UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMAS OPERATIVOS: DISEÑO E IMPLANTACION, UN ENFOQUE ACADEMICO

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A:

FERNANDO ALTAMIRANO RUISECO

Director de Tesis:
MANUEL MANRIQUEZ MIRANDA

México, D. F. 1997

Dedicatoria

A mi esposa. Por estar ahí cuando la necesito, por apoyarme en todo momento y por comprender siempre mi comportamiento.

◆ INTRODUCCION

Para conocer y entender mejor el funcionamiento de las computadoras modernas tales como las computadoras personales, minicomputadoras y sistemas centrales o macrocomputadoras, es necesario conocer como el software administra el hardware entre los diferentes procesos de usuario que se ejecutan en él. Estas funciones de administración son descritas aquí a los largo de 7 capítulos, en los cuales se detallarán las técnicas utilizadas para el diseño e implantación de un sistema operativo moderno.
Comenzaremos nuestro estudio contestando las principales preguntas que se hacen acerca de los sistemas operativos

¿Qué es un sistema operativo?

Existen diferentes autores que han definido a los sistemas operativos como

Un sistema operativo es un programa que actúa como intermediario entre el usuario y el hardware de una computadora, por lo tanto es una parte importante de cualquier sistema de cómputo [1].
Un sistema operativo puede ser también visto como una colección organizada de extensiones del hardware utilizando software, las cuales consisten en rutinas de control que hacen funcionar la computadora y proporcionan un entorno para la ejecución de los programas. [2]
Es evidente que se necesita una nueva definición de sistema operativo. Se puede imaginar un sistema operativo como los programas instalados (el software) que hacen utilizable el hardware. El hardware proporciona la capacidad bruta de cómputo; los sistemas operativos ponen esa capacidad de cómputo al alcance de los usuarios y administran cuidadosamente el hardware para lograr un buen rendimiento.
Los sistemas operativos son ante todo administradores de recursos; el principal recurso que administran es el hardware de la computadora, este hardware está compuesto por:

- Los procesadores y la memoria principal (la mayoría de las microcomputadoras solo tienen un procesador y una memoria principal).
- Los medios de almacenamiento (ya sean discos, memoria central, cintas, CD-ROM, etc.)
- Los dispositivos de entrada y salida (terminales, impresoras, mouse, lápiz óptico, scanner, modem, etc.)

En general los sistemas operativos realizan una gran cantidad de funciones, tales como: proporcionar la interfaz con el usuario, permitir que los usuarios compartan entre si el hardware y los datos, evitar que los usuarios interfieran recíprocamente, planificar la distribución de los recursos entre los usuarios,

facilitar la entrada y salida, recuperarse de los errores que pudieran surgir, contabilizar el uso de los recursos físicos y lógicos, facilitar las operaciones en paralelo, organizar los datos para lograr un acceso rápido y seguro, y manejar las comunicaciones en red [3]
Existen varias definiciones adicionales de lo que es un sistema operativo, para nuestro caso en cuestión consideraremos a un sistema operativo como:

- Una máquina extendida, que presenta el hardware de una manera más clara y más fácil de utilizar. [4]
- Un controlador de recursos, el cual administra repartiendo de una manera adecuada los recursos de cómputo. [4]

Consideramos las dos definiciones anteriores debido a que como usuarios requerimos ver la computadora por medio del sistema operativo (máquina extendida o ampliada) para poder utilizarla de una manera más sencilla y eficiente, y como programadores de sistemas requerimos ver la computadora como un conjunto de recursos los cuales administraremos con la creación del software adecuado (sistema operativo).

¿Para qué sirve?

Como se mencionó en la sección anterior los sistemas operativos tienen dos objetivos básicos los cuales son administrar de una manera eficiente los recursos de cómputo y presentar el hardware de una manera más amigable y fácil de usar. Por lo tanto podemos decir, que un sistema operativo sirve para aprovechar los recursos de cómputo.
Cada computadora moderna posee un sistema operativo, y en caso que no lo tuviera sería muy difícil para un usuario aprovecharla para realizar alguna tarea útil, ya que tendría que escribir programas de bajo nivel para administrar cada uno de los dispositivos conectados a ella (incluyendo el video y el teclado) lo que haría mucho más complicada su utilización.
Los sistemas operativos actuales poseen gran cantidad de funciones que facilitan la utilización de las computadoras modernas, los cuales serán mencionados en la siguiente sección.

¿Qué funciones realiza?

Los sistemas operativos realizan múltiples funciones para proporcionar al usuario las herramientas necesarias para el mejor aprovechamiento del hardware.
El sistema operativo ofrece varias funciones dependiendo de como es éste visto. Si vemos al sistema operativo como un administrador de recursos de cómputo podemos citar las siguientes funciones:

- Mantener la contabilidad del estado de los recursos

- Reforzar las politicas que determinan quien obtiene que recurso, cuando se obtiene dicho recurso y en que cantidad se otorga el recurso
- Alojar el recurso
- Reclamar el recurso

Cada uno de los componentes principales del sistema operativo posee las funciones anteriormente mencionadas; estos componentes según el recurso que administran son en general.

- Administrador de memoria
- Administrador de procesador
- Administrador de dispositivos
- Administrador de archivos [5]

Cada uno de estos componentes serán analizados con detalle incluyendo su implantación.

¿Porqué es útil conocer su funcionamiento?

Consideramos que es importante conocer el funcionamiento de los sistemas operativos debido a que si conocemos su funcionamiento es más fácil aprovechar sus características específicas, seleccionar el sistema operativo de nuestra conveniencia o como nuestro caso diseñar uno que aproveche los recursos de cómputo de la manera que nosotros deseamos o requerimos.
Es importante conocer su estructura para que en caso de que diseñemos aplicaciones para un sistema operativo específico estas aplicaciones se ajusten a sus políticas y tengan una integración adecuada aprovechando al máximo los servicios proporcionados.

Un sistema operativo es en general un proyecto complejo y extenso, pero en él se muestran la mayor parte de los avances de la ingeniería del software, ya que representa por si mismo aproximadamente 40 años de desarrollo y mejora del software, por lo cual se considera muy útil su entendimiento.

◆ CAPITULO I.- EVOLUCION DE LOS SISTEMAS OPERATIVOS

Es importante conocer la estructura básica de un sistema operativo para poder comparar sus cambios a lo largo de la historia.

Estructura de un sistema operativo

Existen varios tipos de estructuras de los sistemas operativos, de los cuales mencionaremos los cuatro tipos más utilizados.

a) Sistemas monoliticos

Este tipo de organización es por mucho la más común. La estructura consiste en que no tiene estructura alguna. El sistema operativo se escribe como una colección de procedimientos, cada uno de los cuales puede llamar a los demás cada vez que así lo requieran. Cuando se usa esa técnica, cada procedimiento del sistema tiene una interfaz bien definida en términos de parámetros y resultados y cada uno de ellos es libre de llamar a cualquier otro, si éste último proporciona cierto servicio útil para el primero.

Sin embargo, incluso en los sistemas monolíticos es posible tener al menos algo de estructura. Los servicios (llamadas al sistema) que proporciona el sistema operativo se solicitan colocando los parámetros en lugares bien definidos, como en los registros del procesador o en la pila, para ejecutar una instrucción especial de trampa conocida como llamada al nucleo o llamada al supervisor.
Esta instrucción cambia la computadora del modo usuario al modo núcleo (también conocido como modo supervisor) y transfiere el control al sistema operativo. (la mayoría de las CPU's tiene dos modos: el modo del nucleo para el sistema operativo, en el que se permiten todas las instrucciones y el modo usuario, para los programas del usuario, en donde no se permiten instrucciones de E/S y ciertas instrucciones privilegiadas).

El sistema operativo examina entonces los parámetros de la llamada primeramente analizando si esta es una llamada válida, posteriormente habiendo determinado la validez de la misma se verifica que el solicitante de la llamada tenga permiso para realizarla, más tarde, para determinar cual de ellas se desea realizar el sistema operativo analiza una tabla que contiene en la entrada *k* un apuntador al procedimiento que realiza la k-ésima llamada al sistema. En esta tabla generalmente se encuentran todas las llamadas implantadas en el sistema operativo, es decir, la tabla tiene n entradas (n llamadas al sistema). Esta operación identifica el procedimiento de servicio solicitado, el cual finalmente es llamado (cargado y ejecutado). Por último, la llamada al sistema termina su ejecución y el control regresa al programa del usuario.

Esta organización sugiere una estructura básica del sistema operativo.

- Un programa principal que llama al procedimiento del servicio solicitado
- Un conjunto de procedimientos de servicio que llevan cabo las llamadas al sistema
- Un conjunto de procedimientos utilitarios que ayudan al procedimiento de servicio

En este modelo, para cada llamada al sistema existe un procedimiento de servicio que se encarga de él. Los procedimientos utilitarios hacen cosas necesarias para varios procedimientos de servicio

b) Sistemas en capas

Este tipo de sistemas consiste en organizar el sistema operativo como una jerarquía de capas, cada una constituida sobre la inmediata inferior. El primer sistema construido de esta manera fue el sistema THE, desarrollado en Holanda por E. W. Dijkstra (1968) y sus estudiantes. El sistema THE era un sistema sencillo de procesamiento por lotes para una computadora holandesa, la Electrológica X8, con 32K de palabras de 27 bits (los bits eran muy costosos en aquel entonces).
El sistema tenía 6 capas, las cuales son:

5	El operador
4	Programas del usuario
3	Control de E/S
2	Comunicación operador-procesos
1	Administración de la memoria y del disco
0	Asignación del procesador y multiprogramación

La capa 0 trabaja con la asignación del procesador y alterna entre los procesos cuando ocurren las interrupciones o cuando expiran los tiempos asignados. Sobre la capa 0, el sistema consta de procesos secuenciales, cada uno de los cuales se podía programar sin tener que preocuparse por el hecho de que varios procesos estuvieran en ejecución en el mismo procesador. En otras palabras, proporcionaba la multiprogramación básica de la CPU.
La capa 1 realizaba la administración de la memoria. Asignaba el espacio de memoria principal para los procesos y un tambor (el equivalente del disco actual) de palabras de 512K se utilizaba para almacenar partes de los procesos (páginas) para las que no existía lugar en la memoria principal. Por encima de la capa 1, los procesos no debían preocuparse si estaban en la memoria o en el recipiente; el software de la capa 1 se encargaba de garantizar que las páginas llegaran a la memoria cuando fueran requeridas.
La capa 2 se encargaba de la comunicación entre cada proceso y la consola del operador. Por encima de esta capa, cada proceso tiene su propia consola de

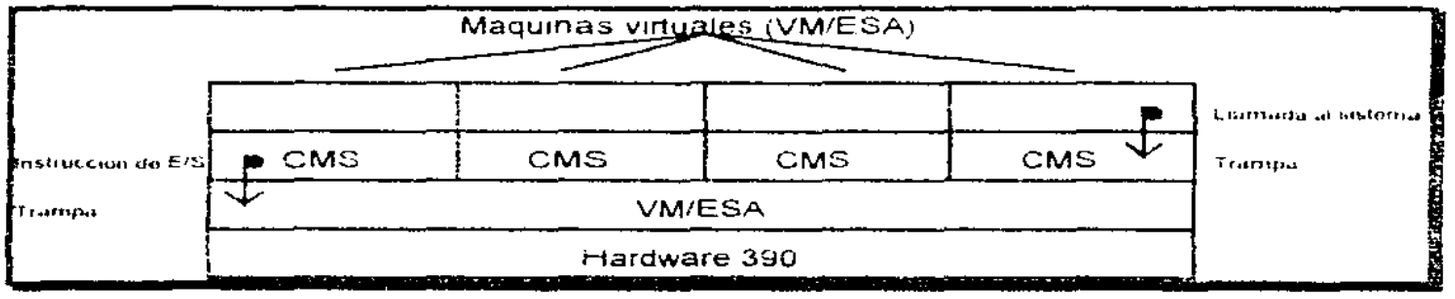
operador. La capa 3 controla los dispositivos de E/S y guarda en almacenes (buffers) los flujos de información entre ellos Por encima de la capa 3, cada proceso puede trabajar con dispositivos virtuales de E/S con propiedades adecuadas, en lugar de dispositivos reales con muchas particularidades. La capa 4 es donde estaban los procesos del usuario Estos no tenían que preocuparse por su proceso, memoria, consola o control de E/S El proceso operador del sistema se localizaba en la capa 5.
Una generalización más avanzada del concepto de capas se presentó en el sistema MULTICS. En lugar de capas, MULTICS estaba organizado como una serie de anillos concéntricos, siendo los anillos interiores los privilegiados. Cuando un procedimiento de un anillo exterior deseaba llamar a un procedimiento de un anillo interior, debía hacer el equivalente a una llamada del sistema; es decir, una instrucción TRAP (trampa) cuyos parámetros eran validados con cuidado antes de permitir que procediera la llamada. Aunque todo el sistema operativo era parte del espacio de direcciones de cada proceso del usuario en MULTICS, el hardware posibilitó el diseño de procedimientos individuales (en realidad segmentos de memoria) de forma protegida contra la lectura, escritura o ejecución.
Mientras que el sistema de capas de THE era en realidad un apoyo al diseño, debido a que todas las partes del sistema estaban, en última instancia, ligadas entre sí en un solo programa objeto, en MULTICS, el mecanismo de anillos estaba más presente durante el tiempo de ejecución y era reforzado por el hardware. La ventaja del mecanismo de anillos es su facilidad de extensión para estructurar subsistemas del usuario.

c) Máquinas virtuales

Las versiones iniciales del OS/360 (sistema operativo de IBM) eran, en sentido estricto, sistemas de procesamiento por lotes. Sin embargo, muchos de los usuarios del hardware de 360 deseaban tener tiempo compartido, de forma que varios grupos, tanto dentro como fuera de IBM, decidieron escribir sistemas de tiempo compartido para tal hardware. El sistema de tiempo compartido oficial de IBM, el TSS/360, tuvo una entrega muy retrasada y cuando finalmente apareció, era tan grande y lento que fue adoptado en muy pocos lugares. Un grupo del centro científico de IBM en Cambridge, Mass., produjo un sistema con diferencias radicales, el cual fue aceptado por IBM como producto y que ahora tiene un uso muy extenso.
Este sistema, cuyo nombre original era CP/CMS y que ahora se llama VM/ESA se basó en una observación muy inteligente; un sistema de tiempo compartido proporciona dos elementos esenciales:

- Multiprogramación
- Una máquina extendida con una interfaz más conveniente que el hardware solo

La esencia del VM/ESA es la total separación de estas dos funciones.
El nucleo del sistema, llamado monitor de la máquina virtual, se ejecuta en el hardware simple y realiza la multiprogramación, proporcionando no una, sino varias máquinas virtuales a la siguiente capa superior, como se muestra en la siguiente figura.

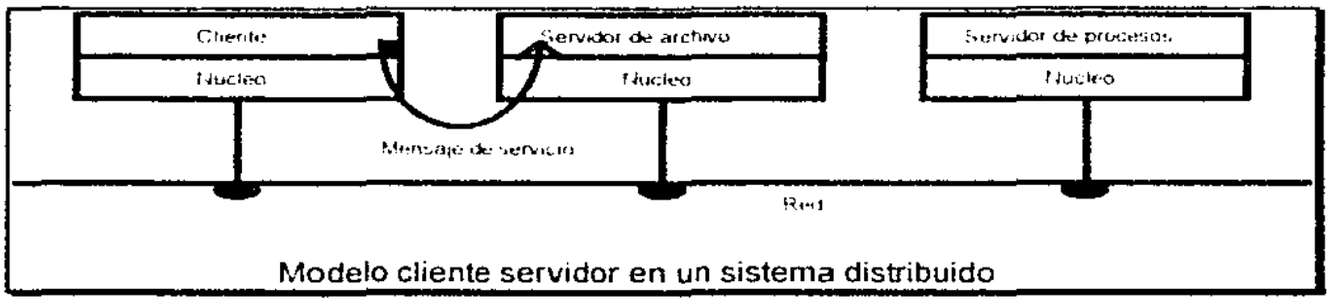


Sin embargo, a diferencia de los demás sistemas operativos, estas máquinas virtuales no son máquinas extendidas, con archivos u otras caracteristicas adecuadas. En lugar de esto, son copias exactas del hardware simple, con su modo nucleo/usuario, E/S, interrupciones y todo lo demás que posee la máquina real.
Puesto que cada máquina virtual es idéntica al hardware real, cada una puede ejecutar cualquier sistema operativo que se ejecute en forma directa sobre el hardware. Las distintas máquinas virtuales pueden, y por lo general lo hacen, ejecutar distintos sistemas operativos. Algunas ejecutan uno de los descendientes del OS/360 para el procesamiento por lotes, mientras que otras ejecutan un sistema interactivo de un solo usuario, llamado CMS (Conversational Monitor System), para los usuarios de tiempo compartido.

d) Modelo cliente-servidor

Una tendencia de los sistemas operativos modernos es la de explotar la idea de mover cada vez más código a capas superiores y eliminar la mayor parte del sistema operativo para mantener un nucleo mínimo. El punto de vista usual es el de implantar la mayoría de las funciones del sistema operativo en procesos de usuario. De esta manera para solicitar un servicio, como la lectura de un cierto bloque de un archivo, un proceso de usuario (denominado proceso cliente) envía la solicitud a un proceso servidor, que entonces realiza el trabajo y entrega el resultado.
Lo único que realiza el nucleo es controlar la comunicación entre los clientes y los servidores. Al separar el sistema operativo en partes, cada una de ellas controla una parte del sistema, como el servicio a archivos, servicio a procesos, servicio a terminales o servicio a la memoria, cada parte es pequeña y controlable. Además, puesto que todos los servidores se ejecutan como procesos en modo usuario y no en modo nucleo, no tienen acceso directo al hardware. En consecuencia, si existe

un error en el servidor de archivos, este puede fallar pero no afectará a toda la máquina.
Otra de las ventajas del modelo cliente servidor es su capacidad de adaptación para su utilización en sistemas distribuidos. Si un cliente se comunica con un servidor mediante mensajes, el cliente no necesita saber si el mensaje se maneja en forma local, en su máquina, o si se envía por medio de una red a un servidor en una máquina remota. En lo que respecta al cliente, lo mismo ocurre en ambos casos: se envió la solicitud del servicio y el servicio fue recibido.
La idea anterior de un nucleo que solo controla el transporte de mensajes de clientes a servidores y viceversa no es completamente real. Algunas de las funciones del sistema operativo (como el cargado de comandos en los registros físicos de los dispositivos de E/S) son difíciles, si no es que imposibles de realizar a partir de programas de usuario. [4]



Modelo cliente servidor en un sistema distribuido

Sistemas operativos actuales

Hoy en día existen múltiples sistemas operativos, cada uno orientado a un nicho específico del mercado, todos tienen características especiales que los distinguen, aquí describiremos brevemente los sistemas operativos utilizados en las plataformas PC. Estas PC's debido a su amplia difusión y bajo precio, es posible encontrarlas en casi todos los nichos de la sociedad; ahora, las PC's las encontramos desde las casas de bolsa, hasta los laboratorios de investigación de las universidades, pasando por estudios arquitectónicos, agencias de publicidad, constructoras, hospitales, centros de servicio automotriz, tiendas de autoservicio, etc.
Comenzaremos con la historia de los sistemas operativos de PC.

La historia de los sistemas operativos para PC es una historia de 2 compañías dominantes, la primera, IBM, conocida como el gigante azul de las computadoras, compañía que toma la decisión de que Microsoft, que es una empresa pequeña que diseña programas, es la empresa ideal para la creación del sistema operativo para su máquina de escritorio (conocida posteriormente como IBM PC).
El MS-DOS (nombrado así por la compañía que lo creó [Microsoft]) apareció con el IBM PC en el año de 1981. Fue el primer sistema operativo de 16 bits utilizado en microcomputadoras. En 1988 se habían vendido más de 10 millones de copias,

pero los números ahora son incalculables El crecimiento y el éxito del MS-DOS no puede separarse de la computadora con la cual nació La IBM PC
La primera microcomputadora verdadera fue la ALTAIR, desarrollada en 1975 por MITS (Micro Instrumentation Telemetry System) de Albuquerque, en Nuevo México. Esta computadora se basaba en el microprocesador 8080 de Intel, esta máquina contaba con 256 bytes de memoria, el aparato se vendía por $ 397 dólares sin teclado, pantalla, disco o cinta Esta computadora estaba dirigida principalmente a personas cuyo pasatiempo era la computación Bill Gates y Paul Allen, fundadores de Microsoft, desarrollaron una versión de Basic para la computadora Altair. Allen se fue a trabajar a MITS y pidió a Gates que desarrollara una versión de BASIC que trabajara con discos flexibles. Esta tarea obligó a la creación de un sistema de administración de archivos basado en el empleo de una tabla de asignación de archivos (File Allocation Table la actual FAT) residente en memoria, y no en disco, que finalmente se convirtió en el pilar del actual MS-DOS.
En esa época, CP/M dominaba lo que era el mercado de sistemas operativos para computadoras de 8 bits, pero ya estaban surgiendo las microcomputadoras de 16 bits; aunque los chips de 16 bits eran más rápidos y más potentes, su precio era extremadamente alto y no era seguro que el mercado de la computación personal pudiera soportar tales costos. Microsoft obtuvo de Bell Laboratories la licencia para un Unix y, en 1980, lo convirtió en XENIX, el cual era un sistema operativo multiusuario y multitarea para microcomputadoras de 16 bits

En 1979, Tim Paterson, que trabajaba en la compañía Seattle Computer Products, necesitaba software para probar un producto basado en el 8086 de Intel. Microsoft contaba con Basic para 8086, y Digital Research estaba trabajando sobre CP/M para 8086, pero Digital Research tardó más de lo necesario en liberar sus sistema. Por esto, Paterson desarrolló su sistema operativo para el 8086, imitando el CP/M, debido a que CP/M se había convertido en el sistema operativo estándar para microcomputadoras basadas en el 8080 y Z80. La primera versión de sus sistema operativo ocupaba solo 6K de memoria.
Mientras tanto, IBM en sus oficinas de Boca Ratón Florida se encontraba desarrollando un microcomputador basado en el 8086. Durante algunos años, IBM había contemplado la evolución de la industria de la computación personal, la cual consideraba como industria juguetera, IBM pensaba que nunca podría opacar el reinado de las macrocomputadoras. Pero al final, IBM decidió participar en la industria de microcomputadoras. Su problema fundamental era el de diseñar una máquina y su software con rapidez, además de que deberían ser de buena calidad para poder competir en el mercado. IBM sabía que su ciclo normal de desarrollo de productos era demasiado lento; no tenía tiempo para desarrollar todos sus componentes, de tal manera que decidió utilizar componentes de hardware comprados, existentes en el mercado de los 80's, por lo cual necesitaba ofrecer programas ya existentes en vez de desarrollar programas nuevos.
IBM solicitó a Microsoft que desarrollara un Basic en ROM para su computadora, que al inicio resultaría de 8 bits. Microsoft sugirió utilizar una máquina de 16 bits,

por lo cual IBM estudió los microprocesadores 8086 y 8088 de Intel IBM decidió finalmente utilizar el 8088 (para su primer diseño), debido a que se podía comunicar con periféricos de 8 bits en lugar de los más caros de 16 bits.
IBM requería que estuvieran disponibles varios lenguajes de programación, así como también un sistema operativo El BASIC tenía la ventaja de ser un producto independiente. Bill Gates sugirió a IBM que utilizara CP/M 86 IBM celebró varias reuniones con Digital Research, pero no fueron provechosas, su sistema operativo iba muy atrasado. Microsoft decidió escribir su propio sistema operativo y adquirió el producto desarrollado por Tim Paterson en Seattle Computer Products. Microsoft sentía la necesidad de desarrollar el sistema operativo, así como también los lenguajes de programación, para poder entregar a IBM los productos a tiempo. Paterson realizó algunas modificaciones mientras se encontraba en Seattle Computer Products. Paterson no tenía la menor sospecha de que IBM era el que requería con tanta urgencia su sistema operativo. Paterson se enteró de ésto cuando ingreso a Microsoft. Los requerimientos de seguridad y discreción de IBM eran muy rígidos, por lo cual el desarrollo del sistema operativo se realizó en una habitación asegurada de 2X3 metros, sin ventanas y con cerradura de seguridad en la puerta. MS-DOS se ejecutó en el prototipo de IBM PC en Agosto de 1981. Poco tiempo después MS-DOS fue convertido en el sistema operativo más importante de la industria de la computación personal. [3]
Hasta hoy en día el sistema operativo más utilizado por la computadoras personales es el MS-DOS de Microsoft, pero debido a los requerimientos cada vez más demandantes, los procesadores más potentes, los equipos más complejos, el MS-DOS no es una buena opción como sistema operativo Los sistemas actuales trabajan con procesadores y memorias de 32 bits, mientras que el MS-DOS para mantener la compatibilidad con las versiones anteriores continua con la política de 16 bits.
Microsoft se dió cuenta de que requería un sistema operativo mejor, pero necesitaba tiempo para su liberación, por lo cual liberó Windows. Este producto no fue muy utilizado hasta que la versión 3.0 llegó al mercado Era un sistema estable, pero hacia falta algo más, velocidad y aplicaciones. La versión 3.1 llegó al mercado ofreciendo mejores opciones, manejo de memoria adecuado para aplicaciones gráficas, capacidad multitarea cooperativa, adicionalmente los proveedores de software se interesaron mucho en este producto liberando gran cantidad de aplicaciones para el mismo. Este sistema nuevo solo tenía un defecto corría bajo el viejo MS-DOS.
Posteriormente Microsoft liberó una versión más de Windows conocida como Windows para trabajo en grupo, esta versión permite compartir recursos de red como discos e impresoras entre usuarios de una LAN.
Más adelante Microsoft liberó un sistema operativo nuevo muy poderoso, pero requería un procesador muy grande y al menos 16 MB de memoria para trabajar con algunas aplicaciones. Este sistema fue adoptado en las corporaciones para manejar la red y sus servicios, ya que además posee interfaces para manejo de red. Era un sistema operativo muy bueno, pero era demasiado grande para el

usuario final, por lo cual Microsoft comenzó con un proyecto conocido como Chicago el cual era un sistema intermedio entre Windows 3.1 y Windows NT. Microsoft tuvo algunos problemas para la liberación de este producto, pero finalmente lo liberó el 26 de Agosto de 1995 y su nombre final fué Windows 95.

Presente, pasado y futuro de los sistemas operativos de Microsoft

Año	Usuarios corporativos	Usuarios caseros
1994	Windows 3.1 Windows for Workgroups	Windows 3.1
1995	Windows NT 3.51 Compatibilidad PowerPC Controles y cajas de diálogo comunes con Windows 95	Windows 95 Nueva interfaz de usuario Multitareas por prioridades Multihilos Protección de memoria Microsoft Network API Win32
1996	Windows NT 3.52 Microsoft Network Interfaz de Windows 95	Nashville Liberación de mantenimiento de Windows 95
1997-1998	Windows NT 4.0, NT 5.0, Arquitectura orientada a objetos Sistema de archivos orientado a objetos OLE de red Servicios de directorio Capacidad ATM integrada Autentificación Kerberos Otras mejoras adicionales	Memphis Último mantenimiento mayor a Windows 95
1999-2000	Windows NT Sistema operativo unificado para todos los usuarios de escritorio	Windows NT Sistema operativo unificado para todos los usuarios de escritorio

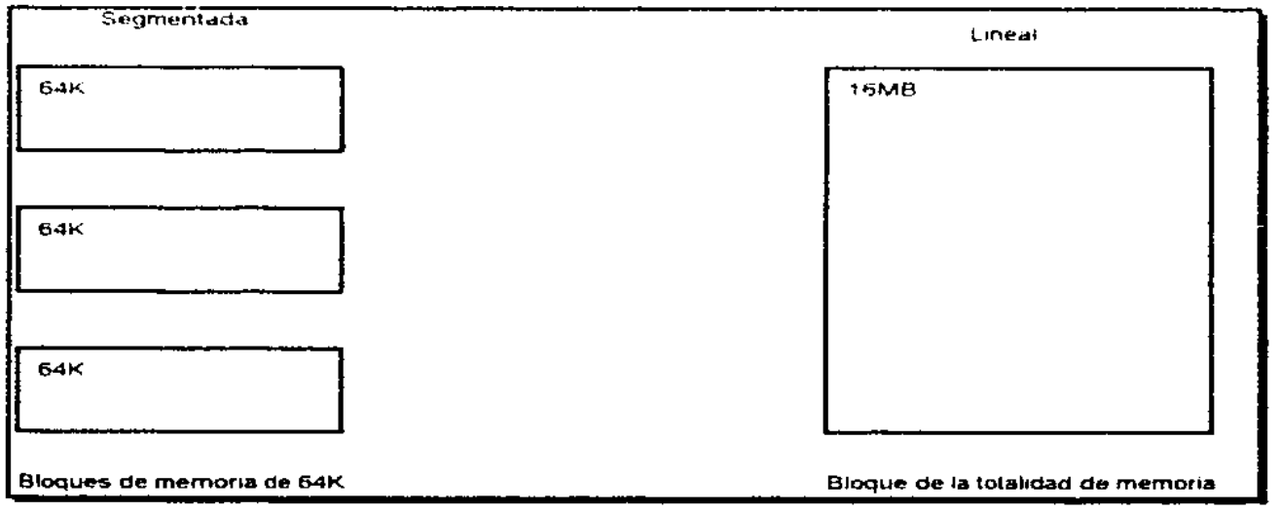
Tecnología de sistemas operativos

Hoy en día es muy común escuchar los términos 32 bits, multihilos, multitareas, cooperativo, por prioridades. Estos son términos utilizados en la definición de sistemas operativos de la nueva generación.

Manejo de memoria

Los sistemas operativos modernos utilizan técnicas de manejo de memoria plana, es decir, manejan toda la memoria como un solo espacio el cual es accedido

mediante una dirección, al contrario de los sistemas operativos anteriores o de las primeras generaciones los cuales multiplexan o brincan entre bloques de memoria separados, administrándolos con técnicas muy complicadas que consumían muchos ciclos de estas CPU's, estas limitaciones son típicas de la arquitectura inicial X86, por esto, los sistemas operativos modernos utilizan un modelo en el cual se puede manejar gran cantidad de memoria de manera lineal, es decir, no tiene porque manejar múltiples bloques de memoria de tamaño fijo (segmentos) ni registros de control (registros de segmento) para poder aprovechar la totalidad de la memoria.



Con el manejo de memoria lineal se obtienen los siguientes beneficios:

- Mejor rendimiento de aplicaciones
- Programas de mayor tamaño en memoria
- Mayor cantidad de programas en memoria
- Facilidad de administración de memoria (sistema operativo)
- Se evitan brincos entre diferentes segmentos de memoria para un programa

En resumen un modelo de memoria lineal beneficia directamente las aplicaciones debido a las características antes mencionadas.

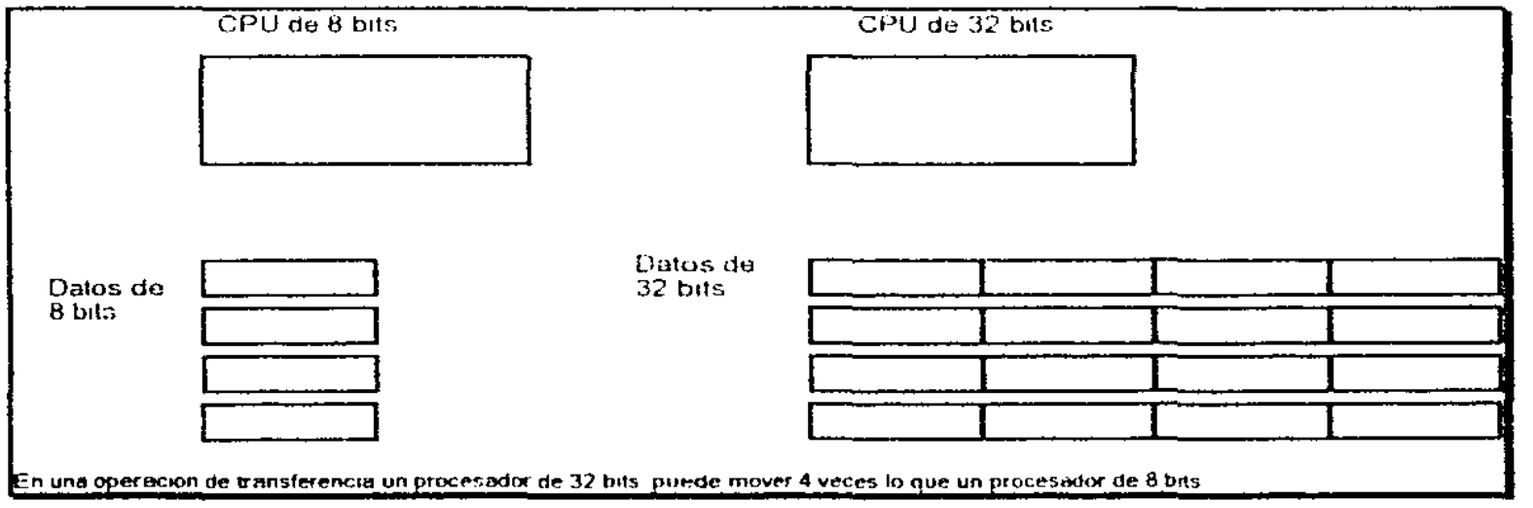
32, 64 o más bits

Otra parte importante del avance de los sistemas operativos es el manejo de 32 bits o más para sus operaciones. En pocas palabras, en una operación de movimiento de datos si se tiene la capacidad de 32 bits, el sistema puede mover o modificar 32 bits, al contrario que en los sistemas anteriores, donde básicamente se movían 8 o 16 bits por cada operación del procesador. Adicionalmente a esto, el tener un manejo de memoria de 32 bits permite trabajar con una cantidad de

memoria mayor; en un sistema tradicional de PC se podía direccionar hasta 20 bits, lo cual daba como resultado 1 MB, con 32 bits se pueden manejar memorias muy grandes, del orden de Gigabytes (4 Gigabytes) Por esto, un sistema de 32 bits tiene mayores capacidades para el manejo de información ya sea para transferencias de datos, para operaciones matemáticas con datos de mayor tamaño o para un mayor direccionamiento de memoria

Cabe mencionar que desde hace algunos años la compañía MIPS and Technologies fabricó uno de los primeros microprocesadores de 64 bits. actualmente cuenta con varias series de microprocesadores de 64 bits disponibles, de entre las cuales se encuentra el poderoso R4000, el R4400. R5000, R8000 y su más reciente creación el R10000, estos microprocesadores son utilizados en las computadoras Silicon Graphics las cuales a su vez son utilizadas en los centros de investigación y en las compañías cinematográficas. con estos equipos se han podido realizar películas tan impresionantes como Jurassic Park, La Máscara o Terminator II

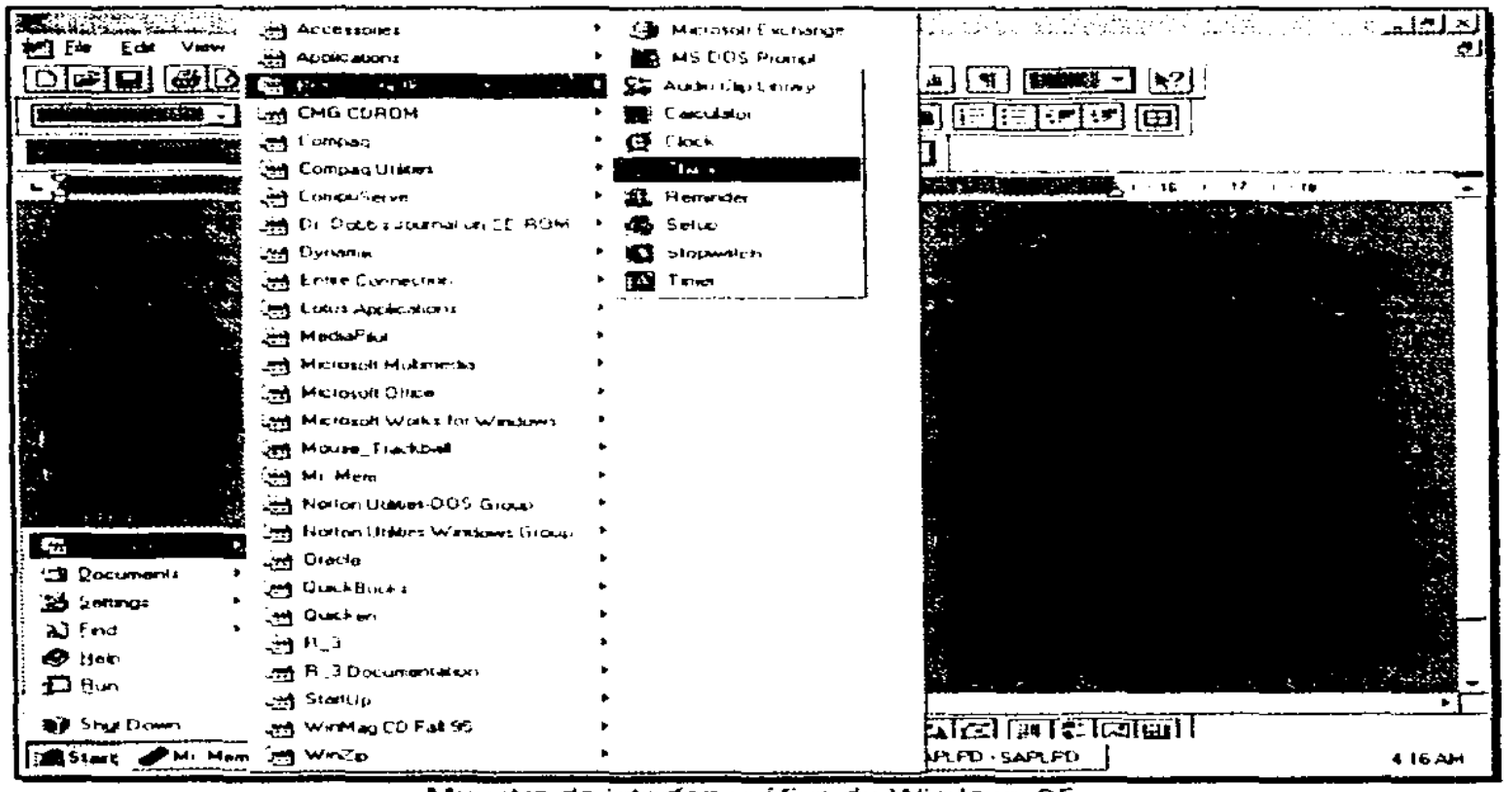
Los fabricantes actuales de microprocesadores se encuentran evaluando utilizar 128 bits en los microprocesadores tanto para transferencia de datos como para manejo de los registros internos (operaciones con registros), así es que, ¡en poco tiempo veremos supermicrocomputadoras con procesadores de 128 bits!



Interfaz gráfica

Uno de los principales avances para poner la computadora al alcance de mayor cantidad de personas es la parte de la interfaz. Una interfaz gráfica facilita el uso de la computadora haciendo analogías con objetos cotidianos de oficina Con esto proporciona una manera ágil y divertida para comunicarse con la computadora Aún las personas más novatas aprenden rápidamente a utilizar y aprovechar la computadora con este tipo de interfaz

Los sistemas operativos modernos proveen una interfaz gráfica clara y sencilla que permite a los usuarios con la utilización del ratón el aprovechamiento total del equipo sin necesidad de aprender comandos complejos con múltiples parámetros



Muestra de interfaz gráfica de Windows 95.

```
Microsoft(R) DOS Version 7.00
  (C)Copyright Microsoft Corp 1981-1996.

C:\>
```

Muestra de interfaz de texto de DOS 7.00.

Capacidad multitareas

Los sistemas operativos modernos deben permitir la ejecución de varias aplicaciones a la vez, esto con la finalidad de aprovechar la totalidad de la CPU no dejando ciclos ociosos en donde la CPU no este ocupada realizando alguna actividad. Esto permite que se realicen múltiples operaciones dentro de una sola computadora obteniendo un mayor beneficio, con esto se puede emitir una impresión y se pueden realizar otras actividades mientras el sistema envia el documento a la impresora.
La ventaja real de la capacidad de multitareas es aprovechar la máquina cuando se deja en ejecución una tarea que toma largo tiempo en completarse, como por ejemplo: una consulta en la base de datos local no bloquea la máquina si no que permite que se realicen otras actividades mientras el sistema concluye la consulta.

Protección de programas

En una ambiente en el cual se ejecutan múltiples aplicaciones es cada vez más importante el concepto de protección de memoria. Si el sistema no protege las aplicaciones unas de otras el sistema puede volverse inestable en cualquier momento y puede destruir el trabajo de las demás aplicaciones.
Por esto, los sistemas operativos modernos deben de proporcionar mecanismos que protejan las aplicaciones unas de otras verificando cada acceso a la memoria validando siempre si el proceso que hace la solicitud tiene la autorización para realizar la operación. Con esta facilidad se puede emitir un fax, recalcular un estado de resultados, emitir un reporte financiero y hacer una consulta de la base de datos, todo al mismo tiempo, haciendo de esta manera más productivo el trabajo sin interferir los procesos unos con otros.

Disponibilidad de aplicaciones

La gente utiliza computadoras porque éstas facilitan el trabajo, pero si no existieran aplicaciones que facilitaran su trabajo las computadoras perderían su utilidad. No importa que sea el sistema operativo perfecto, que tenga un desempeño excelente si no tiene aplicaciones que se ejecuten en el mismo. Por esto, la importancia de que el sistema operativo elegido tenga aplicaciones de utilidad para el negocio es una parte importante para su evaluación.
Las características antes mencionadas son las más deseables para los sistemas operativos, en la actualidad existen varios sistemas que ofrecen las características antes mencionadas, estos sistemas operativos son:

- OS/2 Warp de IBM
- Windows 95 de Microsoft
- Windows NT de Microsoft

- Linux
- NextStep
- MAC OS
- Unix

Describiremos cada sistema a continuación indicando al final de cada descripción las ventajas y desventajas de cada uno de ellos.

