso debido a que no se puede predecir el futuro.
2. Se traen al almacenamiento primario solo las páginas que necesita un proceso.
3. El trabajo extra de búsqueda es insignificante.

Liberación de Páginas
Esta ayuda a sacar del almacenamiento primario las páginas que ya no se necesitan.

Tamaño de Páginas
Consideraciones que afectan la determinación del tamaño óptimo de página para un sistema dado:

1. Un tamaño de página pequeño requiere tablas de páginas más grandes con la consecuente fragmentación de tablas.
2. Un tamaño de página grande hace que se transfieran al almacenamiento primario instrucciones y datos a los que no se hará referencia.
3. Las transferencias de entrada/salida son más eficientes con páginas grandes.

4. La localidad tiende a ser pequeña.
5. La fragmentación interna se reduce con páginas pequeñas.

Los diseñadores sugieren el uso de páginas pequeñas.

Muchos experimentos han examinado el comportamiento de los sistemas de computo con páginación y los resultados han sido interesantes:

1. Cuando un proceso comienza a ejecutarse, por lo regular hace referencia rápidamente aun gran porcentaje de sus páginas.
2. El número de fallas experimentadas por un proceso en ejecución tiende a crecer a medida que aumenta el tamaño de la página suponiendo que permanece constante el almacenamiento primario asignado al proceso.
3. El tiempo entre fallos experimentado por un proceso en ejecución crece a medida que aumenta la cantidad de marcos de página asignados al proceso, una vez asignado al proceso un número suficiénte de marcos de páginas para contener su conjunto de trabajo disminuye la tasa de incremento.
4. El número de instrucciones ejecutadas en una página antes de que el control salga de la página tiende a ser pequeño.

Unidad V

SISTEMAS DE ARCHIVO

Objetivo de esta unidad.- Durante el desarrollo de esta unidad, se analizará como almacenar una gran cantidad de información, que no se pierda hasta cuando el proceso que la generó termina su ejecución y que dos o más procesos tengan acceso concurrente a la información.
El alumno deberá comprender la forma de uso de los archivos y las propiedades que estos poseen; la utilización de los directorios, su organización, sus propiedades y las operaciones que se lleven a cabo con ellos; la implantación de un sistema de archivo, la seguridad de la información y mecanismos de protección debe de tener un sistema de archivos.

Archivo.- Es un conjunto de datos al cual se le asigna un nombre, reside, por lo regular, en el almacenamiento secundario ya sea en cinta ó en disco, se puede manejar como una unidad mediante operaciones como:

- abrir
- cerrar
- destruir
- copiar
- renombrar
- listar

A los elementos individuales de información dentro del archivo se le pueden aplicar las siguientes operaciones:

- leer
- escribir
- modificar
- insertar
- eliminar

Los archivos pueden caracterizarse por su:

Volatilidad.- Se refiere a la frecuencia con la que se añade información a un archivo y se borra de él.

Actividad.- Se refiere al porcentaje de los registros de un archivo al cual se tuvo acceso durante un periodo dado.

Tamaño.- Se refiere a la cantidad de información alamacenada en el archivo

Sistemas de Archivos
Los sistemas de archivo contienen:

Método de acceso.- Se ocupan de la forma en que se obtiene acceso a los datos almacenados en archivos.

4. La localidad tiende a ser pequeña.
5. La fragmentación interna se reduce con páginas pequeñas.

Los diseñadores sugieren el uso de páginas pequeñas.

Muchos experimentos han examinado el comportamiento de los sistemas de computo con páginación y los resultados han sido interesantes:

1. Cuando un proceso comienza a ejecutarse, por lo regular hace referencia rápidamente aun gran porcentaje de sus páginas.
2. El número de fallas experimentadas por un proceso en ejecución tiende a crecer a medida que aumenta el tamaño de la página suponiendo que permanece constante el almacenamiento primario asignado al proceso.
3. El tiempo entre fallos experimentado por un proceso en ejecución crece a medida que aumenta la cantidad de marcos de página asignados al proceso, una vez asignado al proceso un número suficiénte de marcos de páginas para contener su conjunto de trabajo disminuye la tasa de incremento.
4. El número de instrucciones ejecutadas en una página antes de que el control salga de la página tiende a ser pequeño.

Unidad V

SISTEMAS DE ARCHIVO

Objetivo de esta unidad.- Durante el desarrollo de esta unidad, se analizará como almacenar una gran cantidad de información, que no se pierda hasta cuando el proceso que la generó termina su ejecución y que dos o más procesos tengan acceso concurrente a la información.
El alumno deberá comprender la forma de uso de los archivos y las propiedades que estos poseen; la utilización de los directorios, su organización, sus propiedades y las operaciones que se lleven a cabo con ellos; la implantación de un sistema de archivo, la seguridad de la información y mecanismos de protección debe de tener un sistema de archivos.

Archivo.- Es un conjunto de datos al cual se le asigna un nombre, reside, por lo regular, en el almacenamiento secundario ya sea en cinta ó en disco, se puede manejar como una unidad mediante operaciones como:

- abrir
- cerrar
- destruir
- copiar
- renombrar
- listar

A los elementos individuales de información dentro del archivo se le pueden aplicar las siguientes operaciones:

- leer
- escribir
- modificar
- insertar
- eliminar

Los archivos pueden caracterizarse por su:

Volatilidad.- Se refiere a la frecuencia con la que se añade información a un archivo y se borra de él.

Actividad.- Se refiere al porcentaje de los registros de un archivo al cual se tuvo acceso durante un periodo dado.

Tamaño.- Se refiere a la cantidad de información alamacenada en el archivo

Sistemas de Archivos
Los sistemas de archivo contienen:

Método de acceso.- Se ocupan de la forma en que se obtiene acceso a los datos almacenados en archivos.

Administración de archivos.- Se ocupa de ofrecer mecanismos para almacenar, compartir y asegurar archivos, y hacer referencia a ellos.

Administración de almacenamiento secundario.- Se ocupa de asignar espacio a los archivos en dispositivos de almacenamiento secundario.

Mecanismos de integridad de los archivos.- Se ocupan de garantizar que no se corrompa la información en un archivo. Cuando se asegura la integridad de los archivos, cualquier información que deba estar en un archivo estará ahí.

El sistema de archivos se ocupa primordialmente de administrar el espacio de almacenamiento secundario, sobre todo el espacio en disco.

Funciones del Sistemas de Archivos

Además de que los usuarios deben poder crear, modificar y eliminar archivos las funciones normalmente atribuidas a los sistemas de archivos son:

Compartir información.- Los usuarios deben ser capaces de compartir sus archivos entre sí en forma cuidadosamente controlada para aprovechar y continuar el trabajo de los demás.

Mantener privada la información, cifrarla o descifrarla.- En ambientes delicados en los cuales la información debe mantenerse segura y privada, como los sistemas de transferencia de fondos, sistemas de expedientes criminales, sistemas de expedientes médicos, etc, es conveniente ofrecer estas funciones, lo cual hace que la información solo resulte útil a quienes esta destinada.

Obtener acceso a la información.- El mecanismo para compartir debe ofrecer varios tipos de acceso controlado, como acceso para lectura, acceso para escritura, acceso para ejecución o las diversas combinaciones de estos.

Respaldo y Recuperación de la información.- Deben ofrecerse recursos de respaldo y recuperación para evitar la pérdida accidental o la destrucción mal intencionada de información.

Independencia con respecto a los dispositivos.- Los usuarios deben poder hacer referencia a sus archivos mediante nombres simbólicos en vez de tener que usar nombres de dispositivos físicos

Lo más importante de todo es que el sistema de archivos debe ofrecer una interfaz amable con el usuario. Debe proporcionar a los usuarios una vista lógica de sus datos y las funciones que puede realizar con ellos, más que una vista física. El usurio no debe preocuparse de los dispositivos específicos en los cuales esta almacenada la infomación, la

forma que adoptan los datos en esos dispositivos o los mecanismos físicos para transferir datos de esos dispositivos o hacia ellos.

Jerarquía de los Datos

La estructura de datos esta formada por entidades de complejidad creciente:

bits → bytes → campos → registros → archivos → bases de datos

División en Bloques y Empleo de Buffer

Registro físico.- o Bloque físico Es la unidad de información que se lee realmente de un dispositivo o se graba en él.

Registro Lógico.- o Bloque lógico Es un conjunto de datos manejado como una unidad desde el punto de vista del usuario.

Registro sin Bloques.- Cuando cada registro físico contiene sólo un registro lógico se dice que el archivo esta formado por *registros sin bloques.*

Registro en Bloques.- Cuando cada registro físico puede contener varios registros lógicos se dice que el archivo esta formado por *registros en bloques*.

En un archivo con *registros de longitud fija*, todos los registros tienen el mismo tamaño; el tamaño de bloque casi siempre es un múltiplo entero de la longitud del registro.

En un archivo con *registros de longitud variable*, el tamaño del registro puede variar, sin rebasar el tamaño del bloque.

El *empleo de Buffer* (áreas de almacenamiento temporal) hace posible que el cómputo se efectúe de forma simultáneo con la entrada/salida. Se reservan espacios en almacenamiento primario para guardar varios bloques físicos de un archivo a la vez, cada uno de estos espacios se llama buffer. El método más común son los buffers dobles y funciona como siguen (para la salida):

1. Existen dos buffers
2. Los registros generados por un proceso se depositan en el primer buffer hasta que se llena.
3. Se inicia la transferencia del bloque en el primer buffer al almacenamiento secundario.
4. Mientras se realiza esta transferencia, el proceso continúa generando registros que se depositan en el segundo buffer
5. Cuando se llena el segundo buffer y una vez finalizada la transferencia del primer buffer, se inicia la transferencia del segundo buffer.