OS/2

Este sistema operativo fue inicialmente desarrollado en conjunto por IBM y Microsoft. Pero, Microsoft desarrollaba al mismo tiempo un proyecto paralelo (el cual produjo Windows), motivo que provocó el divorcio comercial de IBM y Microsoft quedando IBM como propietario del OS/2. IBM hasta ahora ha lanzado 4 liberaciones importantes del producto, la primera fue la versión 2.0 la cual tenía problemas de manejadores de dispositivos (existía una carencia de manejadores para múltiples proveedores de hardware). Esta versión fue la primera de OS/2 en proporcionar una ventana de WIN-OS2 permitiendo ejecutar aplicaciones de Windows 3.0.
La siguiente liberación importante de IBM fue el IBM OS/2 for Windows (oficialmente OS/2 versión 2.11) la cual utilizaba una parte licenciada de código de Windows para correr las aplicaciones de Microsoft Windows. Esta liberación mejoró el desempeño general, pero el problema seguía siendo la falta de soporte para dispositivos no estándar.
Hace apenas pocos años IBM liberó su OS/2 Warp, el cual además de mejorar el rendimiento con respecto a las versiones anteriores proporcionaba un juego de aplicaciones completas para trabajar con él (Bonus Pack). Ahora OS/2 podía correr aplicaciones de Windows más rápidamente que las versiones anteriores de OS/2, tenía una instalación simplificada que permitía mayor velocidad en la misma. Pero el problema era el mismo, falta de soporte para dispositivos no estándar y por lo mismo la falta de aceptación por la comunidad usuaria. Poco tiempo después de la liberación de este producto IBM hizo un nuevo anuncio, la siguiente versión de OS/2, la cual tendría integrado el manejo red, esta nueva versión hoy es conocida como OS/2 Warp Connect, que viene siendo la respuesta de IBM hacia Windows 95 de Microsoft y Windows NT.

OS/2 Warp posee las siguientes características:

- Modelo de memoria plano de 32 bits
- Capacidad de multitarea por prioridades o multitareas apropiable
- Protección de aplicaciones
- Sistema de archivo de alto rendimiento (HPFS)
- Capacidad para ejecutarse en máquinas con 4 MB de memoria
- Aplicaciones integradas (Bonus Pack)
- Multimedia integrada

- Compatibilidad con aplicaciones de Windows, DOS y OS/2 (en modelos de máquina virtual para aplicaciones de DOS y Windows 3.1 de 16 bits)
- Estabilidad de aplicaciones
- Excelente despliegue gráfico de 32 bits
- Interfaz Orientada a objetos
- Comunicaciones integradas
- LaunchPad; caracteristica que permite ejecutar las aplicaciones con un pulso del ratón

Desventajas

- Desempeño pobre para aplicaciones de Windows
- Pocas aplicaciones nativas disponibles (comparado con Windows 95)
- Interfaz de usuario (Workplace Shell) difícil de aprender para personas que conocen Windows
- Carencia de manejadores de dispositivos
- Configuración e instalación compleja para el usuario novato

Windows NT

Este sistema operativo nació debido a la necesidad de protección de memoria, seguridad, conectividad y compatibilidad de aplicaciones Windows.
Es un sistema operativo robusto, se encuentra certificado por tener una seguridad tipo C2 (solicitada por la Secretaria de Defensa de los Estados Unidos de America). Por lo cual pretende ser la solución en cuanto a ambientes cliente/servidor.
Este producto generalmente es utilizado para administración de redes, servidores, etc. debido a las facilidades antes mencionadas de red y seguridad.
Windows NT ha tenido algunas liberaciones, pero la más importante es la 4.00, en la cual se cambió la interfaz de usuario por una orientada a objetos encaminándose así como estándar de la industria.

Sus caracteristicas principales son:

- Modelo de memoria plano de 32 bits
- Modelo de distribución de capas en el nivel de rutinas básicas, permitiendo de esta manera la reducción de errores
- Capacidad de multitarea por prioridades o multitareas apropiable
- Protección de aplicaciones
- Sistema de archivo de alto rendimiento (NTFS)
- Compatibilidad con aplicaciones de Windows, DOS y Windows NT (32 bits)
- Estabilidad de aplicaciones
- Excelente despliegue gráfico de 32 bits
- Seguridad tipo C2
- Administración de comunicaciones integrada

- Facilidad de uso (igual que Windows 95)
- Estrategia futura de Microsoft
- Sistema operativo multiplataforma, soportado en Intel X86, Alpha AXP, MIPS R4X00, PowerPc
- Soporte a multiprocesadores (SMP)

Desventaja

- Requiere 12 MB en su configuración mínima

Windows 95

Microsoft siempre ha prometido la solución total en sistemas operativos, pero hasta ahora, no lo ha podido cumplir, Windows 95 nació como respuesta a las múltiples solicitudes de usuarios que requerían un sistema operativo poderoso, no solo una interfaz gráfica fácil de utilizar. Microsoft emitió múltiples versiones de prueba (Alfa y Beta) del sistema operativo, las cuales fueron evolucionando desde un cambio de interfaz hasta un manejo gráfico interno mejorado y un sistema de archivos instalable.
Windows 95 es la evolución remarcada de la arquitectura de Windows. Su nueva interfaz es fácil de modificar y de navegar por ella. Ahora se pueden ejecutar aplicaciones de 16 y 32 bits más rápido y mejor que nunca.
La estabilidad de Windows 95 es estupenda comparándola con la de Windows 3.1, ofrece un soporte sin igual en el área de comunicaciones, comparado con DOS/Windows. Pero aún así los usuarios desean más, Windows 95 no ha alcanzado los niveles de estabilidad y desempeño del OpenVMS o Unix.

Sus características principales son:

- Modelo de memoria plano de 32 bits
- Capacidad de multitarea por prioridades
- Protección de aplicaciones (solo 32 bits)
- Sistema de archivo instalable (IFS)
- Compatibilidad con aplicaciones de Windows, DOS y Windows 95 (32 bits)
- Estabilidad de aplicaciones
- Excelente despliegue gráfico de 32 bits
- Administración de comunicaciones integrada
- Mejor desempeño de aplicaciones
- Múltiples aplicaciones disponibles (1997)

Desventajas

- Interfaz modificada, dificultad de uso para los nuevos usuarios
- Mayores requerimientos de hardware (6 a 8 MB de memoria, disco duro de mayor tamaño)

Linux

Linux es la respuesta de los usuarios a la demanda de un Unix económico para PC. Linux fue desarrollado independientemente por Linus Torvalds de la Universidad de Helsinki en Finlandia. El Nucleo de Linux no usa código de AT&T o de algún otro sistema propietario fuente y mucho del software ha sido desarrollado por el proyecto GNU de la fundación del Software Gratis (FSF).
Originalmente Linux inició como un hobby inspirado en el Minix; Linus Torvalds deseaba un mejor Minix (sistema operativo diseñado por Andrew S. Tanembaum para la enseñanza de la materia de sistemas operativos) con más características de Unix, ya que él deseaba probar un sistema robusto con caracteristicas más poderosas, por lo cual fue extendiendo su proyecto hasta completar Linux. Se puede decir que Linux es parte de Internet y muchas de las rutinas básicas han sido creadas a partir de esta red de redes, habiendo participado gente de todo el mundo, incluyendo investigadores mexicanos, también se le ha llamado a **LINUX** el Sistema de los **hackers** (hacker es aquella persona que se dedica a desenmarañar la complejidad del software).

Caracteristicas

Multitareas, multiusuarios, es decir, muchos usuarios pueden ser "registrados" en la misma máquina a la vez, corriendo múltiples programas simultáneamente.
En el nivel fuente incluye características de otros sistemas operativos IEEE POSIX.1, System V, y BSD. Linux fue desarrollado con la idea de la transportabilidad en mente, de hecho existen aplicaciones para UNIX disponibles en Internet que pueden ser compiladas en cualquier hardware que corra LINUX, además se puede obtener el código fuente de Linux incluyendo núcleo, manejadores de dispositivos, bibliotecas, programas de usuarios y herramientas de desarrollo, todo esto gratuitamente.
Linux incluye control de trabajos POSIX (usados en intérpretes de comandos como csh y bash), seudoterminales (dispositivos PTY), soporte para teclados nacionales o personalizados usando "manejadores de teclado cargados dinámicamente". Linux también soporta "consolas virtuales" las cuales permiten conmutar entre múltiples sesiones accesadas desde la consola del sistema en modo texto (los usuarios del programa "screen" tienen este concepto familarizado).
El núcleo es capaz de emular instrucciones 387-FPU propias, así un sistema sin un coprocesador puede correr programas que requieren instrucciones matemáticas de punto flotante.

En linux existen varios tipos de sistemas de archivos de los cuales los más importantes son:

- **Ext2fs**.- Desarrollado especificamente para Linux.

- **Minix-1 y Xenix**.- UNIX compatible.
- **MS-DOS**.- Directos desde la tabla de partición
- **ISO 9660 CD-ROM**.- De formato estándar CD-ROM.

Linux está provisto de una implementación completa del protocolo para uso de redes TCP/IP, incluye manejadores de dispositivos para las tarjetas de red más populares.
El núcleo de Linux fue desarrollado para usar todas las características especiales del modo protegido del 80386, 80486, Pentium, Pentium II y Pentium Pro de Intel, especialmente para los paradigmas de la administración de memoria basados en descriptores de modo protegido y muchas otras características avanzadas de estos procesadores. Este modo de programación como se sabe se utiliza para diseño de sistemas multitareas como UNIX (ó Multics).
El núcleo de Linux soporta "carga ejecutable de demanda de paginación", esto es, que solo aquellos segmentos de un programa que son utilizados son leídos y cargados a la memoria desde el disco. En Linux si varios procedimientos de un programa están ejecutándose a la vez, estos compartirán páginas en memoria física, reduciendo el uso de memoria global.
Así mismo para incrementar la memoria disponible, Linux implementa paginación en disco de hasta 256 MB, esta área reservada del disco es conocida como "espacio de intercambio". Cuando el sistema requiere más memoria física, este intercambia páginas de memoria inactivas a disco permitiendo correr aplicaciones más grandes y soportar más usuarios a la vez. No obstante el intercambio no es sustituto de la memoria física, ya que el proceso de intercambiar páginas de memoria en disco es mucho más lento debido al acceso a manejadores y a la unidad física.
El núcleo también implanta un conjunto de memoria unificado para los programas del usuario y disco cache. De este modo, toda la memoria libre es usada como cache, y el cache se reduce cuando los programas más grandes están corriendo.
Los ejecutables utilizan librerías o bibliotecas compartidas ligadas dinámicamente, la ventaja de esto es que se reduce el tamaño del archivo ejecutable.
También el usuario puede ligar las librerías estáticamente para mantener el ejecutable completo sin compartir librerías o bien puede ligar en tiempo de ejecución permitiendo que el programador reemplace los módulos de las librerías con sus propias rutinas.

Nextstep

Este es uno de los rivales más fuertes dentro del mercado de sistemas operativos, su arquitectura es muy elegante y esta totalmente orientado a objetos, pensado para desarrollo de aplicaciones de misión crítica, su estructura es cliente/servidor, su creador es Steve Jobs, creador técnico del sistema Macintosh de Apple. Nextstep cuenta con una gran cantidad de herramientas para conexión a redes, fue en este sistema operativo donde se desarrolló el formato HTML (Hyper Text Markup Language), muy utilizado en las páginas de World Wide Web de Internet.

Otra de las características que implanta este sistema es la de dispositivos CD-ROM y discos ópticos de lectura y escritura, teniendo la capacidad de almacenar enormes cantidades de información El aspecto visual de este sistema operativo está muy bien cuidado, aunque resulta complejo utilizar video de alta resolución a menos que se tenga el dispositivo adecuado, otra de las desventajas de este sistema operativo es el manejo de impresión, por utilizar el lenguaje Postscript no es compatible con algunas impresoras comerciales
Es importante destacar que a principios de 1997, Apple compró Next, con lo cual las caracteristicas innovadoras de este sistema operativo se verán reflejadas en la nueva generación de sistemas de Apple, el nombre clave de este nuevo sistema operativo es Rapsodia

Sistema operativo de Apple

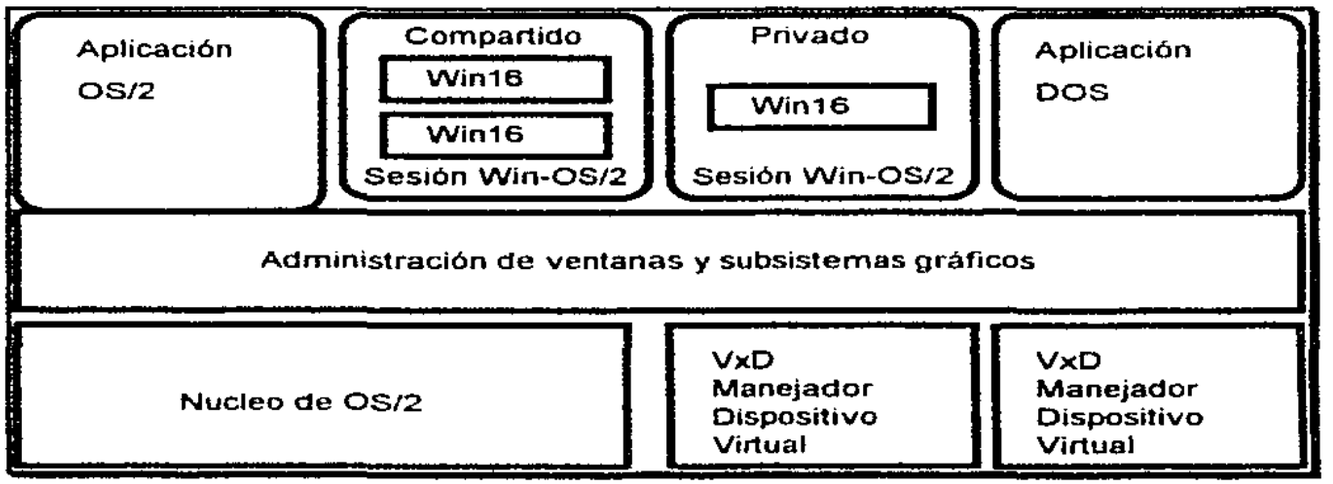
En 1984 se introduce la primera computadora personal Macintosh con una interfaz gráfica para usuarios novatos. Originalmente la Mac venía con dos paquetes incluidos: un procesador de texto y un editor de dibujo, pero esto no bastó para suplir las crecientes necesidades de las empresas No se disponía ni de discos duros ni de impresoras de buena calidad, en el inicio este sistema contaba con solo 128 KB en su memoria principal. La arquitectura cerrada del sistema dificultaba que empresas de hardware y de software diseñaran nuevos aditamentos. Hoy en día Mac OS 7.6 tiene otro enfoque de mercadotecnia, recientemente se ha separado al sistema operativo de la arquitectura del hardware (muy dependientes hasta antes de esto) para ser comercializado por terceros. Entre las mejoras que presenta este sistema se incorpora Quick Draw un entorno orientado a objetos, Power Talk para correo electrónico y nuevas opciones multimedia. Por último se tiene el Apple Script un estándar de automatización de tareas para administración de impresión, instalación de hardware y software nuevo e intercambio de información con otros sistemas operativos como MS-DOS, incluye además conexiones para Internet tipo SLIP/PPP y protocolo TCP/IP.
Una de las desventajas de este sistema operativo es con respecto al manejo de la multitarea pues es del tipo no apropiable, quiere decir que impide que se trabaje en otro proceso a la vez cuando la computadora está ocupada en un primer proceso.
El corazón de este sistema operativo es el procesador 680X0 (actualmente el PowerPC), la memoria RAM, la memoria ROM y diversos circuitos integrados que conjugan la relación software y hardware adecuadamente. El procesador es en esencia de 16 bits, pero en varios aspectos se puede considerar uno de 32 bits ya que cuenta con 16 registros de 32 bits (registros de datos, de direcciones y de apuntadores de pila), su reloj trabaja con pulsos de hasta 80 millones de ciclos por segundo y este procesador puede procesar hasta 20 millones de instrucciones por segundo. Incluye un versátil adaptador que controla dispositivos como teclado, mouse, señales internas, interfaces de disco, sonido, además

puede generar diferentes tipos de interrupciones como actividades del teclado, reloj, borrado vertical de video, etc.
A pesar de ser un sistema de alto rendimiento, en el inicio se quedó inmerso por permanecer hermético (hoy la filosofía es distinta), esto le costó que su difusión fuera escasa y por consiguiente no tuviera apoyo para crear hardware propio con lo cual el público en general optó por otros sistemas operativos.

Estructura interna de los sistemas operativos más populares de PC

OS/2

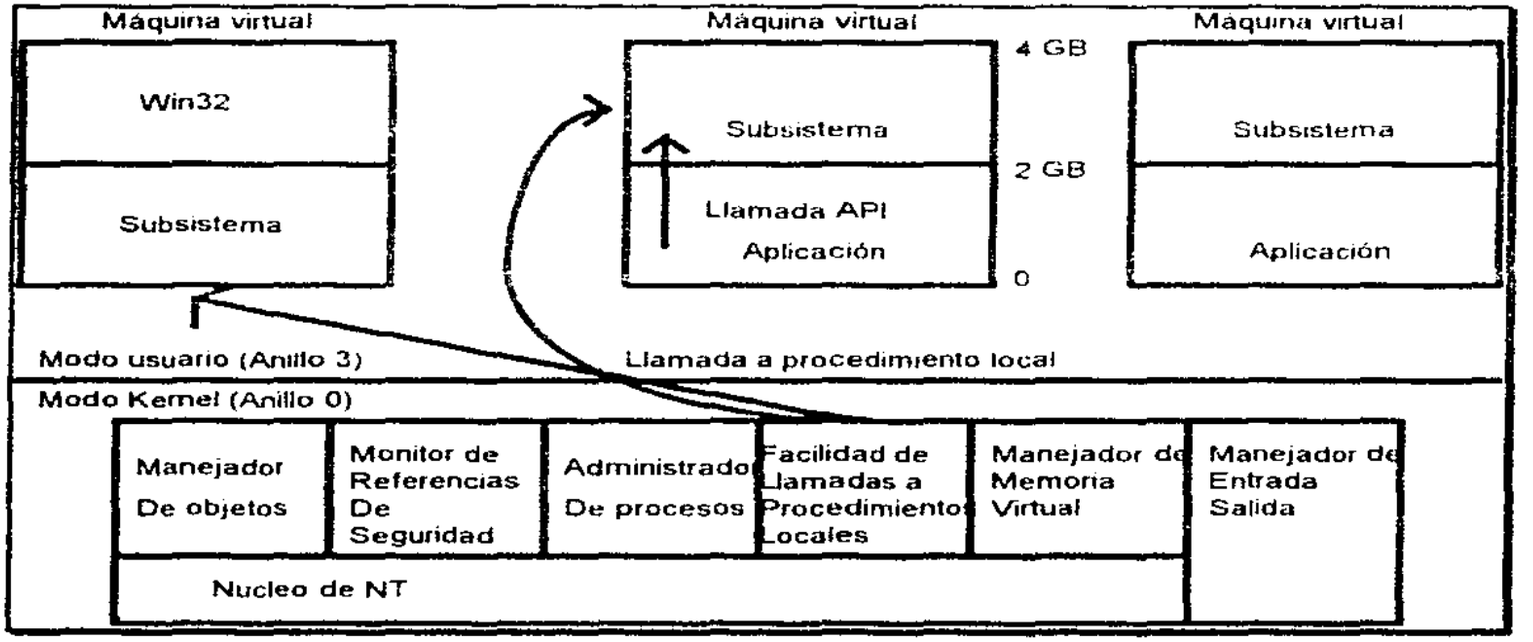
En el OS/2 Warp Connect las aplicaciones de 16 y 32 bits corren en VM's (máquinas virtuales) separadas, de tal manera que es muy difícil que una perjudique a la otra. También se puede seleccionar la ejecución de varias aplicaciones Win16 en el mismo espacio de memoria o en diferentes espacios de memoria, obteniendo el mismo resultado (protección contra aplicaciones con mal comportamiento).
Una de las razones por las cuales OS/2 no soporta el API de Win32 es por el rompimiento de su alianza con Microsoft, rompimiento que establece que las licencias de utilización de código de Windows caducan en 1993.



Windows NT

Windows NT utiliza una arquitectura interna cliente/servidor única entre los sistemas operativos modernos. Las llamadas de hardware y otras manipulaciones de bajo nivel son virtualmente imposibles con esta configuración. Por esto las aplicaciones deben en su lugar invocar los servicios mediante llamadas de procedimientos locales. Mientras que esto es extremadamente seguro, esta configuración reduce un poco el rendimiento, particularmente en juegos, los cuales generalmente tratar de manipular directamente el hardware. Al igual que

en OS/2, la aplicaciones Win32 y Win16 corren en espacios de memoria separados, y usted puede seleccionar cuales aplicaciones Win16 correrán en los mismos espacios de memoria o en diferentes espacios de memoria.

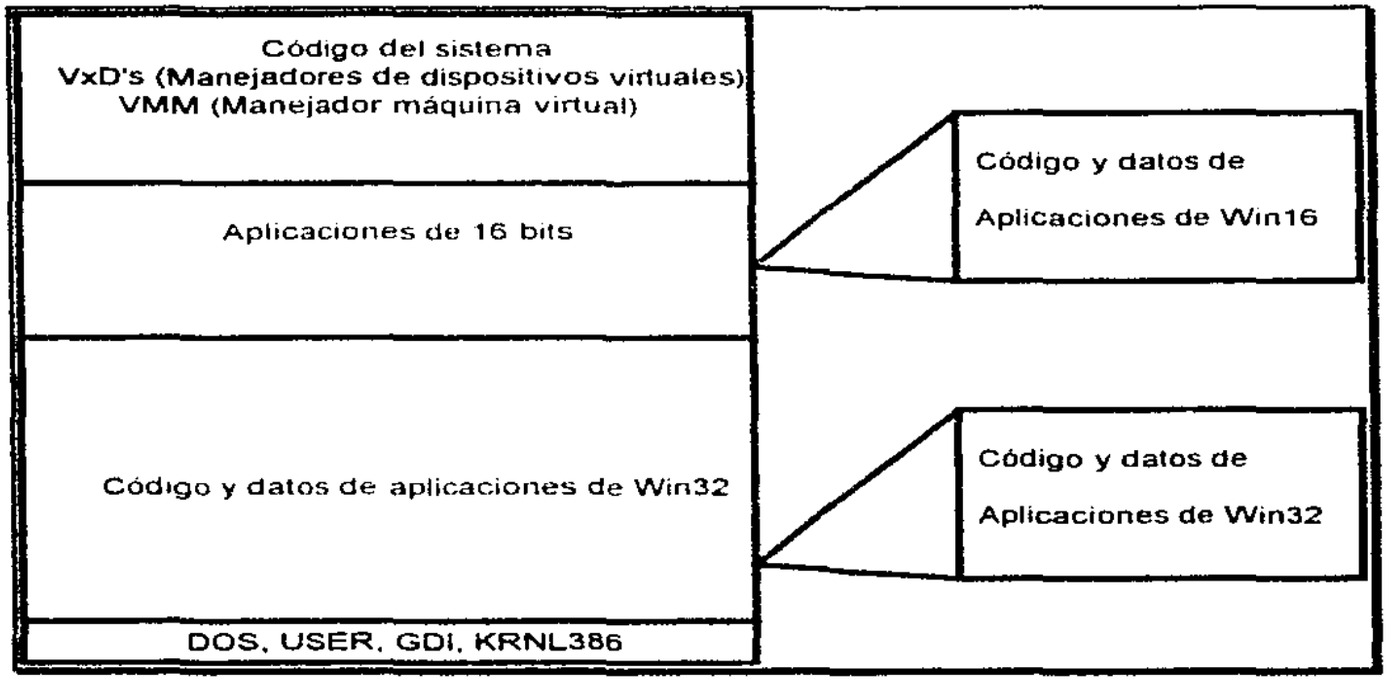


Windows 95

En Windows 95 se modificó poco la estructura de Windows 3.1. Mucho del código del sistema operativo todavía corre en el mismo espacio de las aplicaciones, pero las aplicaciones de Win32 corren dentro de espacios de direcciones privados, lo cual disminuye la probabilidad de que una aplicación mal comportada detenga la totalidad del sistema, esto solo es cierto para las nuevas aplicaciones de 32 bits. Cabe señalar que el DOS continua en la parte baja de los 4 MB de memoria, incluyendo las regiones USER, GDI y KRNL386.

Es importante destacar que Microsoft pretende soportar un solo sistema operativo en el futuro (Windows NT), por lo cual Windows 95 es un sistema operativo de transición el cual permite gran parte de las llamadas al sistema de Win32 de Windows NT, pero solo fue liberado para preparar a los usuarios para el sistema operativo unificado que Microsoft liberára en los próximos años.

Como se puede observar claramente en la gráfica solo las aplicaciones nuevas de Win32 corren protegidas en espacios de direcciones separados.



Otros sistemas operativos modernos son:

- Plan 9 (desarrollado por el equipo original de Unix)
- Spring (Sun Microsystems)
- Solaris (clon de Unix)
- Deskview
- Coherent (clon de Unix)
- 386BSD (clon de Unix)
- FreeBSD (clon de Unix)
- QNX (clon de Unix, tiempo real)
- Copland (Apple)
- Oberon (desarrollado a partir del lenguaje del mismo nombre)
- Inferno
- RTOS
- MicroOS
- Unixware (Novell)

Tipos de sistemas operativos

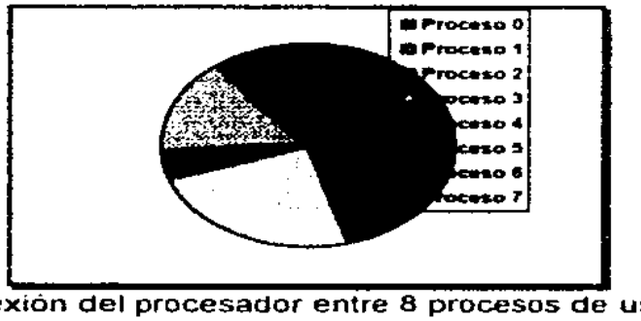
Comenzaremos nuestra descripción de los tipos de sistemas operativos con los sistemas más simples, posteriormente indicaremos cuales son las demandas de los usuarios con respecto a los sistemas operativos e indicaremos los tipos modernos de sistemas operativos.

Sistemas operativos para procesamiento por lotes

El procesamiento por lotes precisa generalmente que el programa, los datos y las órdenes adecuadas al sistema sean remitidos todos juntos en forma de un trabajo único. Los sistema operativos por lotes permiten poca o ninguna interacción entre los usuarios y los programas en ejecución. El procesamiento por lotes tiene un mayor potencial de utilización de recursos que el procesamiento en serie simple en sistemas informáticos que dan servicio a múltiples usuarios [2]. Por lo mismo este tipo de procesamiento no es muy conveniente para el desarrollo y depuración de programas, ya que normalmente no se conocen los resultados de un proceso o programa hasta que éste ha terminado de ejecutarse. En la actualidad el sistema de procesamiento por lotes más utilizado es el MVS (Multiple Virtual Storage) de IBM, el cual se utiliza en las grandes corporaciones para procesar cantidades de información enormes

Sistemas operativos de multiprogramación

La ejecución concurrente de los programas tiene un significativo potencial para mejorar la productividad del sistema y la utilización de los recursos con respecto al procesamiento serie o procesamiento por lotes. Este potencial se consigue, o al menos se explota mejor, con una clase de sistemas operativos que multiplexan los recursos de un sistema informático entre una multitud de programas en ejecución [2].
Esta multiplexión puede ser representada de la siguiente manera:



Multiplexión del procesador entre 8 procesos de usuario

Donde a cada uno de los procesos se les asigna procesador dependiendo de la prioridad que estos tengan, así, los procesos con mayor prioridad tendrán una mayor cantidad de procesador asignado. La multiplexión de procesador no es la única que se realiza en este tipo de sistemas, ya que por ejemplo, el disco también es un recurso que se puede multiplexar entre varios procesos.
En el área de PC los representantes más importantes de este tipo de sistemas operativos son:

- OS/2
- Windows NT

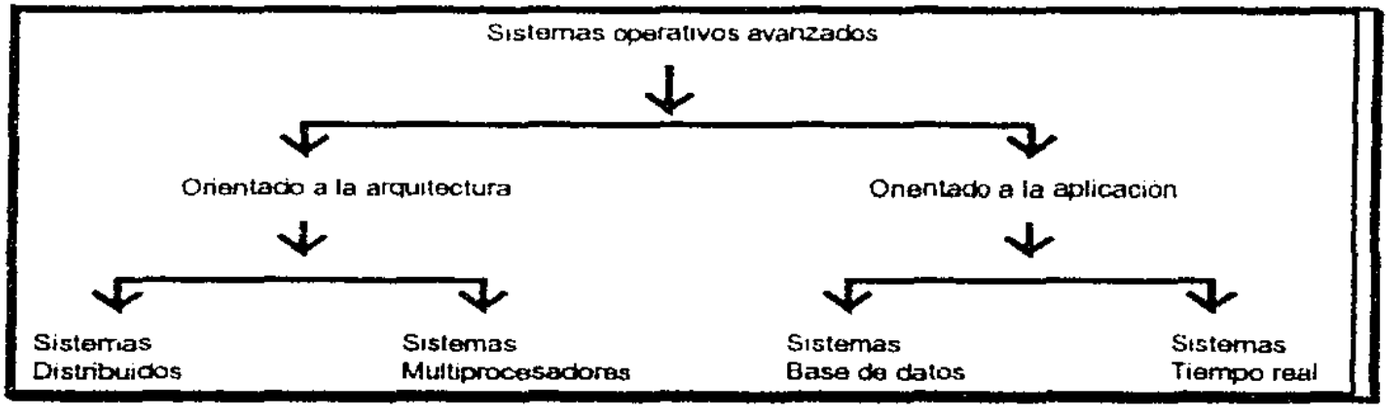
- Windows 95

Sistemas operativos de tiempo compartido

Los sistemas operativos de tiempo compartido son representantes de los sistemas multiprogramados, pero en este caso multiusuario.
Uno de los objetivos esenciales de los sistemas multiusuario en general, y de los de tiempo compartido en particular, es el de proporcionar un buen tiempo de respuesta de terminal a cada uno de los usuarios. Dando de esta manera la ilusión a cada usuario de que dispone de una máquina para si mismo. los sistemas de tiempo compartido intentan con frecuencia lograr una compartición equitativa de los recursos comunes [2].
Uno de los representantes actuales más importantes de este tipo de sistema operativo es el Unix en todos sus dialectos, el cual se ejecuta desde una PC hasta un gran computador central

Tipos de sistemas operativos avanzados

La siguiente figura muestra una clasificación de los sistemas operativos avanzados:



El impetu por los sistemas operativos avanzados ha venido en dos direcciones. Primero, éste ha venido de los avances en la arquitectura de sistemas de multicomputadoras y está ahora orientado por una amplia variedad de arquitecturas de alta velocidad. Estas arquitecturas ofrecen un gran potencial para acelerar los desarrollos pero también presentan un reto substancial a los diseñadores de sistemas operativos.
Una segunda clase de sistema operativos avanzados es la orientada a la aplicación. Existen varias aplicaciones importantes que requieren sistemas operativos especiales para su soporte. Los sistemas operativos de propósito general son demasiado amplios en su naturaleza por lo cual son ineficientes y fallan para proveer un soporte adecuado a tales aplicaciones. Dos aplicaciones

específicas llamadas sistemas de base de datos y sistemas de tiempo real, han recibido atención considerable desde el pasado.

Sistemas operativos distribuidos

Los sistemas operativos distribuidos son sistemas operativos para una red de trabajo de computadoras autónomas conectadas por un canal de comunicaciones. Un sistema operativo distribuido controla y administra los recursos de hardware y software de los sistemas distribuidos los cuales son vistos por los usuarios como un poderoso sistema monolítico Cuando un programa es ejecutado en un sistema distribuido, el usuario nunca se entera en que computadora se ejecuta su programa ni donde se encuentran los recursos utilizados.

Los elementos básicos del diseño de un sistem operativo distribuido son los mismos que en los sistemas operativos tradicionales, sincronización de procesos, programación de eventos, candados (lazos de recursos), sistemas de archivo, comunicación entre procesos, administración de áreas temporales y memoria, recuperación a fallas, etc. Sin embargo, los sistemas operativos distribuidos también poseen debilidades, tales como la carencia de memoria compartida, un reloj global para el sistema, además, debido a que se utilizan canales de comunicación los retrasos en la llegada de la información son impredecibles, con lo cual el diseño de sistemas operativos distribuidos es más difícil que en los sistemas tradicionales.

Sistemas operativos multiprocesadores

Un sistema típico multiprocesador consiste de un conjunto de procesadores que comparten un conjunto de bloques de memoria física sobre una red interconectada. Asi, un sistema operativo multiprocesador es un sistema fuertemente acoplado donde los procesos comparten un espacio de direcciones. Un sistema operativo multiprocesador controla y administra los recursos de hardware y software de tal manera que los usuarios ven al sistema como un poderoso sistema uniprocesador; los usuarios nunca se enteran de la presencia de múltiples procesadores ni de sus sistema de interconexión.

Los elementos básicos del diseño de un sistema operativo multiprocesador son los mismos que en los sistemas operativos tradicionales. Sin embargo, elementos como la sincronización de procesos, programación de tareas, administración de memoria, protección y seguridad, se vuelven más complejos debido a que la memoria principal es compartida por múltiples procesadores.

Sistemas operativos de bases de datos

Los sistemas de bases de datos colocan requerimientos especiales en los sistemas operativos. Estos requerimientos tienen sus raices en el ambiente

específico que los sistemas de bases de datos soportan. Un sistema de base de datos debe soportar: el concepto de transacción; operaciones a almacenar, recuperar, y manipular un volumen muy grande de datos eficazmente; primitivas de concurrencia y control, y posibilidad de recuperación después de fallas. Para almacenar información temporal y recuperar información desde el almacenaje secundario, éste debe tener un esquema efectivo de administración del espacio temporal (buffer).

Sistemas operativos de tiempo real

Los sistemas operativos de tiempo real también colocan requerimientos especiales en los sistemas operativos, los cuales tienen sus raíces en las aplicaciones específicas que los sistemas operativos de tiempo real soportan. Una característica distinta de los sistemas operativos de tiempo real es que los trabajos tienen un plazo específico para completarse. Un trabajo debe completarse antes de que llegue su plazo específico para ser de utilidad (en sistemas operativos de tiempo real *suaves*) o para anunciar un desastre (en sistemas operativos de tiempo real *duros*).

El elemento más importante en el diseño de sistemas operativos de tiempo real es la programación de trabajos de tal manera que el máximo número de trabajos cumpla con su plazo fijado.

Metodología utilizada para el diseño de un sistema operativo

Iniciaremos el diseño utilizando los modelos conceptuales de sistemas operativos más populares ahora existentes. Los sistemas operativos actuales han sido desarrollados generalmente por grupos de ingenieros, los cuales ponen en práctica todos los conceptos y teoría de la programación estructurada para su diseño e implantación. Aquí, utilizaremos la programación estructurada para ofrecer un sistema operativo que sea claro, efectivo, extensible y modular; todo esto es con el fin de facilitar su estudio por estudiantes y personas independientes.

Las divisiones o módulos principales del sistema operativo son:

- Administrador de procesador
- Administrador de memoria
- Administrador de dispositivos
- Administrador de archivos

Cada uno de estos módulos se irá dividiendo a su vez en pequeños componentes para facilitar su programación y comprensión. Se recalcarán los aspectos más problemáticos e interesantes de su diseño e implantación para mejorar la comprensión del sistema.

Finalmente se integrará cada uno de los módulos en su componente mayor (administrador) para conjuntarlos en el sistema operativo funcional.
Es importante mencionar que algunos de los componentes (administradores) requieren dividirse en mayor candidad de módulos debido a su complejidad y extensión.

Metodología de Implantación y Diseño

Cuando se tiene en mente la creación de un sistema operativo, se debe de considerar el siguiente perfil:
La computación en general tiene tres líneas de investigación: la teoría, la abstracción y el diseño
La teoría favorece el aprendizaje académico, sin embargo por si sola representa una carga que no es fácil entender; es necesario que estas tres líneas se retroalimenten generando el enfoque académico.

Principio de abstracción o de ocultación de la información.- Este principio es usado para implementar niveles o capas en aquellos sistemas complejos, haciendo que los detalles de diseño de los niveles más bajos queden ocultos en los niveles superiores, con el fin de presentar un sistema más sencillo y de uso más práctico. Cada nivel superior se construye bajo la base de un nivel inferior. por lo tanto es necesario documentar técnicamente cada nivel para el desarrollo de niveles posteriores y para futuras referencias.

Principio de Modularidad en el diseño.- Además de ocultar la complejidad del sistema, el diseñador debe de seguir un enfoque modular, es decir, debe fraccionar los problemas en pequeñas entidades que realizan una sola función. Con esto se logra aislar las tareas y en determinado momento cuando se quiera modificar o corregir los errores, solo se afectará el módulo que nos interesa.
Teniendo en mente estos dos principios es importante que el diseñador conozca como va a interactuar el sistema operativo con respecto a los procesos, y como será el ciclo de vida de éstos, si serán procesos aislados o compartidos. El sistema operativo deberá ser capaz de separar, controlar y supervisar todas estas características de los procesos, además asignará el tiempo y los recursos hardware y software que el proceso solicite, llevando una tabla de tiempos y recursos, manejando interrupciones para solicitudes especiales y sincronizando la comunicación entre aquellos procesos que comparten información. La clave de un buen diseño radica en la estructura de datos de los procesos y como se conducirán ante las señales de mensajes y de control.
El control de bloques de datos que leen y escriben tanto de entrada como de salida, requiere el uso de instrucciones de control hardware que involucran dispositivos internos y externos, estos se deben de acoplar en una línea de espera para evitar que se solapen e influyan en los procesos existentes.

Uno de los recursos hardware más importantes del sistema es la memoria; una buena administración de la misma consiste en definir que tipo de esquema se utilizará, se debe tener en cuenta si es contigua o no, si las particiones serán estáticas o dinámicas, el conteo exacto de memoria que se tiene, todo esto junto con los algoritmos de planificación adecuados transforman al sistema en uno más eficiente.

Otro aspecto que se debe de considerar en el diseño es el del almacenamiento unido al administrador de archivos. Evitar que el usuario se enrede en diferentes formatos de datos y hacer corresponder de manera abstracta las peticiones de acceso de direcciones de memoria lógicas con direcciones físicas de los dispositivos de almacenamiento secundario hace que el sistema cumpla con el principio de abstracción. Si además de eso se planea darle seguridad al propio administrador de archivos, evitando errores que puedan resultar en la pérdida, robo o actualizaciones no autorizadas de la información, además de resistir fallas generales en el sistema, se tendrá más solidez en la versión final.
La imagen de una interfaz simple al usuario representa un alto porcentaje de éxito o fracaso de un nuevo sistema, si esta interfaz no es manipulable, entonces el usuario emitirá un juicio negativo que provocará que el sistema no sea aceptado en su totalidad.
Todo esto nos habla de planificación; una buena planificación es la de jerarquizar. La jerarquización de los sistemas operativos ha dado buenos resultados tanto en sistemas pequeños como grandes, no es de un sistema operativo en particular sino más bien es un buen inicio para un proyecto de diseño.

Nivel 1	Núcleo
Nivel 2	Entrada y salida
Nivel 3	Administración de Memoria
Nivel 4	Administración de Archivos
Nivel 5	Intérprete de comandos del usuario

Principios adicionales de Diseño

Los puntos arriba mencionados entran en el aspecto técnico de desarrollo, sin embargo factores menos teóricos también deben de ser considerados en el diseño:

Consistencia.- es importante que los objetos que integran el sistema se sustenten unos con otros, que no haya conflictos y pérdida de tiempo al generar una parte que no va de acuerdo con la estructura general del proyecto; si esto sucede entonces es necesario hacer una revisión de donde puede existir el error.

Retroalimentación.- Como ya se comentó, la retroalimentación de las tres líneas de estudio se debe de aplicar en el diseño de los diferentes esquemas que el proyecto presenta. Esto se logra al ver que los niveles de abstracción inferiores

responden a las expectativas de los niveles superiores y estos a su vez concuerdan con la forma teórica que se planeó desde el comienzo Cuando hay demasiadas discrepancias, la retroalimentación es nula y el sistema operativo carece de un buen diseño

Simplicidad.- Aunque suena un poco absurdo la simplicidad en el diseño es uno de los principios que muchas veces no se aplican llevando consigo muchos problemas. La simplicidad no es ser simplista, sino llevar funcionalidad a los términos aceptables; cuando se desperdician recursos para implantar una función que puede ser mejorada con menos no se esta aplicando este principio

Balance en el equipo de diseño.- Otro aspecto de consideración es el del tipo de personas que se involucran para diseñar. Raramente una persona tendrá todos los atributos para una disciplina dada, por lo que se requiere conocer en que área puede cierto individuo ser productivo y más aún si es capaz de trabajar y comunicar con el resto del equipo los avances e ideas que surgen en el camino.

Ciclo de diseño.- Por último el ciclo de diseño independientemente de sus fases es una actitud o comportamiento para controlar los errores que pueden comprometer al sistema: diseño, prototipo, pruebas y repetición, cada vez que se presenta una modificación es importante cubrir este ciclo para probar la confiabilidad de ese cambio, y siempre que se genere una mejora es también conveniente aplicar este principio.