El proceso sigue generando registros, los cuales se depositan ahora en el primer buffer. Esta alternación de buffers permite que se lleve a cabo la entrada/salida en paralelo con los cálculos de un proceso.

Organización de los Archivos
La organización de los archivos se refiere a la forma como se acomodan los registros de un archivo en almacenamiento secundario.

Sistemas de Organización de Archivos más comunes:

Secuencial.- Los registros se colocan en orden físico. Es una organización natural para archivos grabados en cinta mágnetica, medio de almacenamiento que por su naturaleza es secuencial. Los archivos en disco también se pueden organizar secuencialmente, aunque por diversas razones, los registros de un archivo secuencial de disco no se almacenan en forma contigua por fuerza.

Directo.- Se obtiene acceso directo (aleatorio) a los registros por su dirección física en un dispositivo de almacenamiento de acceso directo (*DASD, direct access storage device*)

Secuencial indexado.- Los registros se acomodan en secuencia lógica de acuerdo con una clave contenida en cada registro. El sistema mantiene un índice con las direcciones físicas de ciertos registros principales. El acceso a los registros secuencial indexado puede obtenerse secuencialmente por orden de clave o de manera directa, mediante una búsqueda dentro del índice creado por el sistema. Los archivos secuencial indexado casi siempre se almacenan en discos.

De partición.- Este es en esencia un archivo de subarchivos secuenciales. Cada subarchivo secuencial se llama miembro. La dirección inicial de cada miembro se almacena en el directorio del archivo.

El término *volumen* se usa para referirse al medio de grabación de cada dispositivo auxiliar de almacenamiento en particular. El volumen empleado en una unidad de cinta es el carrete de cinta y el volumen empleado en una unidad de disco es el disco.

Métodos de acceso
El acceso a los archivos se logra mediante funciones de los sistemas operativos llamados *métodos de acceso*, estos se agrupan en dos categorías:

Método de Acceso por Colas.- Se usan con archivos de organización secuencial; realizan *transferencias anticipadas a buffers* y programan la E/S además de ofrecer agrupación en bloques y separación de bloques en forma automática.

Métodos de Acceso Básicos.- Se utilizan por lo regular cuando no es posible anticipar el orden en el que se van a procesar los registros, sobre todo en el caso del acceso directo. El los métodos básicos, el método leer y escribe bloques físicos; si se requiere en la aplicación, el usuario se encarga de la formación y división de bloques.

Asignación y Liberación de Espacio
Con la asignación contigua cada archivo se asigna a un área única del almacenamiento secundario. La asignación contigua facilita el acceso rápido pero adolece de serios problemas de fragmentación.
Si la asignación es no contigua, el archivo puede estar disperso en varias áreas de almacenamiento secundario. La asignación no contigua es más flexible pero puede requerir búsquedas frecuentes.

Descriptor de Archivos
Llamado también Bloque de Control de Archivo. Es un bloque de control con información que el sistema necesita para administrar un archivo. Un descriptor de archivo representativo incluye lo siguiente:
1. Nombre simbólico del archivo.
2. Localización del archivo en el almacenamiento secundario.
3. Organización del archivo.
4. Tipo de dispositivo
5. Datos para el control de acceso.
6. Tipo (Si es archivo de datos, programa objeto, o programa fuente).
7. Tratamiento (Temporal o permanente).
8. Fecha y hora de creación.
9. Fecha de destrucción.
10. Fecha y hora de la última modificación.
11. Conteo De la actividad de acceso.

Los descriptores de archivo se mantienen en el almacenamiento secundario y se transfieren al almacenamiento primario cuando se abre un archivo. El descriptor de archivo es controlado por el sistema operativo, el usuario no puede hacer referencia directa a él.

Tipos de Control de Acceso
El control de acceso a los archivos puede manejarse mediante una matriz para control de acceso que indica cuales usuarios tienen que tipos de acceso a cuales archivos. Lo más frecuente es que el control de acceso se maneje por clases de usuarios donde por ejemplo se puede conceder acceso al propietario ó aun usuario específico, a un miembro de un grupo ó a un miembro del público en general.

Respaldo y Recuperación
Son funciones muy importantes de cualquier sistema de archivos. La técnica más común es el respaldo periódico. Otra técnica es vaciado por incrementos.

Vaciado por Incrementos.- Los archivos modificados por un usuario durante una sesión determinada se respaldan cuando ese usuario sale del sistema.

Bitácora de Transacciones.- Todas la líneas tecleadas por un usuario se copian en un archivo de bitácora. La recuperación implicaría literalmente una nueva aplicación de todas las transacciones realizadas después del último respaldo periódico principal.

Bases de Datos

Es un conjunto integrado de datos controlados centralmente. Unas de la ventajas principales son:

1. Se puede reducir la redundancia.
2. Evitar la inconsistencia.
3. Compartir datos.
4. Imponer Normas.
5. Se puede aplicar restricciones de seguridad.
6. Mantener la integridad.
7. Se puede equilibrar requerimientos en conflicto.

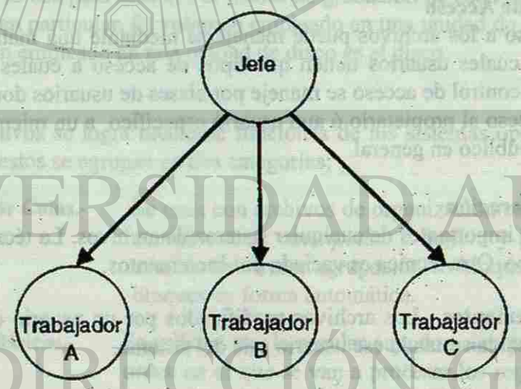
Independencia de los datos.- Hace posible modificar una aplicación y desarrollar nuevas aplicaciones sin tener que alterar la estructura del almacenamiento de los datos y la estrategia de acceso.

Bases de Datos Distribuida.- Está distribuida o dispersa en todos los sistemas de cómputo mediante una red.

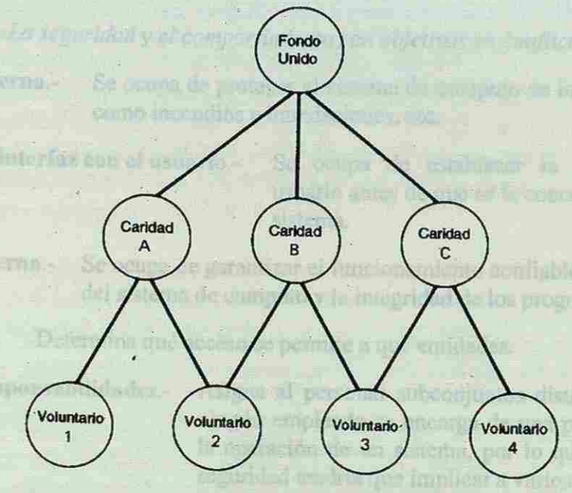
Tipos de Organización de Bases de Datos

1. Jerárquica
2. Red
3. Relacional

En el enfoque *jerárquico* los datos se organizan según interrelaciones padre-hijo, cada hijo tiene un solo padre y cada padre puede tener muchos hijos. Búsquedas y mantenimientos fáciles, pero es limitada la flexibilidad del usuario para diseñar interdependencias complejas de los datos.



En el enfoque de *red* se pueden expresar en forma conveniente interdependencias muy generales, pero las estructuras resultantes pueden ser difíciles de entender, modificar o reconstruir en caso de falla (Red).



El enfoque *relacional* ofrece muchas ventajas con respecto a las anteriores, la representación tabular es más fácil de comprender y llevar a la práctica. Otros esquemas se pueden convertir con mucha facilidad a la organización relacional.

Relación: EMPLEADO

Número	Nombre	Departamento	Salario	Localización
23603	RAMOS, A.	413	1100	SEGOVIA
24568	CORTÉS, R.	413	2000	SEGOVIA
34589	LÓPEZ, P.	642	1800	ZARAGOZA
35761	MENDOZA, B.	611	1400	MADRID
47132	NOGUERA, C.	413	9000	SEGOVIA
78321	SALAS, T.	611	8500	MADRID

Una tupla

Clave primaria

Un dominio

Operaciones útiles como la proyección y la reunión facilitan la creación de nuevas relaciones. Los datos delicados pueden asegurarse colocándolos en relaciones separadas. Las búsquedas son más directas y rápidas, modificaciones directas y mejora la claridad y visibilidad de los datos.

SEGURIDAD EN LOS SISTEMAS OPERATIVOS

La seguridad y el compartimiento son objetivos en conflicto.

Seguridad externa.- Se ocupa de proteger el sistema de computo de intrusos y desastres como incendios e inundaciones, etc.

Seguridad de interfaz con el usuario.- Se ocupa de establecer la identidad de un usuario antes de que se le conceda el acceso a un sistema.

Seguridad interna.- Se ocupa de garantizar el funcionamiento confiable y sin corrupción del sistema de computo y la integridad de los programas y datos.

Autorización.- Determina qué acceso se permite a qué entidades.

División de responsabilidades.- Asigna al personal subconjuntos distintos de deberes; ningún empleado se encarga de una porción grande de la operación de un sistema, por lo que un ataque a la seguridad tendría que implicar a varios empleados.

Vigilancia.- Se ocupa de supervisar el sistema y realizar auditorías así como de verificar la autenticidad de los usuarios.

Supervisión de amenazas.- El sistema operativo controla operaciones delicadas en lugar de ceder el control directamente a los usuarios. Los *programas de vigilancia* ejecutan las operaciones delicadas. Se habla de *amplificación* cuando éstos programas requieren un acceso más amplio para atender las solicitudes de los usuarios.

Protección por contraseña

Hay tres clases de elementos de la verificación de autenticidad con los cuales puede establecerse la identidad de una persona:

1. Algo característico de la persona (huellas digitales, patrones de voz, fotografías y firmas)
2. Algo que la persona posee (credenciales, tarjetas de identificación y claves).
3. Algo que sabe la persona (contraseñas, combinaciones de candados, el apellido de su maestra de tercer año de primaria).

El ciframiento de la lista maestra de contraseñas ayuda a mantener la seguridad de estas aun cuando haya penetración al sistema. Se recomienda cambiar con frecuencia las contraseñas.

Auditoría.- Se realiza por lo general en sistemas manuales *a posteriori.* Se convocan auditores periódicamente para examinar las transacciones recientes de una organización y determinar si se han realizado actividades fraudulentas. En los sistemas de computo puede implicar un procesamiento inmediato en el computador para revisar las transacciones que se acaban de realizar.

Bitácora de auditoría.- Es un registro permanente de eventos importantes que ocurren en el sistema de computo. Se produce automáticamente cada vez que sucede un evento así y se almacena en un área protegida del sistema; si el sistema se ve comprometido, la bitácora deberá permanecer intacta. Es un mecanismo de detección importante. Aunque logren penetrar las defensas de un sistema, las personas puden refrenar sus deseos de hacerlo si temen una detección posterior.

Controles de Acceso.- La clave para la seguridad interna es controlar el acceso a los datos almacenados. Los *derechos de acceso* definen qué acceso tienen varios sujetos a diversos objetos. Los objetos se protegen contra los sujetos.
Los accesos más comunes son el de lectura, escritura y ejecución.

Núcleos de seguridad.- Las medidas de seguridad más vitales se ponen en práctica en el núcleo, el cual se mantiene a propósito lo más pequeño posible. Esto hace más razonable la revisión cuidadosa del núcleo para detectar fallas y demostrar formalmente que este correcto.
La seguridad de un sistema operativo depende sobre todo de asegurar las funciones que se encargan del control de acceso, las entradas al sistema y la supervisión, y que administran el almacenamiento real, el almacenamiento virtual y el sistema de archivos.

Sistemas Tolerantes a Fallas.- Un sistema de cómputo tolerante a fallas continua funcionando aún después de haber fallado uno o más de sus componentes. La tolerancia a fallas se facilita mediante la incorporación de mecanismos a prueba de fallas, el empleo de multiprocesamiento transparente, el uso de múltiples subsistemas de E/S, la incorporación en hardware de gran parte del sistema operativo y la incorporación en hardware de mecanismos para la detección de fallas.

Capacidades y Sistemas Orientados a Objetos

Un derecho de acceso permite a algún sujeto obtener acceso a un objeto de una manera predeterminada. Los sujetos son usuarios de sistemas de computo o entidades que actúan a nombre de los usuarios o del sistema. Los objetos son recursos dentro del sistema. Los sujetos pueden ser cosas como tareas, procesos y procedimientos. Los objetos pueden ser archivos, programas, semáforos, directorios, terminales, canales, controladores, dispositivos, pistas de disco, bloques de almacenamiento primario, etc. Los sujetos también se consideran como objetos del sistema, de modo que un sujeto puede tener acceso a otro. Los sujetos son entidades activas; los objetos son pasivos.

Un *dominio de protección* define los derechos de acceso que tiene un sujeto a los diversos objetos del sistema. Es el conjunto de capacidades pertenecientes a un sujeto. Es importante que los dominios de protección sean pequeños para hacer cumplir el principio de menor privilegio.
Para que un objeto obtenga acceso a un objeto específico debe poseer una capacidad para ello.El problema del objeto perdido se refiere a lo que sucede cuando se elimina la capacidad para tener acceso a un objeto. La renovación de capacidades puede ser difícil, una capacidad podría haber sido copiadas muchas veces. Las capacidades por lo regular no se modifican, pero pueden reproducirse.

Criptografía

Criptografía.- Es el empleo de transformaciones de los datos a fin de hacerlos incomprensibles para todos con excepción de sus usuarios autorizados.

Problema de intimidad.- Se ocupa de evitar la extracción no autorizada de información de un canal de comunicación.

Problema de verificación de autenticidad.- Se ocupa de evitar que algún enemigo modifique una transmisión o inserte datos falsos en una transmisión.

Problemas de disputa.- Se ocupa de ofrecer al receptor de un mensaje una prueba legal de la identidad del remitente o sea el equivalente electrónico de una firma escrita. Como funciona: un remitente cifra texto simple para crear texto cifrado el cual se transmite a un receptor a través de un canal no seguro incluso vigilado por un espía. El receptor descifra el texto cifrado para reconstruir el texto simple original.

Criptoanálisis.- Es el proceso de intentar regenerar un texto simple a partir del texto cifrado pero sin conocer la clave de descifrado.

Clave pública.- En los sistemas de clave pública las funciones de cifrado y descifrado están separadas, cada una requiere una clave distinta, la clave se hace pública si la otra permanece en secreto. Cuando se cifra un mensaje con clave pública de un usuario sólo ese usuario puede descifrar el mensaje. Con los sistemas de clave pública es posible llevar a la práctica firmas digitales que garanticen la autenticidad de un mensaje.

Los esquemas DES y RSA

Dos de los esquemas criptográficos más importantes son la norma de cifrado de datos (DES) y el esquema Rivest, Shamir y Adleman (RSA).

DES es un esquema simétrico de cifrado en el cual se usa una sola clave para cifrar y descifrar y RSA es un esquema asimétrico en el cual se utilizan claves distintas para estos propósitos.

El uso más común en llos sistemas operativos actuales es para proteger la lista maestra de contraseñas de un sistema. El ciframiento también es usado para proteger datos almacenados en archivos y para proteger datos que se transmiten a través de una red. Las cintas y discos de respaldo cifrado no necesitan cuidarse con tanto celo como los no cifrados. El ciframiento de enlaces se ocupa de el ciframiento/descifrамiento en cada nodo de una red de computadores.
Ejemplo: Si se usa ciframiento de extremo a extremo los mensajes se cifran sólo en su punto de origen y se descifran sólo en su punto destino.

Mediante un procedimiento de reto y respuesta, un sistema puede verificar la autenticidad de un usuario cuando trate de entrar, sin necesidad de transmitir una clave.

Penetración en el Sistema Operativo
Las defensas de un sistema operativo deben ser capaces de resistir un intento de penetración por parte de un usuario no privilegiado; hacer que un sistema operativo sea impenetrable es una tarea imposible, lo que podemos esperar es que sea altamente resistente a la penetración.

Defectos Funcionales Genéricos de los Sistemas
Se han encontrado varios defectos comunes a muchos sitemas de computo. Entre ellos están:

Verificación de autenticidad.- En muchos sistemas, los usuarios no pueden determinar si el equipo y los programas son lo que deberían de ser. Esto hace que un penetrador pueda reemplazar con facilidad un programa sin que se entere el usuario. Ejemplo.- Un usuario podría dar su contraseña a un programa falso de entrada al sistema.

El ciframiento.- La lista maestra de contraseñas debe almacenarse en forma cifrada. A veces no se hace.

Realización.- Un diseño bien pensado para un mecanismo de seguridad puede llevarse a la práctica en forma inadecuada.

Confianza implícita.- Una rutina supone que otra está funcionando correctamente, en vez de examinar con cuidado los parámetros suministrados por la otra.

Compartimiento implícito.- El sistema puede depositar sin darse cuenta información vital del sistema en el espacio de direcciones de un usuario.

Comunicación entre procesos.- El penetrador puede usar un mecanismo de transmisión/recepción para probar diversas posibilidades. Ejemplo.- El penetrador puede solicitar un recurso del sistema y suministrar una contraseña; la información de vuelta puede indicar "contraseña correcta", comfirmando la contraseña adivinada por el penetrador.

Comprobación de legalidad.- El sistema quizá no verifique lo suficiente la validez de los parámetros del usuario.

Desconexión de línea.- En sistemas de tiempo compartido y redes, cuando se pierde la línea el sistema operativo deberá clausurar de inmediato la sesión del usuario o poner a éste en un estado tal que sea necesaria una nueva autorización para otorgarle el control. Un penetrador podría obtener control del proceso y utilizar los recursos a los cuales puede tener acceso este último.

Descuido del operador.- Un penetrador puede engañar a un operador para que monte un disco de sistema operativo falso.

Paso de parámetros por referéncia en vez de por valor.- Es más pasar parámetros directamente en registros y no hacer que los registros apunten a localidades donde estan los parámetros. El paso por referencia puede conducir a una situación en la cual los parámetros siguen en el espacio de direcciones del usuario después de haberse realizado la verificación de legalidad; así, el usuario podría sumisnistrar parámetros legítimos, hacer que sean verificados y después modificarlos justo antes de que los utilice el sistema.

Contraseñas.- A menudo las contraseñas son fáciles de adivinar o de obtener por intentos repetidos.

Trampas para el penetrador.- Los sistemas deben incluir mecanismos de trampa para atraer al intruso inexperto, pues constituyen una buena primera línea de detección. La mayor parte de los sistemas tienen mecanismos de trampa inadecuados.