◆ CAPITULO II.- ADMINISTRACION DEL PROCESADOR

Una de las partes esenciales de los sistemas operativos es la administración del procesador; en el administrador del procesador se distribuye la cantidad de CPU que se le asignará a cada uno de los programas en ejecución dentro de una computadora (incluyendo el tiempo de procesador asignado al sistema operativo), aquí mostraremos algunas técnicas para la administración efectiva del procesador en un sistema operativo.

Concepto de proceso

Para poder administrar un procesador primeramente debemos conocer que es un proceso, aquí mostramos algunas de las definiciones clásicas de proceso.

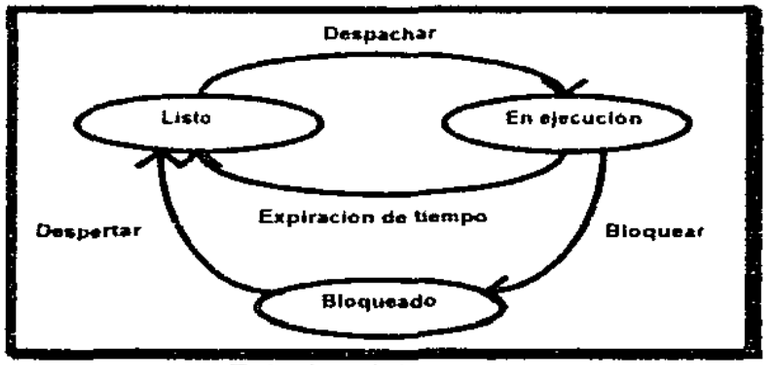
- "Un proceso es básicamente un programa en ejecución. Consta del programa ejecutable, sus datos, pila, contador de programa y otros registros, además de toda la información necesaria para ejecutar el proceso." [4]
- Un proceso es una secuencia temporal de ejecuciones de instrucciones que corresponden en su totalidad a la ejecución de un programa secuencial. Un proceso es una entidad dinámica que nace cuando es cargado a la memoria física, y muere al finalizar la ejecución del programa." [6]
- "Un proceso es un programa cuya ejecución ha iniciado pero todavía no ha sido completada." [7]
- "En esencia, un proceso o tarea es una instancia de un programa en ejecución. Es la unidad más pequeña de trabajo individualmente planificable por un sistema operativo." [2]
- "Un programa en ejecución, una actividad asíncrona, 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 del proceso en el sistema operativo, la entidad a la que se asignan procesadores, la unidad despachable." [3]

De todas estas definiciones la más adecuada y más repetida es la de programa en ejecución, aparentemente estas definiciones no son totalmente diferentes, debido a que hablan prácticamente de lo mismo, el proceso. Aquí indicamos una definición obtenida de todas las anteriores, que es la que nos parece correcta:

Existe un proceso cuando un programa es cargado a la memoria y cada una de sus instrucciones es ejecutada en orden secuencial (se ejecutan de manera secuencial en los procesadores tradicionales).

Cuando existen varios procesos en ejecución dentro de una máquina, dichos procesos se encuentran en diferentes estados dependiendo de la carga del

procesador y de las actividades en si que realice el proceso durante su ejecución, dichos estados son los siguientes:



Estados del proceso

- **En ejecución,** el procesador está ejecutando las instrucciones del proceso correspondiente.
- **Listo,** El proceso se encuentra listo para se ejecutado, pero el procesador no está disponible para la ejecución de este proceso.
- **Bloqueado,** El proceso se encuentra en espera de que ocurra algún evento. Por ejemplo, espera a que una operación de entrada y salida sea completada.

Un proceso puede encontrarse en cualquiera de los estados anteriormente mencionados, pero ¿Porqué nos interesa saber en que estado se encuentra un proceso?

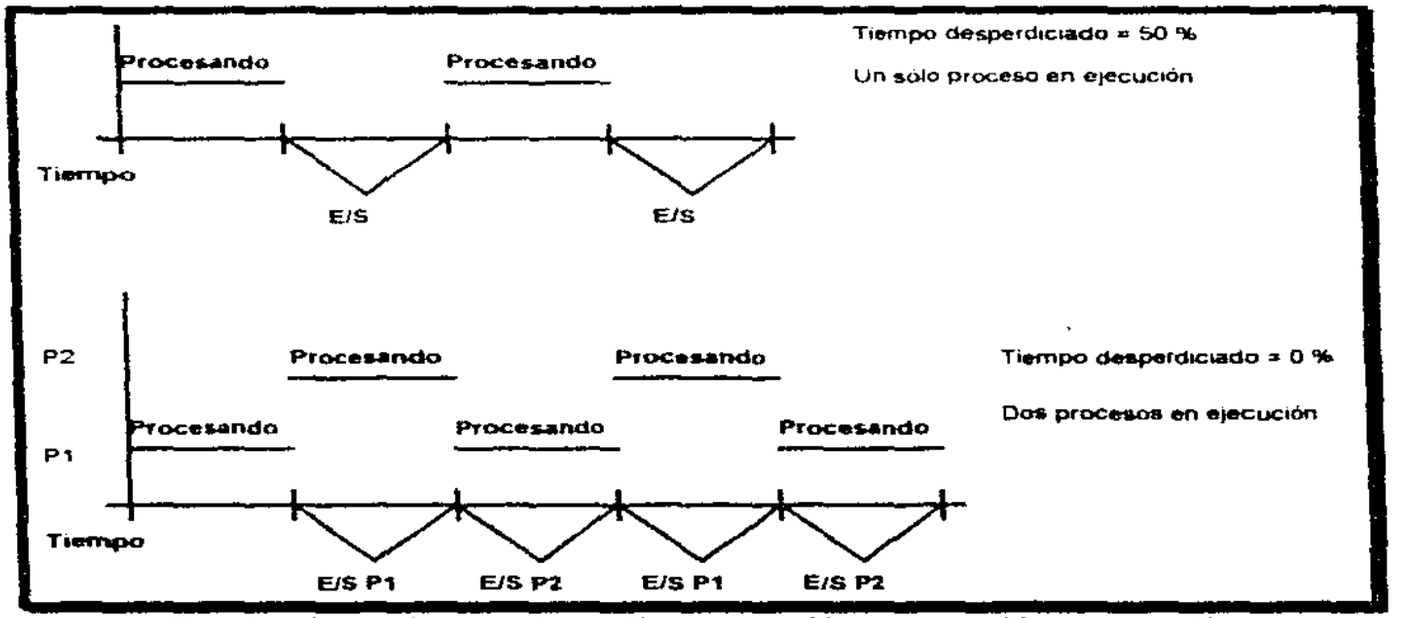
Nos interesa conocer el estado de un proceso porque de esa manera podemos (debiera decir, debemos) aprovechar los recursos del procesador de la mejor manera posible; en los sistemas antiguos donde solo se procesaba un trabajo a la vez la planificación del procesador no era una tarea compleja, esto era debido principalmente a que el procesador era asignado en su totalidad a un único proceso que se encontraba en memoria, cuando ocurría una operación de entrada y salida, el procesador simple y sencillamente se quedaba ocioso, es decir, sin realizar operación alguna mientras era completada la entrada y salida (el MS-DOS es un sistema de este tipo).

Los ingenieros de sistemas operativos de la época de los 40's y 50's diseñaron varias maneras para tratar de mejorar este "defecto" de los primeros sistemas operativos, una de las maneras más efectivas actualmente utilizadas es la multiprogramación.

Por ejemplo, si un proceso realiza cálculos durante un segundo y ejecuta operaciones de entrada y salida durante el siguiente segundo, continuando de esta manera alternadamente hasta concluir después de dos minutos, habrá desperdiciado (no habrá utilizado) el procesador por un minuto, debido a que

exactamente la mitad de su tiempo de ejecución realizó operaciones de entrada y salida.

Este comportamiento no es deseable por el gran desperdicio de procesador que provoca, por lo cual incluiremos en nuestro ejemplo otro proceso que presenta el mismo comportamiento, pero en este caso comenzará su ejecución cuando el primer proceso se encuentre realizando su primera operación de entrada y salida.



Muestra de un sistema sin multiprogramación y con multiprogramación

Multiprogramación

Los diseñadores de sistemas operativos pensaron que si un proceso pasaba la mayor parte del tiempo realizando operaciones de entrada y salida, ese tiempo que el procesador se encontraba ocioso podía ser aprovechado por otro u otros procesos, por lo cual consideraron que debían ejecutarse más de un proceso dentro de una computadora.

Multiprogramación es cuando dentro de una computadora con un solo procesador se pueden ejecutar varios programas simultáneamente, es decir, se multiplexan los procesos para que a cada uno se le asigne un determinado tiempo de procesador, con lo cual se aprovecha mejor el mismo, realizando algún tipo de actividad (de un proceso determinado) mientras los otros se encuentran en espera de operaciones de entrada y salida u otro tipo de eventos.

La multiprogramación es importante (y compleja) ya que como se dijo anteriormente mejora la utilización del procesador de manera drástica, aquí se

mostrarán algunas técnicas para poder aprovechar esta característica de los sistemas operativos modernos.

Algunos sistemas operativos modernos que utilizan multiprogramación son:

- OS/2
- Windows NT
- Windows 95
- Linux
- Plan 9 (descendiente directo de Unix, creado por los autores originales de Unix)
- NextStep
- AIX (clon de Unix para estaciones de trabajo RS/6000)

Estructuras de control de los procesos

Para poder ejecutar varios procesos dentro de una misma computadora se debe llevar el control estricto de cada una de las actividades que realizan los mismos, por lo cual se debe establecer una estructura de datos que conserve el estado de todas las actividades de un proceso para que cuando sea cambiado el control del procesador a otro proceso no se pierda el avance del proceso al que se le expropia el procesador. La estructura que contiene los datos necesarios para restablecer la ejecución de un proceso cuando se le vuelve a asignar el procesador se conoce como estructura de control del proceso o bloque de control del proceso.

Este bloque de control del proceso contiene fundamentalmente la siguiente información:

1. El estado del proceso
2. Identificador del proceso (número del proceso)
3. Apuntador hacia el proceso creador (es decir, al proceso que lo ejecutó)
4. Apuntador hacia los procesos derivados (es decir, procesos ejecutados por él)
5. Prioridad del proceso
6. Registros del procesador (incluyendo el contador del programa)
7. Límites de memoria
8. Lista de recursos asignados
9. Lista de archivos abiertos
10. Información contable

A continuación se detallan cada uno de ellos:

1.- El estado del proceso puede ser, de acuerdo a lo definido:

- En ejecución

- Listo
- Bloqueado

2.- El identificador del proceso es un número entero que le indica al sistema operativo cual es el proceso en cuestión, es decir, es el nombre del proceso cuando el proceso está en ejecución.

3.- Dentro de un ambiente donde se permite la ejecución de múltiples procesos existen mecanismos especiales para la creación de los mismos, en general, un proceso es creado por otro proceso, al creador se le denomina padre, y al proceso creado se le denomina hijo, por lo cual es importante conservar la liga entre un proceso y su proceso padre.

4.- Lo mismo que aplica a los padres aplica a los hijos, esta liga se conserva para mantener un tipo de supervisión sobre los hijos.

5.- Dentro de un sistema pueden existir niveles donde cada nivel tiene asignada una cantidad determinada de recursos, estos niveles son formados por las prioridades.

6.- Se deben conservar los registros del procesador para que cuando se le devuelva el control al proceso este continúe con su ejecución exactamente en el lugar donde quedó anteriormente.

7.- Para poder restablecer su ejecución también es necesario conocer que áreas de memoria tenía asignado el proceso antes de que se le expropiara el procesador, en la lista de límites de memoria generalmente se incluye también los niveles de autorización a las mencionadas áreas.

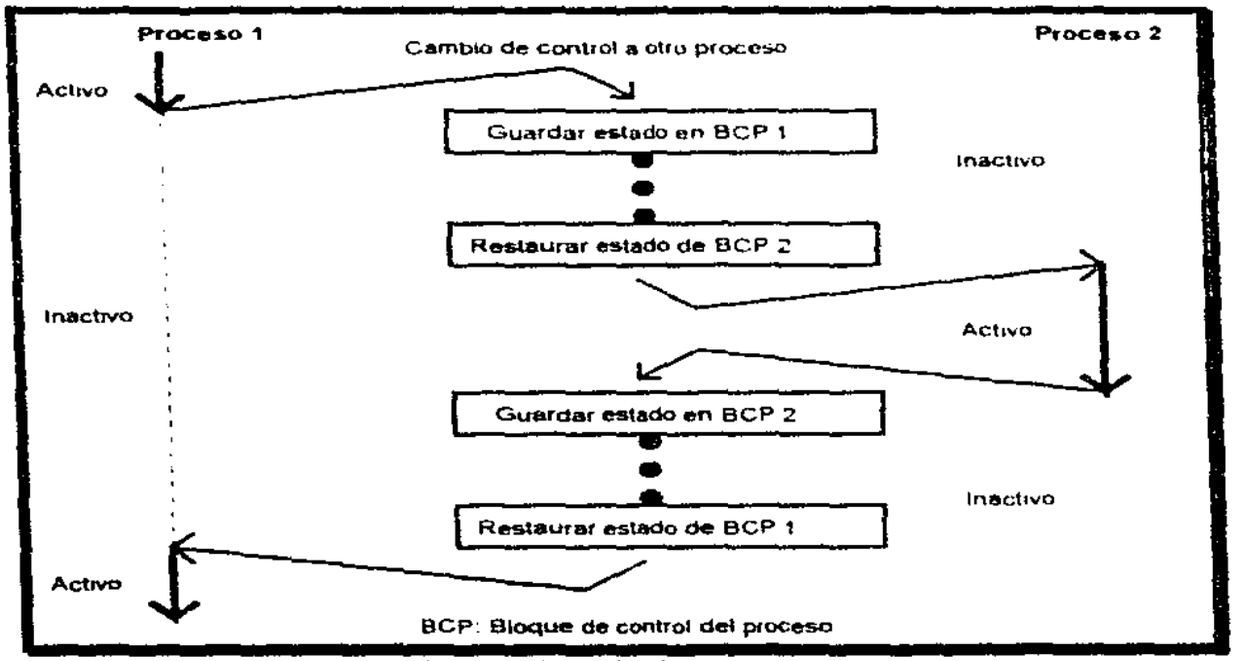
8.- Se debe proporcionar la lista de recursos asignados para poder continuar con la ejecución, ya que aunque se le expropie el procesador no se deben expropiar los recursos dedicados tales como unidades de respaldo, u otro tipo de recursos que solo pueden ser utilizados por un proceso a la vez.

9.- Se conserva una lista de archivos abiertos para conservar todas las referencias de actualización o de lectura de los mismos.

10.- En los sistemas donde se realiza algún tipo de cobro o se lleva un control estricto de los recursos asignados se incluye un campo para la información contable, esta información incluye la cantidad de tiempo de procesador utilizada, cuenta utilizada, límites de tiempo, etc.

La siguiente gráfica muestra como a un proceso en ejecución (Proceso 1) se le expropia el procesador, inmediatamente después se salva toda la información necesaria para su ejecución en su correspondiente bloque de control (BCP 1),

posteriormente se restaura la información del bloque de control (BCP 2) de otro proceso (Proceso 2) e inmediatamente después continúa su ejecución éste proceso (Proceso 2), más adelante se le expropia el procesador al proceso en ejecución (Proceso 2), e inmediatamente se salva toda la información necesaria para su ejecución en el bloque de control correspondiente (BCP 2). posteriormente se restaura la información del bloque de control del proceso inicial (Proceso 1) e inmediatamente después se continúa con la ejecución de este proceso (Proceso 1).
A este intercambio de bloques de control de procesos que permiten la multiprogramación se le conoce como cambio de contexto. La operación de cambio de contexto es muy importante en los sistemas multiprogramados, por lo cual debe de hacerse de una manera efectiva y rápida, esto es debido a que si existe retraso en el cambio de contexto el desempeño general baja debido a que se consume un tiempo "mayor" en una tarea administrativa del sistema operativo. Por esto, muchos fabricantes de microprocesadores proporcionan un apoyo adicional mediante instrucciones especiales del procesador que facilitan (aceleran) el cambio de contexto.



Procesador cambiando de un proceso a otro

Operaciones sobre los procesos

Los sistemas que utilizan procesos para aprovechar los recursos de cómputo deben realizar algunas operaciones con los procesos, las operaciones realizadas con los procesos son:

- **Creación de procesos**

Para la creación de los procesos el sistema operativo define un nuevo bloque de control con los datos del nuevo proceso el cual luego es insertado en la cola de procesos listos para su posterior ejecución. Es importante señalar que para que un proceso sea creado otro proceso debe de solicitar su creación, un proceso (más bien un programa) no puede invocar su creación por si mismo.

- **Destrucción de procesos**

Cuando un proceso termina es necesario remover del sistema (reasignar como libres) los recursos atrapados por el proceso finalizado. Estos recursos son regresados al sistema operativo para su posterior utilización. Un proceso puede ser destruido por el sistema operativo, el mismo proceso u otro proceso que tenga autorización

- **Cancelación de procesos**

Cuando un proceso no se comporta de una manera adecuada es posible forzar su terminación, lo cual provocaría también una liberación de los recursos utilizados por el proceso.

- **Suspensión de procesos**

Cuando un proceso debe esperar por mucho tiempo algún evento (por ejemplo una operación de entrada y salida) es conveniente que no esté consumiendo recursos de procesador (es decir, que no pueda ser ejecutable, al menos durante un tiempo), por esto se le coloca en una lista especial donde se encontrará hasta que se cumpla la condición de espera.

- **Reanudación de procesos**

Cuando un proceso fue suspendido y el evento que espera sucede es necesaria su reanudación, por lo cual el proceso puede ser reanudado (colocado en la lista de procesos listos para ejecución) por otro proceso o por el mismo sistema operativo.

- **Lectura de características de los procesos (atributos)**

Cuando se quieren conocer las características de un proceso específico es necesario leerlas del bloque de control del proceso deseado. Por ejemplo, un atributo importante de los procesos es su prioridad, por lo que si queremos obtener este dato debemos poder consultarlo de su bloque de control.

- **Cambio de características de los procesos (atributos)**

Cuando ya se verificó alguna de las características del proceso y ésta desea cambiarse, es necesario asignar un nuevo atributo al proceso deseado.

- **Despacho de los procesos**

Cuando un proceso se encuentra en ejecución le son asignados los recursos que éste solicita, a esta asignación de recursos se le conoce como despacho.

Planificación

La planificación se fundamenta en un grupo de políticas y mecanismos integrados en el sistema operativo que administran el orden de ejecución de los procesos. El planificador es la parte del sistema operativo que selecciona el siguiente proceso que ingresa al sistema para su ejecución. El objetivo principal del planificador es optimar el rendimiento del sistema de acuerdo con los criterios seleccionados por los diseñadores del sistema.

Colas de planificación

Los procesos que ingresan a los sistemas multitareas son colocados en una cola de procesos constituida por los procesos que se encuentran en almacenamiento secundario, y que esperan asignación de almacenamiento primario. Los procesos que se encuentran ya en el almacenamiento primario (los cuales se encuentran listos para su ejecución), se mantienen en la cola de procesos listos. La cola de procesos listos es generalmente apuntada en dos sitios, uno en el primer proceso de la lista y el otro en el último proceso.

Existen colas adicionales que ayudan a la administración de dispositivos, por lo cual se les conoce como colas de dispositivos. En general cada uno de los dispositivos tiene su propia cola, en la cual se encuentran formados los procesos que solicitan la atención del dispositivo. Existen algunos dispositivos que no pueden compartirse, estos dispositivos no pueden tener más de un proceso esperando en su cola, esto es debido a que el recurso debe utilizarse hasta que termine el proceso, por ejemplo, la unidad de cinta y la impresora son dispositivos que no pueden compartirse cuando se han comenzado a utilizar.

Tipos de planificadores

Durante su ejecución un proceso recorre diversas colas de recursos (memoria, procesador, dispositivos), por lo cual el sistema operativo va seleccionando de acuerdo con sus políticas los procesos de cada una de las colas de recursos.

Debido a la naturaleza tan distinta de los recursos, los planificadores de los mismos generalmente son muy diferentes en cuanto a su frecuencia de ejecución. Por ejemplo, el planificador de la CPU selecciona alguno de los procesos listos para posteriormente asignarle el procesador, este planificador es de corto plazo, pues en un sistema multiprogramado la selección de procesos para su ejecución se realiza múltiples veces en periodos de tiempo muy cortos, en cambio, el planificador de procesos por lotes selecciona alguno de los procesos en disco para posteriormente cargarlo en memoria y después solicitar su ejecución, este planificador es de largo plazo, pues el planificador selecciona los procesos para

ejecutarlos cuando terminan los procesos anteriores del mismo tipo (procesos por lotes), lo cual no ocurre muy frecuentemente.

Algunos sistemas operativos, como aquellos que tienen tiempo compartido, presentan un nivel intermedio de planificación, aquí las peticiones del usuario son emitidas al planificador el cual "vacía" (envía al almacenamiento secundario a los usuarios menos activos) a algún usuario de la memoria para atender las peticiones de los usuarios con mayor número de solicitudes consecutivas.

El despachador

Este es el componente del sistema operativo que entrega realmente el control del procesador al proceso elegido por el planificador a corto plazo. La función del despachador implica:

- Cambiar el contexto del proceso
- Cambiar el procesador a modo usuario
- Reiniciar el proceso en el mismo lugar donde se le expropió el procesador

Debido a que las tareas del despachador son críticas este debe ejecutarlas lo más rápido posible para que no exista una baja en el rendimiento.

En la planificación se debe decidir a cuál proceso de la cola de procesos listos se le debe asignar el procesador, por lo cual, dependiendo de las necesidades de la instalación se deben utilizar distintas políticas para su asignación.
Los algoritmos de planificación tienen características distintas, por lo cual pueden favorecer a algunos tipos de procesos en lugar de a otros, así que la elección del algoritmo se deja a los diseñadores del sistema operativo.
Existen diversos criterios que son utilizados para la comparación de los algoritmos de planificación, estos criterios son básicamente los siguientes:

- **Utilización del procesador**. Se requiere que el procesador este ocupado la mayor cantidad de tiempo posible. La utilización varía entre el 0% y el 100% (ya que solamente existen dos estados para el procesador, procesando y desocupado), en un sistema real debe llegar al 40% (utilización promedio en un lapso dado de tiempo) para un sistema con poca carga y 90% para un sistema con gran carga. [1]

- **Productividad del procesador**. Otra medida es la cantidad de procesos que se ejecutan (más bien completan) por una unidad de tiempo. Esta medida se denomina productividad. Para procesos de larga duración la tasa puede ser de 1 o 2 procesos por hora, mientras que para transacciones breves, la tasa debe ser de 10 a 100 transacciones por segundo. [1]

- **Tiempo de retorno del procesador**. Uno de los criterios más importantes es el de duración, es decir, cuanto tiempo tarda en ejecutarse determinado proceso. El intervalo de tiempo que transcurre desde la emisión del proceso hasta su conclusión se le llama tasa de retorno, es decir, la suma de los periodos transcurridos en la memoria, esperando en la cola de procesos listos, ejecutándose en el procesador y efectuando operaciones de entrada y salida. [1]

- **Tiempo de espera del procesador**. Los algoritmos de planificación del procesador no afectan la cantidad de tiempo que realmente el proceso se ejecuta o lleva a cabo operaciones de entrada y salida. Los algoritmos únicamente afectan la cantidad de tiempo que se encuentran los procesos en la cola de listos, de modo que podemos considerar sólo el tiempo de espera para cada proceso en lugar del tiempo de retorno. [1]

- **Tiempo de respuesta del procesador**. En los sistemas en linea, es posible que el tiempo de retorno no sea el criterio más adecuado. Algunos procesos producen alguna salida útil durante los primeros instantes de su ejecución. Por esto, otra medición es el tiempo transcurrido desde la emisión de la solicitud hasta que se produce alguna respuesta. Esta medición es denominada tiempo de respuesta, es la cantidad de tiempo que el usuario espera antes de que el sistema responda la petición, pero no el tiempo necesario para mostrar dicha respuesta. [1]

Para aprovechar mejor los recursos es deseable maximizar la utilización del procesador y mejorar la productividad, minimizar los tiempos de retorno, espera y respuesta. La mayoría optima el promedio, pero en determinadas ocasiones puede ser deseable optimar los valores máximos y mínimos en lugar del promedio.

Se ha propuesto que para sistemas en linea es más importante minimizar la variación en los tiempos de respuesta en lugar del promedio. Esto es debido a que un sistema con tiempo de respuesta razonable y predecible es más deseable que un sistema que en promedio sea más rápido. [1]

Algoritmos de planificación

Primero en llegar, primero en salir

El algoritmo más sencillo para la planificación del procesador es el primero en llegar primero en salir, en este algoritmo el proceso que solicita primero el procesador es el primero que lo tiene asignado. Esta política se implanta fácilmente con una cola de primeras entradas, primeras salidas.

Este algoritmo no es apropiativo (es decir, no remueve un proceso de su ejecución para ejecutar otro). Una vez que al proceso se le asigna el procesador, éste lo conserva hasta que el proceso la libere, termine o realice una operación de entrada y salida. Este algoritmo es muy fácil de programar, pero ofrece un tiempo promedio de espera muy largo. [1]

Primero el proceso más breve

Otro enfoque para la planificación es el algoritmo del primero el proceso más breve. Este algoritmo asocia a cada proceso la cantidad de procesador requerida la próxima vez que se le asigne el procesador. Cuando el procesador está disponible, al proceso que tenga el menor consumo de procesador estimado se le asigna el procesador. Si existen varios procesos que tienen el mismo consumo de procesador, se utiliza el algoritmo de primero en llegar primero en salir para planificar los procesos empatados.

El algoritmo de primero el proceso más breve puede ser apropiativo o no apropiativo, esto se plantea cuando un nuevo proceso llega a la cola de procesos listos, mientras se está ejecutando otro proceso. El nuevo proceso puede tener un consumo menor que el que se encuentra en ejecución, si el algoritmo es apropiativo, se le quitará el procesador al proceso en ejecución, en caso de que no sea apropiativo, el algoritmo permitirá que el proceso se ejecute hasta que termine su consumo. [1]

Planificación por prioridades

El algoritmo de primero el proceso más breve es un caso especial del algoritmo general para la planificación por prioridades. En este algoritmo se asocia una prioridad a cada proceso y el procesador es asignado al proceso con mayor prioridad. Los procesos con igual prioridad se planifican utilizando el algoritmo de primero en llegar, primero en salir.
Las prioridades pueden definirse interna o externamente. Las prioridades internas utilizan alguna cantidad o cantidades medibles para calcular la prioridad instantánea del proceso (es decir, ésta puede cambiar). Las prioridades externas se fijan empleando criterios ajenos al sistema operativo.

La planificación por prioridades puede ser apropiativa o no apropiativa, cuando un proceso llega a la cola de procesos listos su prioridad puede ser comparada con la del proceso en ejecución, en caso de que el algoritmo fuera apropiativo, se le quitará el procesador al proceso en ejecución, en caso de que no sea apropiativo, el algoritmo permitirá que el proceso se ejecute hasta que termine su consumo. Adicionalmente a esto, si el algoritmo no es apropiativo el planificador colocará al proceso con más alta prioridad en el inicio de la cola de procesos listos.
Un problema muy serio con este algoritmo es que se puede presentar la inanición o bloqueo indefinido.

Una solución para el problema del bloqueo indefinido puede ser el envejecimiento, con esta técnica se aumenta gradualmente la prioridad de los procesos que esperan durante mucho tiempo el procesador para su ejecución. [1]

Planificación circular

Este algoritmo está diseñado para sistemas de tiempo compartido. Aquí se define una pequeña unidad de tiempo que se denomina cuanto de tiempo o quantum (generalmente varía entre 10 y 100 milisegundos, pero puede tener valores diferentes, ya sean mayores o menores). La cola de procesos listos se trata como una cola circular en la cual el planificador le asigna un quantum a cada proceso. Cuando un proceso nuevo llega al sistema, éste se agrega al final la lista.
El planificador toma el primer proceso de la cola, programa el cronómetro para que se interrumpa después de un quantum y despacha el proceso, al terminar su quantum el planificador coloca al proceso recién atendido al final de la lista y toma al siguiente proceso para ser despachado.
El problema de este algoritmo radica en que el tiempo de retorno es muy grande conforme aumentan los procesos dentro del sistema. [1]

Planificación de colas de múltiples niveles

Existen otra clase de algoritmos de planificación para aquellas situaciones en las que se pueden clasificar a los procesos en varios grupos.
El algoritmo de planificación de colas de múltiples niveles divide la cola de procesos listos en varias colas. Los procesos se asignan en forma permanente a una cola, generalmente esto se realiza distinguiendo alguna propiedad del proceso. Cada cola tiene un algoritmo de planificación determinado.
Existe un algoritmo maestro para la planificación de las múltiples colas, el cual generalmente es un algoritmo apropiativo de manejo de prioridades fijas.
Otro algoritmo muy utilizado es el de porciones fijas de tiempo a cada cola, con lo cual cada cola recibe cierta porción de tiempo de procesador (no es propiamente un quantum ya que puede ser de diferente tamaño para cada cola), la cual es a su vez asignada a los procesos de la cola en cuestión. [1]
Este algoritmo es utilizado en el sistema operativo MVS para la distribución de sus recursos (no únicamente del procesador).

Planificación de colas de múltiples niveles con realimentación

En un algoritmo para la planificación de colas de múltiples niveles los procesos se asignan de manera definitiva a una cola al ingresar al sistema, por esto no es posible moverlos a otras colas. Este algoritmo tiene la ventaja de provocar poca sobrecarga administrativa durante la planificación, pero es inflexible.
La planificación de colas de múltiples niveles con realimentación permite cambiar a los procesos de una cola a otra. La idea fundamental es la de separar los procesos con diferentes consumos de procesador. Con lo cual si un proceso

consume mucho tiempo de procesador, este proceso será cambiado a una cola de menor prioridad. Este algoritmo permite que los procesos en línea y los procesos limitados por entrada y salida se encuentren en las colas de mayor prioridad.
Un planificador de colas de múltiples niveles con realimentación define los siguientes parámetros:

- Número de colas
- Algoritmo de planificación para cada cola
- Método utilizado para promover un proceso
- Método utilizado para degradar un proceso
- Método utilizado para determinar a que cola ingresará un proceso que requiere servicio

Este es el algoritmo más general para la planificación del procesador, pero por lo mismo es el más complejo en cuanto a su configuración e implantación.

Semáforos y sincronización

La sincronización entre procesos concurrentes cooperativos es esencial para preservar las relaciones de precedencia (importancia y orden de ejecución de actividades especiales) y para evitar los problemas de temporización relacionados con la concurrencia. Los procesos cooperativos deben sincronizarse unos con otros cuando van a utilizar **recursos compartidos,** tales como estructuras de datos comunes o dispositivos físicos (como discos). Debido a que el sistema operativo no conoce ni debe conocer el orden de las actividades realizadas por un proceso, los procesos cooperativos deben encargarse de sincronizar adecuadamente sus operaciones. [2]

El problema de la sincronización es fácilmente demostrable. Supongamos dos procesos en ejecución P1 y P2, ambos procesos tienen acceso a una variable compartida que denominaremos **turno**, la cual será utilizada para indicar cuando se tiene el control de la impresora del sistema. Al inicio de la ejecución la variable compartida **turno** tendrá un valor de nulo. Primeramente el proceso P1 comprueba el contenido de la variable turno para verificar que nadie se encuentre utilizando la impresora, comprueba que nadie la está usando, pero desafortunadamente expira su tiempo y es intercambiado su bloque de control por el del proceso P2, ahora P2 también desea obtener acceso a la impresora, verifica su estado, el cual es nulo (debido a que P1 no alcanzó a modificar el valor para indicar propiedad de la impresora), con esto P2 se asigna la impresora y empieza a emitir un reporte, cuando expira su tiempo, el bloque de control del proceso P2 es intercambiado con el bloque de control del proceso P1, P1 no verifica nuevamente la variable compartida ya que fue verificada con anterioridad, por lo cual P1 se asigna la impresora y comienza a utilizarla emitiendo un reporte hacia la misma provocando que los reportes resulten mezclados; aquí vemos que la secuencia de sincronización para accesar a una variable compartida puede

presentar fallas, ya que en el peor de los casos ambos procesos se asignan el recurso causando un caos en la impresora.
Este ejemplo demuestra que la ejecución concurrente no sincronizada de procesos cooperativos puede dar lugar a errores (en algún caso puede no ocurrir el error, pero esto no es siempre verdadero) de sincronización, los cuales perjudican la confiabilidad del sistema. [2]

Exclusión mutua

Las dificultades anteriormente presentadas demandan un método alternativo que proporcione la metodología y herramientas necesarias para tratar con problemas de sincronización entre los procesos concurrentes.
En cierto sentido la actualización de una variable compartida puede ser considerada como una sección crítica (por los resultados que pueden obtenerse si no es controlado adecuadamente su acceso a la variable compartida). La sección crítica es una secuencia de instrucciones que delimita la actualización de una o más variables compartidas. Cuando un proceso ingresa en su sección crítica debe de completar todas las instrucciones incluidas en ésta antes de que se permita el acceso a cualquier otro proceso entrar en la misma sección crítica. Sólo el proceso que ejecuta la sección crítica tiene permitido el acceso a la variable compartida; los procesos restantes deben tenerlo prohibido hasta la terminación de la sección crítica. A esto se le denomina exclusión mutua, en la cual un solo proceso excluye temporalmente a todos los demás de utilizar un recurso compartido con el fin de asegurar la integridad del sistema.
Si el recurso compartido es una variable, la exclusión mutua asegura que como máximo un proceso tendrá acceso a la variable durante las actualizaciones. Como resultado, los procesos restantes solo obtienen valores consistentes de las variables compartidas.
Para que la exclusión mutua sea aceptable como herramienta general, las soluciones deben presentar las siguientes características:

- Asegurar la exclusión mutua de los procesos que accesan la variable compartida.
- No hacer suposiciones con respecto a las velocidades y prioridades relativas de los procesos en conflicto.
- Garantizar que el aborto o terminación de cualquier proceso fuera de su sección crítica no afecte la capacidad de los restantes procesos contendientes para acceder al recurso compartido.
- Cuando más de un proceso desee entrar en la sección crítica, conceder la entrada a uno de ellos durante un tiempo corto (finito).

El método más simple para asegurar la exclusión mutua es el de descartar la concurrencia. Este método no funciona debido a que anula las ventajas proporcionadas por la multiprogramación. Lo que realmente se desea es conceder

temporalmente a un proceso que necesita completar una sección crítica el acceso exclusivo a un recurso compartido.
En varias estrategias de exclusión mutua los procesos observan el siguiente protocolo básico:

- Negociación del protocolo {El ganador continúa}
- Sección crítica {Uso exclusivo del recurso}
- Protocolo de salida {Liberación del recurso}

Algoritmos de exclusión mutua para dos procesos

Los procesos son numerados como P0 y P1. Por comodidad, cuando hablemos de Pi utilizaremos Pj para referirnos al otro proceso.

Algoritmo 1

La primera aproximación consiste en permitir que los procesos compartan una variable entera común conocida como *turno* con un valor inicial dado (ya sea 0 o 1, dependiendo de quien se quiere que inicie). Si turno = i, entonces se permite que el proceso Pi se ejecute en su sección crítica
Esta solución asegura que solo un proceso a la vez puede encontrarse en su sección crítica, pero no cumple con el requisito de progreso, ya que se requiere de una alternación estricta de los procesos en ejecución. Esto puede provocar problemas de retraso si un proceso requiere más ingresos a su sección crítica que el otro.

```
while verdadero
{
while turno ¬= i ;              /* Hacer nada si no es turno i */
        entra_sección_crítica;
turno = j;
        entra_sección_restante;
}
```

Estructura del proceso Pi

Algoritmo 2

Para remediar el problema del algoritmo se substituye la variable turno con un arreglo.

```
booleano      indicador[2]={falso,falso};
```

Como se puede observar los elementos del arreglo toman un valor inicial de falso. Si indicador[i] == verdadero, esto indica que Pi está listo para entrar en la sección crítica.

En este algoritmo, el proceso Pi primero asigna verdadero a indicador[i] señalando que está listo para entrar en su sección crítica; después de esto, Pi verifica que el proceso Pj no este también listo para entrar en su sección crítica Si Pj estuviera listo, entonces Pi esperaría hasta que Pj indicara que ya no necesita estar en la sección crítica. En ese momento, Pi entraría en su sección crítica. Al salir de su sección crítica Pi asigna falso a indicador[i], permitiendo que el otro proceso entre en su sección crítica.
En esta solución se satisface el requisito de exclusión mutua, aunque los procesos pueden entrar en un bloqueo (sin avance), si ambos asignan un valor de verdadero a su indicador correspondiente.

```
while verdadero
{
indicador[i] = verdadero;
while indicador[j]==verdadero  ;   /* Hacer nada si indicador[j] es verdadero */
        entra_sección_crítica;
indicador[i] = falso;
        entra_sección_restante;
}
```

Estructura del proceso Pi

Algoritmo 3

Combinando las ideas claves de los algoritmos 1 y 2 se obtiene una solución correcta para el problema de la sección crítica. Los procesos comparten dos variables:

```
booleano     indicador[2]={falso,falso};
bit          turno;                          /* 0 o 1 */
```

Las variables indicador[i] e indicador[j] se inicializan con falso, el valor de turno puede ser 0 o 1 (al iniciar no importa que valor tenga).
Para entrar en la sección crítica, el proceso Pi primero asigna verdadero a indicador[i] y luego afirma que es el turno del otro proceso para ingresar a la sección crítica si así lo desea (turno = j). Si ambos procesos tratan de entrar a la vez, se asignará turno como i y j aproximadamente al mismo tiempo. Solo una de estas asignaciones durará; la otra ocurrirá, pero será reemplazada de inmediato. El valor temporal de turno decide a cual de los dos procesos se le permite el ingreso a su sección crítica.

```
while verdadero
{
indicador[i] = verdadero;
turno=j;
while ((indicador[j]==verdadero) && (turno==j) );
        /* Hacer nada si indicador[j] es verdadero y turno es j */
entra_sección_crítica;
indicador[i] = falso;
entra_sección_restante;
}
```

Estructura del proceso Pi

Semáforos

Las soluciones al problema de exclusión mutua no son fáciles de generalizar para problemas más complejos. A fin de superar este obstáculo es posible utilizar una nueva herramienta de sincronización conocida como semáforo. Un semáforo S es una variable entera a la que salvo la asignación de sus valores iniciales, solamente puede accederse mediante operaciones atómicas (indivisibles, que solo utilizan una instrucción del procesador para realizar toda la operación o que inhabilitan las interrupciones antes de su ejecución para no ser suspendidas durante la misma ya que acceden a recursos compartidos) comunes conocidas como: espera (wait) y señal (signal).
Las definiciones clásicas de espera y señal son:

```
espera(S): while S <= 0 ;          /* Mientras sea menor o igual a 0 hacer nada */
                S = S - 1;

señal(S): S = S + 1;               /* No importa el valor, solo lo incrementa */
```

Las modificaciones al valor entero del semáforo en las operaciones espera y señal se ejecutan indivisiblemente. Es decir, cuando un proceso modifica el valor del semáforo, ningún otro proceso puede modificar simultáneamente ese mismo valor. Además, en el caso de espera(S), la evaluación del valor entero de S (S<=0) y su posible modificación (S=S-1), también deben ejecutarse sin interrupción.

Utilización de semáforos

Los semáforos pueden ser utilizados para resolver el problema de la sección crítica para n proceso; estos n procesos comparten un semáforo común, mutex (en todos los textos encontrados sobre el tema el nombre del semáforo es mutex, esto es debido a que implica **mutua exc**lusión), con valor inicial de 1. Cada proceso se organiza de la siguiente manera:

```
while true
{
espera(mutex);
        sección_crítica;
señal(mutex);
        sección_restante;
}
```

Implantación de semáforos

La principal desventaja de las soluciones presentadas para el problema de sección crítica con y sin semáforos es que todas estas soluciones requieren espera activa. Mientras un proceso se encuentra en su sección crítica, cualquier otro proceso que intente entrar en esa sección deberá ejecutar un ciclo continuo en la sección de ingreso a la sección crítica. Esto es un problema en un sistema multiprogramado, ya que se desperdician preciosos ciclos de CPU en actividades de espera. A este tipo de semáforos (con espera activa) se les conoce como cerradura giratoria. La ventaja de una cerradura giratoria es que no es necesario efectuar un cambio de contexto cuando el proceso debe esperar una cerradura. Por esto, cuando se espera que las cerraduras tengan breve duración, la cerradura giratoria es de utilidad.
Para resolver el problema de espera activa, es posible modificar la definición de las operaciones espera y señal del semáforo. Cuando un proceso ejecuta la operación espera y encuentra que el valor del semáforo no es positivo, debe esperar. Sin embargo, en lugar de esperar activamente, el proceso se puede bloquear a si mismo. La operación de bloqueo coloca al proceso en una cola de espera asociada con el semáforo y cambia el proceso al estado de bloqueo. Con lo cual el planificador puede seleccionar otro proceso para su ejecución.
Un proceso bloqueado, en espera de un semáforo S, debe reiniciarse con la ejecución de una operación señal realizada por otro proceso. El proceso se reinicia con una operación despertar (wake up), que cambia el proceso del estado bloqueado al estado de listo, para que el proceso sea colocado en la cola de procesos listos.
Para esto definimos a un semáforo siguiendo esta definición:

```
struct semáforo {
int valor;
lista_procesos L;
} semáforo;
```

Cada semáforo tiene un valor entero y una lista de procesos; cuando un proceso debe esperar en un semáforo, se añade a la lista de procesos en espera por el

semáforo. Una operación de señal elimina uno de los procesos de la lista de bloqueados y lo despierta.
Así podemos definir las operaciones del semáforo como.

```
espera(S) :   S.valor = S.valor -1;
              if S.valor < 0 {
                    añadir proceso a S.L;
                    bloquear proceso;
                    }

señal(S):     S.valor = S.valor +1;
              if S.valor <= 0 {
                    sacar proceso de S.L;
                    despertar proceso;
                    }
```

La operación bloquear suspende al proceso que la invoca. La operación despertar(P) reanuda la ejecución de un proceso P bloqueado. Estas dos operaciones son proporcionadas por el sistema operativo como llamadas del sistema.
Cabe señalar que una diferencia con la definición inicial de semáforos es que esta si permite valores negativos del mismo. Si el valor del semáforo es negativo indica que este valor negativo es el número de procesos esperando por el semáforo. Este factor es el resultado de cambiar el orden de la reducción y de la comprobación en la implantación de espera.
La lista de procesos puede implantarse fácilmente con un campo de enlace en cada bloque de control del proceso. Cada semáforo contiene un valor entero y un apuntador a una lista de bloques de control de procesos bloqueados en el semáforo. Una manera de añadir y eliminar procesos de la lista, que asegura la espera limitada, sería por orden de llegada (cola del tipo primero en entrar primero en salir), donde el semáforo contiene apuntadores al inicio y al final de la cola.
Es importante indicar que con la definición última de espera y señal no se ha eliminado por completo la espera activa. Lo que se ha hecho es eliminar la espera activa de la entrada a las secciones críticas.