Privilegios.- En algunos sistemas, son demasiados los programas que tienen demasiados privilegios. Esto va contra el principio del menor privilegio.

Confinamiento de programas.- Un programa prestado por otro usuario puede actuar como Caballo de Troya; podría robar o alterar los archivos de quien lo pidió prestado.

Prohibición.- Muchas veces se indica a los usuarios que se abstengan de usar ciertas funciones porque los resultados pueden ser "indeterminados". No obstante, estas funciones siguen siendo accesibles para los usuarios.

Residuo.- El penetrador puede encontrar una lista de contraseñas examinando el cesto de la basura. En ocasiones se deja residuo en el almacenamiento después de ejecutarse una rutina del sistema. Toda información confidencial deberáreemplazarse o destruise antes de liberar o desechar el medio que ocupa.

Blindaje.- Una corriente en un alambre genera un campo magnético alrededor de este; los penetradores pueden intervenir de hecho una línea de transmisión o un sistema de computo sin hacer contacto físico. El blindaje eléctrico puede servir para evitar estas "intrusiones invisibles".

Valores de Umbral.- El propósito de estos es refrenar intentos repetidos de entrar en el sistema. Ejemplo: Después de un cierto número de intentos de entrada no válidos, ese usuario deberá bloquearse, notificando al administrador del sistema. Muchos sistemas no cuentan con esta característica.

Ataques Genéricos a los Sistemas Operativos.- Metodologías de penetración:

Asincronía.- Cuando varios procesos avanzan en forma asíncrona, es posible que un proceso modifique parámetros cuya validez ha sido verificada por otro, aunque este último todavía no los haya usado; así un proceso que tiene valores malos a otro aunque el segundo realice una verificación exhaustiva.

Hojeo.- Un usuario revisa el sistema de computo intentando localizar información privilegiada.

Entre Líneas.- Se usa una terminal especial para intervenir una línea de comunicaciones empleada por un usuario inactivo que haya entrado en el sistema.

Código clandestino.- Se instala un parche con la pretensión de corregir un error en el sistema operativo; el código contiene escotillones, a través de los cuales se puede entrar después en el sistema sin autorización.

Rechazo de acceso.- Un usuario escribe un programa para hacer que se caiga el sistema, para ponerlo en un ciclo infinito o para monopolizar sus recursos. La intención en este caso es impedir que usuarios legítimos obtengan acceso o servicio.

Interacción de procesos sincronizados.- Los procesos usan las primitivas de sincronización del sistema para compartir o pasara información entre ellos.

Desconexión de línea.- El penetrador intenta obtener acceso al trabajo de un usuario después de una desconexión de línea, pero antes de que el sistema reconozca la desconexión.

Disfraz.- El penetrador asume la identidad de un usuario legítimo después de haber obtenido la identificación correcta por medios clandestinos.

Ataque NAK.- Muchos sistemas permiten a un usuario interrumpir un proceso en ejecución(utilizando la tecla "negative acknowledge"), realizar otra operación y después continuar el proceso interrumpido. El penetrador puede "atrapar" al sistema en un estado no protegido y adueñarse del control con facilidad.

Engaño del operador.- Un penetrador astuto a menudo puede engañar al operador del computador para que realice una acción que ponga en peligro la seguridad del sistema.

Parásito.- El penetrador utiliza una terminal especial para intervenir una línea de comunicación. El penetrador intercepta mensajes entre el usuario y el procesador y los modifica o bien los reemplaza por completo.

Caballo de Troya.- El penetrador coloca código en el sistema que le permitirá un acceso posterior no autorizado. El Caballo de Troya puede dejarse en el sistema permanentemente o puede borrar todos los indicios de su existencia después de una penetración.

Parámetros inesperados.- El penetrador suministra valores inesperados en una llamada al supervisor para aprovechar un punto débil en los mecanismos de verificación de legalidad del sistema.

La confidencialidad de un computador no existe.
Aquí la idea es: Si no quieres que lean algo
no lo metas en el sistema.

Unidad VI

ENTRADA / SALIDA

Objetivo de esta unidad.- Durante el desarrollo de esta unidad, se analizará la forma de controlar los dispositivos de entrada/salida de la computadora.
El alumno deberá comprender los principios de hardware de E/S y de software de E/S.

Dispositivos de E/S

De manera general se dividen en dos categorías:

Dispositivo de bloque.- La propiedad esencial de un dispositivo de bloque es la posibilidad de leer o escribir en un bloque de forma independiente de los demás, es decir, en todo momento, el programa puede leer o escribir en cualquiera de los bloques. Los discos son dispositivos de bloque.

Dispositivos de caracter.- Un dispositivo de caracter envía o recibe un flujo de caracteres, sin sujetarse a una estructura de bloques. No se pueden utilizar direcciones ni tienen operación de búsqueda. Las terminales, impresoras de línea, cintas de papel, tarjetas perforadas, interfaces de red, mouse y otros dispositivos no parecidos a los discos son dispositivos de caracter.

Controladores de dispositivos.
Las unidades de E/S cuentan por lo general con un componente mecánico y un componente electrónico. El componente electrónico se llama controlador de dispositivo o adapatador, este toma con frecuencia la forma de tarjeta de circuitos impresos que se puede insertar en la computadora. El componente mecánico es el dispositivo mismo.

La labor del controlador es convertir el flujo de bits en serie en un bloque de bytes y llevar a cabo cualquier corrección de errores necesaria. Lo común es que el bloque de bytes se ensamble, bit a bit, en un buffer dentro del controlador. Después de verificar la suma y declarar al bloque libre de errores, se le puede copiar en la memoria principal.

El controlador de una terminal CRT también funciona como un dispositivo serial de bits en un nivel igual de bajo. Lee bytes que contienen los caracteres a exhibir en la memoria y genera las señales utilizadas para modular la luz CRT para que esta escriba en pantalla. El controlador también genera señales para que la luz CRT vuelva a realizar un trazo horinzontal después de terminar una línea de rastreo, así como las señales para que vuelva a hacer un trazo vertical después de rastrear en toda la pantalla. De no ser por el controlador CRT, el programador del sistema operativo tendría que programar en forma explícita el rastreo análogo del tubo de rayos catódicos.
Con el controlador el sistema operativo inicializa éste con pocos parámetros, tales como el número de caracteres por línea y el número de líneas en la pantalla, para dejar que el controlador se encargue de dirigir en realidad el rayo de luz.

Cada controlador tiene unos cuantos registros que utiliza para la comunicación con el CPU. En ciertas computadoras, estos registros son parte del espacio normal de direcciones de la memoria. Este esquema se llama E/S mapeada a memoria.

El manejador de disco es la única parte del sistema operativo que conoce el número de registros de un controlador de disco y el uso de que tienen estos. El se encarga de los sectores, pistas, cilindros, cabezas, movimiento del brazo, factores de separación, control de motor, tiempos de descenso de la cabeza y el resto de la mecánica del funcionamiento adecuado del disco.

Acceso Directo a Memoria (DMA)

Muchos controladores, en particular los correspondientes a los dispositivos de bloques, permiten el **a**cceso **d**irecto a **m**emoria, si no se utilizara éste, la lectura del disco se haría de la siguiente forma:

1. El controlador lee en serie el bloque de la unidad, bit por bit, hasta que todo el bloque se encuentra en el buffer interno del controlador.
2. Se calcula la suma de verificación para corroborar que no existen errores de lectura.
3. El controlador provoca una interrupción.
4. Cuando el sistema operativo empieza su ejecución, puede leer el bloque del disco por medio del buffer del controlador, un byte o una palabra a la vez, en un ciclo, en el que durante cada iteración se lee un byte o una palabra del registro del controlador y se almacena en memoria.

Un ciclo programado en el CPU para la lectura de bytes desde el controlador (uno a la vez) desperdicia tiempo de CPU.

Al utilizar DMA, el CPU le proporciona al controlador, dos elementos de la información, además de la dirección del bloque en el disco, la dirección en memoria donde debe ir el bloque y el número de bytes por transferir.

Después de que el controlador ha leído todo el bloque del dispositivo a su buffer y ha corroborado la suma de verificación, copia el primer byte o palabra a la memoria principal, en la dirección especificada por medio de la dirección de memoria DMA.

Entonces incrementa la dirección DMA y decrementa el contador DMA en el número de bytes que acaba de transferir. Este proceso se repite hasta que el contador se anula, momento en el cual el controlador provoca una interrupción. Al iniciar su ejecución el sistema operativo, no tiene que copiar el bloque en la memoria, ya esta ahí.

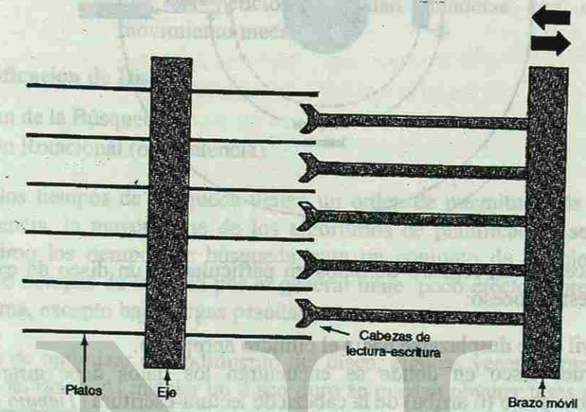
Interfaz uniforme para los manejadores de dispositivos
Nombres de los dispositivos
Protección del dispositivo
Proporcionar un tamaño de bloque independiente del dispositivo
Uso de buffers
Asignación de espacio en los dispositivos por bloques
Asignación y liberación de los dispositivos de uso exclusivo
Informe de errores

Funciones del software de E/S independiente del dispositivo

DISCOS

El uso de discos para almacenar información tiene tres ventajas con respecto al uso de la memoria principal como almacenamiento:

1. La capacidad de espacio de almacenamiento disponible es mucho más grande.
2. El precio por bit es más barato.
3. La información no se pierde al apagar la computadora.



Operación de un Almacenamiento de Disco de Cabeza Móvil.

Los datos se graban sobre una serie de *discos magnéticos* o *platos*. Estos discos están conectados por un *eje* común que gira a una velocidad muy alta (algunos ejes alcanzan a girar a 3600 revoluciones por minuto).

El acceso a los datos es mediante una serie de *cabezas de lectura-escritura*. Una cabeza de lectura-escritura sólo puede tener acceso a los datos que estén adyacentes a ella. De este modo, antes de que pueda obtenerse acceso a los datos, la porción de la superficie del disco de la que se leerán los datos (ó se escribirán) debe girar hasta que se encuentre justo abajo (ó arriba) de la cabeza de lectura-escritura.

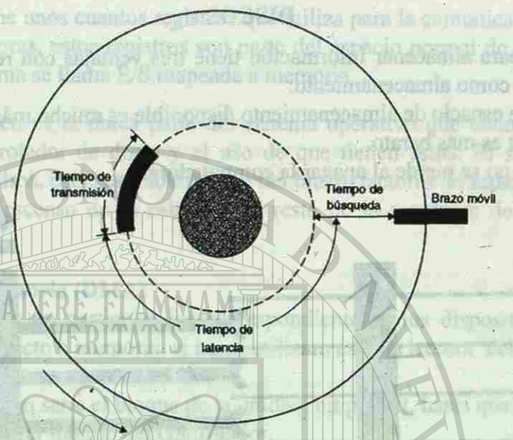
El tiempo que le toma a los datos girar desde la posición en que se encuentran hasta una adyacente a la cabeza de lectura-escritura se le llama: *Tiempo de Latencia*.

Cada una de las diferentes cabezas de lectura-escritura, mientras están fijas en una posición, determinan una *pista* circular de datos sobre la superficie de un disco.

Todas éstas cabezas están sujetas a una sola unidad de *brazo móvil* la cual puede moverse hacia adentro ó hacia afuera. Cuando el brazo móvil desplaza las cabezas hacia una nueva posición puede obtenerse acceso a otro conjunto de pistas.

Para una posición dada del brazo móvil, el conjunto de pistas definido por todas las cabezas forma un *cilindro* vertical.

El proceso de desplazar el brazo móvil hacia un nuevo cilindro se conoce como *operación de búsqueda (tiempo de búsqueda).*



Para obtener acceso a un registro de datos en particular en un disco de cabeza móvil se hacen las sigue este proceso:

1. El brazo móvil debe desplazarse hacia el cilindro apropiado.
2. La porción del disco en donde se encuentran los datos debe girar hasta quedar inmediatamente abajo (ó arriba) de la cabeza de lectura-escritura (*Tiempo de Latencia*).
3. El registro, cuyo tamaño es arbitrario (el máximo tamaño es una pista) debe girar para pasar por la cabeza (*Tiempo de transmisión*).

En síntesis, un acceso a disco (lectura o escritura) implicará tres acciones importantes:

1. Una búsqueda
2. Un retraso rotacional (latencia)
3. Una transmisión de registros

Cabe señalar que cada operación implica movimientos mecánicos, teniendo como resultado un tiempo total de acceso de una fracción de segundo (0.01 a 0.1 segundos). Estas velocidades pudieran parecer lentas si las comparamos con las velocidades de procesamiento.

En los sistemas multiprogramados, muchos procesos pueden estar haciendo solicitudes de lectura y escritura de registros en discos. En ocasiones estos procesos realizan peticiones más rápido de lo que pueden ser atendidas por los discos de cabeza móvil, se forman colas de espera para cada dispositivo. Por lo general algunos sistemas se limitan a atender las peticiones según el esquema *FCFS (First-Come-First-Served),* primero que llega, primero que se atiende, que es un método justo para asignar servicio, lamentablemente, cuando aumenta la carga, puede dar lugar a tiempos de espera muy largos. Además, exhibe un patrón de búsqueda aleatorio, provocando búsquedas de los cilindros más internos a los más externos, lo que hace que se consuma mucho tiempo.

La ineficiencia se debe al uso inapropiado de recursos de almacenamiento rotacional como discos y tambores.

Planificación de Disco.- El acomodo de solicitudes pendientes de acceso a disco para reducir las búsquedas se le conoce como planificación de disco e implica un examen cuidadoso de las peticiones pendientes para determinar la forma más eficiente de atenderlas.
Un planificador de disco examina las relaciones de posición entre las peticiones en espera. La cola de espera se reordena para que las peticiones puedan atenderse con un mínimo de movimiento mecánico.

Tipos de planificación de Disco

- Optimización de la Búsqueda
- Optimización Rotacional (o de latencia)

Debido a que los tiempos de búsqueda tienen un orden de magnitud más grande que los tiempos de latencia, la mayor parte de los algoritmos de planificación se concentran en reducir al mínimo los tiempos de búsqueda para un conjunto de peticiones y como la reducción de los tiempos de latencia por lo general tiene poco efecto sobre el rendimiento global del sistema, excepto bajo cargas pesadas.

En la búsqueda de optimizar el rendimiento del disco, algunas veces es preciso retardar el procesamiento del disco. Al instalar un disco duro se pueden obtener tasas de transferencia de disco mayores que las que puede manejar un computador personal, así que los datos excedentes se deben almacenar temporalmente en el controlador de disco. La alternación puede ser útil en estos casos. Para retardar la tasa efectiva de transferencia, los registros consecutivos de un archivo secuencial están separados por *n-1* bloques de disco a fin de dar oportunidad al procesador de alcanzar el disco. Esto da como resultado lo que se conoce como *alternación de disco de n vías*. Pero las velocidades de transferencia de disco son una cuestión independiente de la reducción de número de búsquedas mínimas; estos últimos aspectos siguen dominando las consideraciones de optimización del acceso a disco.

Características Deseables de las las Políticas de Planificación de Disco

Algunos criterios que se siguen para clasificar las políticas de planificación son:

La productividad.- El mayor número posible de peticiones atendidas por unidad de tiempo.

El tiempo promedio de respuesta.- Tratar de reducir el tiempo promedio de respuesta, es decir, el tiempo promedio de espera más el tiempo promedio de servicio. Como, la planificación reduce el tiempo de espera por búsquedas, debe ser ciertamente capaz de mejorar el tiempo promedio de respuesta de FCFS.

La planificación mejora a menudo el rendimiento global pero reduce el nivel de atención para ciertas peticiones.

La varianza de los tiempos de respuesta.- (predecibilidad)

La varianza es una medida matemática de cuanto se desvían elementos individuales del promedio de los elementos. Utilizamos la varianza para indicar la predecibilidad, esto es, a menor varianza mayor predecibilidad.

Si una política de planificación sólo trata de aumentar la producción sin reducir al mínimo la varianza, podría procesar nada más las peticiones fáciles y hacer caso omiso de las difíciles.

Optimización de la Búsqueda

SSTF.- (Primero en el menor tiempo de búsqueda)

El brazo del disco se traslada en seguida (en cualquier dirección) a la petición que requiere un movimiento mínimo. Esto es, atiende las solicitudes de acuerdo a su proximidad a la última solicitud atendida. La siguiente solicitud atendida es la más cercana a la última sin tener en cuenta la dirección en que se deba desplazar el brazo móvil.

Una desventaja importante es que aumenta la varianza de los tiempos de respuesta debido a la discriminación contra las pistas exteriores e interiores. En un caso extremo, podría causar aplazamiento indefinido de las solicitudes alejadas de las cabezas de lectura-escritura. Sin embargo, resulta útil en sistemas de procesamiento por lotes, donde la productividad es la consideración más importante. Aunque la elevada varianza de los tiempos de respuesta (impredecibilidad) lo hace inaceptable en sistemas interactivos.

SCAN .- El brazo del disco se mueve hacia adentro y hacia afuera, atendiendo todas las peticiones que encuentra a su paso, cambia su dirección sólo cuando no hay más peticiones que atender en la dirección actual. Es decir, funciona como SSTF excepto que el brazo se sigue moviendo en una dirección preferida hasta que se atienden todas las solicitudes en esa dirección; después se invierte el proceso.

A causa del movimiento oscilante de las cabezas de lectura-escritura las pistas más exteriores se visitan con menos frecuencia que las de la parte media, pero esto no es tan grave como la discriminación de SSTF.

C-SCAN.- (Scan circular)

El brazo se mueve en una sola dirección sobre la superficie del disco hacia la pista más interior, cuando no hay más peticiones en esa dirección regresa para atenderla petición más cercana a la pistqa exterior y de nuevo se mueve hacia adentro. Es decir, elimina la discriminación del SCAN contra las pistas de la parte media realizando el barrido en una sola dirección; al terminar un barrido el brazo móvil salta al extremo opuesto del disco y continua el barrido en la misma dirección

Algunos resultados de simulaciones en la literatura indican que la mejor política de planificación de disco podría operar en dos etapas. Cuando la

carga es ligera, la política SCAN es la mejor. Cuando la carga es mediana o pesada, C-SCAN produce los mejores resultados. C-SCAN con optimización rotacional maneja en forma efectiva las situaciones de carga pesada.