Soporte de hardware para sincronización

Debido a que la sincronización de procesos es una parte muy importante en los sistemas donde existe la multiprogramación los fabricantes de hardware han **cooperado** proporcionando instrucciones especiales que facilitan la multiprogramación (en nuestro caso solamente trataremos la exclusión mutua).
El problema de la sección crítica puede solucionarse si se evitan interrupciones cuando es modificada una variable compartida, pero generalmente no es posible esta solución. Por esto, varios proveedores de hardware proporcionan juegos de

instrucciones especiales que permiten evaluar y modificar el contenido de una variable o registro o intercambiar el contenido de dos registros Estas instrucciones pueden ser utilizadas para resolver el problema de la sección crítica de una manera sencilla. Aquí mostramos una abstracción de los conceptos principales que existen detrás de estos dos tipos de instrucciones.

La instrucción evaluar y asignar se puede definir de la siguiente manera:

```
boolean evaluar_y_asignar(boolean *objetivo)
        {
        evaluar_y_asignar = *objetivo;
        *objetivo = verdadero.
        }
```

La característica más importante es que la instrucción se ejecuta atómicamente (indivisiblemente).
Aquí mostramos como se utilizaría la instrucción evaluar_y_asignar para mantener la exclusión mutua en la sección crítica:

```
while verdadero {
        while evaluar_y_asignar(cerradura) ;     /* Hacer nada mientras alguien se
                                                    encuentre en la sección crítica */
                sección_critica;
        cerradura = falso;
                sección_restante;
        }
```

Implantación de exclusión mutua con evaluar_y_asignar

La instrucción intercambiar intercambia atómicamente el contenido de dos palabras y se define de la siguiente manera:

```
void intercambiar(boolean *a, *b) {
        boolean temp;
        temp = *a;
        *a = *b;
        *b  =temp;
        }
```

La exclusión mutua puede ofrecerse como se describe a continuación. Se declara una variable booleana global llamada cerradura y se le asigna un valor inicial de falso. Cada proceso tiene una variable local llamada clave. Aquí mostramos como se utilizaría la instrucción intercambiar para mantener la exclusión mutua en la sección crítica:

```
while verdadero {
        clave = verdadero;
        while clave = verdadero intercambiar(cerradura,clave);
                sección_crítica,
        cerradura = falso;
                sección_restante;
        }
```

Implantación de exclusión mutua con intercambiar

Bloqueos y recursos

En un entorno de multiprogramación varios procesos pueden competir por un número reducido de recursos. Un proceso emite su petición de recursos al sistema operativo, si en ese momento no se encuentran disponibles, el proceso se bloquea hasta que el recursos o los recursos deseados estén disponibles. Puede suceder que los procesos bloqueados por recursos nunca cambien de estado debido a que los recursos que solicitan están en poder de otros procesos que también se encuentran bloqueados en espera de otros recursos. A esta situación se le denomina bloqueo mutuo.

Caracterización de bloqueos mutuos

Los bloqueos mutuos son indeseables, pero cuando éstos se presentan, los procesos no concluyen su ejecución y los recursos del sistema se congelan impidiendo que otros procesos se inicien o continúen.

Una situación de bloqueo mutuo puede surgir si y solo si en un sistema se presentan simultáneamente las siguientes cuatro condiciones:

Exclusión mutua.- Por lo menos un recurso debe retenerse en modo no compartido; es decir, sólo un proceso a la vez puede utilizar el recurso. Si otro proceso solicita el recurso, este proceso deberá esperar hasta que se haya liberado.
Retención y espera.- Debe existir un proceso que retenga por lo menos un recurso y espere adquirir otros recursos retenidos por otros procesos.
No apropiación.- Los recursos no se pueden quitar; es decir, un recurso solo puede ser liberado voluntariamente por el proceso que lo retiene.
Espera circular.- Debe haber un conjunto (P0, P1, ..., Pn) de procesos en espera tales que P0 espera por un recurso retenido por P1, P1 espera un recurso retenido por P2, ... Pn-1 espera un recurso retenido por Pn, Pn espera un recurso retenido por P0.

Es importante recalcar que se deben presentar las cuatro condiciones para que pueda aparecer el bloqueo mutuo. La condición de espera circular implica la

condición de retención y espera. por lo que las cuatro condiciones no son completamente independientes Sin embargo es bastante útil considerar por separado cada condición

Métodos para manejar bloqueos mutuos

Existen dos métodos principales para tratar el problema de los bloqueos mutuos. Es posible la utilización de protocolos para asegurar que el sistema nunca entre en un estado de bloqueo mutuo, o se puede permitir un bloqueo mutuo y luego provocar que se recupere La recuperación de un bloqueo mutuo es difícil, y costosa en cuanto a recursos (detección y cura), por lo cual se recomienda el uso de protocolos.

Prevención de bloqueos mutuos

Para que ocurra un bloqueo mutuo deben de presentarse cada una de las cuatro condiciones necesarias. Al asegurar que al menos una de éstas no se cumpla es posible la prevención de los bloqueos mutuos. Por esto, se examinarán cada una de las condiciones necesarias por separado.

Exclusión mutua. Ésta debe conservarse para recursos no compartibles Los recursos compartibles no requieren acceso mutuamente excluyente, por lo tanto son candidatos que no participan en un bloqueo mutuo. Un proceso nunca necesita esperar un recurso compartible. Sin embargo, por lo general no es posible evitar bloqueos mutuos negando la condición de exclusión mutua. Es importante señalar que algunos tipos de recursos no pueden compartirse.

Retención y espera. Para asegurar que la condición de retención y espera nunca suceda se debe garantizar que cuando un proceso solicita un recurso, no deben ser retenidos los otros recursos que están (más bien estuvieron) en posesión del proceso. Un protocolo que puede utilizarse en este caso es el de asignación total de recursos, es decir, al proceso solicitante de recursos se le asignan todos los recursos que requiera durante su ejecución.
Un protocolo alternativo permite que un proceso solicite recursos sólo cuando no tenga alguno asignado con anterioridad. Un proceso puede solicitar recursos y utilizarlos, pero antes de que puede solicitar más recursos debe de liberar el recurso anteriormente asignado.

Los protocolos tienen dos desventajas básicas:

- Utilización de recursos baja
- Asignación de recursos sin utilizar durante largo tiempo

Otro problema de los protocolos es que pueden generar un bloqueo indefinido. Un proceso que requiere varios recursos de alta utilización quizá tenga que esperar

indefinidamente porque algunos recursos que necesita están asignados a otro proceso.

No apropiación. La tercera condición necesaria es que no se presente la apropiación de los recursos que ya se hayan asignado. Para asegurar que no se de esta situación, es posible la utilización del siguiente protocolo: Si un proceso que retiene algún recurso solicita otro que no se le puede asignar de inmediato, entonces se le expropian todos los recursos que tiene. Los recursos apropiados se agregan a la lista de recursos que espera el proceso, el cual iniciará únicamente cuando pueda recuperar la totalidad de recursos requeridos.
Este protocolo puede aplicarse a recursos cuyo estado puede almacenarse y recuperarse más tarde con cierta facilidad, tales como los registros del procesador y espacio de memoria. No es aplicable a todos los tipos de recursos.

Espera circular. Una forma de asegurar que no se presente esta condición es imponer una ordenación total de todos los tipos de recursos y requerir que cada proceso solicite los recursos siguiendo un orden de numeración ascendente. El detalle aquí está en una numeración correcta de los recursos, con una numeración correcta la condición de espera circular no puede existir.

◆ CAPITULO III.- ADMINISTRACION DE MEMORIA

Concepto de memoria

Uno de los recursos más importantes de los sistemas de cómputo es la memoria, pero, ¿Que es la memoria?

La memoria es el lugar donde se almacenan temporalmente datos y programas, ésta es básicamente de dos tipos distintos:

- Memoria real, o memoria principal
- Memoria secundaria

La memoria real tiene las siguientes caracteristicas:

- Rapidez de acceso
- Volatilidad, refiriéndonos a que su contenido varía mucho con el tiempo
- Alto precio
- Capacidad limitada de almacenamiento (esta caracteristica se desvanece poco a poco con el descenso de los precios de la memoria gracias a los avances de la tecnología)

La memoria secundaria tiene las siguientes características:

- Lentitud de acceso (esta característica cada vez se reduce más debido a las altas velocidades de los disco modernos)
- Permanencia, refiriéndonos a que su contenido no varia mucho con el tiempo
- Bajo precio
- Capacidad casi ilimitada

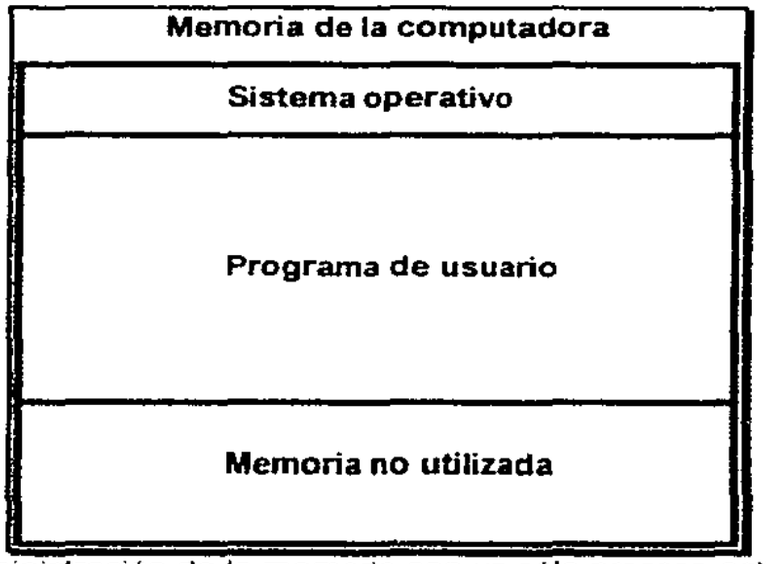
En este capítulo nos dedicaremos a la administración de la memoria real o primaria, ya que la administración de memoria secundaria se explicará en el capítulo de administración de dispositivos y administración de archivos.

Existen dos grandes divisiones para la administración de la memoria, una administra toda la memoria requerida por un proceso o usuario colocando la totalidad de ésta en forma contigua, otra administra la memoria dividiéndola generalmente en bloques pequeños, pudiendo colocar a un proceso o usuario en áreas de memoria totalmente distintas.

En los primeros sistemas de cómputo utilizaban el método de memoria contigua, porque era mucho más fácil su administración, posteriormente se vió la necesidad de incrementar la efectividad en el uso de la memoria con lo cual nacieron

diversas políticas de administración que resolvían unos problemas y creaban otros, culminando con lo que ahora conocemos como memoria virtual.

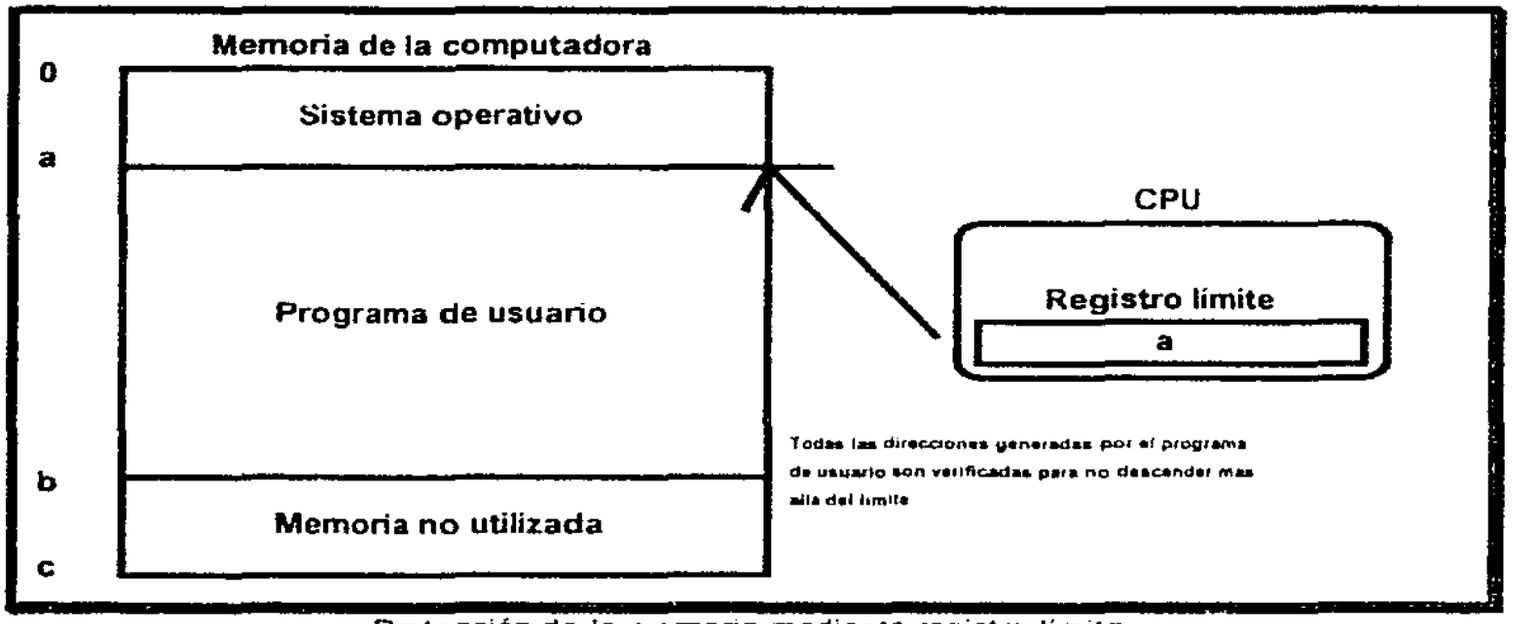
Alrededor de los años 60's donde las computadoras eran muy grandes y costosas el método de administración de memoria era muy sencillo, un sólo proceso a la vez, el cual ocupaba toda la memoria (y a veces no toda). El problema de este tipo de administración de memoria era que la persona que utilizaba la computadora normalmente programaba también las rutinas de acceso a los dispositivos que utilizaba durante la corrida de su programa, lo que dificultaba su uso y disminuía además la posibilidad de que varias personas utilizaran los equipos por la dificultad de programación.



Administración de la memoria con un sólo proceso activo

Posteriormente los programadores de sistemas agregaron múltiples rutinas de bajo nivel que ayudaban a que los usuarios no accesaran a los dispositivos periféricos directamente, pero el problema seguía siendo el mismo, sólo un programa a la vez era ejecutado en la computadora.

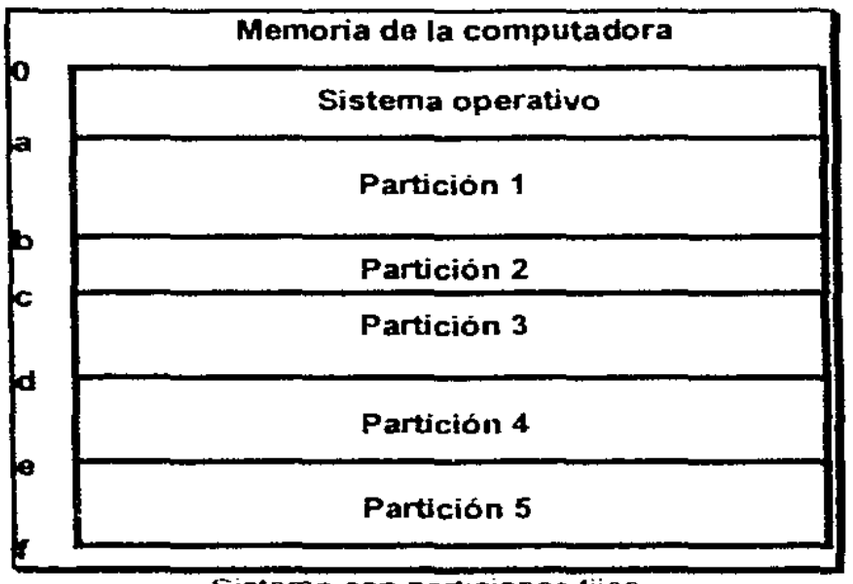
Los algoritmos de administración eran muy sencillos, ya que solo incluían verificación del límites para no sobrepasar el área del sistema operativo. Estos registros límites se volvieron tan populares, que algunos fabricantes de computadoras de la época los incluyeron en el hardware de las mismas para acelerar las operaciones de verificación de éstos.



Protección de la memoria mediante registro límite

Al proteger al sistema operativo lo único que se evitaba es tener que recargarlo cada vez que era dañado por el usuario, la mayor parte de los programas realizaban muchas operaciones de entrada y salida por lo que los sistemas eran altamente desperdiciados (esto era debido a que las operaciones de entrada y salida no utilizan o casi no utilizan el procesador). Por ésto, se creó un nuevo método de administración de almacenamiento que permitia tener al mismo tiempo en la memoria varios programas, este método se denominó administración de memoria con particiones fijas.

En este método de administración la memoria se dividia en varias particiones (pudiendo ser de diferentes tamaños), en la primera y más baja se encontraba siempre el sistema operativo y en las siguientes se encontraban los procesos en ejecución. La configuración de las particiones era asignada cuando se levantaba el sistema y no podía ser modificada mientras éste estuviera activo. Existe un sistema operativo con las caracteristicas antes mencionadas que todavía hoy se encuentra en uso en algunas instalaciones, este es el VSE/ESA de IBM.



Sistema con particiones fijas

Dentro de estos sistemas existian varias modalidades de ejecución y creación de los procesos, estas son:

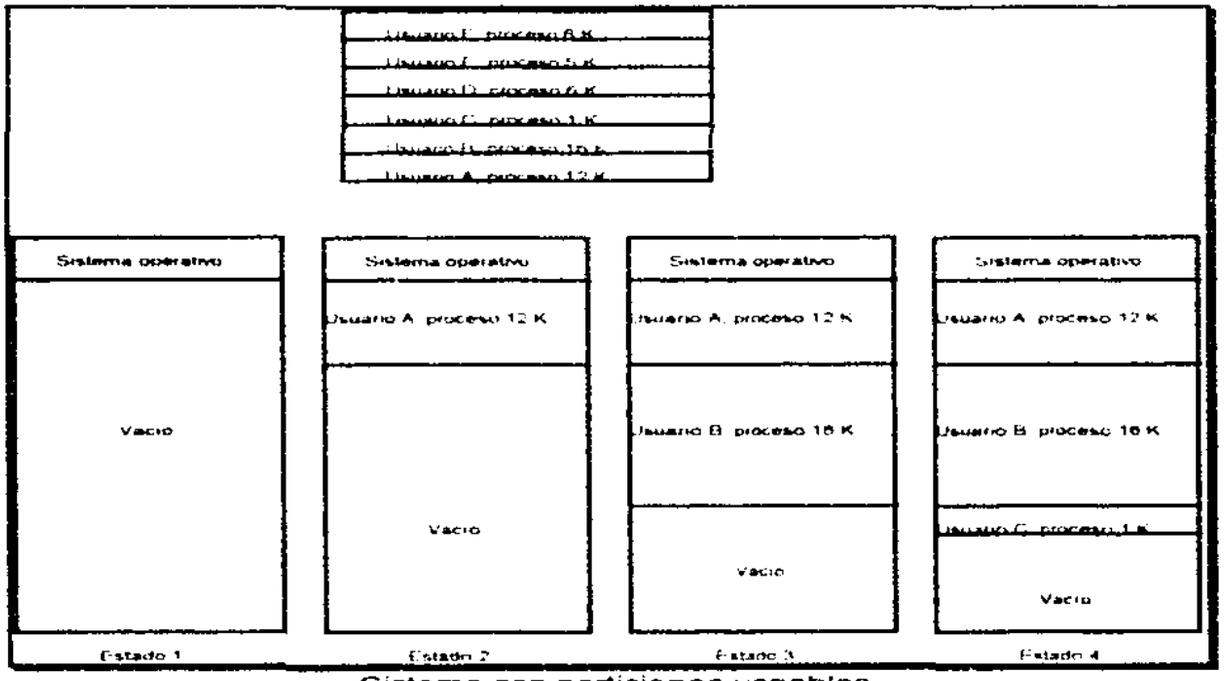
- Traducción y carga absoluta
- Traducción y carga con relocalización

La traducción y carga absoluta forzaba a que los programas se ejecutaran siempre en una partición determinada, esto debido a que al momento de su compilación y enlace se fijaban direcciones absolutas, que no podían cambiarse en tiempo de ejecución. Este tipo de modalidad podía causar problemas, ya que si muchos programas habían sido compilados y enlazados con la misma dirección de partición, esa partición tendría muchos procesos en espera para ejecución pudiendo estar alguna o algunas particiones adicionales libres.

La traducción y carga con relocalización permitía la ejecución de los programas en cualquier partición de tamaño suficientemente grande para contenerlos, con lo cual se evitaba la saturación de determinadas particiones.

En los sistemas con particiones fijas los esquemas de protección de memoria se limitaban a generar varios registros límites y a verificar si las peticiones de acceso a memoria cumplían con los mismos, es decir, para cada partición se generaban dos registros límites, uno superior y otro inferior, para el caso del inferior no se aceptaban peticiones de memoria menores a este, y para el superior, no se aceptaban peticiones de memoria mayores a éste. Otra manera de proteger la memoria era utilizar un solo registro límite por partición y la longitud de la partición como límite, con lo cual solo se necesitaba conocer donde empezaba la partición y su longitud para poder proteger las particiones de accesos inválidos.

Los sistemas de particiones fijas estaban muy limitados debido a que solo aceptaban procesos que se ajustaran a los tamaños establecidos cuando se arrancaba el sistema, por lo cual se ideó una nueva manera de administrar la memoria (sistema con particiones variables), este esquema no limitaba las particiones (donde se ejecutaban procesos de usuario) a un tamaño único, al contrario, según el proceso requiriera memoria (y la memoria alcanzara) se generaban particiones adecuadas para los procesos en tiempo de ejecución.



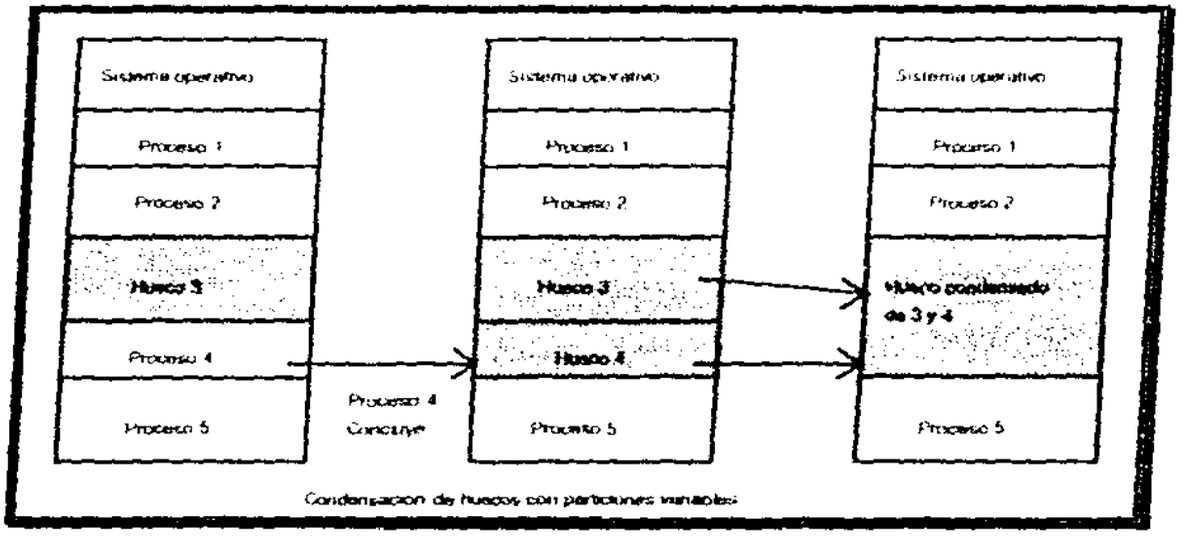
Sistema con particiones variables

Aquí notamos que la administración de memoria con particiones variables es un poco más complicada, puesto que implica que algunas veces quedarán huecos entre particiones ocupadas, estos huecos son generados por procesos que concluyen y son desalojados de memoria dejando los ya antes mencionados huecos.
Con esto la utilización de registros límite no basta, pués ahora existen varios problemas adicionales, tales como la condensación de huecos, y la fragmentación de la memoria.

Condensación de huecos

Cuando dos o más procesos que utilizan memoria contigua terminan (no es necesario que terminen al mismo tiempo, lo único que es importante es que las áreas de memoria que se han dejado de utilizar sean contiguas) surge el

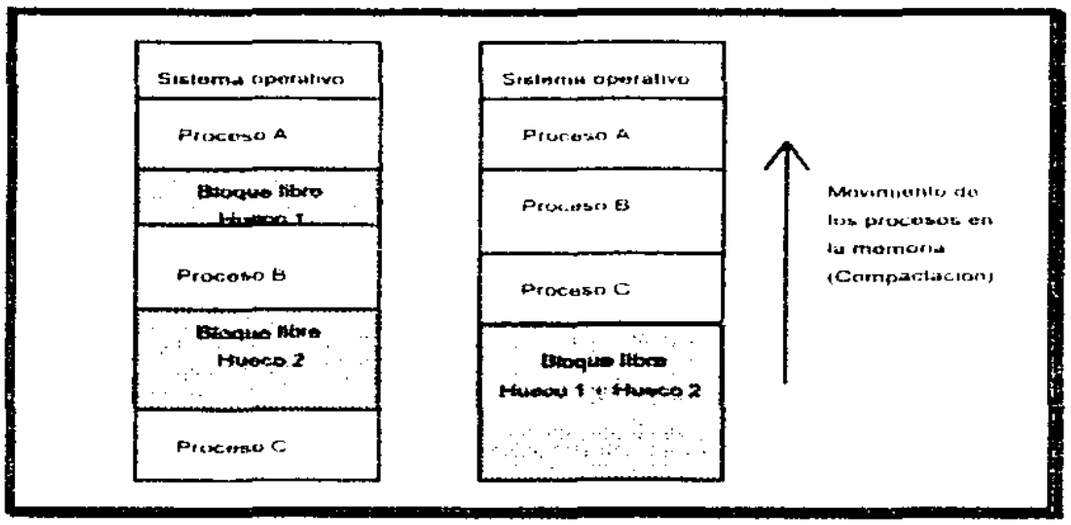
problema de que se tienen varios apuntadores hacia esas áreas de memoria contigua, estos apuntadores podrían ser uno solo unificando toda la memoria libre contigua en un solo bloque.



Este proceso ayuda a manipular menos áreas de memoria cuando es posible su unificación en un solo bloque.

Compactación de huecos

Cuando se tienen muchos huecos existe mucha fragmentación, es decir, existen dentro de la memoria muchas áreas que anteriormente fueron utilizadas y ahora están libres, que no son necesariamente contiguas, es posible que los procesos más demandantes de memoria no quepan en el bloque de memoria libre de mayor tamaño, pero sumando todas las áreas libres si exista espacio disponible para contenerlo, esto da lugar a la posibilidad de mover todos los bloque usados hacia abajo de la memoria unificando de alguna manera los huecos dejando un bloque de mayor tamaño de memoria libre, el cual es capaz de contener el proceso que demanda memoria; al movimiento de los bloques de memoria utilizados hacia la parte baja se le denomina compactación o reorganización de memoria.



Uno de los sistemas operativos actuales que utilizan compactación de memoria es el OS/2, pero solo lo hace cuando es estrictamente necesario, es decir, cuando un proceso no cabe en el bloque de memoria de mayor tamaño.

Políticas de administración de memoria

El manejo de particiones variables implica algunas complicaciones en el manejo de la memoria, una de estas complicaciones es la necesidad de establecer políticas de uso de memoria, es decir, se debe elegir cuando un bloque de memoria es asignado. Estas políticas son:

- Primer ajuste
- Mejor Ajuste
- Peor ajuste

Aunque existen otras políticas para la administración de memoria con particiones variables las aquí mencionadas son las más utilizadas.

Primer ajuste

Cuando se utiliza esta política se busca dentro de la lista de bloques libres de memoria el primero que sea capaz de contener el proceso que se desea colocar en memoria. Aún si el proceso es pequeño y el primer bloque donde cabe es muy grande esta política asigna el bloque. La ventaja que posee esta política es la velocidad, ya que, cuando encuentra el primer bloque libre que pueda contener al proceso lo asigna y termina. La desventaja aquí es la fragmentación, ya que si el proceso en cuestión requiere poca memoria y el primer bloque libre es muy grande, como se dijo anteriormente, este bloque será asignado y fraccionado, es

decir, dividido en un bloque pequeño usado por el proceso y la parte restante no usada del bloque se marcará como bloque libre

Mejor ajuste

Cuando se utiliza esta política se busca dentro de la lista de bloques libres el bloque en el cual se ajuste mejor el proceso, es decir, en el cual la memoria no utilizada sea menor. Este proceso tiene varias desventajas, es lento, ya que examina todos los bloques libres, además de la lentitud, los bloques que se quedan sin asignar generalmente son muy pequeños provocando que solo pocos procesos con demandas de memoria bajas se puedan ejecutar, logrando así una alta fragmentación de la memoria. Se ha demostrado con pruebas estadísticas que el mejor ajuste no es el que ofrece más ventajas en la utilización de memoria, ya que como se dijo anteriormente el tiempo es alto al igual que la fragmentación.

Peor ajuste

Cuando se utiliza esta política se busca dentro de la lista de bloques libres el bloque en el cual se ajuste peor el proceso, es decir, en el cual la memoria no utilizada sea la mayor. Este proceso tiene varias desventajas, al igual que el mejor ajuste es lento, ya que recorre toda la lista de bloque libres, pero su aparente desventaja (peor ajuste) hace que la fragmentación no sea tan crítica, ya que se busca el bloque de mayor tamaño, y se asigna la memoria necesaria para el proceso, marcando el resto del bloque como un bloque completo libre, el cual generalmente será un bloque grande.

Intercambio

Otra política de administración de memoria es el intercambio en la cual se asigna toda la memoria a un proceso y cuando este no puede continuar se transfiere totalmente a memoria secundaria, es decir, disco y posteriormente se carga el siguiente proceso. En esta política toda la memoria es asignada a un proceso; excepto las partes de memoria ocupadas por el sistema operativo, por unos momentos, la desventaja más evidente es que este proceso es extremadamente lento debido al intercambio de los procesos completos hacia el disco. Esta política no es muy utilizada actualmente pero dió las bases para el diseño de lo que se llama memoria virtual.

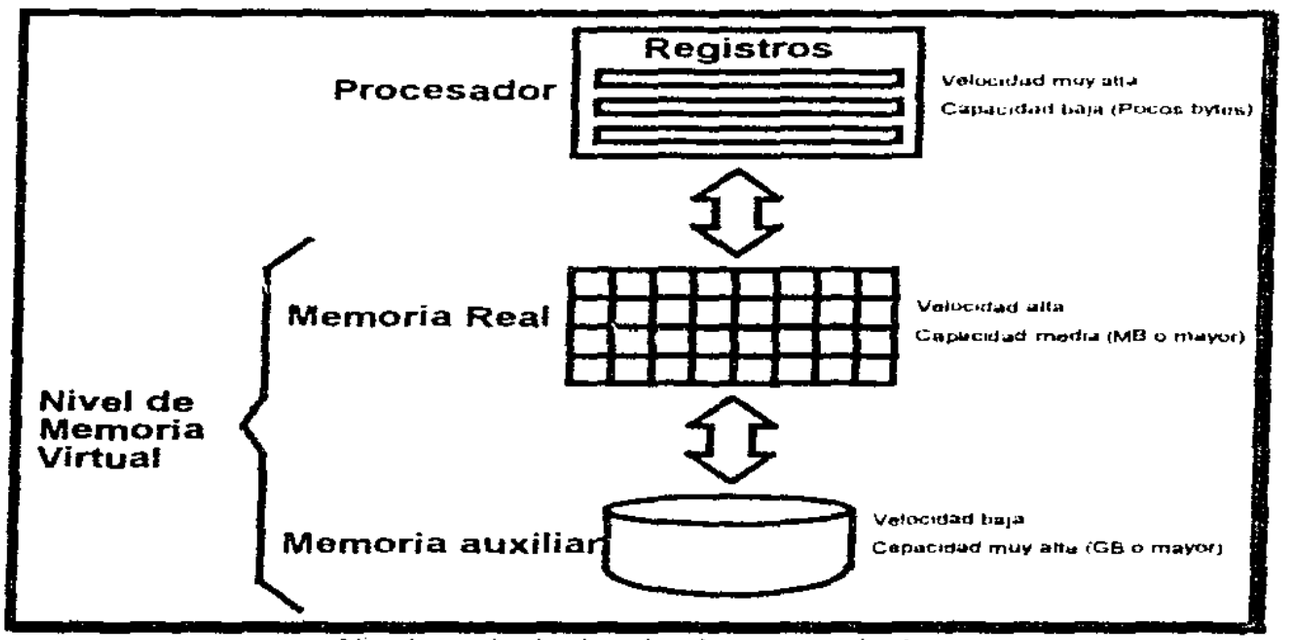
Memoria virtual

Cuando la cantidad de memoria existente dentro de una computadora no fue suficiente para almacenar los procesos se ideó una nueva manera de administrar la memoria, esta nueva manera de ver a la memoria se le llamó memoria virtual. El término de memoria virtual generalmente se asocia con la capacidad para obtener acceso a direcciones en un espacio de memoria mucho mayor que el

disponible en memoria primaria. Esto se consigue asignando un área fija de disco denominada según su administración área de paginación o segmentación.
Cabe mencionar que el manejo de memoria virtual requiere una cantidad mayor de pasos para satisfacer los requerimientos de memoria de algún proceso en ejecución, por lo cual es importante establecer el balance entre la cantidad de memoria y el desempeño del sistema.

El almacenamiento virtual es útil porque solo una pequeña parte de los procesos en ejecución es la que se encuentra activa, por lo cual la parte restante del proceso no se requiere en un momento determinado, con lo cual si solo se mantienen las direcciones de memoria necesarias para un proceso durante su ejecución se puede incrementar drásticamente el número de procesos residentes en memoria incrementando de esta manera el nivel de multiprogramación

Existen varios niveles de almacenamiento cada uno de los cuales posee características diferentes en cuanto a tamaño y velocidad, aquí mostramos estos niveles e indicamos en que niveles se encuentra la memoria virtual.



Niveles principales de almacenamiento

La clave del almacenamiento virtual es la disociación de direcciones a las que hacen referencia los procesos en ejecución de las direcciones disponibles en el almacenamiento real.
A las direcciones que hacen referencia los procesos en ejecución se conocen como direcciones virtuales, pero a las direcciones que existen en el almacenamiento primario (que es donde se encuentran realmente los procesos en

ejecución, o piezas de los procesos en ejecución) se le conoce como direcciones reales.
Los procesos que se ejecutan en un sistema con memoria virtual solo pueden hacer referencia a direcciones virtuales, por esto, es importante establecer las reglas de correspondencia entre las direcciones virtuales y las direcciones reales, adicionalmente a lo anterior la traducción de direcciones virtuales a reales debe hacerse con mucha rapidez, ya que de lo contrario el rendimiento del sistema estaría gravemente afectado o degradado y los beneficios de la memoria virtual no valdrían la pena.
Los diseñadores de sistemas operativos están muy preocupados por las desventajas que pueden tener los sistemas con almacenamiento virtual, por lo cual, se han desarrollado varios métodos para la asociación de direcciones de memoria virtuales con direcciones de memoria reales
Uno de los mecanismos más utilizados para la conversión de direcciones virtuales a reales es el método de traducción dinámica de direcciones (Dynamic Address Translation, DAT) en el cual se traducen las direcciones durante la ejecución de los procesos, cabe mencionar que para que esta traducción sea rápida se requiere soporte adicional de hardware.
El mecanismo de traducción dinámica de direcciones debe tener un *mapa de correspondencia para la traducción de direcciones* que muestre cuales direcciones de almacenamiento virtual se encuentran en el almacenamiento real y donde están localizadas, es decir, cuales direcciones y donde
Si la correspondencia de direcciones fuera de 1 a 1 sería necesaria una gran cantidad de memoria sólo para la administración lo cual no es deseable, por esta razón generalmente las direcciones son agrupadas en bloques con lo cual el mapa de correspondencia puede ser de un tamaño aceptable. Cabe señalar que entre mayor sea el tamaño del bloque, el mapa de correspondencia será menor, pero entre mayores sean los bloques mayor será el tiempo de recuperación. En este caso, también el consumo de almacenamiento real es mayor por lo cual el nivel de multiprogramación se reduce, adicionalmente la fragmentación es mayor ya que generalmente en el último bloque de cada proceso se desperdicia gran cantidad de memoria.
En la actualidad no existe un acuerdo en cuanto al tamaño de los bloques por lo cual si los bloques son todos de igual tamaño se llaman páginas y su método de administración se denomina paginación. Cuando los bloques tienen (pueden o no tener) tamaños distintos se llaman segmentos y su método de administración se denomina segmentación. Cabe mencionar que algunos sistemas combinan ambas técnicas, con segmentos de tamaño variable compuestos de páginas de tamaño fijo

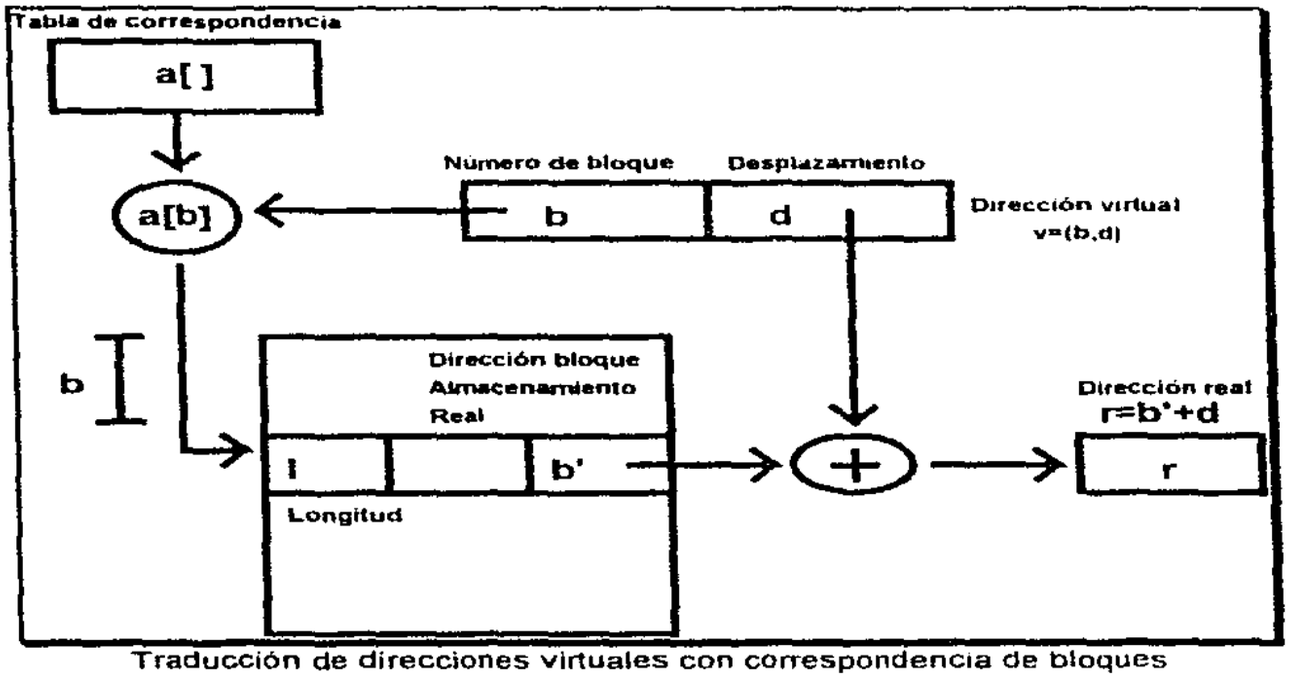
Las direcciones virtuales están compuestas de un par ordenado que indica básicamente lo siguiente:

Número de bloque	Desplazamiento
b	d

Dirección Virtual v=(b,d)

Donde v es la dirección virtual, **b** el número de bloque en el que se encuentra el elemento al que se hace referencia y **d** es el desplazamiento a partir del inicio del bloque.
Para la traducción de una dirección virtual a real se sigue el siguiente procedimiento:

a) Primeramente se localiza el inicio de la tabla (arreglo) de correspondencia, esta dirección la denominaremos **a[ ]**
b) Se desplazan **b** registros a partir del inicio **a[ ] (a[b])** de la tabla de correspondencia.
c) Se obtiene la dirección del bloque **b'** que se encuentra en almacenamiento real.
d) Se suma el desplazamiento **d** a la dirección del bloque **b'** para obtener la dirección real deseada.



Traducción de direcciones virtuales con correspondencia de bloques

Paginación

Debido a que los sistemas que manejan tamaños múltiples de bloque son bastante complejos comenzaremos nuestra explicación para cuando el tamaño de

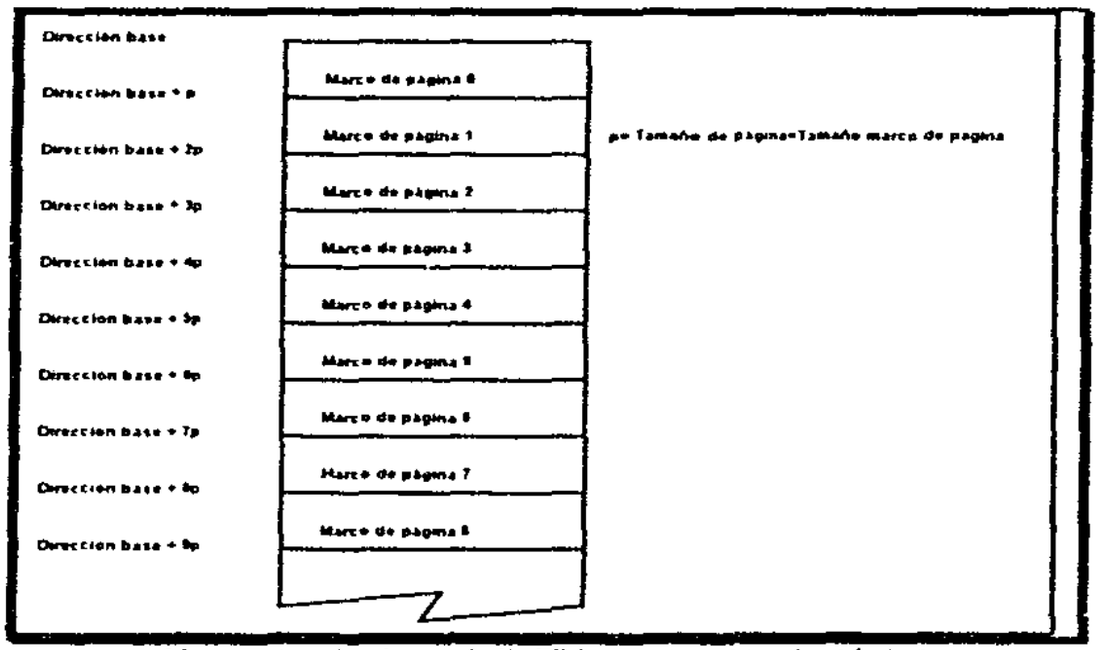
los bloques es fijo. A la correspondencia de direcciones cuando los bloques son todos de igual tamaño se le conoce como paginación o paginación pura.
Una dirección virtual en un sistema paginado es un par ordenado (p,d) en el cual **p** es el número de la página dentro del almacenamiento virtual, y **d** es el desplazamiento dentro de la página **p**.

Número de página	Desplazamiento
p	d

Dirección Virtual **v=(p,d)**

Dirección virtual en paginación

Es posible que un proceso se encuentre activo dentro del sistema (es decir, en ejecución) si su página activa se encuentra en el almacenamiento primario. La transferencia de las páginas se realiza desde el almacenamiento secundario (cuando la página no se encuentra en almacenamiento real) hasta el almacenamiento primario colocándose dentro de bloques conocidos como marcos de página, los cuales tienen el mismo tamaño que las páginas que son ingresadas a éstos. Es importante destacar que los marcos de página comienzan en direcciones del almacenamiento real que son múltiplos enteros del tamaño fijo de la página. Las páginas que ingresan a la memoria real pueden colocarse en cualquier marco de página libre dentro de la misma.

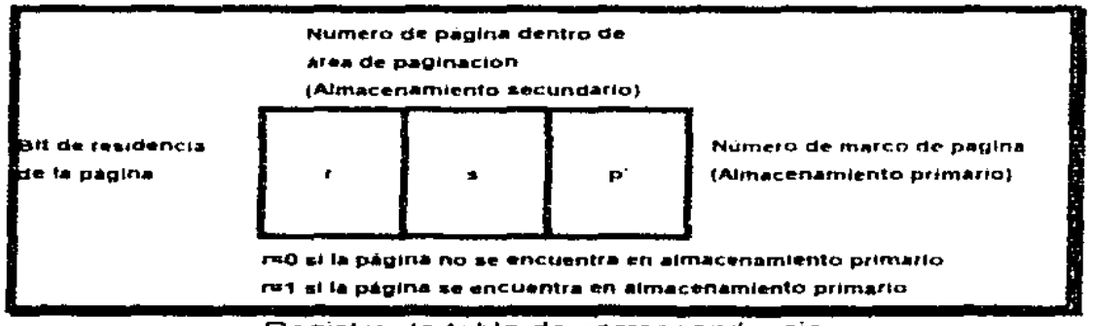


Almacenamiento real dividido en marcos de página

Cuando se realiza la traducción dinámica de direcciones en un sistema con paginación los procesos en ejecución hacen referencia a una dirección virtual v (v=p,d), esta dirección es tratada por un mecanismo especial de correspondencia de páginas. Este mecanismo busca la página p en una tabla de correspondencia, es decir, la tabla de correspondencia del proceso que requiere la dirección, para determinar en que marco de página se encuentra la página p', finalmente la dirección completa se obtiene sumando el desplazamiento d a la dirección de inicio de p'. Cabe mencionar que p' no es una dirección, ya que es el número de un marco de página en el almacenamiento real.

Es importante señalar que no se ha mencionado que sucede cuando la página solicitada no se encuentra en el almacenamiento primario. Para poder manejar una situación en la cual la página solicitada no se encuentra en el almacenamiento real se debe agregar un bit r a la tabla de correspondencia para indicar si la página se encuentra o no en el almacenamiento primario, este bit se le conoce comúnmente como bit de residencia, adicionalmente (en caso de que no se encuentre dicha página en el almacenamiento primario) se debe tener la localización en el disco (área de paginación) de la página solicitada, por lo cual será necesario otro parámetro más que denominaremos s, el cual indicará el número de página solicitada dentro del área de paginación

Así la correspondencia de direcciones queda como sigue:



Registro de tabla de correspondencia

Segmentación

En los sistemas con segmentación se elimina la restricción de que los programas en ejecución en conjunción con sus datos se tengan que ejecutar en un bloque de localidades contiguas en el almacenamiento primario, así, estos programas en ejecución pueden ocupar múltiples bloques separados en el almacenamiento primario. En la segmentación los bloques no tienen que ser del mismo tamaño, pero si deben estar formados por localidades contiguas (segmentos), pero no es forzoso que distintos segmentos del mismo programa en ejecución se encuentren en localidades de memoria adyacentes.

Debido a esta libertad en el tamaño de los bloques resulta ahora ineficiente la simple utilización de un par de registros límites, por lo cual se deben elaborar esquemas de protección de memoria que garanticen la integridad de los procesos y sus datos. Uno de los esquemas de protección de programas y datos es el de inclusión de claves de protección del almacenamiento.

Al igual que en la paginación en los sistemas con almacenamiento virtual por segmentación es necesario un par ordenado para realizar las solicitudes de almacenamiento, este par ordenado es v=(s,d), donde **s** es el número de segmento en el almacenamiento virtual, y **d** es el desplazamiento a partir del inicio del segmento.

Número de segmento	Desplazamiento
s	d

Dirección Virtual **v=(s,d)**

Dirección virtual en segmentación

Un proceso solamente puede ejecutarse si su segmento activo se encuentra en el almacenamiento primario. Al igual que en la paginación los segmentos se recuperan del almacenamiento secundario como unidades completas, es decir, cada vez que un segmento no se encuentra en memoria primaria éste es requerido del almacenamiento secundario recuperándose en su totalidad. Es importante destacar que todo el segmento debe colocarse en localidades del almacenamiento primario contiguas. Este segmento que es recuperado del almacenamiento secundario puede ser colocado en cualquier espacio del almacenamiento primario que tenga el suficiente tamaño para contenerlo. Las políticas de administración de los segmentos que ingresan al sistema son las mismas que en los sistemas con particiones variables, las políticas más utilizadas son las de primer y mejor ajuste.
En la segmentación la traducción dinámica de direcciones se realiza de la siguiente forma:

a) Un proceso en ejecución hace referencia a una dirección del almacenamiento virtual, v=(s,d).
b) Posteriormente se localiza el inicio de la tabla (arreglo) de correspondencia, esta dirección la denominaremos **b[ ]**.
c) Se desplazan **s** registros a partir del inicio **b[ ] (b[s])** de la tabla de correspondencia.
c) Se obtiene la dirección del segmento **s'** que se encuentra en almacenamiento real.
e) Se suma el desplazamiento **d** a la dirección del segmento **s'** para obtener la dirección real deseada.

Aquí se puede ver claramente que el proceso es idéntico al realizado para la paginación, con la excepción de que los segmentos son de tamaño variable

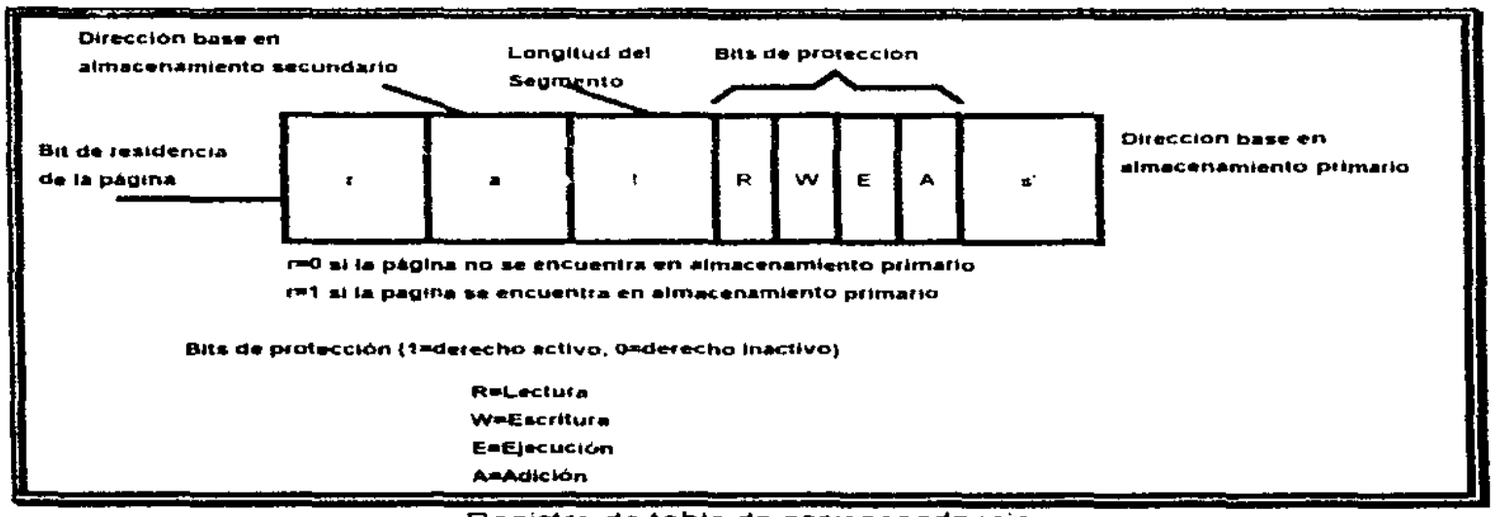
Protección de los segmentos

Como se indicó anteriormente es importante evitar accesos prohibidos a los segmentos, por ésto, se diseñó un sofisticado control de acceso para los mencionados segmentos. El control de acceso asigna a cada proceso determinados derechos en cada uno de los segmentos, estos derechos pueden ser:

Abreviatura	Acceso	Detalle
R	Lectura	Segmento leible
W	Escritura	Segmento escribible
E	Ejecución	Segmento ejecutable
A	Adición	Segmento adicionable

Si un proceso tiene derecho de lectura para el segmento, el proceso puede leer la totalidad de información contenida en el segmento.
Si un proceso tiene derecho de escritura para el segmento, el proceso puede escribir en el segmento, pudiendo inclusive cambiarlo en su totalidad.
Si un proceso tiene derecho de ejecución para el segmento, el proceso puede ejecutar el segmento como cualquier programa (subrutina). Cabe mencionar que el derecho de ejecución se encuentra negado para los segmentos de datos.
Si un proceso tiene derecho de adición para el segmento, el proceso puede agregar información al final del segmento, pero de ninguna manera puede modificar los datos que el segmento contenía antes de agregar información.
Estos derechos de acceso se pueden combinar para establecer reglas de seguridad confiables.
Para establecer una correspondencia adecuada con las direcciones virtuales y las direcciones reales, los sistemas con segmentación tienen consideraciones adicionales, es decir, cada registro de la tabla de correspondencia contiene más información, ya que al menos se requieren los siguientes campos:

Bit de residencia, el cual indica si el segmento se encuentra en almacenamiento primario o no.
Dirección de almacenamiento secundario, la cual indica en que parte del disco se encuentra el segmento.
Longitud del segmento, la cual indica la longitud del segmento.
Bits de protección (4 bits), los cuales indican los derechos de acceso al segmento.
Dirección base del almacenamiento primario, la cual indica en que parte del almacenamiento primario se encuentra el inicio del segmento.



Registro de tabla de correspondencia

◆ CAPITULO IV.- ADMINISTRACION DE DISPOSITIVOS

Concepto de Dispositivo

Un dispositivo es en el concepto estricto, el nombre que recibe un componente hardware, físico en un sistema. Se puede decir que el nombre del dispositivo es la parte abstracta del mismo, este nombre bien definido en el sistema operativo dará las características necesarias para hacerlo útil en su contraparte real. Es tan necesario el uso de dispositivos en un sistema que la tarea de definición de los mismos radica en el propio diseño del sistema operativo, localizando su administración en el propio núcleo del sistema operativo. Los dispositivos más conocidos dentro de un sistema son el monitor, teclado, disco, la impresora, ratón, pluma electrónica, unidad de CD-ROM, etc., todos estos reciben el nombre de dispositivos periféricos

Manejadores de Dispositivos y papel que desempeñan

Los manejadores de dispositivos son los que le "dicen" al sistema como comunicarse con los diferentes dispositivos periféricos; los dispositivos periféricos y los manejadores de dispositivos deberán de tener la adecuada comunicación y compatibilidad para que resulten útiles al sistema.
Existen algunos manejadores de dispositivos que pueden redireccionar y hacer referencia a otros manejadores de dispositivos, sin tener que pasar por el núcleo.
En realidad un manejador de dispositivo es código escrito para determinado dispositivo, este se puede ser compartido entre las diferentes aplicaciones en ejecución para una mejor utilización de los mismos.
En algunos sistemas, para evitar que el código de los administradores de dispositivos sea transgredido se protege de los usuarios y de los programas que lo utilizan; si se elabora bien, se podrán adicionar o remover dispositivos del sistema sin tener que cambiar todo el sistema. Así, un sistema operativo representaría una librería compartida general y privilegiada. Pero la tendencia de los sistemas operativos modernos ubica a los manejadores de dispositivos en el núcleo como se verá más adelante.

Un manejador de dispositivo se describe también como el software que opera un dispositivo, no solo se refiere a "dispositivos hardware" o dispositivos periféricos, si así fuera, se excluirían muchos manejadores de "dispositivos software"; estos últimos son usados para imitar la interfaz de un manejador de dispositivo y simular los efectos del "pseudodispositivo", como ejemplo se tienen la emulación de terminales TTY (Término heredado de las viejos computadoras centrales, TTY es sinónimo de teletipo; pantalla con teclado, anteriormente solo era un teclado similar al de una máquina de escribir, con una impresora integrada) bajo una sesión telnet (Programa que permite la conexión a un sistema Unix mediante la emulación de una terminal).

El núcleo es la parte del sistema operativo que se ejecutará siempre en modo superviso(modo supervisor es el modo del procesador en el que se permite la ejecución de todo tipo de instrucciones, incluyendo las instrucciones de entrada salida, al contrario del modo usuario, en donde solo se permite un conjunto limitado de instrucciones), esto es para proteger tanto las estructuras de datos críticas como los registros de dispositivos de los programas de los usuarios.

Un manejador es un conjunto de funciones, compiladas dentro del núcleo. El núcleo se conecta al mecanismo de interfaz de dichos manejadores para permitir la utilización de los dispositivos. Frecuentemente, la parte de un dispositivo con la cual la computadora actúa directamente es llamada "controlador", este controlador es el que recibe las órdenes del sistema operativo para luego operar el dispositivo.

Si los dispositivos pueden trabajar independientemente durante un rango determinado de operaciones, son llamados dispositivos "esclavos", ya que comparten responsabilidad con el controlador principal denominado generalmente "maestro", esto se realiza principalmente en las operaciones de entrada y salida.

El papel que realizan los manejadores de dispositivos para los diferentes sistemas operativos puede variar en responsabilidad y forma. Por ejemplo, para MS-DOS, se tienen manejadores implementados en el BIOS o como archivos que en el proceso de inicialización de alguna manera fueron cargados, como por ejemplo ANSI.SYS; o bien están residentes en memoria, como por ejemplo, el manejador del ratón.

En un sistema MAC (Acrónimo de Macintosh) los manejadores de dispositivos corren fuera del núcleo en procesos separados, como entidades completamente separadas del sistema operativo, lo cual permite que se modifiquen dinámicamente, es decir, permiten que se conecten y desconecten dispositivos sin tener que reinicializar el sistema.

Generalmente los administradores de dispositivos ya sea que corran fuera del núcleo o que estén incluidos en el mismo son los responsables de todos los aspectos de reconocimiento, inicialización, operación y recuperación de errores de los dispositivos conectados al sistema de cómputo.

La constante relación entre el sistema y la arquitectura es una de las razones por las que el código de algún manejador tiene que reinventarse (Reescribirse) la mayoría de las veces. El ejemplo tradicional para este caso es el de los sistemas UNIX, con manejadores para la misma arquitectura que requieren ser reescritos si se quieren portar a otra versión de UNIX de otro proveedor como SVR3, SVR4, MACH, BSD, etc.

Concepto de Interrupción

Para describir el concepto de interrupción, vayamos a los primeros sistemas, en donde los dispositivos de entrada y salida como lo eran las cintas, las terminales remotas y las tarjetas perforadas por nombrar sólo algunos, realizaban la tarea de una manera total, es decir, de principio a fin, si en ese momento el procesador del sistema estaba inactivo, era imposible asignarle otra tarea mientras la tarea o el proceso del dispositivo de entrada y salida estaba en ejecución, además el sistema operativo debería de estar en un estado busy-wait (Ocupado en espera) hasta que la liberación del dispositivo fuera hecha, es en esa deficiencia donde nacerı las interrupciones.

Uso de interrupciones dentro de un sistema

- Como se mencionó anteriormente, los dispositivos actuales mantienen un controlador que puede ser configurado como maestro o como esclavo dependiendo la tarea, haciendo que el dispositivo trabaje autónomamente.

- El CPU envía una serie de comandos o interrupciones al dispositivo de entrada y salida, y posteriormente se dedica a realizar otras funciones.

- Si el dispositivo esta ocupado, esta señal es tomada por el controlador que espera que el dispositivo finalice su tarea, al finalizar la tarea, el dispositivo envía una señal de interrupción al CPU y este deja todas sus funciones pendientes, guardando su configuración y atendiendo los requerimientos que hace el dispositivo. Es importante hacer notar que el dispositivo emite una interrupción cuando termina alguna operación debido a que generalmente estos dispositivos son de baja velocidad comparados con la CPU, por lo cual atender al dispositivo en el momento que termina alguna operación redunda en un beneficio en el desempeño general del sistema.

Dado que estas señales entre los dispositivos y el procesador o procesadores usados es constante y muy utilizada, es necesario tener una consideración especial para su manejo, surgen entonces algunos problemas que se tienen que resolver, por ejemplo, es necesario evitar que una interrupción destruya el trabajo que ella misma interrumpió, además, se debe controlar la situación cuando se generan dos interrupciones al mismo tiempo, evitando que provoquen conflictos.

Los dos mecanismos básicos usados para la comunicación entre el sistema operativo y el hardware son:

- Interrupciones de dispositivo: que llevan el trayecto hardware -> sistema operativo
- Registro de dispositivos: que llevan el trayecto sistema operativo -> hardware

Cuando una computadora es ensamblada y configurada, cada dispositivo es reconocido por un "número de interrupción" especifico, también llamado nivel, con el fin de emitir las señales al procesador. Cuando sea posible, este número deberá ser único para cada dispositivo, pero algunos sistemas (Como por ejemplo DOS y las PC's) no lo hacen, acarreando con esto complicaciones en su manejo (Esto es debido principalmente a que en las PC's se le deja la parte de asignación de interrupciones a los usuarios, con lo cual se producen confusiones y errores en la asignación de interrupciones).

Cuando al dispositivo que le fue dado el número de la interrupción "i" genera una interrupción, el CPU automáticamente salta a la función denominada "manejador de interrupción", que se encuentra en la i-ésima posición en el "vector de interrupciones" (O tabla de interrupciones). El vector de interrupciones es una tabla de apuntadores (Direcciones) a las funciones necesarias para llevar a cabo la interrupción, el cual generalmente se encuentra localizado en un espacio conocido en el área de datos del núcleo. Este vector contiene las direcciones del manejador de interrupciones de dispositivos individuales, por ejemplo:

Vector de interrupciones		
0	0x2ff00000	teclado
1	0xfc0000b0	ratón
2	0x2df00000	Reloj
3	0x2ffc6810	disco 1

En este ejemplo, el teclado se ha configurado para generar la interrupción 0, y el manejador del teclado se localiza en la dirección 0x2ff00000. En otras palabras, una función con un nombre como "HandleKeyboardInterrupt()" se localiza en la dirección 0x2ff00000. Cuando el hardware del teclado genera la interrupción 0 y esta es recibida por la CPU, este saltará a la función HandleKeyboardInterrupt(), después de haberla ubicado en el vector de interrupciones.

El proceso del manejo de interrupciones lleva consigo varios procedimientos:

1. Salvar el estado crítico de la CPU (En el más mínimo detalle, esto es a nivel del hardware)
2. El contador del programa apunta hacia la interrupción requerida en el vector de interrupciones (PC=vector_de_interrupción[i])
3. Salvar el estado del CPU (Por ejemplo, registros y estado general del proceso que se encuentra en ejecución antes de que ocurra la interrupción) ya que los registros serán modificados por el manejador de interrupción; y además si es necesario:
    3a. Inhabilitar otras interrupciones (Generalmente se inhabilitan todas las interrupciones)
4. Manejar la interrupción (Ejecutar código de la interrupción)

5. Restaurar el estado del CPU antes de la ejecución de la interrupción; y además si es necesario:
   5a. Habilitar otras interrupciones (Generalmente se habilitan todas las interrupciones)
6. Restaurar el estado del CPU en cuanto al hardware y regresar a la rutina original

Los pasos 1, 2 y 6 se hacen en el nivel de hardware; los pasos 3, 4 y 5 se hacen en el manejador de interrupción a nivel software.
Como parte de la inicialización del sistema operativo es de vital importancia inicializar el vector de interrupciones, por ejemplo:

```
vector_de_interrupción[0]= HandleKeyboardInterrupt;
vector_de_interrupción[1]= HandleMouseInterrupt;
Etc.
```

Alguna de las rutinas del núcleo del sistema operativo que maneja las interrupciones se podrían ver de la siguiente manera:

```
HandleKeyboardInterrupt()
{
/* salvar registros y realizar operaciones necesarias, se recomienda ensamblador
para optimar esta parte */
asm(...);
/* realizar todo lo que el dispositivo necesita*/
...
/* restaurar registros*/
asm(...);
}
```

Tipos de Dispositivos

Dentro de las clasificaciones de los dispositivos, se tienen las siguientes:

Dispositivos independientes

Son capaces de operar con una gran variedad de periféricos hardware, por ejemplo: un generador de reportes que puede imprimir en diferentes clases de impresoras es un dispositivo independiente.
Los programas de aplicaciones que usan convenientemente manejadores de dispositivos estándares e interfaces públicas de sistemas operativos, y así evitan usar características poco usuales de un dispositivo particular, también son dispositivos independientes.
Como se podrá observar, los dispositivos independientes entran dentro de la categoría de hardware lógico pues corresponden a programas que administran

dispositivos. En este contexto también podemos hablar de lo que se conoce como dependencia de la versión que es una causa por la cual los dispositivos independientes llegan a fallar, la dependencia de la versión consiste en tomar ventaja de características no publicadas o no documentadas de algún sistema operativo o programa, un ejemplo muy conocido de dependencia de la versión surgió cuando Microsoft liberó la versión 4 de MS-DOS, en esta versión, ciertas direcciones en el núcleo eran diferentes a las utilizadas en la versión 3. Aunque el conocimiento de estas direcciones no era parte de la interfaz legal entre los programas y las aplicaciones, muchos productos no pudieron trabajar bajo esta nueva versión ya que habían tomado ventaja usando la parte no oficial y no documentada de la versión anterior. Considerando esto muchos dispositivos pueden fallar en una nueva versión de sistema operativo, pero esta dependencia en ocasiones puede resultar mala o buena dependiendo su uso:

Primeramente puede resultar mala porque complica la portabilidad de software a otras plataformas, peor aún, es un obstáculo para mantener la compatibilidad con futuras versiones del mismo sistema operativo, compilador, hardware y software.

Esta dependencia de implantación resulta en ocasiones innecesaria, por las funciones propias del software, sin embargo muchos desarrolladores, especialmente vendedores de productos de competencia, en ocasiones les resulta ideal explotar la dependencia por estas razones:

- Hacer programas que corran más rápido que lo que se permite con el uso legal de la herramienta usada.
- Hacer trampa en las limitantes de una opción no propia de un lenguaje de programación.
- Engañar al sistema operativo o a la aplicación, haciendo algo para lo cual no fue diseñado para hacer.

Para crear manejadores de dispositivos en este medio, se puede minimizar la dependencia tanto de versión como de implantación, creando un pequeño conjunto de módulos que puedan ser modificados por otros ambientes operativos (Encapsulación y código portable).

Dispositivos de carácter

Operan en la base de transferir carácter por carácter, el teclado es un ejemplo de estos dispositivos. La mayoría de las veces no utiliza un buffer (almacenamiento temporal para mejora en el rendimiento) pues mantiene uno propio dentro del hardware, adicionalmente no son dispositivos de acceso aleatorio, por el contrario, utilizan funciones del tipo read() y write() y no finalizan la rutina hasta que el proceso termina.

Dispositivos de bloque

Operan en la base de transferir por bloques de información, por ejemplo, los discos y discos lógicos RAM representan a este tipo de dispositivos. A diferencia de los dispositivos de carácter estos son de acceso aleatorio. Otra de las características de estos dispositivos es que no implementan las funciones read() y write(), en vez de esto se tiene una función que históricamente se le ha llamado "rutina de estrategia". La lectura y escritura son realizadas desde el buffer por medio de 3 funciones genéricas que permiten reconocer cuando leer y escribir bloques solicitados, además de ejecutar en segundo plano una función asincrona que solicita esta rutina de estrategia para planear leer algo que no ha sido solicitado, en espera de que se requiera más tarde. De alguna manera predicen que información va a ser solicitada en el futuro (Leyendo más bloques de los solicitados en cada operación), con lo cual si los bloques traídos con anticipación son solicitados ya no es necesaria una operación de entrada y salida.

Dispositivos Estándar

Los dispositivos estándar son aquellos con los cuales la computadora actúa más comúnmente, es decir, son incluidos en casi todas las computadoras, ellos son el disco, el video, el teclado pudiendo considerar además el uso de un dispositivo de cronómetro como es el reloj y otros no menos importantes como lo son el ratón y un puerto para impresora. Nos enfocaremos a mencionar los 4 primeros dispositivos.

Disco

Uno de los dispositivos estándar más utilizados es el disco, este es de gran utilidad debido a que puede almacenar grandes cantidades de información, a continuación se describe la estructura típica de un disco:

Platos.- son discos hechos de material magnético, como el de cintas pero con mayor densidad.

Superficie.- es uno de los lados del plato.

Pista.- es un círculo concéntrico en la superficie.

Cilindro.- es un conjunto de pistas con el mismo diámetro ubicadas en diferentes platos.

Sector.- es la sección de una pista y es de tamaño fijo

Cluster.- es la agrupación de 2 o más sectores, dependiendo del tamaño en bytes del sector

El disco contiene un motor que gira a alta velocidad, comúnmente a 3600 r.p.m. (actualmente 9600 o más), aunque esta puede variar, cada disco puede contener de 2 a 20 platos y las cabezas lectoras pueden ser fijas o con movimiento, aunque estas últimas son las más utilizadas. Los tamaños más utilizados de sector son los que están en el rango de 512 a 8192 bytes. Los sectores son leídos o escritos de manera individual o en grupos adyacentes, por medio del movimiento de la cabeza lectora, buscando la pista correcta y esperando a que el disco gire a la posición correcta sobre el sector deseado
El tiempo de búsqueda de un sector es alto (5 a 50 milisegundos), debido al proceso mecánico que se realiza al girar el disco. Existe un tiempo denominado retraso rotacional que depende de la rapidez con que el disco gira y la localidad donde se encuentra el sector deseado, este tiempo en discos de alta tecnología es de 0 a 16 milisegundos. El tamaño de los discos ha sido reducido, y ha llevado a efecto que el costo baje y la capacidad de almacenamiento aumente.

Los CD-ROM han venido a incrementar la popularidad de los equipos multimedia, este dispositivo presenta muchas más características interesantes que considerar:

- Tiempos de búsqueda más pequeños.
- Promedio de transmisión de datos inferior (por ahora, velocidad máxima 16X, es decir, 1.6 MB por segundo).
- Mucho mayor capacidad (650 MB o más)

Se debe de considerar que en un disco la lectura y escritura de la información se realiza por bloques y no por simples bytes, haciendo comparaciones con otros dispositivos podemos sacar las siguientes características:

Eficiencia de almacenamiento.- Cintas, comúnmente utilizadas para hacer respaldos hoy en día, aunque fueron muy utilizadas antes de la venida del disco, se tiene que en ellas se pueden almacenar de 1600 hasta 6250 bits/pulgada en los equipos más antiguos, es decir, un registro de 80 bytes usaría 0.05 pulgadas, el espacio entre registros o bloques de información es de 0.6 pulgadas, como se puede apreciar es un gran desperdicio de espacio por lo que la eficiencia resulta casi nula.

En discos con un tamaño de sector de 512 bytes, casi el 50% de la capacidad total del disco contiene información acerca del inicio, fin y contenido de los sectores en un área denominada encabezado del sector, que es información de cientos de bits por lo que esta clase de discos no es la mejor y por lo tanto su eficiencia de almacenamiento es bastante baja, debido básicamente al alto

desperdicio. Hoy en día los discos con sectores de mayor tamaño son los más eficientes en almacenamiento. Hay que notar que el encabezado de cada sector es información vital, no se puede prescindir de ella y es la que marca la diferencia entre un disco formateado y uno no formateado.

Eficiencia de acceso.- En discos es muy usual una espera de 12-25 milisegundos antes de empezar una transferencia de lectura o escritura, el tiempo de transferencia típica es de solo 1 microsegundo/byte, es decir, para transferir 1000 bytes solo se requiere de 1 milisegundos, esto llevaría un total de 13-26 milisegundos para leer los datos, con esto se aprecia que el tiempo de búsqueda es demasiado y podría ser un punto de investigación para eliminar o reducir a lo mínimo este tiempo que en términos de sistemas es un gasto excesivo.

Una manera de reducir en tiempo el proceso de transferencia de datos en un disco es intentar poner la información en bloques grandes de manera secuencial, pero es importante destacar que entre mayor sea el tamaño de bloque mayor será el desperdicio del último bloque de datos.

Las operaciones realizadas en un disco son las siguientes:

- Leer_Sector(dirección_en_disco, buffer)
- Escribir_sector(dirección_en_disco, buffer)

Ambas operaciones funcionan de manera similar

La dirección_en_disco incluye o implica considerar la superficie, sector y cilindro del disco.
El buffer es una localidad en el área de memoria comúnmente accesada por los canales de acceso directo a memoria DMA (Acceso directo a memoria, en inglés Direct Memory Access).

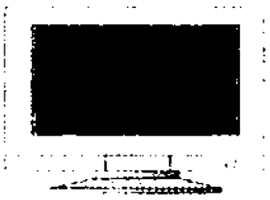
Las operaciones de lectura y escritura como ya se explicó anteriormente llevan un tiempo de búsqueda en el cual se realiza los siguientes pasos:

- Girar el disco sobre su eje manteniendo un cilindro de revolución este tiempo oscila entre 10 y 100 milisegundos
- Seleccionar la superficie y cabeza lectora apropiada para realizar la lectura o escritura, con un tiempo de 0 milisegundos
- Es en este momento cuando ocurre el retraso rotacional:
- Esperar a que el sector aparezca bajo la cabeza lectora (3600 r.p.m.) esto es de 0 a 16 milisegundos
- Por último hacer la transferencia de datos (1000 bytes cada milisegundo)

Después de iniciadas estas operaciones se deberá tener cuidado de que el sistema operativo no se bloqueé por otros procesos hasta que el procesador sea

liberado, es decir, hasta que se haya emitido la solicitud de entrada y salida; más aún se deben de utilizar los canales DMA y las interrupciones para realizar este trabajo en segundo plano, es decir, mientras el procesador se encuentra realizando algún otro trabajo útil.

Video



El video es considerado el dispositivo de salida principal, a continuación se explica como se realizan las funciones básicas para su utilización.
Existen muchos estándares de video para trabajar en sistemas diferentes, el subsistema de video es independiente del sistema operativo que se utiliza, sin embargo las funciones básicas que radican en el núcleo del sistema hacen mención del subsistema de video; nos basaremos en el uso de MS-DOS para explicar estas rutinas.

El subsistema de video es el encargado de controlar y manejar las señales que acepta el monitor, todo esto se encuentra comúnmente en una tarjeta electrónica llamada adaptador de video. Este adaptador de video es un dispositivo del tipo de mapeo de memoria, el cual mantiene un bloque de memoria específico desde donde lee el contenido que después transforma en caracteres o dibujos en la pantalla del monitor. Este bloque de memoria se denomina buffer de video.

El buffer de video está directamente conectado a los circuitos del monitor, aunque es parte de la memoria RAM en un espacio reservado de esta. En años anteriores este espacio reservado era fijo, hoy en dia es variable.

Almacenando y desplegando datos en el video

Un sistema basado en PC manipula el video desde el circuito controlador del tubo de rayos catódico 6845 (CRTC). Este circuito realiza importantes tareas que los desarrolladores no tienen acceso a ellas:

- Detección de señales de dispositivos externos (Light-pen)
- Incrementar contador de dirección del buffer de video
- Sincronizar despliegue y temporizador
- Seleccionar el buffer del video
- Determinar el tamaño y la localización del cursor de hardware

El diseño del sistema es en concepto muy simple. Un video de una PC es un dispositivo que hace una transformación de lo que existe en memoria (Mapeador

de memoria). El buffer de video almacena información que aparece en el monitor La dirección de inicio del buffer es variable, dependiendo del tipo de video que se usa, el modo al que esta configurado (Texto o gráfico. cga, vga. svga. etc.) y la cantidad de memoria asignada para la pantalla. Además el buffer del video es del tipo puerto dual, es decir, tanto el CPU como los circuitos del monitor pueden accesar datos al mismo tiempo.

Los adaptadores de video generalmente contienen diferentes cantidades de memoria que va de los 256K hasta los 2 MB y hoy en dia existen adaptadores que contienen hasta 4 MB o más MB. Ya que la necesidad de los datos de requerir toda la memoria no es muy común, algunos adaptadores pueden controlar más de una pantalla a la vez. Las pantallas a desplegar o páginas son la representación en pantalla de lo que hay en memoria y estas páginas varian dependiendo del adaptador que se use

Para todos los adaptadores, independientemente de cual se use, el número de páginas disponibles para el modo texto es el resultado de 2 bytes por posición en la pantalla, con 80 caracteres de texto por linea con 25 lineas; esto es 2*80*25, es decir, 4 KB por pantalla completa. Estos 2 bytes contienen la información tanto de los pixeles que van a ser prendidos como sus atributos.

El circuito CRTC, independientemente del sistema de operativo que se este usando, hace una búsqueda en el área de memoria de video. y basado en la información almacenada allí, y un código en memoria ROM para caracteres que compara y actualiza la pantalla. Esta actualización es el resultado de un rayo de electrones que prende o apaga los pequeños puntos que conforman a la pantalla (Pixeles). La ruta de este rayo es de izquierda a derecha y de arriba a abajo (vista del usuario) sobre la pantalla completa.

Para poder dar una imagen uniforme, la pantalla debe de ser refrescada, es decir, el rayo de electrones hará un ciclo completo en la pantalla en promedio 60 veces por segundo. Al final de cada línea, el rayo deberá moverse de derecha a izquierda, a este periodo de tiempo se le llama intervalo de retraso horizontal (HRI). Igualmente, después de que el rayo completa un ciclo, este deberá moverse del borde inferior derecho de la pantalla hasta el borde superior izquierdo de la misma para poder iniciar un nuevo ciclo, este periodo de tiempo es llamado intervalo de retraso vertical (VRI). Durante estos lapsos el rayo de electrones es apagado y no se escribe en la pantalla. Como ya se mencionó anteriormente, el buffer de video es un puerto dual que se puede accesar al mismo tiempo, tanto para leer como para escribir. El retraso vertical es un periodo de tiempo muy importante para poder actualizar datos, ya que en ese periodo de tiempo no se escribe en la pantalla y se evita el conflicto que sucederia si se tratara de leer algo que se está actualizando en el mismo momento en el buffer de video (muy común en los adaptadores de video CGA. efecto de nieve).

Existen tres maneras de accesar el buffer del video:

1. Desde llamadas a funciones DOS. Este método es el más fácil pero el más lento de accesar, con la aparición de la Versión 2.0 del sistema operativo se incluyó un manejador ANSI.SYS que permitía el control del video desde secuencias de código de control, es decir, la solicitud de los servicios de despliegue se hacía de una manera más rápida.
2. Desde llamadas a funciones del BIOS. Este es el método más compatible de accesar la pantalla mucho más rápido que el DOS. Muchos sistemas aunque otros no, son compatibles con este método de accesar la pantalla. Desde estas llamadas al BIOS están disponibles efectos visuales y gráficos que de otra manera en las funciones del DOS no se pueden accesar.
3. Directamente desde el nivel de hardware. Este método es incompatible por la gran variedad de diferencias que existen en el hardware de los sistemas, los programas que generalmente usan estos métodos son en realidad incompatibles cuando se decide instalarlos en otro hardware, además este método no es compatible en sistemas multiusuarios o multitareas. La razón y ventaja de su frecuente uso es su rápida operación en el momento de su despliegue.

Esta última opción es dejada como último recurso para la realización de programas, en realidad las dos primeras opciones son las más utilizadas a nivel programación, tanto a nivel usuario como multiusuario y multitareas.

En el mercado existen una gran variedad de tipos de monitores:

Monitores Monocromáticos de manejo directo.- Son los primeros monitores que aparecen, producto de la serie de grandes computadoras centrales de IBM 3270, son monitores que pueden ser controlados con una adaptador MDA, HGA y hasta un EGA emulando un adaptador monocromático; manejan texto de alta resolución y nivel gráfico de caracteres de color verde, blanco o ámbar. Aun son utilizados en oficinas que mantienen terminales.

Monitores Monocromáticos compuestos.- Estos monitores pueden manejarse desde un adaptador CGA y pueden desplegar gráficos sin color, son baratos y son de color ámbar, verde y blanco. Ya no son muy comunes hoy en día.

Monitores a color compuestos y de TV.- Son monitores que utilizan una simple señal combinada que produce colores pero con limitantes, ya que su resolución es demasiado pobre con una configuración de 80 columnas; por lo que para poder obtener un resultado satisfactorio es necesario utilizar solo 40 columnas haciendo la imagen legible pero muy tosca.

Monitores RGB.- Son los monitores que se consideraban hasta hace pocos años como los mejores, su calidad tanto en texto como en gráficos es alta. Su

característica radica en que mantienen 3 líneas diferentes de señal con los tres colores primarios: rojo, verde y azul y combinando sus intensidades para lograr hasta 256 o más tonos de colores.

Monitores de frecuencia variable (Multisincrónicos).- Es una extensión de los monitores RGB pero con tonalidades que van más allá de los 16 millones de colores, se dice que son de frecuencia variable porque pueden emular cualquier otro tipo de monitor manejando cualquier tipo de adaptador. Hoy en día son los más utilizados en la industria de la computación.

Hay que considerar que cuando se programen las rutinas para manejo del monitor es conveniente ver el grado de compatibilidad que debemos mantener. Si el sistema operativo puede realizar todas las funciones específicas en un adaptador diferente al que se consideró es un sistema robusto que evitará problemas, pero hará incrementar el número de líneas de código del núcleo. Sin embargo el diseñador es el que tiene la última palabra y el que decide a fin de cuentas cual será la mejor opción.

Teclado

El teclado es considerado el dispositivo de entrada principal; su estructura resulta ser más sofisticada de lo que parece, en realidad es una pieza avanzada de ingeniería.
Trabaja con funciones de bajo nivel en el núcleo, cuenta con un buffer de caracteres y utiliza interrupciones que realizan la conversión del código de la tecla pulsada en el carácter deseado, resultando todo esto invisible al usuario.
Para la mayoría de los programadores el teclado es el dispositivo de entrada de la computadora más familiar y conocido pero es quizá el menos entendido, de hecho existe muy poca información técnica acerca de éste.

En vez de describir el hardware se tratará de explicar la secuencia de eventos que suceden; se describirán estos eventos en base a uno de los sistemas operativos más usados MS-DOS y los servicios esenciales del sistema básico de entrada y salida (BIOS).