SCAN de N Pasos.- El brazo del disco se mueve igual que en Scan, pero las peticiones llegan durante el barrido en una dirección se almacenan y reordenan para darles un servicio óptimo durante el barrido de retorno. Esto es, evita los retrasos y posiblemente un aplazamiento indefinido al obligar a las solicitudes que llegan a que esperen el barrido en la dirección opuesta para ser atendidas.

El Scan de n pasos elimina la posibilidad de que ocurra un aplazamiento indefinido si llega un gran número de peticiones para el cilindro actual. Este guarda dichas peticiones para atenderlas en el barrido de regreso.

Esquema de Eschenbach.- En cada cilindro se atiende toda una pista de información sin importar si existen o no peticiones para ese cilindro. Las peticiones dentro de un cilindro se reordenan para atenderlas aprovechando su posición rotacional, pero si existen dos peticiones traslapadas dentro de un cilindro se atenderá sólo a una en ese barrido del brazo del disco. Fue diseñado para cargas en extremo pesadas, aunque C-SCAN con optimización rotacional ha demostrado ser más efectivo bajo todas las cargas.

Fue uno de los primeros en tratar de optimizar no solo el tiempo de búsqueda, sino también el retraso rotacional.

Optimización Rotacional

En condiciones de carga pesada aumenta la probabilidad de múltiples referencias a un cilindro específico por lo que resulta útil tener en cuenta la optimización rotacional además de la optimización de la búsqueda. Ha sido utilizada por años en dispositivos de cabeza fija como los tambores.

SLTF .- (shortest-latency-time-first)

La estrategia de primero el tiempo de latencia mas corto ó para la optimización rotacional es análoga de la estrategia SSTF para la optimización de la búsqueda. Una vez que el brazo del disco llega a un cilindro determinado, puede haber muchas solicitudes pendientes para las diversas pistas de dicho cilindro. Examina todas estas solicitudes y atiende primero la que tiene el retraso rotacional más corto.

La optimización rotacional se denomina a veces *puesta en cola de sectores*; los sectores se colocan en una cola según su posición alrededor del disco y se atiende primero los sectores más cercanos.

Consideraciones en los sistemas

Cuando el almacenamiento de disco resulta un cuello de botella, se tiende a agregar más discos al sistema. Hacerlo no siempre resuelve el problema porque el cuello de botella puede deberse a una carga pesada de solicitudes sobre un número relativamente pequeño de discos. Si se detecta este situación, la planificación de disco puede servir para mejorar la eficiencia y eliminar el cuello de botella.

Memoria Caché de Disco.- Es un área de almacenamiento primario en la que se conservan los registros de acceso frecuente para ayudar a evitar la necesidad de búsquedas largas de disco.

Cuando se realiza una operación de escritura, podría pensarse que el registro se graba de inmediato en el disco. Así sucede en algunos sistemas, pero en otros con memoria caché de disco la escritura sólo hace que el registro se almacene en un buffer en almacenamiento primario; el registro permanece ahí hasta que el sistema se queda sin espacio de buffer para escrituras subsecuentes, y en ese momento se graba el registro en el disco.

Si fuera necesario leer un registro escrito recientemente, se puede obtener del buffer de memoria caché de disco en almacenamiento primario mucho más rápido que si tuviera que leer del disco.

La clave para aprovechar la memoria caché de disco es mantener los registros de acceso frecuente en el buffer de memoria caché de disco de almacenamiento primario. Desde luego esta técnica sólo funciona bien cuando es posible identificar los regirtros de acceso frecuente. Utilizando la heurística de localidad, un registro al que se hizo referencia en el pasado reciente talvez será requerido en el fututo cercano.

Otras técnicas para mejorar el desempeño

La optimización del desempeño de los dispositivos rotacionales de almacenamiento se ha tratado de lograr empleando métodos de hardware, de sistemas operativos y de sistemas de aplicación. Por ello podríamos decir que otras técnicas para mejorar el desempeño son:

- La reorganización del disco para reducir la fragmentación;
- La partición del disco en la que se confinan los archivos a zonas pequeñas a fin de reducir la fragmentación;
- Colocar varias copias de datos estables de referencia frecuente en muchas partes del disco para reducir las distancias de búsqueda;
- La duplicación de datos estables de referencia frecuente en unidades de disco de acceso individual para lograr una mayor concurrencia;
- La agrupación de registros en bloques para reducir el numero de búsquedas;
- El mantenimiento de datos de acceso frecuente en posiciones de acceso más rápido dentro de la jerarquía de almacenamiento;
- La colocación de datos en las pistas de la zona media en dispositivos que utilizan una planificación tipo SCAN;
- La lectura de una pista completa a la vez para aprovechar la localidad espacial
- La compresión de datos para reducir el espacio requerido y por consiguiente, los tiempos de acceso

En general se ha evitado el aumento en las velocidades de rotación de los discos por causa de limitaciones físicas.

Discos de Ram.- Es un dispositivo de disco simulado en memoria convencional de acceso aleatorio. Elimina por completo los retrasos que se dan en los discos convencionales a causa de los movimientos mecánicos inherentes a las búsquedas y a la rotación del disco (los disco en RAM no implican movimientos mecánicos). Son útiles sobre todo en las aplicaciones de alto rendimiento.

Están separados de la memoria principal, de modo que no utilizan espacio requerido por el sistema operativo o las aplicaciones. Los tiempos de referencia a datos individuales son uniformes, sin la amplia variabilidad de los discos convencionales.

Los discos en RAM son caros. Son volátiles, es decir pierden su contenido cuando se apaga la computadora o se interrumpe el suministro de energía, podríamos solucionar esto con el uso de baterías de respaldo (UPS), pero si el corte de energía se prolonga estas se pueden agotar por lo que siempre es más recomendable hacer respaldos frecuentes en discos convencionales.

Discos de Opticos.- Pueden contener enormes cantidades de datos bajo un régimen de grabacion-única-lectura-múltiple o WORM write-once-read-many (los primeros discos láser). Los discos láser regrabables podrían desplazar los discos magnéticos de cabeza móvil.

Resultan apropiados para aplicaciones de archivo, pero no son muy útiles para aplicaciones que requieren actualización regular. Las capacidades enormes de estos discos (varios gigabytes, quizá 100 veces mayor que los discos duros empleados normalmente en las PC's) han hecho que resulten realmente efectivos en algunas aplicaciones

RELOJES

También llamados cronómetros son esenciales para la operación de cualquier sistema de tiempo compartido. El sofware de reloj toma la forma de un manejador de dispositivo, aunque un reloj toma por la forma de un manejador de dispositivo, aunque un reloj no es un dispositivo de bloque, como un disco, ni un dispositivo de caracter, como una terminal.

Cronómetro de intervalos o reloj de interrupciones

Los relojes no tiene direcciones por medio de bloques, tampoco generan o aceptan flujos de caracteres, lo único que hacen es provocar interrrupciones a intervalos bien definidos.

Un proceso que tiene asignado el CPU esta en ejecución. Sí el proceso pertenece al sistema operativo, se dice que el sistema operativo esta en ejecución y puede tomar decisiones que afectan la operación del sistema. Para evitar que los usuarios monopolicen el sistema (accidental o deliberadamente), el sistema operativo tiene mecanismos para arrebatar el CPU al usuario.

El sistema operativo mantiene un *reloj de interrupciones o cronómetro de intervalos* para generar interrupciones en algún momento futuro específico (o después de cierto tiempo). El reloj de interrupciones ayuda a garantizar tiempos de respuesta aceptables para los usuarios interactivos, evita que el sistema quede bloqueado en un ciclo infinito de algún usuario y permite que los procesos respondan a eventos dependientes de tiempo. Los procesos que deben ejecutarse periódicamente dependen del reloj de interrupciones.
Por lo general las labores de un reloj son:

1. Mantener la hora del día
2. Evitar que los procesos se ejecuten más tiempo del permitido.
3. Mantener un registro del uso del CPU
4. Controlar la llamada al sistema ALARM por parte de los procesos del usuario.
5. Proporcionar cronómetros guardianes de partes del propio sistema.
6. Realizar resúmenes, monitoreo y recolección de estadísticas

TERMINALES

Cada computadora tiene una o más terminales que se utilizan para comunicarse con ella. Las terminales tienen un gran número de formas distintas. El manejador de terminal se encarga de ocultar todas estas diferencias, de forma que la parte independiente del dispositivo en el sistema operativo y los programas de usuario no tienen que volverse a escribir para cada tipo de terminal.
Los editores de pantalla y muchos otros programas sofisticados deben poder actualizar la pantalla de maneras complejas que le simple recorrido del texto en la partre inferior de la misma. Para lograr esto, muchos manejadores de terminales soportan varias secuencias de escape. A continuación se listan algunas:

1. Mover el cursor hacia arriba, abajo, a la izquierda o a la derecha una posición.
2. Mover el cursor a *x, y*
3. Insertar un caracter o una línea n el cursor
4. Eliminar un caracter o una línea del cursor
5. Recorrer la pantalla havia arriba o hacia abajo *n* líneas
6. Limpiar la pantalla desde el cursor hasta el final de la línea o hasta el final de la pantalla.
7. Trabajar en modo video inverso, subrayado, parpadeo o normal
8. Crear, destruir, mover o controlar ventanas.

Cuando el manejador ve el caracter que inicia la secuencia de escape, activa una bandera y espera a que llegue el resto de la secuencia. Cuando toda la secuencia ha llegado, el manejador debe llevarla a cabo en software. La inserción y eliminación de texto requiere el movimiento de bloques de caracteres en el video RAM. El hardware no ayuda más que en recorrer descendente o ascendentemente texto y en exhibir el cursor.