Cuando el teclado se utiliza genera una señal que viaja a través del cable que lo une a la computadora, esta señal es interpretada como un único número para cada pulsación de tecla, estos números son conocidos como "códigos analizadores" (scan codes).

Cada vez que se presiona una tecla se genera el "código analizador" que es un número de 8 bits (donde el bit más significativo es siempre cero para diferenciar entre una tecla presionada y una tecla liberada), incluso las teclas SHIFT IZQ y SHIFT DER se representan por diferente código. Estos códigos indican exactamente que tecla fue presionada.
Cuando la tecla es liberada, es generado otro "código analizador", este código es el mismo que el que se generó al apretar la tecla, excepto que el bit más significativo es puesto a 1, es decir, se le suman 80H al "código analizador", con esto se le indica al BIOS que la tecla ha sido liberada.
Cuando se mantiene una tecla presionada por más de medio segundo, el teclado genera una secuencia de "código analizador" repetitiva que corresponden a la tecla presionada, en la misma razón de tiempo. El BIOS entonces interpreta esto como que la tecla esta siendo presionada y retenida (de ninguna manera presionada repetidamente) y por lo tanto no se genera código de liberación para esta acción.
El teclado solo reconoce las teclas y no los caracteres sobre ellas que se presionan, y de esta manera solo define el código analizador correspondiente. El BIOS en ROM es el que interpreta y determina que carácter ASCII le corresponde a la tecla presionada.
Siempre que es presionada una tecla, el teclado no solo genera el código analizador, además de esto genera una interrupción (09H), que le indica al BIOS en ROM que una tecla ha sido presionada. El control del sistema es entonces transferido momentáneamente al manejador de la interrupción para determinar que tecla fue presionada. La rutina de servicio lee el código analizador y lo convierte a un código de 16 bits que representa la pulsación de la tecla, este código contiene en el byte menos significativo el valor ASCII de la tecla presionada y en el byte más significativo contiene el código analizador.

Las teclas especiales como son las de funciones y del teclado numérico contienen en el byte menos significativo un cero, indicando que no es un carácter ASCII y que debe ser procesada de una manera diferente.
Cuando el manejador de interrupción finaliza el proceso, fija este código de 16 bits en el buffer del teclado, donde permanece hasta que un programa lo solicita.

Teclas especiales como lo son CTRL+BREAK o CTRL+ALT+DEL son procesadas por otras interrupciones del BIOS.

El buffer del teclado tiene una longitud para almacenar 15 caracteres, si el buffer se llena (sucede cuando la computadora esta ocupada en otra tarea) utiliza un "beep" para indicar que el buffer esta lleno y descarta la pulsación.
Cuando se programan las rutinas que usará este dispositivo se debe de especificar que tipo de teclado se utilizará, pues al utilizar un teclado diferente al que se programó el sistema operativo se encontrarán graves errores difíciles de superar, de los cuales el error más frecuente es el de teclear un caracter y obtener otro.

Reloj

El reloj aunque no entra en la característica de dispositivo de bloque o de carácter se considera un manejador de dispositivo, pues gracias a sus funciones hace que el sistema se mantenga sincronizado y trabajando de manera eficiente. Su uso va desde mantener la hora y fecha exactas como cualquier reloj, hasta asignar tiempos de uso del CPU para sistemas de tiempo compartido o tiempos de control a dispositivos periféricos o asignación de tiempos a las diferentes tareas en un sistema multitareas.

En las computadoras se utilizan dos tipos de relojes, el común es el que realiza un ciclo cada vez que la fuente de poder estándar de 110 ó 220 genera su propio ciclo, es decir, de 60 Hz. El otro reloj o cronómetro es un reloj de cristal de cuarzo que mantiene dos registros: un registro contador y un registro de retención de almacenamiento del valor del reloj. El funcionamiento de este reloj es relativamente sencillo: el reloj oscilará en forma indefinida, el contador de reloj disminuirá en cada oscilación su valor que inicialmente es el mismo que el del registro, cuando el contador llega a cero se produce una interrupción que le indica al CPU que el tiempo ha expirado.

Los tiempos asignados al reloj se ocupan principalmente para:

- Conservación de la hora y el día
- Asignar tiempos específicos a los procesos
- Contador para uso exclusivo del CPU
- Cronómetros guardianes para cuidar partes críticas del sistema
- Monitoreo en general y manejo estadístico
- Generación de señales de alarma

Los relojes pueden trabajar en modo no repetido que es cuando el contador llega a cero y no es vuelto a inicializar hasta que se hace a nivel software; el modo de onda cuadriforme es contrario al modo no repetido, en este modo el registro de retención es copiado en el contador automáticamente cuando este llega a cero y genera la interrupción, iniciando un nuevo tiempo de reloj; a este tipo de interrupciones periódicas se les denomina pulsaciones de reloj.

Ya que físicamente no se puede tener un reloj para cada proceso es necesario emular relojes virtuales con un solo reloj físico, esto es realizado a través de una tabla que lleva el conteo para los procesos y su tiempo total, esta tabla es creada

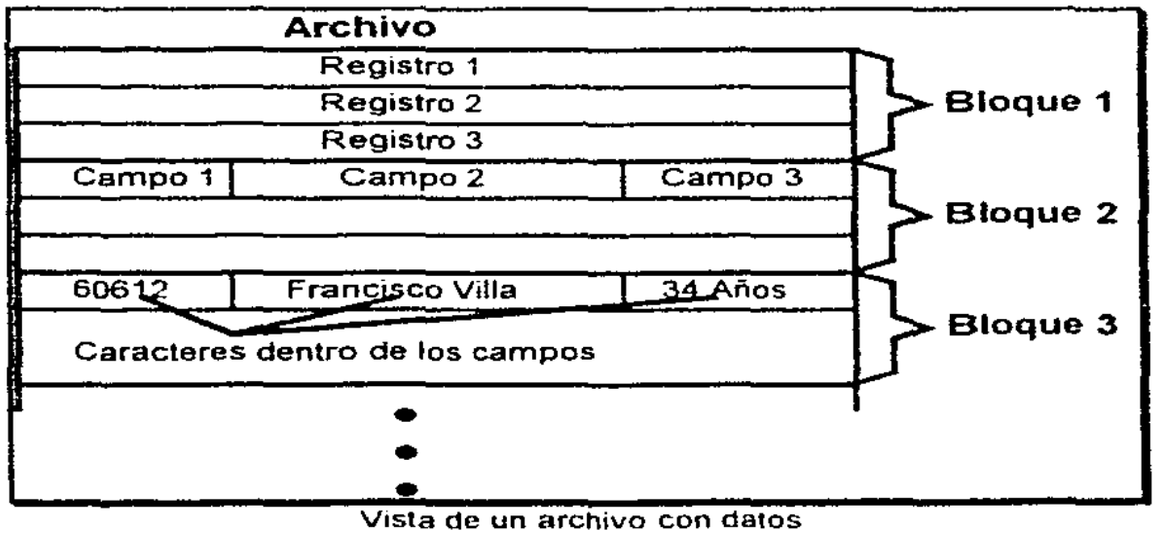
para administrar la información de tiempos de cada proceso así como su duración, cuando un proceso llega a su tiempo límite la tabla será accesada y se generará una interrupción que avise al sistema que el tiempo del proceso ha finalizado.
Los manejadores de relojes son útiles también para el acceso a dispositivos periféricos tales como impresoras, las cuales requieren supervisión. Si la operación hacia el dispositivo falla, el sistema debe de enviar una señal, esta señal solo es enviada si existe un tiempo límite controlado por los relojes que indique el tiempo de espera antes de generar una señal.
Para implementar relojes es necesario utilizar el acceso al reloj físicamente y de ahí implementar estructuras de datos que lean los ciclos que genera el reloj, estas estructuras de datos básicamente deben de contar con lo que se mencionó anteriormente: un registro contador, un límite de conteo de ciclos y una señal de interrupción al procesador.
Los relojes son una clase de dispositivos diferente a los demás pero con características propias útiles en el diseño de los sistemas operativos.

◆ CAPITULO V.- ADMINISTRACION DE ARCHIVOS

Una de las partes más importantes de un sistema operativo con la que el usuario tiene mayor contacto, exceptuando por supuesto las nuevas interfaces gráficas, es el sistema de archivos, el sistema de archivos muestra y manipula la información de un usuario de acuerdo con sus requerimientos al mismo Aquí iniciaremos indicando qué es un archivo, y porqué es útil en los sistemas de cómputo, adicionalmente mostraremos como se diseña, construye y administra un sistema de archivos.

Concepto de archivo

Un archivo es un conjunto de bloques físicos relacionados entre sí. Se debe pensar en un archivo como en una caja que puede localizarse en el trabajo o en el hogar, la cual contiene información de utilidad para nosotros Los archivos están constituidos por **bloques**. Los bloques pueden definirse como folders de archivo que están dentro de la caja. Como folders de archivo, los bloques pueden tener o no el mismo tamaño o capacidad. Dentro de cada bloque existen **registros lógicos**. Los registros lógicos son parecidos a páginas de papel dentro del folder de un archivo. Cada registro lógico se compone de **campos**. Los campos son como zonas de la página, el principio puede ser el título. El número de página puede ir situado en la última línea de la página. Los campos pueden estar compuestos de **caracteres**. Los caracteres son la parte más pequeña, es decir, son el equivalente de las letras, igual que en una hoja escrita.



Vista de un archivo con datos

En pocas palabras los archivos son mecanismos de abstracción, es decir, son los equivalentes dentro de la computadora de los almacenes de información, son una

manera de almacenar grandes cantidades de información en un disco para que luego puedan ser utilizadas. Las operaciones de recuperación y generación (creación) deben de hacerse de una manera que el usuario quede protegido contra los detalles de forma y lugar de almacenamiento de la información, así como del funcionamiento real de los discos. La característica más importante de cualquier mecanismo de abstracción es probablemente la forma de nombrar los objetos por administrar.

Cuando un proceso crea un archivo se le da un nombre. Cuando el proceso concluye el archivo sigue existiendo, con excepción de los archivos de trabajo [temporales], los cuales son utilizados en algunos sistemas operativos como MVS, por lo cual otros procesos pueden tener acceso a estos archivos mediante su nombre.

Las reglas exactas para los nombres de los archivos varían un poco de sistema a sistema aunque todos los sistemas operativos permiten cadenas de hasta ocho letras como nombres de archivos.

Algunos administradores de archivos distinguen entre las letras mayúsculas y minúsculas, como es el caso de UNIX, pero otros no, como es el caso de MS-DOS.

Muchos sistemas operativos utilizan nombres de archivos con dos partes, separadas por un punto. La parte que sigue al punto es conocida como extensión del archivo e indica por lo general alguna característica especifica del archivo. Por ejemplo, el MS-DOS acepta nombres de archivos compuestos por 8 posiciones que indican el nombre del archivo y tres posiciones que indican la extensión del archivo (tipo de archivo). Otros sistemas operativos como el MVS van más allá permitiendo un número elevado de componentes para el acceso a los archivos, en el caso de MVS estos componentes se conocen como calificadores, estos calificadores son posicionales, es decir, su nombre refleja la posición dentro del nombre del archivo, al calificador inicial se le conoce como calificador de más alto nivel.

En muchos de los casos las extensiones asignadas a los archivos son meras convenciones y no son forzosas. Un archivo con extensión **.txt** es probablemente un archivo de texto, pero el nombre tiene más la intención de servir como recordatorio al propietario que proporcionar información especifica a la computadora para su tratamiento.

Por otro lado, una biblioteca con primer calificador **SYS1** puede indicar que es una biblioteca del sistema operativo MVS, pudiendo negar el acceso de lectura y escritura o tratamiento a los usuarios que no son administradores del sistema.

Otro ejemplo es el tratamiento de archivos para su compilación, un compilador de lenguaje C podría insistir en que los archivos a compilar tuvieran una extensión .c y negarse rotundamente a compilar los que no tengan esta extensión.

Convenciones como las anteriormente mencionadas son de utilidad si el mismo programa puede manipular y operar con archivos de diferentes tipos. Lo que hace que la extensión se convierta en un factor esencial para indicarle al programa que manipula el archivo como debe de hacerlo.

Aquí mostramos algunos tipos de archivos utilizados:

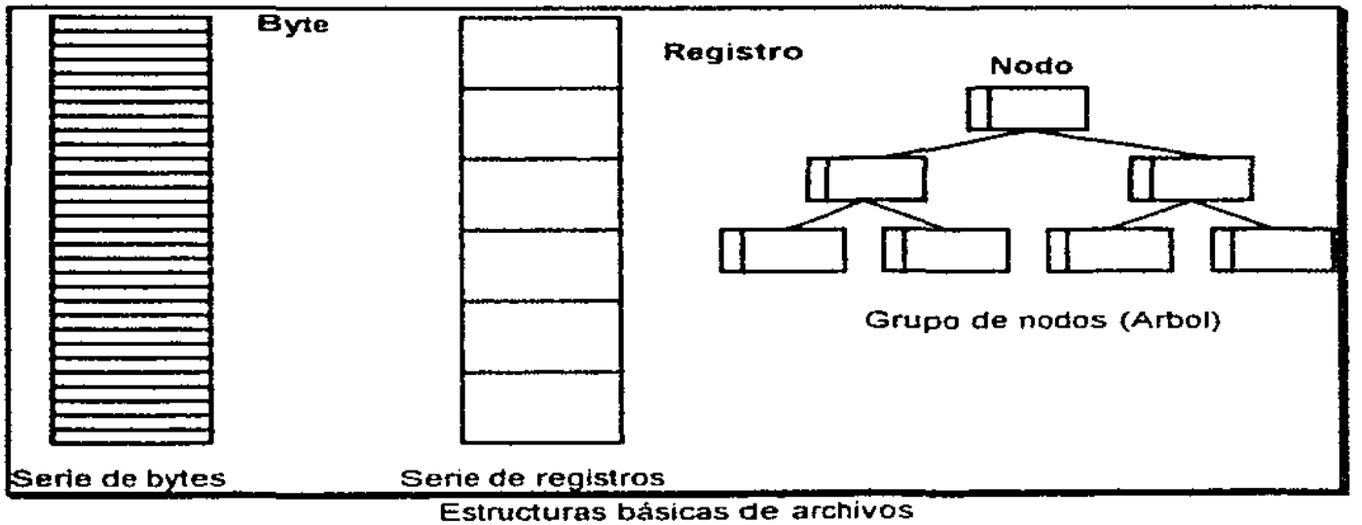
Extensión	Descripción
.BAK	Archivo de respaldo
.BAS	Programa en lenguaje Basic
.BIN	Programa binario
.C	Programa en lenguaje C
.DAT	Archivo de datos
.DBF	Archivo de base de datos
.DOC	Archivo de documentación
.DLL	Archivo de librería de enlace dinámico
.FTN	Programa en lenguaje Fortran
.HLP	Archivo de ayuda
.LIB	Archivo de biblioteca con programas a enlazar
.TXT	Archivo de texto
.PAS	Programa en lenguaje Pascal
.SYS	Archivo con parámetros del sistema

Extensiones de archivos

Estructura básica de un archivo

Los archivos pueden tener varios tipos de estructuras, pero existen tres estructuras básicas de archivos:

a) Series de bytes
b) Series de registros
c) Árbol



Estructuras básicas de archivos

a) El primer tipo de archivo es una secuencia no estructurada de bytes, en este caso al sistema operativo no le interesa el contenido del archivo. Lo único visible son los bytes; el significado se lo dan los programas que utilizan al archivo. UNIX y MS-DOS utilizan este tipo de estructura. El hecho de que el sistema operativo considere a los archivos como una secuencia de bytes proporciona una flexibilidad absoluta. Los programas de usuario pueden colocar lo que sea dentro de los archivos y nombrarlos de la manera que crean más conveniente. El sistema operativo no realiza algún tipo de manipulación sobre los archivos, por lo cual el usuario puede realizar las operaciones que quiera sobre esta estructura de archivo.

b) En este tipo de estructura, los archivos son secuencias de registros de longitud fija, cada uno con su propia estructura interna. Un aspecto importante de este tipo de estructura es el hecho de que las operaciones de lectura regresan un registro y las operaciones de escritura escriben o añaden un registro. Cuando los sistema operativos trabajan con tarjetas de 80 columnas, los archivos también tenían la misma longitud de registro (80 posiciones). Estos sistemas operativos antiguos podían utilizar también registros de 132 posiciones, esto con la intención de utilizarlos con las impresoras de líneas (las cuales en aquella época manejaban 132 posiciones). Un sistema operativo que todavía maneja este tipo de estándares es el MVS, en el cual los archivos de programas generalmente tienen 80 posiciones y los archivos de spool tienen 132 posiciones. Uno de los sistemas operativos más recientes (ya descontinuado) es el CP/M de Digital Research. Este sistema manipula registros de 128 caracteres.

c) En este tipo de estructura un archivo consta de un árbol de registros, los cuales no necesariamente tienen la misma longitud; cada registro contiene un campo llave en una posición fija del registro. El árbol se ordena utilizando este campo llave, con la finalidad de realizar las operaciones con el archivo con mayor rapidez. La operación básica en esta estructura de archivo es la de obtener un registro con la llave deseada. Esta estructura de archivo es claramente distinta de las de las series de bytes no estructuradas que utilizan los sistemas operativos como UNIX o MS-DOS. Esta estructura se utiliza ampliamente en grandes computadoras centrales en programas de administración de bases de datos, donde es crítica la obtención eficiente de la información. Dentro del MVS existe una estructura de archivo manipulada por llaves, esta estructura es denominada VSAM KSDS.

Tipos de archivos

Los sistemas operativos actuales soportan generalmente varios tipos de archivos. UNIX, MS-DOS, Linux, NexStep y MacOS, tiene archivos regulares, archivos especiales de caracteres y archivos especiales de bloques. Los archivos regulares son aquellos que contienen información del usuario. Los directorios son

archivos del sistema que se utilizan para dar seguimiento a la manera como se encuentran realmente colocados todos los archivos dentro del disco (regulares, directorios, especiales de bloques y caracteres). Los archivos especiales de caracteres tienen relación con las operaciones de entrada y salida y se utilizan para modelar dispositivos seriales de entrada y salida, tales como las terminales o las impresoras Los archivos especiales de bloques se utilizan para modelar básicamente discos.

Los archivos regulares son generalmente archivos ASCII, o binarios, o en otro código de caracteres como EBCDIC. Los archivos ASCII constan de líneas de texto. En algunos sistemas, cada línea termina con un caracter de retorno de carro. En algunos otros cada línea termina con un caracter de alimentación de línea. En algunos más se utilizan los dos caracteres para concluir cada línea Estas líneas de texto pudieran o no tener la misma longitud.
Una de las grandes ventajas de los archivos en código ASCII es que se pueden ver e imprimir tal como se encuentran y pueden modificarse con un editor de textos común. Además, si un gran número de programas utilizan archivos en código ASCII para realizar operaciones de entrada y salida, es fácil conectar la salida de un programa con la entrada de otro, como en las tuberías de un shell (intérprete de comandos).

Otro tipo de archivos son los binarios, los cuales no son comprensibles o interpretables a simple vista. Si se envian a la impresora producen un listado incomprensible lleno de saltos. Por lo general los archivos binarios tienen una estructura interna especial.

Utilizaremos como ejemplo un archivo binario de UNIX. Aunque desde el punto de vista técnico, el archivo solo es una secuencia de bytes, el sistema operativo sólo ejecuta un archivo que contenga el formato adecuado. En UNIX el formato tiene 5 secciones:

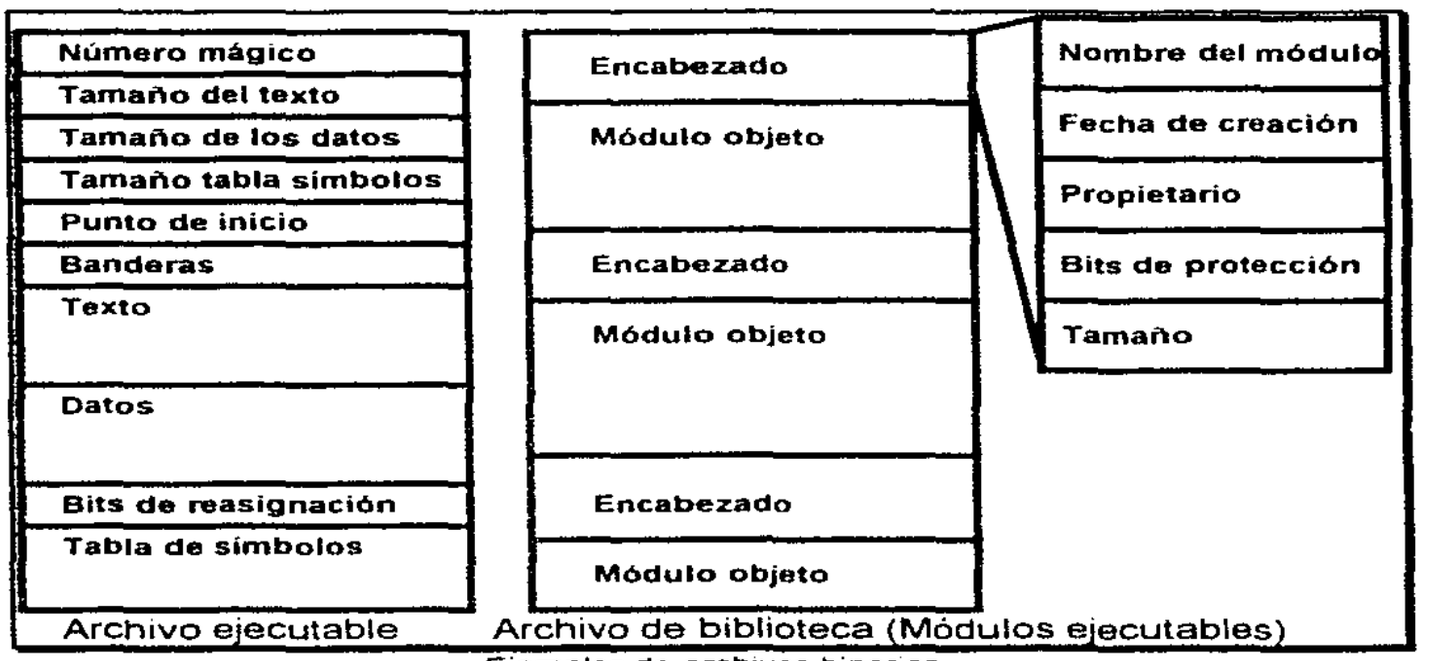
- Encabezado
- Texto
- Datos
- Bits de reasignación
- Tabla de símbolos

El encabezado inicia con un número especial denominado número mágico, el cual identifica el archivo como archivo ejecutable (esto es para evitar la ejecución [Intento de ejecución] de un archivo que no sea de ese formato). Después vienen varios enteros de 16 bits (actualmente 32 bits) con los tamaños de las distintas partes de los archivos, la dirección de inicio del programa ejecutable y algunos bits de estado. Después del encabezado vienen el texto y los datos del mismo programa. Estos se cargan en memoria y se reasignan mediante los bits de reasignación. La tabla de símbolos se utiliza durante la depuración

Como segundo ejemplo mostraremos un archivo binario del tipo biblioteca de UNIX. Este archivo consta de una colección de procedimientos de biblioteca o módulos compilados pero no ligados. Cada uno de los cuales inicia con un encabezado en el que se encuentra la siguiente información:

- Nombre
- Fecha de creación
- Propietario
- Código de protección
- Tamaño

Al igual que en los archivos ejecutables, los encabezados de los módulos son binarios, por lo que si se copian a la impresora o se tratan de leer producen salidas incomprensibles.



Ejemplos de archivos binarios

Tipos de acceso a los archivos

En los primeros sistemas operativos solo existía un tipo de acceso a los archivos, este acceso era secuencial. En estos sistemas, los procesos podían leer un determinado registro de un archivo, pero para hacer esto tenían primero que leer todos sus antecesores. Sin embargo los archivos secuenciales tenían la característica de que podían ser rebobinados, es decir, regresar al inicio del archivo para comenzar una nueva lectura. Actualmente los archivos secuenciales

se utilizan para almacenar información en cintas y cartuchos, ya que carecen de una utilidad práctica en los discos debido a que estos se pueden leer de manera aleatoria. Uno de los sistemas operativos que todavía maneja los archivos secuenciales es el MVS, pero generalmente estos archivos son solamente utilizados como archivos de paso (de trabajo) o de conexión con distintas aplicaciones.

Cuando se comenzó con la utilización de los discos para el almacenamiento de grandes cantidades de información, fue posible leer los bytes o los registros de los archivos en cualquier orden, o bien tener acceso a los registros mediante una llave, en lugar de una posición. Los archivos cuyos bytes o registros pueden leerse en cualquier orden se le llaman archivos de acceso aleatorio.

Los archivos de acceso aleatorio son esenciales en muchas aplicaciones modernas (aplicaciones de misión crítica); por ejemplo en los grandes sistemas de bases de datos

Se utilizan dos operaciones para realizar una lectura en un archivo de acceso aleatorio, primeramente una operación de búsqueda (SEEK) coloca el apuntador de registro en el registro deseado, posteriormente una operación de lectura (READ) obtiene la información del archivo.

En algunos sistemas operativos de grandes computadoras centrales los archivos se siguen clasificando en secuenciales y de acceso aleatorio. Esto permite que el sistema operativo utilice técnicas distintas para la administración de estos archivos. En la gran mayoría de los sistemas de cómputo modernos no se hace la distinción, ya que los archivos siempre son de acceso aleatorio.

Atributos de un archivo

Cada archivo tiene varias características entre las cuales las más conocidas son su nombre y tipo de datos. Además, todos los sistemas operativos asocian información adicional a cada archivo; por ejemplo, su tamaño, la fecha y hora de su creación. Estos elementos que definen alguna característica de los archivos son denominados atributos. La lista de atributos de un archivo varía de un sistema operativo a otro. Aquí mostramos una lista de los atributos considerados por varios sistemas operativos:

Campo	Significado
Contraseña	Clave de acceso al archivo
Creador	Identificador de la persona que creó el archivo
Propietario	Propietario actual
Solo lectura	0 Lectura/escritura, 1 solo lectura
Archivo escondido	0 Visible, 1 invisible
Bandera binario	0 ASCII, 1 binario
Tamaño máximo	Tamaño máximo que puede crecer

Atributos de un archivo

Las banderas son campos de 1 bit que controlan o permiten ciertas propiedades determinadas.
Los campos de la longitud del registro, posición de la llave, y longitud de la llave están presentes en los archivos en cuyos registros se pueden hacer búsquedas mediante una llave Estos campos proporcionan la información necesaria para encontrar las llaves de manera eficiente.

Los campos de creación, última modificación y última referencia son útiles por varias razones. Por ejemplo, un archivo fuente que ha sido modificado después de crear el ejecutable debe volverse a compilar.
El campo de tamaño indica el tamaño actual del archivo, algunos sistemas operativos de grandes computadoras centrales requieren que se especifique el tamaño máximo del archivo (MVS).

Operaciones con archivos

Los archivos son útiles porque permiten almacenar información y recuperarla de una manera efectiva. Los sistemas operativos proporcionan diversas operaciones para permitir el almacenamiento y la recuperación. Aquí mostramos las llamadas al sistema más comunes relacionadas con las operaciones sobre archivos:

- **Crear (CREATE).-** El archivo se crea sin datos. El propósito de la llamada al sistema es asignar la entrada en el directorio y establecer los atributos iniciales.

- **Eliminar (DELETE).-** Cuando ya no es necesario el archivo, éste debe eliminarse para liberar espacio en disco. Esta llamada borra la entrada en el directorio y marca todos los bloques utilizados por el archivo como bloques libres.

- **Abrir (OPEN).-** Antes de utilizar un archivo es necesario abrirlo. La finalidad de esta llamada al sistema es permitir que se trasladen los atributos y la lista de direcciones en disco a la memoria principal para un acceso más rápido en llamadas posteriores. Adicionalmente se asocia un área de memoria principal para colocar los bloques leídos del archivo.

- **Cerrar (CLOSE).-** Cuando se concluyen las operaciones con el archivo se libera el espacio ocupado por los atributos y áreas de memoria utilizadas por el archivo. Adicionalmente si existen operaciones pendientes de escritura estas son emitidas hacia el archivo.

- **Leer (READ).-** Con esta llamada al sistema se lee un grupo de bytes (generalmente uno o más bloques) del archivo a partir de la posición actual. Quien realiza la llamada debe especificar la cantidad de datos necesarios a leer.

- **Escribir (WRITE).-** Con esta llamada al sistema el grupo de bytes y registros modificados o agregados al área temporal son vaciados en el archivo. Si la posición actual está al final del archivo, el tamaño del mismo se aumenta. Si la posición es intermedia, se modifican los datos (sobreescriben) existentes

- **Añadir (APPEND).-** Esta es una forma restringida de escritura, ya que solo se pueden agregar información al final del archivo.

- **Buscar (SEEK).-** Para los archivos de acceso aleatorio se necesita un método para especificar el punto de la siguiente operación sobre el archivo. Después de llevar a cabo esta llamada se pueden leer o escribir datos a partir de la nueva posición.

- **Obtener atributos (GET ATTRIB).-** Es frecuente que los procesos deban leer los atributos para realizar su trabajo. Por ejemplo, el programa make de UNIX se utiliza para la administración de proyectos de desarrollo de software que constan de múltiples archivos fuentes. Al invocar la ejecución de make, el programa examina los tiempos de modificación de todos los archivos fuentes y objetos para llevar a cabo el mínimo de compilaciones necesarias para la actualización de todo el proceso. Para realizar esta tarea el programa lee y compara todos los atributos; en particular los tiempos de modificación

- **Cambiar atributos (SET ATTRIB).-** Algunos de los atributos de los archivos pueden ser determinados por el usuario, por lo cual pueden modificarse después de la creación de los archivos. Como ejemplo se encuentran las banderas de autorización de acceso al archivo

- **Copiar (Duplicar) un archivo (COPY).-** Es una operación especial en la cual se lee un archivo en forma secuencial y se escribe en uno nuevo en forma secuencial, obteniéndose después de la operación una copia idéntica del original.

- **Cambiar de nombre (RENAME).-** Con frecuencia ocurre que un usuario requiere modificar el nombre de un archivo existente. Esta llamada realiza la tarea de modificación o alteración del nombre. Un método alternativo para cambiar el nombre es el de primeramente copiar el archivo en otro con un nuevo nombre y luego eliminar el original, pero si el archivo es muy grande esta operación lleva más tiempo que el simple cambio de nombre.

Archivos mapeados en memoria

El acceso a los archivos dentro de los discos es muy lento si es comparado con el acceso a memoria; para solucionar este problema se han propuesto múltiples soluciones; desde las soluciones por hardware en las cuales un conjunto muy

grande de memoria simula un disco (discos de estado sólido) hasta las soluciones por software tales como el mapeo de los archivos en memoria o el disco RAM (en el cual una parte de memoria simula un disco, incluyendo archivos, directorios y subdirectorios).
Por esto, algunos sistemas operativos, tales como el MULTICS, proporcionan una manera de asociar los archivos con el espacio de direcciones de un proceso en ejecución. La idea fundamental del mapeo es la siguiente: Se crean dos llamadas nuevas al sistema las cuales denominaremos MAPEAR y ELIMINAR_MAPA. La primera utiliza el nombre de un archivo y una dirección virtual provocando que el sistema operativo asocie al archivo con una dirección virtual dentro del espacio de direcciones del proceso.
Supongamos que un archivo **f** de longitud de 64 KB es asociado con la dirección virtual 512 KB. Entonces, cualquier instrucción de la máquina que lea el contenido de la dirección virtual 512 KB obtendrá el primer byte del archivo. En forma análoga, al alterar la dirección 512 KB + 1100 se modifica el byte 1101 del archivo mapeado. Al concluir el proceso, el archivo modificado se graba en disco de la misma manera como si hubiera sido modificado con sentencias **buscar y escribir**.
Aunque el mapeo de archivos elimina múltiples operaciones de entrada y salida con lo que facilita mucho la programación, incluye algunos problemas.

a) Es muy difícil que el sistema conozca la longitud exacta del archivo después de las modificaciones, ya que si se agregan datos, es poco probable que estos sean conocidos de antemano, al menos su longitud final es difícil estimarla. Por lo cual, el crecimiento del archivo y del espacio de direcciones por consiguiente puede ser excesivo y por lo tanto se puede perder el control de la memoria o provocar múltiples fallos de página ocasionando que la ventaja del mapeo sea una desventaja por un crecimiento excesivo del archivo.

b) Si un archivo es mapeado dentro de un espacio de direcciones y otro proceso intenta leer el mismo archivo mapeado por el otro proceso el segundo proceso que mapeo o leyó directamente el archivo no leerá la versión más actualizada del archivo, ya que el primer proceso no reflejará los cambios en el disco, o en la memoria virtual hasta que se termine de utilizar una página o que se libere al archivo en su totalidad. Los sistemas operativos deben ser diseñados para que garanticen que varios procesos no utilicen versiones inconsistentes de un solo archivo.

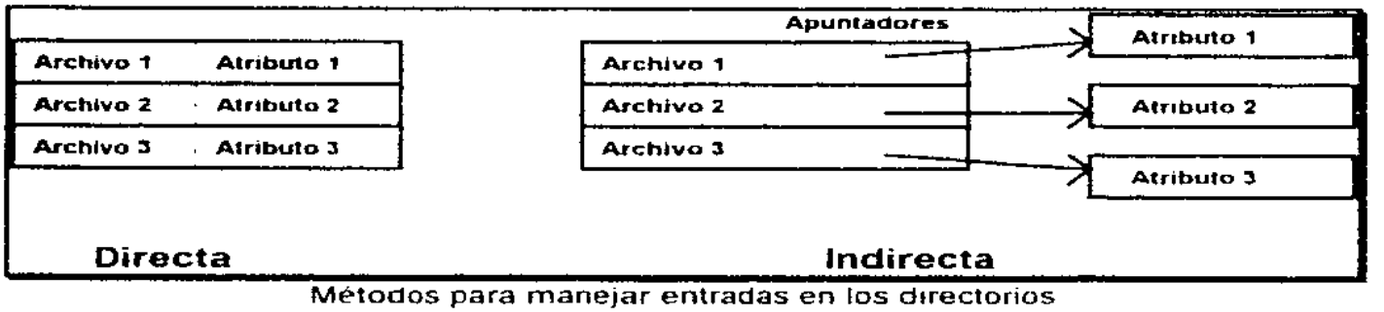
En la actualidad DEC (Digital Equipment Corporation), Oracle (Oracle Corporation) y SAP (System Application Programs) diseñaron un modelo en el cual toda una aplicación SAP construida dentro de Oracle puede estar contenida en su totalidad en la memoria. Este modelo permite tener aplicaciones de SAP de hasta 16 Gigabytes en memoria, provocando con esto que las actualizaciones, consultas y procesos de la aplicación en general se ejecuten a velocidades impresionantes, el único inconveniente es el costo excesivo de la memoria requerida.

Directorios

Cuando los sistemas primitivos de archivos fueron más difíciles de manipular debido al exceso de entradas contenidas en su catálogo se ideó un método de administración que permitía llevar un registro de los archivos mediante varios niveles, estos niveles de administración de catálogos se denominaron directorios.

Sistemas jerárquicos de directorios

Un directorio contiene varios datos, generalmente un registro de datos por archivo. Una posibilidad es que un dato contenga el nombre de un archivo, sus atributos y las direcciones en disco donde se almacenan los datos. Otra posibilidad es que la entrada en el directorio contenga el nombre del archivo y un apuntador a otra estructura de datos, en donde se encuentran los atributos y las direcciones en disco. Estos dos métodos son de uso común

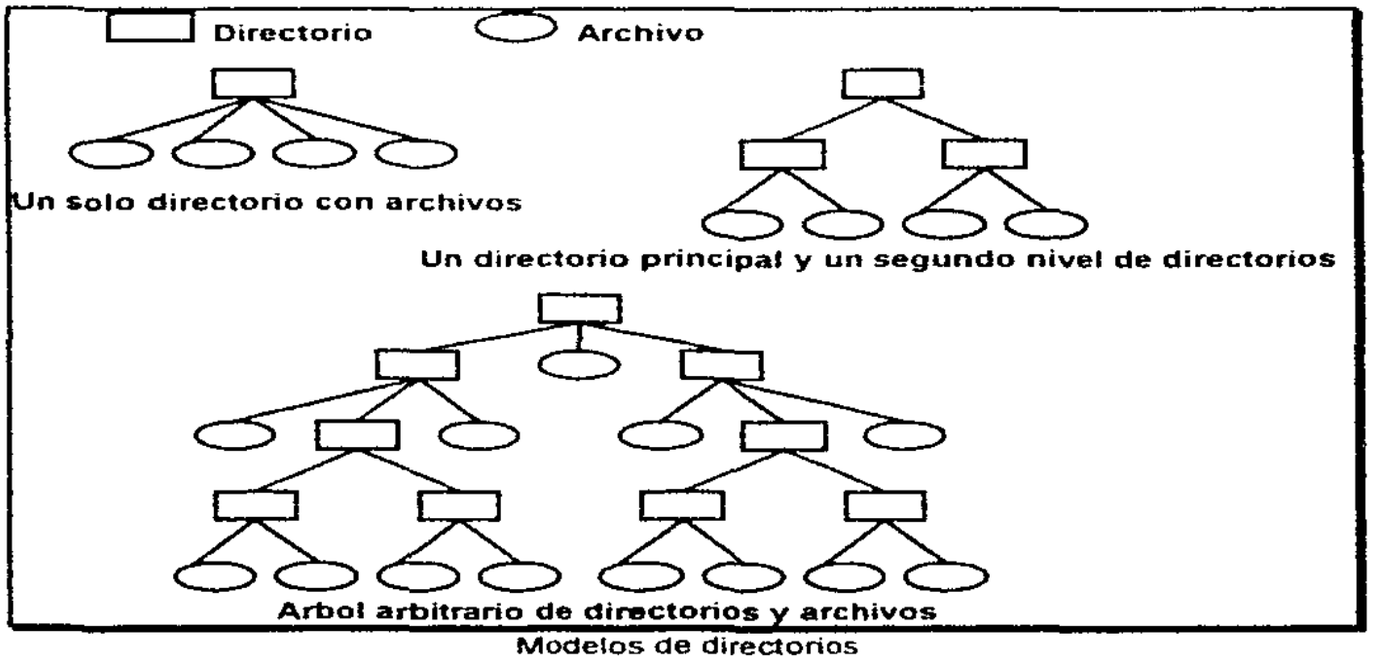


Métodos para manejar entradas en los directorios

Al abrir algún archivo el sistema operativo busca en las entradas de directorio hasta encontrar el nombre del archivo por abrir. Obtiene los atributos y direcciones en disco, ya sea en forma directa de los datos del directorio o de manera indirecta leyendo los atributos de la estructura de datos a la cual apunta la entrada en el directorio. Todos los datos relacionados al archivo son colocados en memoria principal para que las referencias subsecuentes al archivo utilicen la información colocada en memoria principal.
El número de directorios (niveles de directorios) varía ampliamente de un sistema a otro. El diseño más simple es aquel donde solamente existe un directorio principal en el cual se colocan todos los archivos. Si existen muchos usuarios para estos archivos es posible que los usuarios seleccionen nombres similares para sus archivos (tarea_1, programa_1), lo cual provocará confusiones que terminarán con una falla general en el sistema de archivo. Este sistema de un solo directorio solamente se utiliza en microcomputadoras primitivas, un ejemplo es el CP/M de Digital Research.

Una mejora al diseño de un directorio es incrementar el número de directorios por cada usuario dentro del sistema, es decir, agregar un nivel más de directorios

Con lo cual cada usuario puede ponerle el nombre que quiera a cada uno de sus archivos y aunque se repitan los nombres de los archivos (tarea_1, programa_1) entre varios usuarios no interferirán con los demás ya que los archivos de cada usuario se encuentran en sus propios directorios.
Este diseño no elimina los problemas para los usuarios que tienen muchos archivos, ya que es común que los usuarios agrupen sus archivos en formas lógicas, con lo cual un solo nivel adicional de directorios no es suficiente
Para solucionar el problema se necesita una jerarquía general, es decir, múltiples niveles de directorios. Con este punto de vista, cada usuario puede tener tantos directorios como requiera, de manera que pueda agrupar sus archivos en forma ordenada.



Modelos de directorios

Rutas de acceso

Cuando el sistema de archivos se encuentra organizado como un árbol de directorios, se necesita una forma de determinar los nombres de cada uno de los archivos. Se utilizan básicamente dos métodos. En el primero, cada archivo tiene una trayectoria absoluta, la cual consta de la ruta de acceso desde el directorio raíz (directorio principal o inicial, a partir del cual surgen todos los demás directorios) hasta el archivo. Los nombres absolutos siempre inician en el directorio raíz y son únicos.
El otro tipo de nombre es la ruta de acceso relativa. Esta se utiliza junto con el concepto directorio de trabajo, también llamado directorio actual o directorio activo. Se puede designar a un directorio como directorio de trabajo, en cuyo caso

todos los nombres que no comiencen en el directorio raiz se toman en relación con el directorio de trabajo
La mayoría de los sistemas operativos que soportan un sistema jerárquico de directorios tienen dos entradas especiales en cada directorio "." y ".." (punto y punto punto). Punto indica al directorio de trabajo, es decir, el directorio actual, y punto punto indica a su predecesor o padre. Las entradas punto y punto punto evitan la necesidad de memorizar las rutas absolutas de los archivos, ya que permiten referenciar el nivel actual o un nivel anterior de directorios

Operaciones con directorios

Las llamadas al sistema permitidas para el manejo de directorios varían más de sistema a sistema que las relacionadas con operaciones de archivos. Sin embargo aquí mostramos algunas que la mayor parte de los sistemas operativos utilizan.

- **Crear (CREATE).-** Se crea el directorio, el cual se encuentra vacio excepto por las referencias punto y punto punto.

- **Eliminar (DELETE).-** Se elimina el directorio. En la mayoría de los casos solo se puede eliminar un directorio vacío. Un directorio que contenga las referencias punto y punto punto se considera vacio.

- **Abrir directorio (OPEN).-** Se pueden leer los directorios. Por ejemplo, para exhibir una lista de todos los archivos de un directorio, un programa puede abrir el directorio para leer los nombres de todos los archivos. Esta operación se requiere para leer el directorio, ya que también es un archivo.

- **Cerrar directorio (CLOSE).-** Cuando se han realizado las operaciones necesarias con un directorio, este debe cerrarse para liberar espacio de las tablas internas del sistema.

- **Leer directorio (READ).-** Esta llamada regresa la siguiente entrada de un directorio abierto. Anteriormente existía la posibilidad de utilizar la misma llamada al sistema para leer archivos con directorios, pero debido a que el programador debía conocer la estructura de los registros de las entrada del directorio se creo una nueva llamada que no requiere conocer la estructura

- **Cambiar nombre (RENAME).-** Debido a que los directorios son archivos pueden cambiar su nombre con la misma llamada al sistema.

- **Ligar (LINK).-** La liga es una técnica que permite a los archivos aparecer en varios directorios a la vez. Esta llamada al sistema especifica un archivo existente y el nombre de una nueva ruta de acceso. De esta manera un archivo puede aparecer en varios directorios.

- **Desligar (UNLINK).-** Se elimina una liga a un archivo. Si el archivo que se desea desligar aparece solo en un directorio, el cual es el caso más común, se elimina del sistema de archivos Si está presente en varios directorios, solo se elimina la liga especificada, por lo cual las demás ligas permanecen.

Asignación de espacio

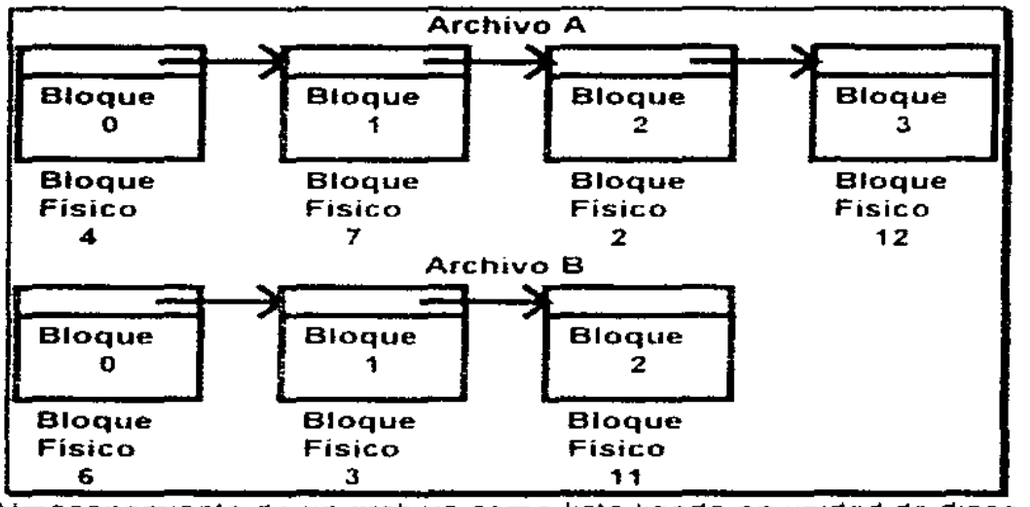
Uno de los problemas con que se encuentran los diseñadores de sistemas operativos es la manera como se debe asignar el espacio a los archivos dentro de los dispositivos de almacenamiento, así como también las estructuras de control que administrarán los métodos de asignación de espacio, comenzaremos discutiendo el método más simple de asignación de espacio.

Asignación adyacente

En este esquema de asignación de espacio toda la información de un archivo se almacena de forma contigua o adyacente, es decir, cada bloque del archivo se encuentra uno tras de otro formando un bloque mayor que contiene toda la información. Por ejemplo, si los bloques del disco son de 4 KB, un archivo de 16 KB ocupará 4 bloques consecutivos. Este esquema de administración tiene dos ventajas fundamentales, la primera es su fácil implantación, ya que los registros de localización de los archivos se limitan a almacenar únicamente el número de bloque inicial de cada archivo, la segunda es que proporciona un rendimiento excelente, ya que todo el archivo se encuentra contiguo, es decir, no existe la necesidad de recorrer la cabeza lectora a través del disco para leer varios bloques del archivo.
Pero este método presenta dos problemas fundamentales, el primer problema es que no es realizable, debido a que el sistema debe conocer en el momento de su creación el tamaño final del archivo, lo cual es imposible, debido a que los archivos generalmente se utilizan para almacenar información que cambia y crece con el tiempo, aunque en unas pocas ocasiones disminuye. El segundo problema es que se presenta una fragmentación muy alta, por ejemplo, si se borra un archivo queda un hueco donde anteriormente se encontraba el archivo, pero si los siguientes archivos a crear no se ajustan en este espacio o hueco, el espacio es desperdiciado, por lo que después de una serie de creaciones y eliminaciones de archivos el disco queda totalmente fragmentado con un porcentaje significativo del espacio desperdiciado

Asignación en forma de lista ligada

Otro método de almacenamiento muy utilizado es el de mantener una lista ligada de los bloques en disco. El primer registro de cada bloque se utiliza como apuntador al bloque siguiente, el resto del bloque contiene datos.



Almacenamiento de un archivo como lista ligada en unidad de disco

A diferencia de el método de asignación adyacente, aquí si es posible la utilización de todos los bloques del disco. No se desperdicia espacio, debido a que cada bloque puede ser utilizado. Adicionalmente, la entrada en el directorio solo debe contener el número inicial del bloque de cada archivo, a partir del cual se puede procesar la totalidad del archivo.
Pero al igual que en la asignación adyacente existen desventajas en este método. Una lectura secuencial es directa (aunque existen las búsquedas de los bloques apuntados por el primer registro del bloque), la lectura aleatoria se vuelve extremadamente lenta. Adicionalmente, la cantidad de espacio asignada para el almacenamiento de la información ya no es una potencia de dos, puesto que el apuntador (registro inicial de cada bloque) ocupa algunos bytes. Aunque este último punto no tiene consecuencias fatales, el hecho de tener un tamaño real de bloque de datos de tamaño diferente a una potencia de dos resulta en una administración menos eficiente.

Asignación en forma de lista ligada y un índice

Las desventajas que tiene el método de la lista ligada pueden ser eliminadas si se toma el primer registro de cada bloque y se almacena en una lista en memoria en lugar de utilizar el primer registro de cada bloque.
Mediante este esquema, todo el bloque está disponible para los datos, adicionalmente el acceso aleatorio es mucho más sencillo, aunque existe la necesidad de verificar la lista en memoria para realizar las operaciones con los archivos, pero la operación de búsqueda del bloque deseado es rápida ya que toda la información necesaria se encuentra en memoria.
La principal desventaja de este método es que toda la tabla debe conservarse en memoria. Con un disco de tamaño grande digamos de 500 MB, la tabla tendría alrededor de 500,000 entradas, cada una de las cuales tendría cuando menos tres bytes. Con lo cual la tabla mediría 1.5 MB, si el sistema intenta optimar el

espacio y el tiempo de acceso esto resulta inaceptable simplemente imaginemos máquinas con 640 KB de memoria principal, la tabla simple y sencillamente no cabe en memoria.

Asignación por Nodos-i

Este método utiliza una tabla para controlar los bloques correspondientes a un archivo, esta tabla es llamada nodo-i o nodo índice, la cual contiene los atributos y las direcciones en disco de los bloques de un archivo.
Las primeras direcciones de disco se almacenan en el propio nodo-i, de manera que en caso de que los archivos sean pequeños, la información necesaria se encuentra en el nodo-i, el cual se traslada del disco a la memoria principal al momento de abrir el archivo Para los archivos un poco más grandes, entre las últimas direcciones contenidas en el nodo-i se encuentra la dirección en disco de un bloque llamado **bloque simplemente indirecto.** Este bloque contiene las direcciones adicionales de bloques en el disco. Si este bloque simplemente indirecto no alcanza a cubrir todas las direcciones de disco del archivo, otra dirección en el nodo-i apunta al **bloque doblemente indirecto,** el cual contiene una dirección que presenta una lista de los bloques simplemente indirectos. Cada uno de estos bloques simplemente indirectos apuntan a cientos de bloques de datos. Para los archivos mayores se puede utilizar un bloque conocido como **bloque triplemente indirecto** el cual apunta a bloques doblemente indirectos que a su vez apuntan a bloques simplemente indirectos que finalmente apuntan a bloques de datos permitiendo tamaños muy grandes para los archivos.

Administración de espacio en disco

Los archivos se almacenan por lo general en discos, por lo que el manejo de los discos es una preocupación fundamental para los diseñadores de los administradores de archivos. Existen dos estrategias generales para almacenar un archivo de n bytes: asignar n bytes consecutivos en disco o dividir el archivo en cierto número de bloques no necesariamente adyacentes. La misma disyuntiva se presenta en los sistema de administración de memoria, en la segmentación pura y la paginación.
El almacenamiento de un archivo como una serie adyacente de bytes tiene el problema de que si un archivo crece, será muy probable que deba moverse en el disco. El mismo problema ocurre con los segmentos de memoria, excepto que el desplazamiento de un segmento de memoria es una operación relativamente rápida en comparación con el desplazamiento de un archivo de una posición a otra del disco. Por esto la mayoría de los sistemas operativos modernos dividen a los archivos en bloques de tamaño fijo.

Tamaño del bloque

Una vez que se ha tomado la decisión de almacenar los archivos en bloques de tamaño fijo, surge la pregunta del tamaño de dichos bloques, aunque es conveniente revisar los manuales de hardware del disco para seleccionar un tamaño de bloque que sea múltiplo del tamaño del sector, del tamaño de la pista o del tamaño del cilindro
Si se selecciona una unidad de asignación grande, digamos un cilindro, esto significa que un archivo, incluso el más pequeño ocupará todo un cilindro Por otra parte la utilización de una unidad muy pequeña para la asignación implica que cada archivo constará de muchos bloques. La lectura de cada bloque requiere por lo general de un retraso por la búsqueda y rotación del dispositivo, por lo que la lectura de un archivo con múltiples bloques pequeños será extremadamente lenta.
Las unidades recomendadas para el tamaño de bloque son de

- 512 Bytes
- 1 KB
- 2 KB
- 4 KB

Esta recomendación se fundamenta en que la mayor parte de los fabricantes de discos utilizan estos valores para el tamaño del sector

Registro de los bloques libres

Una vez elegido el tamaño de bloque, el siguiente aspecto a considerar es la forma de llevar un registro de los bloques libres. Se utilizan por lo general dos métodos:

- El primero consiste en el uso de una lista ligada de bloques en el disco, en la que cada bloque contiene tantos números de bloques libres como pueda.
- El segundo consiste utilizar un mapa de bits. Donde cada bit representa el estado de un bloque dentro del disco.

El mapa de bits requiere generalmente menos espacio, ya que solamente utiliza un bit para cada bloque.
Si existe suficiente espacio dentro de la memoria principal para contener el mapa de bits, este métodos es preferible.

Cuotas de disco

Dentro de un sistema multiusuario es necesario establecer reglas claras en cuanto a la utilización de espacio en disco. Por ésto, para evitar que los usuarios

del sistema de cómputo se apropien de un espacio excesivo se deben proporcionar mecanismos que permitan establecer cuotas de utilización de disco. La idea fundamental es que el administrador del sistema asigne a cada usuario una proporción grande de archivos y bloques que permitan completar sus tareas, estas cuotas deben garantizar que si ingresa un nuevo usuario al sistema encontrará el espacio suficiente para realizar algún trabajo útil.

Confiabilidad del sistema de archivos

La pérdida y destrucción de un sistema de archivos es generalmente un desastre mayor que el de la destrucción de una computadora.
Si el sistema de archivos de una computadora se pierde será difícil restaurar toda la información, esta restauración tardará generalmente mucho tiempo y en algunos casos críticos será imposible. Las consecuencias son catastróficas para las personas que pierden información. Aunque el sistema de archivos no puede ofrecer protección alguna contra la destrucción física del equipo y los medios de almacenamiento, puede ayudar a proteger la información.

Manejo de bloques defectuosos

Los discos duros actualmente se fabrican con procesos especiales que no permiten en la mayoría de los casos que contengan bloques defectuosos. Los discos flexibles son generalmente perfectos cuando salen de la fábrica, pero pueden desarrollar bloques defectuosos durante su uso normal. Aunque los bloques defectuosos son indeseables y cada vez son más difíciles de encontrar es necesario considerar su existencia y manejo. Existen dos tipos de soluciones para el manejo de bloques defectuosos, una es en hardware y otra en software. La solución de hardware consiste en dedicar un sector de disco a los bloques defectuosos. Al iniciar el controlador por primera vez éste lee la lista de bloques defectuosos y elige un bloque, en algunos casos es una pista, de reserva para reemplazar los bloques defectuosos, registrando la asociación en la lista de bloques defectuosos. Las nuevas solicitudes del bloque defectuoso utilizarán el bloque de repuesto.
La solución por software requiere que el usuario o el sistema de archivos construyan con cuidado un archivo con todos los bloques defectuosos. Esta técnica los elimina de la lista de los bloques libres, de forma que nunca pueden aparecer en los archivos de datos. Mientras no se lea o escriba en este archivo de bloques defectuosos no surgirán problemas. Los problemas empiezan si un bloque ya asignado a un archivo se daña, aquí dependiendo de la administración del disco se puede perder solo ese bloque de información o todos los bloques del archivo que siguen lógicamente al bloque dañado

Respaldos

Aún con una estrategia brillante para librar el problema de los bloques dañados, es importante el respaldo de los archivos. Los sistemas de archivos de los discos flexibles se pueden respaldar mediante una copia en otro disco de igual capacidad. Los sistemas de archivos en los discos duros se pueden respaldar en unidades de cinta especializadas de 8 mm., 4 mm., unidades DLT y en discos ópticos de gran capacidad. Otra forma de respaldar el sistema de archivos consiste en que los respaldos sean realizados por incrementos, los cuales solo respaldan la información que haya sido modificada desde el respaldo anterior. Aunque cabe mencionar que de todas maneras es muy recomendable que se obtengan respaldos completos al menos una vez al mes.

Consistencia del sistema de archivos

Otra área donde la confiabilidad es muy importante es en la consistencia del sistema de archivos. Muchos administradores de archivos leen bloques y posteriormente escriben en ellos después. Si el sistema falla antes de escribir los bloques modificados, el sistema de archivos puede quedar en un estado inconsistente. Este problema es particularmente crítico si alguno de los bloques en los que no se han escrito las modificaciones son los bloques de nodos-i, bloques de directorios o bloques con la lista de bloques libres, esto es debido principalmente a que estos bloques de alguna manera son bloques de control que ayudan a la administración del espacio en disco, por lo cual una pérdida de datos en los mismos provocaría una falla mayor en los archivos o datos que administran. Para enfrentar el problema de los sistemas de archivos inconsistentes, la mayoría de las computadoras tienen un programa de utilería que verifica la consistencia del sistema de archivos. Se puede ejecutar al iniciar el sistema; en particular después de una falla general. Se pueden realizar dos tipos de verificaciones de consistencia: para bloques y para archivos. Para verificar la consistencia de bloques, el programa construye una tabla con dos contadores para cada bloque, con un valor inicial de 0. El primer contador mantiene un registro del número de veces que el bloque está presente en un archivo; el segundo registra el número de veces que está presente en la lista de bloques libres o en el mapa de bits de bloques libres. El programa lee entonces todos los nodos-i. Ya que, a partir de un nodo-i, es posible construir una lista de todos los números de los bloques utilizados en el archivo correspondiente. Al leer cada número de bloque, se incrementa su contador en la primera tabla. El programa examina a continuación la lista o mapa de bits de los bloques libres para localizar todos los bloques que no se estén utilizando. Cada aparición de un bloque en la lista de bloques libres hace que se incremente su contador en la segunda tabla. Estos programas de verificación obtienen finalmente la lista del estado de todos los bloques del sistema, en esta lista se destacarán los bloques ausentes, duplicados, no referenciados o con alguna característica especial que signifique inconsistencia.

◆ CAPITULO VI.- IMPLANTACION DE UN SISTEMA OPERATIVO

La confiabilidad de un sistema operativo depende no solamente de proveer el resultado lógico esperado sino que también depende del tiempo en el cual son producidos los resultados, en este caso la predictibilidad y no la velocidad es la meta más importante en el diseño de los sistemas operativos.

Un estudio de sistemas operativos no está completo si la teoría no viene respaldada con una profunda práctica, por lo tanto aquí incluimos un sistema operativo pequeño pero funcional; el cual puede ser utilizado para reforzar la teoría modificando y recompilando el código de dicho sistema para observar los resultados. El proceso de compilación y modificación del sistema se examinará posteriormente en este mismo capítulo.

Es importante destacar que en esta implantación se incluyen todos los componentes mayores de los sistemas operativos comerciales, estos componentes son detalladamente explicados a lo largo de los capítulos 2 a 5 de la presente tesis, los componentes incluidos en este sistema operativo son:

- Administrador de procesos, capítulo 2
- Administrador de memoria, capítulo 3
- Administrador de dispositivos, capítulo 4
- Sistema de archivos, capítulo 5

Descripción de OMNI

Este sistema operativo está diseñado para correr en una plataforma Intel 286 o superior; se ha probado en sistemas 286, 386 y Pentium, encontrándose problemas con algunos de los equipos con procesador 486 y Pentium, estos problemas básicamente son provocados por las diferencias en los BIOS de las máquinas. Se recomienda una máquina 100 % compatible con IBM para su utilización y recompilación, aunque se ha probado en otras plataformas teniendo éxito en la mayoría de las pruebas.

El paquete del sistema operativo consiste en el código fuente y algunas utilerías para construir un disco cargable con un sistema operativo multitareas similar a Unix. ***Este sistema operativo no es por algún motivo un sistema totalmente probado, por lo cual se ofrece como una ayuda experimental para el curso de Sistemas Operativos más que como sistema productivo.***

Algunas de las características importantes de este sistema operativo al cual llamaremos **OMNI** son:

- Creación rápida y fácil de procesos

- Creación de un disco cargable de alta densidad, 1.44 MB, 3.5 pulgadas
- Sistema de archivos similar a Unix, sin manejo de directorios
- Servicios de Sistema Operativo para manejo de teclado, pantalla, sistema de archivo y administración de procesos
- Hasta 8 terminales virtuales
- Librería de C para construcción de ambiente y programas de pruebas
- Un intérprete de comandos simple

En el diseño de este sistema operativo se tomaron algunas decisiones fundamentales; estas decisiones son para facilitar la programación y minimizar el código del sistema operativo, aún a pesar de eso **OMNI** es un programa grande que consta de alrededor de 7300 líneas entre C y ensamblador. Cabe mencionar que el diseño está muy adecuado a la plataforma Intel por lo que gran parte del código se escribió en ensamblador.

En esta implantación de un sistema operativo utilizamos un algoritmo de administración del procesador de planificación circular, en el cual a cada uno de los procesos se le asigna exactamente el mismo tiempo de procesador, es decir, utilizamos una política en la cual no existen prioridades para los procesos. Esta política de administración del procesador se encuentra más detallada en el capítulo de administración del procesador de la presente tesis [Capítulo 2].

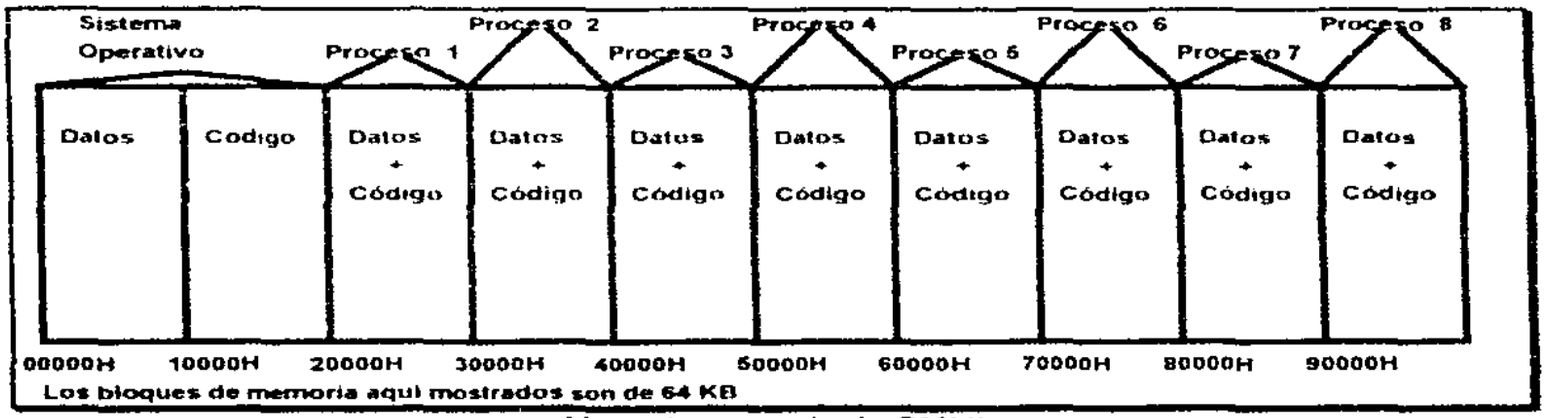
Se cuenta con dos servicios para la creación de procesos, los cuales son exec() y fork(), adicionalmente se incluyen servicios para la alojación de memoria compartida, además de una serie de primitivas para el manejo de semáforos, las cuales son wait() y signal(), se cuenta también con servicios para la terminación de procesos, los cuales son exit() y kill(), adicionalmente se cuenta con servicios para detener momentáneamente la ejecución de procesos, esto es realizado con las funciones tsleep() y sleep(). Los servicios anteriomente mencionados se fundamentan en la teoría localizada en el capítulo de administración del procesador (operaciones con procesos y sincronización con wait() y signal()), y administración de memoria para la carga y desalojo de procesos de la memoria de la computadora.

Para el manejo del sistema de archivos se incluyeron fopen() para abrir un archivo, fclose() para cerrar un archivo, fgetc() para obtener un caracter de un archivo, fputc() para colocar un caracter en un archivo, fread() para leer un registro de un archivo, fwrite() para escribir un registro de un archivo, fremove() para borrar un archivo, fsck() para verificar la consistencia del sistema de archivos, y frd_dir() para leer todas las entradas del directorio raíz. La parte del sistema de archivos es una parte compleja que requiere una base teórica amplia, aquí indicamos que la base teórica del sistema de archivos aquí implantado se encuentra en los capítulos de administración de dispositivos y administración de archivos, ya que para crear la abstracción de archivo dentro de la computadora es

necesaria la administración lógica y física del disco, para posteriormente estabiecer las interfaces de nivel superior que culminan con los archivos.

Los servicios incluidos para la administración del video y teclado son getch() el cual obtiene un caracter del teclado, gets() que obtiene una cadena terminada con un nulo, kbhit() para detectar si se ha presionado una tecla, putc() para colocar un caracter en la pantalla, puts() para colocar una cadena en la pantalla, adicionalmente se incluye la rutina printf() de C para impresión con formato, pero ésta no soporta el tipo de datos de punto flotante. Los programas utilizados para probar el sistema deben ser escritos en C y deben ser enlazados con las bibliotecas proporcionadas así como también con la cabecera del proceso escrita en ensamblador. Cabe mencionar que los programas nativos de DOS no son soportados, por lo cual si se quiere probar algún programa tendrá que enlazarse con las bibliotecas proporcionadas. Los servicios aquí mencionados tienen su base teórica en el capitulo de administración de dispositivos, ya que tanto el teclado como el video son los dispositivos más utilizados en las computadoras, por lo cual en todos los sistemas operativos se incluye un administrador específico para cada uno de estos

En el sistema operativo aquí mostrado se pueden ejecutar hasta 8 procesos concurrentes, estos procesos son procesos de 64 KB que se ejecutan en segmentos distintos múltiplos de 10000H dentro de la memoria de la PC. El tamaño del sistema operativo es de 128 KB el cual se encuentran localizado en los dos primeros segmentos de la máquina utilizando el segmento 0000H para datos y el 1000H para el código. A continuación mostramos un mapa con la distribución de memoria utilizada por OMNI.
Es importante mencionar que dentro de la PC los valores de los segmentos son: 0000H, 1000H, 2000H, etc., pero al ser procesados por la PC se multiplican por 16 para obtener la dirección real dentro de la memoria, es decir, el segmento 1000H se refiere a la dirección de memoria 10000H.



Mapa de memoria de OMNI

Aquí se observa que los dos primeros segmentos son utilizados por el sistema operativo, en el primer segmento, 0000H, se encuentran los vectores de interrupción, en el segundo segmento, 1000H (dirección de memoria 10000H), se encuentra lo que es propiamente el código del sistema operativo. Los 8 segmentos siguientes son utilizados para colocar los programas en ejecución, observamos que tanto el código como los datos de los procesos que corren en esos segmentos se encuentran en el mismo segmento. Finalmente observamos que todos los segmentos son del mismo tamaño, 10000H y que además se encuentran en una posición fija en memoria, con lo cual se facilita en mucho la administración de memoria. Este modelo de particiones fijas es explicado con mayor detalle en el capítulo de administración de memoria.

Servicios del sistema operativo

Es importante destacar que debido a que solo se tiene una consola, es decir, solamente sobre una pantalla se pueden desplegar caracteres a la vez, es necesario explicar lo siguiente:

Un proceso principal es aquel que tiene actualmente conectado el teclado y la terminal virtual o consola o terminal visible en este momento. Un proceso de fondo es aquel que puede ejecutarse sin teclado conectado. Una terminal virtual es uno de los 8 buffers de video de 80X25 caracteres a la cual se le puede conectar un proceso.

Por ejemplo, un proceso conectado a la terminal virtual 3 puede enviar su salida al tercer buffer de video, y puede obtener datos desde el teclado si el proceso está marcado como proceso principal, en caso de que no este marcado como principal esperará hasta que se convierta en principal y pueda recibir caracteres desde el teclado.

Servicios de teclado

Una de las actividades más importantes cuando utilizamos computadoras es el ingreso de datos a los programas. OMNI cuenta con la rutina getch() para recibir caracteres desde el teclado. Esta rutina trabaja de la siguiente manera, primeramente espera a que exista algún caracter ingresado desde el teclado, si no existe algún caracter ingresado desde el teclado o si el proceso que trata de utilizar la rutina de lectura de caracteres no es el proceso principal, es decir, no tiene asignado el teclado en ese momento, el proceso se pone en un estado de espera por teclado. La rutina gets() es similar excepto que esta espera por una cadena terminada con un nulo.

La rutina kbhit() simplemente consulta el estado actual del buffer del teclado para determinar si existen caracteres disponibles. Esta rutina siempre verifica el estado del buffer de entrada de caracteres, ya que no se pueden leer caracteres si no se

ha detectado la pulsación de una tecla, esto es válido hasta para las teclas que no imprimen carácter como control o shift.

Servicios de video

Las rutinas putc() y puts() escriben un caracter y una cadena respectivamente en la terminal virtual a la cual el procesos solicitante se encuentra conectado. La función printf() provista en la librería de funciones utiliza la rutina puts() para escribir caracteres en el video. Es importante recalcar que la rutina de impresión printf() no soporta el tipo datos de punto flotante Estas rutinas soportan la escritura en terminales virtuales proporcionada por OMNI, es decir, escriben al buffer de video del proceso que solicita la salida a video

Servicios del sistema de archivo

El sistema de archivos es bastante parecido al sistema de archivos de Unix, con funciones del mismo nombre. La rutina fopen() regresa un descriptor del archivo abierto, el cual es mantenido en la cabecera del proceso, por lo que las lecturas y escrituras subsecuentes son ejecutadas utilizando el descriptor. La rutina frd_dir() proporciona la lista de archivos existentes en el directorio. Se cuenta también con las rutinas fread() y fwrite() para leer y escribir datos en un archivo. Finalmente la rutina fsck() es utilizada para verificar la consistencia del sistema de archivos.

Servicios para la administración de procesos

La rutina exec() toma el nombre del programa, una lista de argumentos, una terminal virtual, un indicador de proceso de fondo/principal como los parámetros para la ejecución del programa. El nombre del programa es el nombre del archivo ejecutable. Este programa debe existir como un archivo ejecutable dentro del sistema de archivos, si no existe o es un archivo de texto no será posible su ejecución. La lista de argumentos consiste en los parámetros que son pasados al proceso para su ejecución. El número de terminal virtual es la terminal a la cual se conectará el proceso en ejecución. El parámetro de proceso de fondo/principal especifica cuando el proceso ejecutado deberá correr como proceso de fondo o principal, es decir, si tendrá o no el teclado asignado. Si el proceso es ejecutado como proceso de fondo, el proceso invocante es bloqueado hasta que el proceso invocado concluya su ejecución.

La rutina fork() es muy parecida a la rutina exec() excepto que fork() obtiene una nueva terminal virtual, si es que existe alguna disponible. El proceso ejecutado con fork() corre concurrentemente con el proceso invocante y realiza sus operaciones de entrada y salida con la terminal virtual asignada en el momento de su ejecución.

La rutina get_sem() toma una cadena como argumento de entrada, es decir un identificador o ID que se utilizará para referenciar al semáforo, creando un semáforo para el identificador dado regresando el apuntador al semáforo o regresando un apuntador a un semáforo existente con el identificador dado
La rutina free_sem() decrementa la cuenta del uso de semáforos y libera su memoria alojada si no existe algún proceso utilizando el semáforo dado

La rutina wait_s() proporciona el operador de exclusión mutua para un semáforo dado. Si el semáforo no se encuentra bloqueado es bloqueado en el momento y el control es regresado al proceso invocante. Si el semáforo se encuentra bloqueado el proceso invocante es puesto en un estado de espera por semáforo hasta que el semáforo quede desbloqueado. La función de signal_s() libera el aseguramiento de un semáforo retenido por el proceso

Las rutinas wait_t() y wait_p() ponen al proceso invocante en un estado de espera por un tiempo y por un proceso respectivamente. La rutina wait_t() genera un apuntador a una variable contador, la cual contiene el valor inicialmente dado, el proceso espera hasta que la variable llega al valor de cero.

La rutina exit() permite que el proceso se termine a si mismo. La rutina kill() permite que otros procesos terminen a procesos especificos. Ambas rutinas realizan una limpieza de semáforos y áreas de memoria compartidas por los procesos eliminados. Adicionalmente se cierran también los archivos utilizados por los procesos terminados.

La rutina hsleep() simplemente utiliza una variable cronómetro dedicada y llama a la rutina wait_t() para colocar al solicitante en espera por un tiempo dado. La rutina tsleep() coloca al proceso invocante en espera hasta que el tiempo del sistema esperado haya ocurrido, hora especifica.

Servicios para la administración de memoria compartida

La rutina de get_shm() toma una cadena como identificador o ID, un tamaño de memoria y una locación de memoria alojada como argumento y puede regresar un apuntador a una memoria compartida con la misma identificación y tamaño o puede crear un área de memoria compartida, utilizando la memoria alojada pasada a la función, con la identificación y tamaño solicitado. La rutina free_shm() decrementa el contador de procesos que se encuentran utilizando un área de memoria compartida especifica y en caso de que no existan más procesos apuntando al área ésta es liberada.

Lista de servicios incluidos en el sistema operativo OMNI

Teclado

- getch()
- gets()
- kbhit()

Video

- putc()
- puts()
- printf()

Manejo de datos en disco

- fopen()
- fr_dir()
- fsck()
- fopen()
- fclose()
- fread()
- fwrite()
- fgetc()
- fputc()

Manejo de procesos

- exec()
- fork()
- wait_s()
- wait_p()
- wait_t()
- signal_s()
- kill()
- exit()
- get_sem()
- free_sem()
- sleep()
- hsleep()

Manejo de memoria

- get_shm()
- free_shm()

Implantación del sistema operativo

Como se mencionó anteriormente la maquina utilizada es una compatible 100 % con IBM con un procesador 286 o superior. Los servicios proporcionados por el sistema básico de entrada y salida de la PC, BIOS, son utilizados tanto como es posible para proporcionar los servicios de teclado, despliegue en pantalla, y de la manipulación de la información en disco flexible. Sin embargo, los servicios del BIOS de la PC no fueron diseñados para soportar software multitareas, por lo cual el acceso a estas funciones es serializado utilizando semáforos dedicados. Es importante destacar que no se incluye soporte alguno para el coprocesador matemático.

Aquí comenzaremos con la descripción de como se aprovecha la arquitectura del intel X86 para la implantación de este sistema operativo.
En primera instancia definiremos como se administrará la memoria dentro de nuestro sistema para posteriormente continuar con cada uno de los componentes mayores que lo conforman

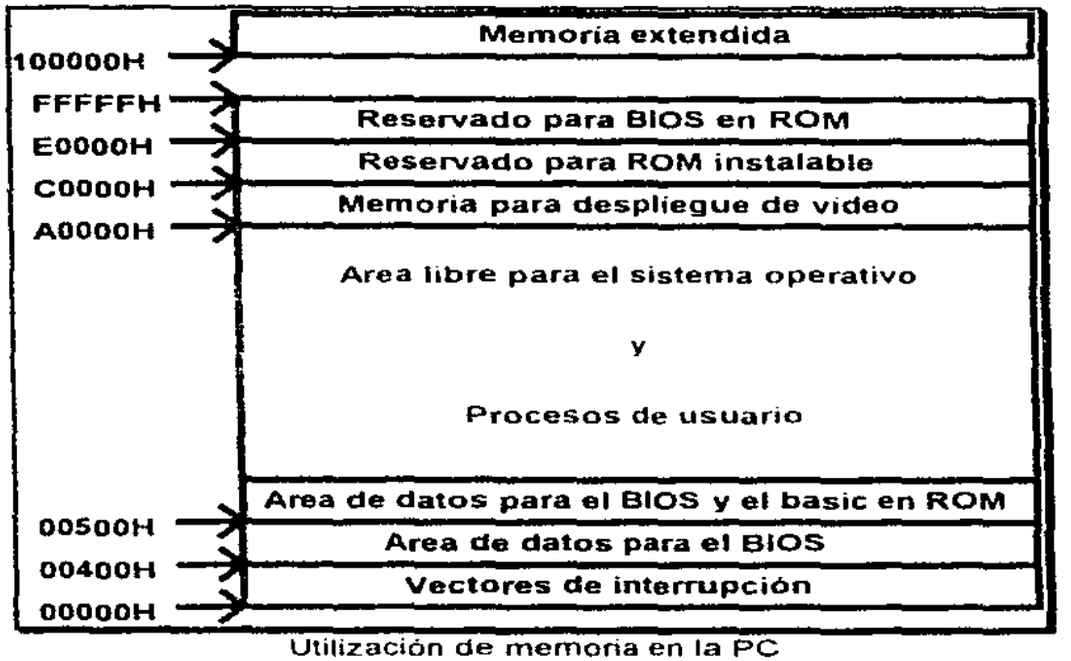
Uno de los componentes más importantes de un sistema operativo es el administrador de memoria, el administrador de memoria dicta las reglas de como se utiliza la memoria y de como es ésta asignada a los procesos. Como se indicó anteriormente este sistema operativo utiliza las facilidades proporcionadas por el procesador Intel x86 para facilitar lo más posible la programación de este sistema operativo.

Utilización de memoria

La arquitectura segmentada de Intel es aprovechada para mantener 8 segmentos separados de 64 KB para código y datos para cada uno de los 8 procesos que se pueden ejecutar en nuestro sistema operativo. El sistema operativo aloja 64 KB de datos, segmento 0000H, y 64 KB para el espacio de código, segmento 1000H. Es importante destacar que los segmentos A000H al F000H se encuentran reservados en la arquitectura de la PC para los buffers de video y el BIOS

El sistema operativo OMNI utiliza los primeros dos segmentos de memoria de la computadora, es decir, 128 KB, el segmento 0000H es utilizado por loso vectores de interrupción, así como también algunas áreas de datos de OMNI, el segmento 1000H es utilizado por el texto del sistema operativo.

Aquí mostramos una gráfica con la distribución de memoria en una computadora PC:



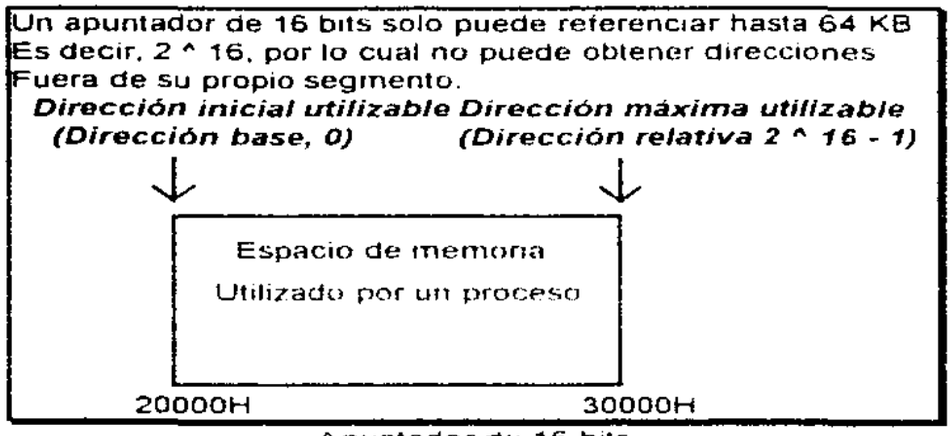
Utilización de memoria en la PC

Cada proceso se aloja en un segmento de 64 KB el cual se encuentra dividido en tres partes principales:

- Datos de la cabecera del proceso
- Código
- Datos

Cuando el proceso se encuentra activo, la pila del sistema se encuentra alojada en el espacio de datos del proceso. Cuando ocurre un cambio de contexto, el estado actual de el proceso es salvado en la pila actual para su próxima restauración, y el apuntador de la pila del proceso es salvado en la cabecera del proceso inmediatamente antes de entregar el control a otro proceso del sistema.

Los apuntadores de los procesos pueden ser tipicamente de 16 bits o apuntadores NEAR, cercanos, los cuales no permiten el acceso a todo el espacio de memoria de los procesos ya que limitan el acceso de los procesos a su propio segmento de datos y código. Esto proporciona un cierto nivel de protección de memoria por hardware, pero no permite la manipulación de segmentos de un tamaño distinto a los 64 KB. Sin embargo, es importante destacar que cuando se efectúan llamadas al sistema operativo los apuntadores para la ejecución de las mismas son de 32 bits para permitir una llamada a un procedimiento fuera del segmento original donde se genera la llamada al sistema



Apuntador de 16 bits

En nuestro diagrama indicamos que se utiliza una dirección base, relativa al segmento que generalmente es considerada como 0000H, es decir, el origen o inicio del segmento, y una dirección relativa máxima direccionable que es 2 ^ 16 - 1; si utilizamos un apuntador de 16 bits y tomamos como base el inicio del segmento, es decir, la dirección relativa 0000H, podemos utilizar el siguiente rango de direcciones de memoria relativas al inicio del segmento

- Dirección 0000H, relativa al inicio del segmento, desplazamiento 0000H a partir del inicio del segmento

Hasta

- Dirección FFFFH, relativa al inicio del segmento, desplazamiento FFFFH a partir del inicio del segmento

Comprobando que de 0000H a FFFFH son 64 KB, 65536 direcciones de memoria incluyendo la 0000H.

Otra parte importante en la construcción de un sistema operativo es la secuencia de carga, conocida como IPL, Initial Program Load; este proceso de carga inicial del sistema operativo es realizado en la arquitectura Intel x86 por un programa muy pequeño, menor a 512 bytes, que después de ser leído del disco flexible intenta cargar al sistema operativo. Aquí mostramos una descripción detallada de este proceso de carga del sistema operativo.

Inicio del sistema, secuencia de carga

El proceso de carga inicial de un sistema operativo es un proceso complejo en el cual intervienen múltiples factores. Uno de los factores principales es la arquitectura del equipo que cargará el sistema operativo, para nuestro caso esta es la arquitectura de los procesadores Intel, en conjunción con un diseño básico de carga inicial creado por IBM, es decir, el BIOS. Este BIOS posee características muy especiales que permiten leer un sistema operativo ya sea del

disco duro, disco flexible, CD-ROM, etc. Aquí mostramos como el BIOS originalmente creado por IBM coloca información en lugares predeterminados de la memoria y como entrega el control a estas direcciones para poder efectuar una carga exitosa del sistema operativo.

Cuando se enciende o resetea el equipo, el BIOS de la computadora intenta leer el primer sector del disco, sector 00H, y si es exitosa la lectura coloca este sector a partir de la dirección 0000:7C00H de la memoria. Cuando termina de leer el sector 0 entrega el control al programa cargado en la dirección de memoria 0000:7C00H, con lo que el programa de carga del sistema operativo procede a leer la primera entrada del directorio del disco para determinar la longitud del sistema operativo OMNI, que no es mayor a 40 KB. Posteriormente el sistema operativo es transferido del disco flexible a la dirección de memoria 1000:0000H, es importante destacar que todos los sectores donde se encuentra el sistema operativo deben ser contiguos dentro del disco flexible, en caso contrario la carga fallará. Cuando termina la carga el control es transferido a la dirección de memoria 1000:0000H donde las primeras operaciones del sistema operativo OMNI son efectuadas para mover el segmento de datos a la dirección 0000:0000H, cuidando de no destruir los vectores de interrupción existentes ni los parámetros iniciales del BIOS. Protegiendo los datos importantes del BIOS así como también sus vectores de interrupción se pueden utilizar sus funciones básicas disminuyendo de manera considerable el código del sistema operativo.