DESEMPEÑO, COPROCESADORES, RISC Y FLUJO DE DATOS

Tres objetivos comunes de la evaluación del rendimiento:

1) **Evaluación para la Selección**.- El evaluador del desempeño debe decidir si conviene adquirir un sistema de cómputo de un proveedor específico.

2) **Proyección del Rendimiento.-** El objetivo del evaluador e n este caso es estimar el desempeño de un sistema inexistente. Puede tratarse de un sistema de cómputo totalmente nuevo ó de un nuevo componente de hardware o software.
3) **Supervisión del Desempeño**.- el evaluador acumula datos de un sistema ó componente ya existente para asegurarse de que cumple con sus objetivos de desempeño, para ayudar a estimar el impacto de las modificaciones y para ofrecer a los administradores la información que necesitan para tomar decisiones estratégicas tales como modificar o no un sistema existente de prioridades de trabajo.

Evaluación y Predicción del Desempeño.- Se necesitan desde los primeros momentos de la concepción de un nuevo sistema, en la operación cotidiana del sistema después de la instalación y cuando se estudia la modificación o posible sustitución de un mejor sistema.

Desempeño.- Es la eficiencia con la que un sistema cumple sus objetivos

Medidas de desempeño

Tiempo de Retorno.- En un sistema de procesamiento por lotes se define como el tiempo transcurrido desde la entrega de un trabajo hasta la devolución al usuario.

Tiempos de Respuesta.- Es el tiempo de retorno en un sistema interactivo y a menudo se le define como el tiempo transcurrido desde que el usuario presiona el "enter" ó "botón del mouse" hasta que el sistema comienza a imprimir o exhibir una respuesta.

Tiempo de Reacción del Sistema.- Es el tiempo transcurrido desde que el usuario presiona "enter" ó el "botón del mouse" hasta que se otorga la primera tajada de tiempo de servicio a solicitud del usuario.

La Varianza en los Tiempos de Respuesta.- Es una medida de predecibilidad.

Producción.- Es la medida del rendimiento de trabajo por unidad de tiempo

Carga de Trabajo.- Es la medida de la cantidad de trabajo que se ha introducido al sistema, cantidad que el sistema debe procesar en condiciones normales para que su funcionamiento se considere aceptable.

Capacidad.- Es la medida de producción máxima que puede tener un sistema.

Utilización.- Es la fracción de tiempo en que está en uso un recurso (es una medida engañosa porque al parecer lo mejor es tener un porcentaje de utilización alto y ésto podría ser señal de un aprovechamiento ineficiente).

Técnicas para Evaluar el Desempeño

Tiempos.- Son útiles para realizar comparaciones rápidas entre equipos.

Mezcla de Instrucciones.- Emplean un promedio ponderado de diversos tiempos de

instrucciones más adecuadas para una aplicación específica.

Programa Núcleo.- Es un programa representativo que podría ejecutarse en una instalación. Se cronometra para una máquina dada empleando las estimaciones de tiempo de instrucciones del fabricante y así se pueden hacer comparaciones entre máquinas distintas de acuerdo con la velocidad de ejecución esperada del programa núcleo.

Modelos Analíticos.- Son representaciones matemáticas de sistemas de cómputo o de sus componentes. Existe un volumen considerable de resultados matemáticos que puede aplicar el evaluador para ayudar a estimar el desempeño de un sistema de cómputo dado (Ejemplo: Teoría de colas y modelo de Markov).

Bancos de Prueba.- Es un programa real que el evaluador ejecuta en un sistema de cómputo en evaluación. El evaluador conoce las características del rendimiento del banco de prueba en un equipo ya existente de tal modo que cuando se ejecuta en un equipo nuevo el evaluador puede sacar conclusiones significativas.

Programas Sintéticos.- Son programas reales diseñados a la medida para ejercitar funciones específicas de un sistema de cómputo. Son útiles sobre todo cuando no existen bancos de prueba que realicen dichas funciones.

Simulación.- Es una técnica con la cual el evaluador desarrolla un modelo computarizado del sistema de evaluación. Luego el modelo se ejecuta en un sistema de cómputo con lo que se refleja el comportamiento del sistema en evaluación.

- Hay simuladores que se manejan por eventos que se producen en el simulador de acuerdo con distribuciones de probabilidades.
- Los simuladores manejados por libreto son controlados por datos derivados empíricamente y manipulados con cuidado de manera que reflejen el comportamiento esperado del sistema simulado.

Supervisión del Desempeño.- Es la obtención y el análisis de información acerca del rendimiento de sistemas ya existentes.

Cuellos de Botella y Saturación

Cuello de Botella.- Un recurso se convierte en *cuello de botella* y limita el desempeño total del sistema cuando no es capaz de manejar el trabajo que se le envía.

Saturación.- Los recursos que operan cerca de su capacidad máxima tienden a saturarse, es decir, los procesos que compiten por la atención del recurso comienzan a interferir unos con otros (Ejemplo: Hiperpaginación)

Ciclo de Retroalimentación.- Es un caso en el que cierta información acerca del estado

actual del sistema puede afectar las solicitudes que llegan. Si la retroalimentación indica que dichas solicitudes pueden tener problemas para ser atendidas, quizas se les pueda enviar por otro conducto.

Retroalimentación negativa.- La tasa de llegada de retroalimentación negativa contribuye a la estabilidad en sistemas manejados por colas (La tasa de llegada de solicitudes nuevas puede disminuir como resultado de la información retroalimentada).

Retroalimentación positiva.- En estos sistemas, la información retroalimentada origina un aumento en algún parámetro, puede provocar inestabilidad es sistemas manejados por colas.

Coprocesadores.- Son procesadores de aplicación especial que se agregan a los sistemas de cómputo para realizar operaciones que no están incluidas en el procesador o procesadores originales.

RISC .-
(Computación con un conjunto reducido de instrucciones)

Las arquitecturas de un RISC casi siempre tiene conjuntos de instrucciones en lenguaje de máquina poco abundantes relativamente sencillas de carga y almacenamiento para transferir datos entre la memoria y los registros con ductos profundos y memoria caché.

Aunque los programas RISC suelen ser mas tardos que sus equivalentes en CISC, casi siempre se ejecutan en forma rápida.

COMPUTACIÓN DISTRIBUIDA: LA PERSPECTIVA DE LA COMPUTACIÓN EN PARALELO.

Canalización.- Es una técnica para mejorar el rendimiento permitiendo que varias instrucciones en lenguaje de maquina estén en diferentes etapas de ejecución al mismo tiempo. Cada etapa de la canalización de instrucciones realiza una etapa diferente sobre la instrucción y después la instrucción avanza a la siguiente etapa. La canalización hace posible que muchas instrucciones, cada una en diferente fase de ejecución, progresen al mismo tiempo.

Multiprocesadores.- Uno de los atractivos de un sistema con múltiples procesadores o multiprocesamiento es que si falla un procesador por lo regular se puede seguir trabajando con los procesadores restantes. El sistema operativo debe darse cuenta de que un procesador especifico ha fallado y ya no está disponible para ser asignado.

Hay dos técnicas comunes para detectar el paralelismo:

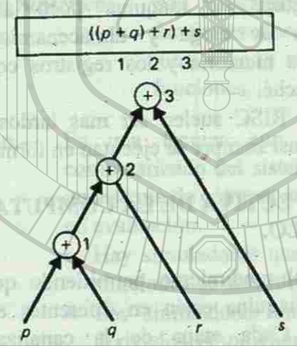
1. Distribución de ciclos: Considere la siguiente instrucción que suma los elementos de los arreglos "b" y "c" y coloca las sumas en el arreglo "a". Este ciclo hace que un procesado secuencial realice las 4 interacciones del ciclo una tras otra.

```
for i = 1 to 4 do
a(i) = b(i) + c(i)
cobegin
        a(1) = b(1) + c(1)
        a(2) = b(2) + c(2)
        a(3) = b(3) + c(3)
        a(4) = b(4) + c(4)
coend
```

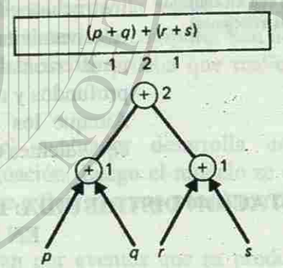
Todas las interacciones se pueden realizar independientemente una de otra separándose en unidades ejecutables de manera concurrente (al mismo tiempo).

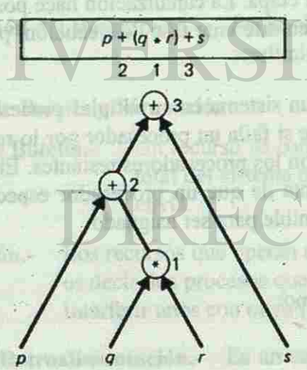
2. Reducción de Altura de Arboles.- Aplica la propiedades de conmutividad, asociatividad y propiedad distributiva de las operaciones aritméticas para reacomodar las expresiones algebraicas y hacerlas más adecuadas para la ejecución concurrente.

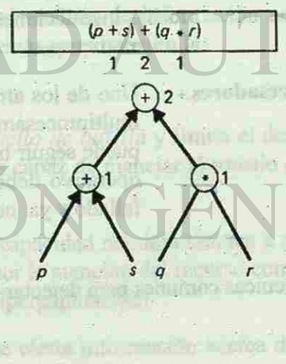
((p + q) + r) + s

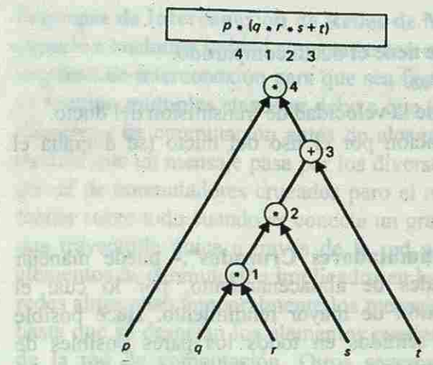


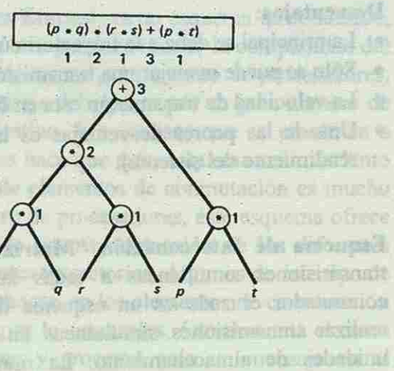
(p + q) + (r +s)











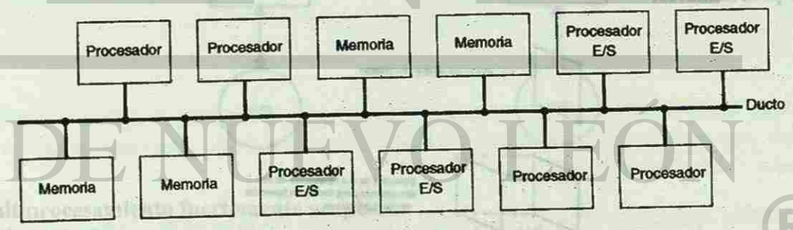
Regla de Nunca Esperar.- Establece que es mejor darle al procesador una tarea que puede ser utilizada o no posteriormente que dejar al procesador sin trabajo.

ESQUEMAS DE INTERCONEXION DE LOS PROCESADORES

Los problemas fundamentales en el diseño de sistemas de multiprocesamiento es determinar la forma de conectar los múltiples procesadores y los procesadores de E/S con las unidades de almacenamiento.

Esquema de interconexión "Ducto Compartido".- utiliza una sola trayectoria de comunicación con todos los procesadores, las unidades de almacenamiento y los procesadores de E/S. Esencialmente es una unidad pasiva.

Ejemplo: Esquema de red de área local: Ethernet.



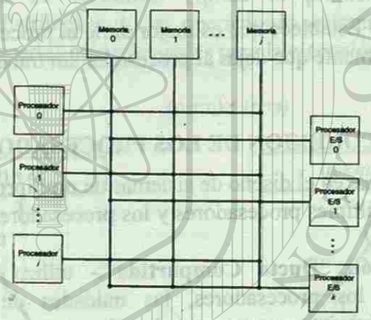
Ventajas:

- Se le pueden fácilmente adicionar nuevas unidades, basta conectarlas directamente al ducto para que ocurra la comunicación, cada unidad debe saber cuales otras unidades están conectadas al ducto, esto se maneja por software.

Desventajas:

- La principal se debe a la trayectoria única que tiene el ducto compartido.
- Sólo se puede manejar una transmisión a la vez.
- La velocidad de transmisión esta en función de la velocidad de transmisión del ducto.
- Una de las peores desventajas es la contención por el uso del ducto (se degrada el rendimiento del sistema).

Esquema de Interconexión "Matriz de Conmutadores Cruzados".- puede manejar transmisiones simultáneas a todas las unidades de almacenamiento, por lo cual el conmutador cruzado es un esquema de conexión de mayor rendimiento. Hace posible realizar transmisiones simultáneas en ambos sentidos en todos los pares posibles de unidades de almacenamiento. La mayor desventaja es el grado de complejidad del conmutador cruzado.



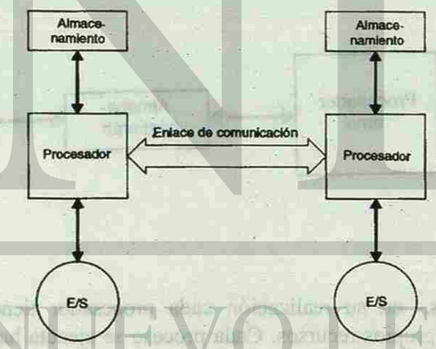
Esquema de Interconexión "Hipercubo".- la red de interconexión hipercubo hace posible conectar un gran numero de procesadores en forma relativamente económica. Utiliza un esquema de conexión muy económica y un hipercubo de 16 dimensiones puede conectar 65,536 procesadores.

Esquema de Interconexión de Redes de Múltiples Etapas.- es un esquema de concesión, permite a cualquier unidad conectarse con cualquier otra y reduce mucho la complejidad del esquema de interconexión para que sea factible conectar un gran numero de procesadores. El término múltiples etapas se debe a que cada mensaje debe pasar por un gran numero de elementos de conmutación antes de alcanzar su destino. Los retrasos que se presentan a medida que un mensaje pasa por los diversos tramos hace que éste esquema sea más lento que el de conmutadores cruzados pero el numero de elementos de conmutación es mucho menor sobre todo cuando se conecta un gran numero de procesadores, éste esquema ofrece una trayectoria única a través de la red pero puede ocurrir contención de los diferentes elementos de conmutación implicados en las diferentes trayectorias. Algunos de éste tipo de redes almacenan temporalmente los mensajes que chocan en los elementos de conmutación hasta que se despejan los elementos pueden proseguir los mensajes, ésto aumenta el costo de la red de computación. Otros sistemas aceptan y procesan solo un mensaje en el conmutador y devuelven los demás mensajes que chocan para ser retransmitidos posteriormente con un retraso aleatorio.

Sistemas Fuertemente y Débilmente Acoplados

Multiprocesamiento débilmente acoplado

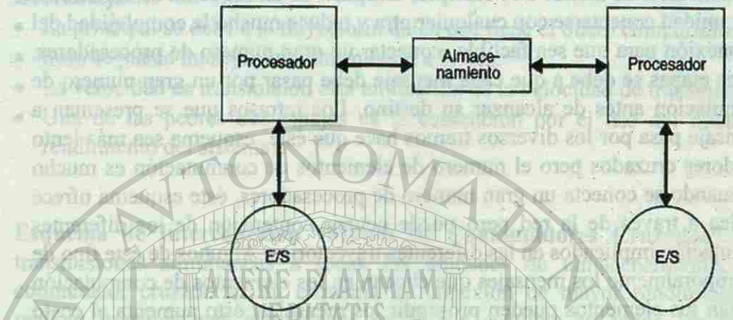
Implica conectar dos o mas sistemas de computo independientes mediante un enlace de comunicación. Cada sistema tiene su propio (procesador) sistema operativo y su almacenamiento. Los sistemas pueden funcionar en forma independiente y se pueden comunicar entre sí si es necesario.



Multiprocesamiento fuertemente acoplado

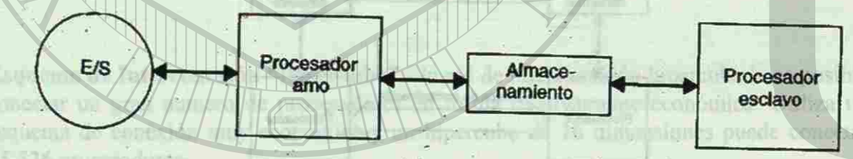
Utiliza un almacenamiento único compartido por los diferentes procesadores y un solo sistema operativo que controla todos los procesadores y el hardware del sistema. La comunicación se realiza con la memoria compartida.

Un aspecto clave de los sistemas fuertemente acoplados es la contención por la memoria compartida. Algunos estudios indican que si se equilibra la arquitectura y se distribuye la carga de trabajo a lo largo del sistema la contención es mínima.



Organizaciones basicas de los sistemas operativos para multiprocesadores
La capacidad para aprovechar el paralelismo en el hardware y en los programas es fundamental. La adición de más procesadores así como las complejas conexiones con la memoria y los procesadores de E/S aumenta considerablemente el costo del hardware.

Amo-Esclavo.- es la más fácil de llevar a la practica, solo el Amo puede ejecutar el sistema operativo el cual no tiene que ser reentrante y se simplifica mucho la exclusión mutua en el acceso a los datos del sistema. Si el Amo falla el sistema ya no podrá efectuar operaciones de E/S.



Ejecutivos Individuales.- en su realización cada procesador tiene su propio sistema operativo y controla sus propios recursos. Cada proceso se ejecuta hasta completarse en el procesador al que fue asignado.

Organización Simétrica.- el multiprocesador simétrico es la organización mas compleja de llevar a la práctica pero también es la mas poderosa. El sistema operativo administra un conjunto de procesadores. La falla de algún procesador hace que el sistema operativo lo retire del conjunto de procesadores pero sigue funcionando. Un proceso dado se puede ejecutar en distintas ocasiones en procesadores diferentes.

La contención podría convertirse en un problema grave debido a que varios procesadores pueden estar en estado supervisor al mismo tiempo.CC

Bibliografía

- **Sistemas Operativos**
  William Stallings
  Editorial Limusa / Megabyte
  1ª Edición

- **Sistemas Operativos**
  Milan Melenkoviç
  Editorial Mc Graw Hill
  2ª Edición

- **Sistemas Operativos**
  H. M. Deitel
  Addisson-Wesley Iberoamericana
  2ª Edición

- **Sistemas Operativos Modernos**
  Andrew S. Tanenbaum
  Prentice-Hall Hipanoamericana