Es importante destacar que para que se efectúe una carga correcta y completa del sistema operativo, el núcleo del sistema operativo debe ser el primer archivo dentro del disco de arranque, es decir, el sistema operativo debe ocupar la primera entrada del directorio del disco flexible. Si se crean los discos del sistema operativo con las utilerías proporcionadas no deben existir problemas de carga del sistema operativo.

Después de explicar el proceso de carga inicial del sistema operativo es necesario definir que parte del sistema operativo es cargada durante este proceso de carga inicial, por lo cual procederemos con la definición del núcleo del sistema operativo.

Núcleo del sistema operativo

El núcleo del sistema operativo es la parte que realiza las tareas de más bajo nivel relacionadas con la computadora, es decir, interactúa directamente con todos los componentes de la computadora, por lo cual es necesario definir cada uno de sus componentes. Comenzaremos indicando primeramente cuales son los archivos que conforman este núcleo, para posteriormente indicar detalladamente su contenido y utilidad.

Las rutinas más importantes que conforman el núcleo del sistema operativo fueron escritas en ensamblador, esto debido a que en C no es fácil interactuar directamente con el hardware de la computadora. El archivo que contiene la mayor parte de estas rutinas es el OMNINIT.ASM, existe otro archivo con rutinas adicionales en ensamblador, este archivo se llama MISCASM.ASM, este archivo contiene muchas rutinas útiles que son utilizadas por varios componentes del sistema operativo.
Las rutinas más importantes no escritas en ensamblador se encuentran en los archivos OMNI.C, PROCESS.C, y MISC.C.

Archivo OMNINIT.ASM

Este archivo contiene el punto de entrada del sistema operativo, así como también el manejador de interrupción para el cambio de contexto y varias funciones adicionales para la alojación de memoria

En el punto de entrada se inicializa el hardware y se cargan los vectores de interrupción de hardware y software apropiados para el sistema OMNI, adicionalmente se prepara un ambiente para soportar las llamadas a la rutina principal de C que está contenida en el archivo OMNI.C Otras rutinas incluidas en este archivo son el depurador del núcleo, el manejador de interrupción de paso a paso y las rutinas para el control de las interfaces de teclado y disco flexible.

Dentro de este archivo se incluye el manejador de interrupción del cambio de contexto, debido a que el cambio de contexto es un punto muy importante que debe de considerarse aquí indicamos con mayor detalle que sucede en este sistema operativo.

Secuencia de cambio de tarea

La secuencia para el cambio de tarea inicia con una rutina en ensamblador llamada TMR_ISR() la cual manipula el cambio de tarea y la interrupción de hardware para el cronómetro. El estado completo de la máquina es salvado en la pila del proceso actual. Si la interrupción es debida a un cronómetro entonces el cronómetro del sistema es incrementado. Si la interrupción es ocasionada por software o debida a la expiración del quantum del proceso actual, entonces la rutina process_switcher() es invocada con el apuntador de pila actual (SP) como parámetro, el process_switcher() entonces salva la pila del proceso actual en su cabecera para posteriormente buscar un proceso listo para ejecución en la lista de procesos. Si la rutina encuentra un proceso que esté listo para ejecución, entonces el apuntador de pila actual (SS:SP, segmento del nuevo proceso SS, apuntador de pila SP) es ajustado con el apuntador de pila del proceso listo para ejecución. Después de esto, el estado de la máquina es restaurado de la nueva pila comenzando su ejecución a partir de la última instrucción ejecutada antes del cambio de contexto.

Pasos del cambio de contexto

Aquí resuminos de manera explicativa los pasos resalizados por OMNI para el cambio de tarea, indicando inicialmente cual es el evento que inicia dicho cambio. El evento que ocasiona el cambio de contexto es la expiración del tiempo de ejecución del proceso, aquí indicamos los pasos del cambio de contexto:

a) Salvar registros del proceso en su pila
b) Salvar el apuntador de la pila del proceso en su cabecera
c) Buscar en la lista de procesos uno que se encuentre listo
d) Leer de la cabecera del proceso listo el apuntador de pila
e) Seleccionar el apuntador de pila como apuntador de pila actual
f) Restaurar los registros de la pila del proceso
g) Ejecutar el proceso seleccionado hasta que espire su tiempo de ejecución

Es importante señalar aquí que este cambio de contexto ocurre muchas veces en un periodo muy corto de tiempo, por lo cual su implantación debe ser eficiente para no desperdiciar demasiados ciclos de reloj en lo que es la administración de procesos.

Archivo OMNI.C

Este archivo contiene el punto de entrada en C a la rutina principal omni(), y el código del sistema operativo llamado por la interrupción de cambio de tarea, el process_switcher(), rutina en la cual se implementan los aspectos de alto nivel de la operación de cambio de tarea incluyendo la rutina de programación de procesos pick_ready_process().

La rutina principal omni() empieza por la inicialización de las rutinas del disco flexible, los componentes del sistema de archivo en segunda instancia y finalmente inicializa las cabeceras de los procesos y sus estructuras de datos asociadas. La rutina termina arrancando el proceso sh, es decir, el shell o intérprete de comandos del sistema operativo utilizando la llamada al sistema exec().

La rutina process_switcher() comienza manejando cualquier solicitud pendiente para cambiar a la siguiente terminal virtual y entonces procede a actualizar el tiempo de CPU utilizado por el proceso actual. El apuntador de pila del proceso existente, el cual es pasado a esta rutina, es salvado en la cabecera del proceso actual. Si la interrupción actual fue una interrupción de hardware, entonces el tiempo de CPU asignado al actual proceso ha terminado y su estado es cambiado de ejecución a listo. En este punto, todos los procesos que están en estado de espera por algún recurso son actualizados y cambiados a estado de listo si la condición de espera se ha cumplido.

Un nuevo proceso es entonces seleccionado para ejecutarse Si un proceso en estado de listo es encontrado en la lista de procesos, entonces el proceso es cambiado a estado de ejecución y el apuntador de pila (SS:SP) de este proceso es regresado a la rutina invocante. Si no existen procesos listos para ejecutarse la rutina regresa un apuntador de pila NULL, nulo, y el nuevo proceso actual es el proceso desocupado, idle(), el cual solamente habilita las interrupciones esperando un nuevo proceso o alguna interrupción en el sistema

Archivo PROCESS.C

Este archivo contiene las rutinas necesarias para administrar los procesos del sistema. Este incluye todas las funciones de semáforos y memoria compartida, así como también las rutinas exec(), fork(), exit() y kill()
Cada proceso arrancado en el sistema tiene una cabecera de proceso al principio del segmento de código del proceso, la cual tiene una longitud de 100H. La excepción a esta regla es el núcleo del sistema operativo, el cual comienza su código al inicio del segmento
Los estados del proceso son los siguientes.

- READY Listo
- WAITING Esperando
- RUNNING Ejecución
- HOLD Retenido
- NONE Ninguno

Estos estados se encuentran especificados en el archivo OMNI.H. Si el estado actual es esperando, es decir, en espera por algún recurso de cómputo, entonces el código de espera especifica el tipo de recurso necesario para continuar con la ejecución, estos estados de espera son:

- KBD Teclado
- SEM Semáforo
- TMR Cronómetro
- WTIME Hora especifica
- PROC Proceso

El apuntador de pila final al terminar un intercambio de proceso es almacenado en SP del proceso que ahora se encuentra en ejecución. El número de la terminal virtual a la que el proceso está conectado es mantenida en la estructura de terminales.
Dentro de los estados de espera por recursos se manejan apuntadores a las estructuras que controlan el recurso por el cual se está esperando. El apuntador w_cntptr apunta a la variable contador por la cual se está esperando si el proceso se encuentra en un estado de espera por cronómetro. De igual manera, w_semptr es un apuntador a la estructura del semáforo si el proceso se encuentra en un

estado de espera por semáforo La entrada w_pid es el identificador del proceso que el proceso en cuestión se encuentra esperando para que termine.
La variable errno es el número de error de los procesos en ejecución. El nombre del archivo que el proceso representa se encuentra en pname, es decir, el nombre del archivo ejecutable a partir del cual éste se genero La variable cpu_time contiene el tiempo de CPU total acumulado por el proceso Finalmente, los descriptores de archivo de cualquier archivo abierto por el proceso son mantenidos en el arreglo fd[].

El pseudo código para la rutina exec() es mostrado en el siguiente cuadro, en el ejemplo se muestra la ejecución en el punto en el que el usuario ingresa un comando desde el shell al punto donde el proceso es inicializado y colocado en la tabla de procesos marcándolo como listo para ejecución

```
test1                   Comando ingresado por el usuario
"sh.c"                  Codigo del shell
        Asegura tabla de procesos y busca un area de memoria libre
        Abre y lee archivo "test1" colocandolo en su area de codigo
        Inicializa la cabecera del nuevo proceso
        Asigna el nuevo apuntador de pila (SP) y coloca los parámetros ingresados en el tope de la pila
        Asigna el apuntador de pila como si hubiera ocurrido un cambio de tarea,
                salva el apuntador de pila resultante en la cabecera del proceso
        Libera la tabla de procesos
        if ( ejecución como proceso de fondo )
                {
                estado actual del proceso = espera por proceso
                estado del nuevo proceso = listo
                genera una interrupción para cambio de tarea
                }
        else
                estado del nuevo proceso = listo
```

pseudocódigo de la rutina exec()

El nombre del programa test1 es ingresado desde el shell lo cual resulta en la llamada a la función exec(). La función exec() inicia asegurando el acceso a la tabla de procesos utilizando el semáforo dedicado para ello generado al momento de arranque del sistema, procediendo posteriormente a localizar alguna área de memoria libre para el nuevo proceso. El archivo del programa "test1" es entonces abierto para lectura y es cargado en su totalidad en el segmento de código asignado al proceso. La cabecera del proceso es entonces inicializada. Un apuntador de pila es inicializado en el tope del área de pila del nuevo proceso y los argumentos, si es que existen, son colocados en el tope de la pila. El estado inicial de la PC es colocado en el tope de la pila como si hubiera sucedido un cambio de tarea. el apuntador de pila resultante (SP) es entonces salvado en la cabecera del proceso. El candado colocado en la tabla de procesos es entonces liberado y es cambiado el estado del proceso nuevo como listo. Si la solicitud de proceso es como proceso de fondo, entonces el proceso actual es colocado (el proceso que llamo a la rutina exec()) en estado de espera por proceso y la interrupción de cambio de tarea es generada

Dentro de este sistema operativo son incluidas también dos rutinas muy importantes para la administración de semáforos, estas rutinas son wait_s() y signal_s() En la rutina wait_s(), las interrupciones son primeramente inhabilitadas para proporcionar acceso exclusivo al semáforo solicitado Si el semáforo no está actualmente asegurado entonces el semáforo es asegurado y el proceso bloqueante es salvado en la estructura del semáforo Las interrupciones son entonces habilitadas y el control es regresado al proceso solicitante Si el semáforo se encontraba asegurado inicialmente, entonces el estado del proceso actual es colocado como esperando por semáforo y el apuntador al semáforo que se está esperando es almacenado en la cabecera del proceso. Un intercambio de tarea es generado para liberar la CPU a otro proceso Cuando el semáforo se desbloquea, la rutina process_switcher() en su actualización de procesos asegura el semáforo y marca el proceso que se encontraba esperando como listo para ejecución

Las rutinas exit() y kill() simplemente limpian cualquier recurso alojado por el proceso a ser terminado y además cambian el estado del proceso a ninguno, indicando que el área de memoria utilizada ahora se encuentra disponible, finalmente genera una interrupción para cambio de tarea para atender al siguiente proceso del sistema o en caso de que no existan más procesos ingresará al ciclo idle() anteriormente descrito.

Archivo MISC.C

Este archivo contiene varias rutinas para el manejo del teclado y la pantalla adicionalmente se incluyen algunas rutinas de utileria. La rutina wtty() es implantada para permitir que la escritura tipo teletipo, tty sea realizada en la página de video virtual especificada La rutina hputc() escribe un solo caracter a la terminal virtual del proceso solicitante obteniendo primeramente acceso exclusivo a la INT10 del BIOS para el manejo de video utilizando una llamada wait_s() y un semáforo exclusivo para la INT10. La rutina wtty() es entonces utilizada para escribir el caracter y la rutina signal_s() es invocada para liberar el semáforo exclusivo para la INT10
La rutina getch() primeramente comprueba si existen caracteres disponibles desde el teclado. Si existe al menos uno, entonces este es obtenido con la rutina _getch() escrita en ensamblador. Si no existen caracteres disponibles o si el proceso solicitante es un proceso de fondo el proceso es puesto entonces en estado de espera por teclado generando la interrupción de cambio de tarea.

Despliegue de salida en terminal virtual

Este sistema operativo soporta múltiples terminales virtuales para el despliegue de caracteres Estas terminales virtuales son simplemente los 8 buffers de video existentes en las PC's con tarjetas de video VGA o superiores, estos buffers

pueden ser seleccionados independientemente para su despliegue. Solamente uno de los buffers de video puede ser mostrado en la pantalla física a la vez. Pulsando la tecla de Impr/Pant, imprime pantalla, se puede seleccionar la siguiente pantalla virtual en la secuencia, si es que existe algún proceso adicional con una pantalla virtual en uso, en caso de que no exista un proceso con pantalla virtual asignada, el sistema no realizará cambio alguno, es decir, dejará activo el buffer de video existente en el momento del pulsado de la tecla Impr/Pant. Como se indicó anteriormente, se pueden manipular hasta 8 terminales virtuales con este sistema operativo

Sistema de archivos

Uno de las partes mas importantes del sistema operativo es el sistema de archivos, se puede decir que el sistema de archivo es la parte con la que el usuario interactúa de manera más frecuente, ya que con éste puede darle un uso más práctico a su información.
El sistema de archivo utilizado es una versión primitiva similar al sistema de archivo de Unix sin el manejo de directorios, el código utiliza la interrupción INT13 del BIOS que proporciona una interfaz para el manejo de la unidad de disco flexible, este manejo está dado por la localización de cabeza, cilindro, sector, byte.

El acceso a la interrupción INT13 es serializado utilizando un semáforo dedicado, esto es debido principalmente a que el BIOS no está diseñado para ser multitareas.

En el sistema de archivo se incluyen las llamadas tradicionales de lectura, escritura, apertura, cierre de archivos, lectura de directorio, y comprobación de sistema de archivos, estas rutinas fueron implantadas haciendo uso extensivo de la INT13 para manejo de disco flexible del BIOS.

Programas de usuario y librería de C

El acceso a las funciones descritas anteriormente del sistema operativo por los programas de usuario es provisto por funciones llamadas similarmente en una librería de C que es enlazada con el programa objeto del usuario. Esta librería de funciones resuelve el direccionamiento de llamadas al sistema operativo utilizando el mecanismo de Intel para manejo de interrupciones de software. La interrupción de software 80H (INT80) es utilizada para este propósito. Los parámetros del procedimiento son pasados en registros del procesador con una función única de código para cada rutina específica del sistema operativo.

Como ejemplo, un programa de usuario emite una llamada a la función gets(s) (obtención de una cadena desde el teclado) la cual está localizada en el archivo en lenguaje C OMNIF.C. Esta rutina coloca el código de la función (4) dentro del

registro de la PC AH, el apuntador lejano a la dirección de la cadena destino en los registros ES:DX, segmento ES y desplazamiento DX, entonces emite una interrupción de software tipo 0x80. Esta interrupción es atrapada por la rutina en ensamblador INT80_TRAP en el archivo "INT80.ASM", el cual es parte del núcleo del sistema operativo y utiliza el segmento de datos del núcleo Este código hace el equivalente de un switch sobre el código de la función invocada en el archivo utilizando AH como desplazamiento para localizar la rutina gets(), entonces apila el apuntador lejano con la dirección de la cadena destino, y llama a la rutina gets() del sistema operativo.

La librería "programa.lib", debe ser enlazada con los programas de usuario, con la rutina de ejecución de C y con el código de inicialización en ensamblador para construir un archivo ejecutable funcional

El archivo en ensamblador CRT.ASM contiene la rutina de inicialización implementada para este sistema operativo, la cual es ejecutada antes de iniciar la rutina principal main(). Una cabecera de tamaño fijo es reservada para cada uno de los procesos al inicio de su área de código. El punto de ingreso para la ejecución inmediatamente sigue esta cabecera y comienza por analizar cualquier argumento de la línea de comandos pasado a través de la llamada para ejecución Después de esto se invoca a la función main() pasando los parámetros argc y argv. Es importante destacar que los parámetros argv y argc son incluidos para todas las llamadas a la función main() de C, esto es debido que en la reglamentación de C se indica que la función main() siempre tiene los dos parámetros antes mencionados.

Si se regresa de la función main() la rutina exit() es automáticamente invocada para terminar el proceso. Un programa ejecutable es entonces construido enlazando el archivo objeto de el programa CRT.ASM, el cual debe ser el primer archivo a enlazar, con el programa objeto del usuario que tenga la llamada a la función main() y con la librería programa.lib.

El shell

El shell es un programa que acepta un conjunto limitado de comandos desde el teclado para poder interactuar con el sistema operativo, en nuestro caso el shell es un programa pequeño que utiliza algunas de las funciones implantadas en el sistema operativo para poder ejecutar programas de una manera simple.
El programa de shell implementado para este sistema operativo es el "sh.c", el cual es un programa simple que espera ingreso de datos desde el teclado en un ciclo infinito.
Este shell soporta los siguientes comandos:

- **open** ***programa parámetros*** Ejecuta el programa dado en una nueva terminal virtual con los parámetros indicados

• **pt**	Lista de estados de los procesos
• **time**	Tiempo de ejecución de los procesos
• **ps**	Lista de los procesos en ejecución
• **ls**	Lista el directorio del disco
• **kill** ***número de proceso***	Elimina al proceso del número dado
• ***programa parámetros***	Ejecuta el programa dado con los parámetros indicados en la misma terminal virtual
• **Unmount**	Desmonta disco actual, permite cambio de disco

Cualquier comando que no está descrito en la lista es considerado como un programa ejecutable dentro del sistema de archivo, por lo cual se hace el intento de ejecutar este comando con la función exec(). El programa del shell utiliza las funciones en la librería omnif.c como interfaz del sistema operativo con excepción de el comando "ps" el cual accesa directamente las cabeceras de proceso de cada proceso para determinar rápidamente la información de éstos.

El parámetro ***programa parámetros*** para el comando **open** puede ser cualquier programa de usuario incluyendo el programa sh.c.
El comando **unmount** es necesario para purgar cualquier petición parcialmente realizada sobre los archivos del disco flexible o para forzar nuevamente la lectura del directorio del disco flexible cuando este es reemplazado

Utilerías para creación del sistema operativo

Se proporcionan algunas utilerías para la compilación y puesta en marcha del sistema operativo, incluyendo la creación de un disco cargable.
Para compilar el sistema operativo se necesita básicamente el siguiente software:

- Turbo C 2.0 o superior
- Macro Assembler 6.0 o superior
- Programa EXE2BIN.EXE de la versión 3.3 de DOS, el cual elimina las cabeceras de los programas para prepararlos para el sistema operativo OMNI
- Programa format.com, para formatear los discos del sistema operativo
- Librería PROGRAMA.LIB (incluida en el disco con los programas fuentes), librería con la cual se deben enlazar los programas que se utilicen en el sistema operativo
- Librería OMNIOS.LIB (Incluida en el disco con los programas fuentes), librería con la cual se deben enlazar los programas del núcleo del sistema operativo para permitir su creación (SISTEMA OPERATIVO)
- DOS versión 5.0 o superior, OS/2 o Windows 95 o Windows NT, (en una ventana de DOS)
- Programa OCP2.EXE, para copiar los programas preparados para el sistema operativo en el disco de arranque
- Programa MKDSKT.EXE, el cual copia el sector de arranque al disco y luego procede a inicializar el sistema de archivo

Se incluyen adicionalmente varios archivos batch para compilar los programas del sistema operativo, estos son:

- **disco.bat** Crea un disco flexible cargable de 1 44 MB, el disco requiere ser formateado con anterioridad
- **lnk.bat** Enlaza los archivos del sistema operativo creando el archivo final del sistema operativo
- **m.bat** El cual compila los programas de usuario que se utilizan dentro del sistema operativo, tales como el shell

Se incluyen adicionalmente dos proyectos para construir las utilerías OCP2.EXE y MKDSKT.EXE, estos proyectos pueden ser leídos por turbo C y pueden ser compilados por éste, los proyectos son:

- MKDSKT.PRJ
- OCP2.PRJ

Los archivos fuentes en ensamblador proporcionados son los siguientes:

- BOOT.ASM Registro de carga del sistema operativo
- INT80.ASM Interfaz con el sistema operativo
- MISCASM.ASM Rutinas varias utilizadas por el sistema operativo
- OMNINIT.ASM Punto de entrada inicial del sistema operativo
- DSKT.ASM Manejo del disco desde el sistema de archivos
- CRT.ASM Cabecera de todos los procesos que corren en el sistema operativo

Los archivos fuentes en C proporcionados son los siguientes:

- DEBUG.C Rutina de depuración del sistema operativo
- MISC.C Rutinas varias para utilizar el sistema operativo
- OMNI.C Rutinas de alto nivel del sistema operativo
- PROCESS.C Servicios de administración de procesos del sistema operativo
- SH.C Shell del sistema operativo
- ALLOC.C Rutinas de alojación del sistema de archivos
- BLKWR.C Rutinas de lectura escritura por bloques del sistema de archivos
- CLOSE.C Rutina close() para cerrar un archivo
- DIR.C Rutinas del sistema de archivo para manejo del directorio
- GETC.C Rutinas getc() y fread() del sistema de archivos
- MKDSKT.C Rutina para crear discos cargables
- OCP2.C Rutina para copiar archivos ejecutables preparados para el sistema operativo
- OPEN.C Rutina open() para abrir un archivo

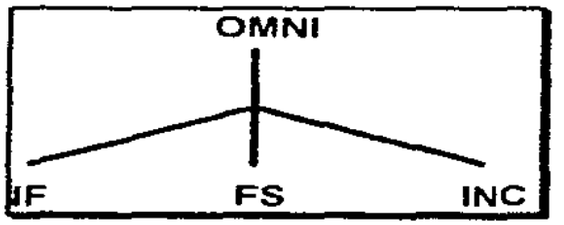
- PUTC.C Rutina putc() y fwrite() del sistema de archivos
- REMOVE.C Rutina remove() del sistema de archivos
- OMNIF.C Librería fuente en C

Los archivos de cabecera .H proporcionados son los siguientes:

- ERRNO.H Archivo de cabecera de los errores
- FS.H Archivo de cabecera para el sistema de archivos
- OMNI.H Archivo de cabecera para el sistema operativo
- OMNIF.H Archivo de cabecera para la librería fuente en C

Adicionalmente se incluye un archivo de configuración **tcconfig.tc** de turbo C el cual debe ser utilizado para compilar las dos utilerías OCP2.C y MKDSKT.C.

El árbol de directorios está dado como sigue:



Donde:

- OMNI Incluye los archivos fuente del núcleo del sistema operativo
- IF Incluye los archivos fuentes de las utilerías, y los archivos de los programas de pruebas
- FS Incluye los archivos fuentes del sistema de archivos
- INC Incluye las cabeceras de los archivos fuentes en C

Es importante destacar que se incluye un archivo en formato **make** el cual puede ser utilizado para compilar todo el sistema operativo. Este archivo se llama:

- **OMNI.MAK**

Se puede utilizar la utilería make.exe de Turbo C para compilar todo el sistema operativo, pero en caso de que los directorios de Turbo C y Macro Assembler tengan otras trayectorias dentro del disco será necesario ajustarlas en el archivo OMNI.MAK. Es importante hacer notar que se necesitan los programas Turbo C y Macro Assembler para la compilación exitosa del sistema operativo.

Detalles de implantación

El mayor esfuerzo de esta implantación está basado en las áreas dependientes de la arquitectura, áreas programadas en lenguaje ensamblador. De las 6000 líneas aproximadamente que tiene el sistema operativo 24 % son del depurador del núcleo, 40 % son rutinas varias en lenguaje ensamblador y el 36 % restante forman el núcleo del sistema operativo
Aquí no se considera el sistema de archivo debido a que por si mismo es un programa grande con funciones muy específicas, el cual tiene una longitud de alrededor de 1300 líneas de código

En esta implantación se entregan los siguientes discos:

1. - Un disco con el código fuente del sistema operativo así como también algunas utilerías para su compilación y prueba. A partir de este disco se puede generar un disco cargable con el cual se pueden probar los programas de ejemplo incluidos en el disco.

2.- Un disco con el sistema operativo funcional, incluyendo el shell y algunos programas de ejemplo como el de hola mundo, las tablas de multiplicar del 1 al 100 y otros programas de utilidad.

- **Modificación del sistema operativo**

La modificación de este sistema operativo no es una cuestión trivial, ya que cualquier modificación por pequeña que parezca puede causar un problema con el funcionamiento del sistema operativo, por esta razón, adicionalmente a la revisión de la tesis aquí expuesta, se recomienda investigar algunos temas para modificar de manera exitosa el sistema operativo OMNI.
También es recomendable realizar una copia de seguridad de los archivos fuentes tanto en ensamblador como en C.
Los temas de estudio adicionales son indicados a continuación.

Es necesario un nivel medio de conocimiento de los siguientes lenguajes de programación:

- Ensamblador de procesador 8086, 80286, 80386, 80486, Pentium, Pentium Pro
- Lenguaje C

Adicionalmente se necesitan conocimientos en las siguientes áreas:

- Arquitectura de computadores basados en Intel X86 compatibles con IBM
- Utilización del BIOS y gestión de interrupciones
- Estructuras de datos

Manual de utilización de OMNI

La parte final de la implementación es la utilización, en esta sección mostraremos como se pueden utilizar los archivos por lotes proporcionados, archivos .BAT, para la creación de un disco cargable, creación de algunos programas y utilización de los mismos dentro del sistema operativo OMNI. Antes de iniciar las pruebas con el sistema operativo es necesario copiar el disco proporcionado en el disco duro del equipo donde se quiera probar este sistema operativo, esto se realiza desde DOS utilizando el comando xcopy. Ya que se puede seleccionar cualquier unidad lógica para instalar todo el paquete del sistema operativo es necesario ajustar las trayectorias de los archivos por lotes, para que se pueda realizar la creación y utilización del sistema operativo. Para hacer la copia del disco flexible al disco duro **c:** incluyendo todos los directorios se utiliza xcopy como sigue:

xcopy a:\ c:\ /s /e

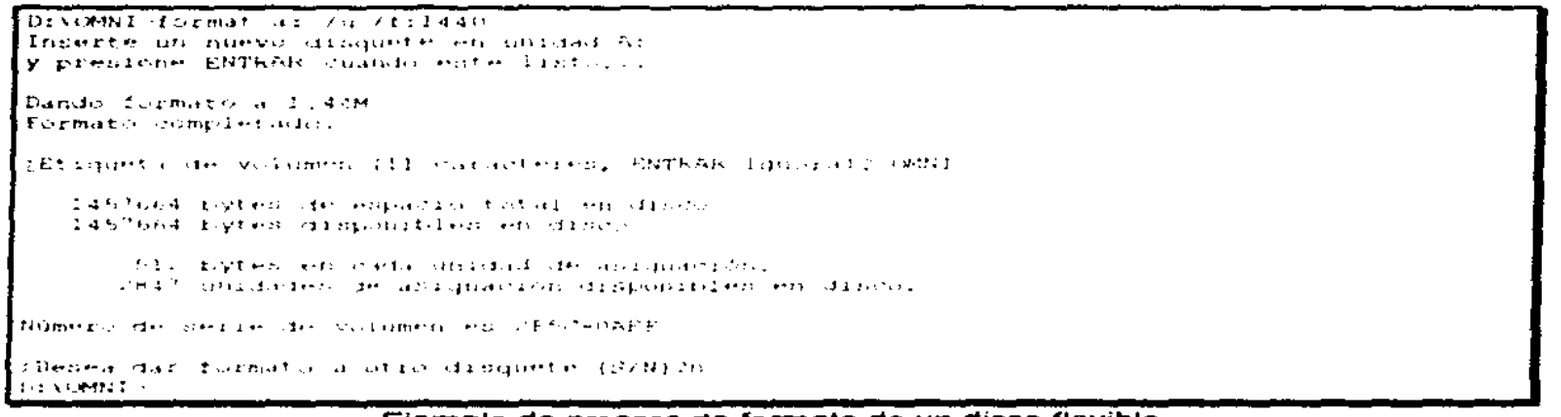
Con lo cual se copiarán todos los subdirectorios del disco **a:** al disco **c:** de la computadora. El programa xcopy indica durante su ejecución los nombres de los archivos que se copian al disco duro, y al terminar indica el número total de archivos copiados.

Creación de un disco cargable

La creación de un disco cargable con el sistema operativo OMNI es relativamente sencilla, primeramente se debe formatear un disco de 3.5 pulgadas con una densidad de 1.44 MB, es necesario que el formato de este disco se realice desde DOS, adicionalmente es importante que el formato del disco sea totalmente exitoso, es decir, que no contenga sectores o bytes dañados dentro del disco. Para dar formato al disco con las características antes mencionadas se emite el siguiente comando desde DOS:

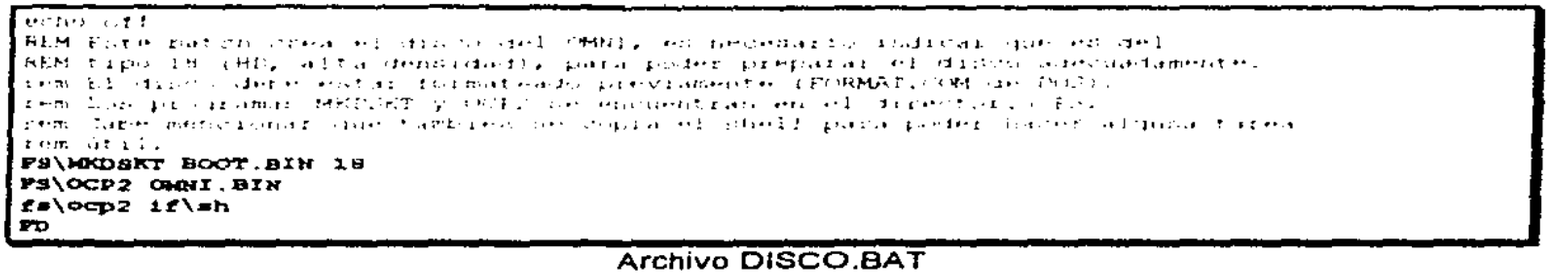
format a: /u /f:1440

Después del cual el sistema operativo DOS solicita se le inserte un disco nuevo para darle el formato deseado. Posteriormente, el DOS solicita una confirmación a la cual se le dará intro para después de unos momentos DOS solicite una etiqueta para el disco. Seleccione la etiqueta de su preferencia para el disco, por ejemplo: SO, OMNI, UNAM, etc. Al terminar de darle formato al disco flexible el DOS muestra una estadística del espacio libre del disco formateado y adicionalmente nos pregunta si deseamos dar formato a otro disco.



Ejemplo de proceso de formato de un disco flexible

Ya que contamos con el disco flexible formateado, es necesario cargar el sistema operativo OMNI en el mismo, para esto es necesario adecuar el archivo DISCO.BAT, dependiendo de donde se encuentren los archivos que se copiarán al disco, el siguiente es un ejemplo de este archivo:



```
FS\MKDSKT BOOT.BIN 18
FS\OCP2 OMNI.BIN
fs\ocp2 if\sh
FD
```

Archivo DISCO.BAT

Aquí observamos que los programas MKDSKT y OCP2 deben encontrarse en el subdirectorio FS para que puedan ser ejecutados. Si se siguió el proceso de instalación mencionado al inicio de la seccion de MANUAL DE UTILIZACIÓN DE OMNI, no es necesario realizar ajustes a este archivo, pero si el sistema fue instalado en el disco duro con los nombres de las trayectorias cambiados deberá ajustarse este archivo por lotes para crear exitosamente el disco cargable.

Es importante destacar que la creación de un disco cargable tarda varios minutos, en algunas máquinas este proceso puede tardar hasta 20 minutos, mientras que en otras puede tardar solamente medio minuto, esto depende básicamente del BIOS de la máquina en la que se esté generando el disco cargable.

Posteriormente se debe ejecutar el archivo por lotes DISCO.BAT. Durante su ejecución aparecen los siguientes datos en la pantalla:

```
C:\OMNI>disco

C:\OMNI>echo off
Tamaño del registro de carga = 450
Numero cilindros:                80
Numero de cabezas:                2
Sectores por pista:              18
Sectores por cilindro:           36
Sectores por FAT:                 2
Numero de archivos:             224
Sectores por directorio:          7
Unidad de diskette para boot: b:
Sector inicial de datos:         31
Escribiendo el registro de arranque ...
Inicializando entrada del directorio ...
Inicializando FAT ...
Todo listo.
Copiando 37826 bytes desde OMNI.BIN a A:\OMNI.BIN ...
   100%    37826 bytes copiados
Copiando  5875 bytes desde kernel.exe a A:\ ...
   100%     5875 bytes copiados
C:\OMNI>
```

Estadística del archivo ejecutable DISCO.BAT

Hasta este momento ya tenemos un disco cargable que podemos utilizar para arrancar el sistema operativo OMNI, lo siguiente será rearrancar la máquina con el disco instalado en la unidad de disco flexible, con lo cual aparecerán los siguientes mensajes:

```
AX:1020 BX:0F9E CX:0000 DX:0000 SI:E500 DI:E500 FLAGS:0287
DS:0900 ES:0900 SS:0900 SP:FFFE BP:FFFE CS:1000 IP:00FE 00EB
1000:00FE EBFA
:
```

Mensajes iniciales de OMNI

Aquí se le debe proporcionar una **g** (letra g) y la tecla de **intro**, para que continúe con la lectura del sistema operativo y del intérprete de comandos.
Después de haber proporcionado la **g** y posteriormente **intro**, el sistema muestra el siguiente mensaje:

```
Iniciando OMNI
OMNI Version 0.0, Fernando Altamirano 10/11/96
Programa sh para OMNI, Version 0.0, Fernando Altamirano Ruiseco
OMNI 0:
```

Mensajes del sistema operativo OMNI después de un arranque exitoso

En este punto ya es posible utilizar los comandos mencionados en la sección de comandos del shell. Por lo cual comenzaremos con una breve descripción del uso de los comandos más utilizados dentro de este sistema operativo.

Aquí mostraremos algunos ejemplos con los comandos del shell.

Comando ls

El primer comando a utilizar es el comando **ls**, el cual muestra las entradas de directorio, es decir, los archivos existentes dentro del disco

Desde la línea de comandos tecleamos **ls**.

Resultado del comando ls

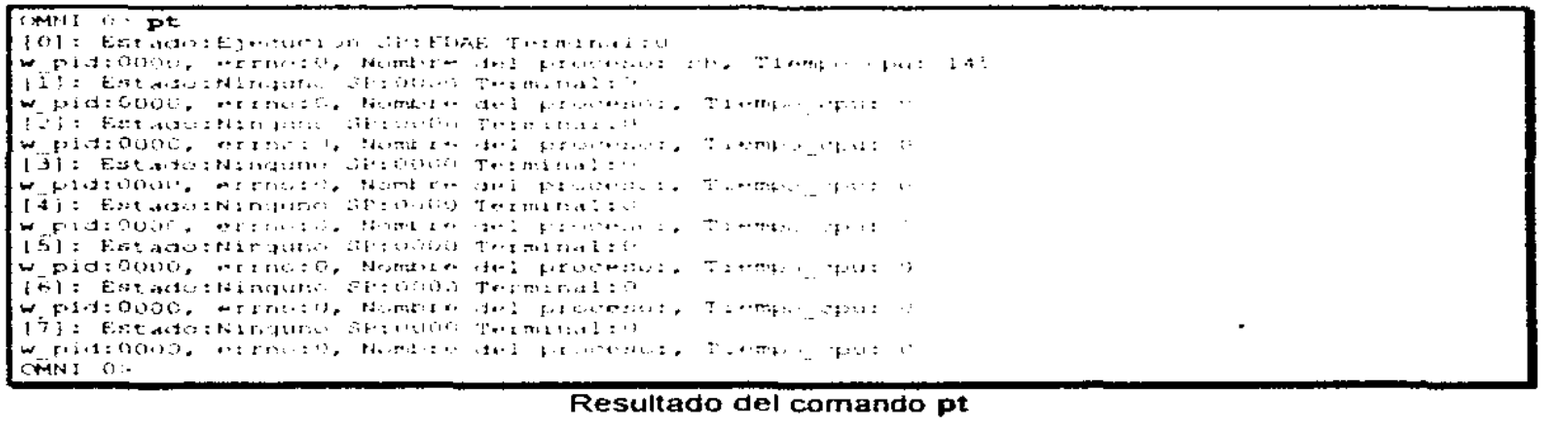
Aquí se observa que en el disco solamente se encuentran dos programas, uno de ellos es el sistema operativo, y el otro es el intérprete de comandos.

Es posible que el comando muestre más archivos, ya que se incluirán dentro de la liberación final varios archivos ejecutables adicionales, estos archivos serán proporcionados para facilitar las pruebas del sistema operativo.

Comando pt

El comando **pt** muestra el estado de todos los procesos, estén o no en ejecución, así como también su tiempo de ejecución, terminal asignada, error actual, nombre del proceso, dirección del apuntador de la pila y el número del área de memoria ocupada.

Desde la línea de comandos tecleamos pt.



Resultado del comando **pt**

Aquí se observa que solo se encuentra un proceso en ejecución, este proceso es el shell, sh, el cual se arranca después de que se concluye exitosamente la carga inicial del sistema operativo.

Comando ps

El comando **ps** muestra los procesos en ejecución, así como también su tiempo de ejecución, terminal y nombre del programa.

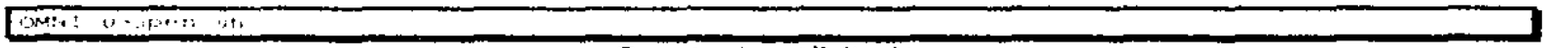
Desde la línea de comandos tecleamos ps.

Resultado del comando ps

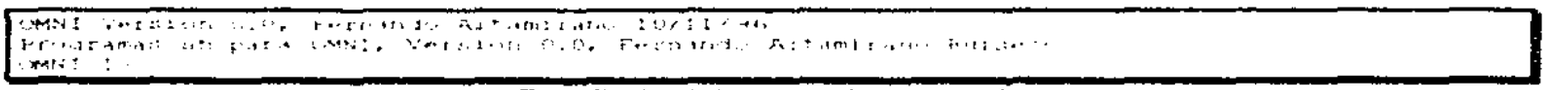
Comando open

El comando **open** carga un programa del disco abriendo una nueva terminal virtual para su ejecución.

Desde la línea de comandos tecleamos **open *programa***. Donde **programa** es el nombre del archivo ejecutable contenido en el disco que se desea utilizar.



Comando solicitado



Resultado del comando **open *sh***

El comando open en este caso primeramente abre una nueva terminal virtual y posteriormente carga y ejecuta el programa solicitado, en nuestro caso el sh, aquí se nota que se abrió una nueva terminal para el proceso debido a que el prompt del shell cambió de **OMNI 0>** a **OMNI 1>**.
Es importante destacar que en este momento ya se encuentran dos programas ejecutándose al mismo tiempo en la computadora, por lo cual si emitimos el comando ps vemos lo siguiente:

Resultado del comando **ps** en terminal virtual 1

Con lo cual comprobamos que existen dos procesos en ejecución al mismo tiempo en la computadora. Es importante destacar que no es de gran utilidad ejecutar múltiples copia del programa shell para comprobar que el sistema es multitareas, por lo cual, como se mencionó anteriormente, es necesario crear programas

propios, enlazarlos con las utilerías proporcionadas, y finalmente copiarlos al disco flexible donde se encuentra el sistema operativo OMNI.

Cambio de terminal virtual, cambio de proceso

Para cambiarse entre las diferentes terminales virtuales, programas en ejecución conectados con diferentes terminales virtuales asignadas, es necesario pulsar la tecla de imprime pantalla, **PrtScrn**, la cual mostrará la siguiente pantalla virtual activa. Cabe mencionar que solo se puede cambiar entre terminales virtuales diferentes, y esto solo se consigue si los programas han sido ejecutados con **open**, ya que si solamente se escribe el nombre del programa a ejecutar no se abre ninguna terminal virtual adicional

Creación de programas de usuario

El proceso de programas de usuario dentro del sistema operativo OMNI es muy similar al proceso de creación de programas en DOS, con la diferencia de que los programas utilizados dentro del sistema operativo OMNI tiene que ser enlazados con las bibliotecas proporcionadas para ello.
Es importante destacar que OMNI no es un sistema operativo muy grande, por lo cual no maneja algunos tipos de datos, tales como los números de punto flotante, esto es debido a que OMNI fue diseñado más que como un sistema productivo, como sistema académico que muestra todas las características de un sistema operativo pequeño pero funcional, aún a pesar de esto pueden construirse algunos programas útiles con las siguientes indicaciones.

Primeramente mostramos el prototipo de programa C que puede compilarse para ejecutarse dentro de OMNI:

```
#include "c:\omni\inc\omnif.h"                    /* Necesaria para utilizar las funciones del
                                                  OMNI */
#include "c:\omni\inc\fs.h"                       /* Necesaria para utilizar las funciones
                                                  del sistema de archivos */
#include "c:\omni\inc\omnilib.h"                  /* Necesaria para acceder las llamadas
                                                  primitivas del sistema operativo, no debe
                                                  ser utilizado para leer las estructuras
                                                  privadas del sistema operativo */

main()
{
Cuerpo del programa
}
```

Prototipo de programa en c dentro de OMNI

Aquí mostramos varias sentencias #include, las cuales no todas son necesarias para los programas de pruebas.

- Primeramente tenemos el archivo omnif.h, el cual es necesario para utilizar las funciones del sistema operativo, tales como alojar memoria, utilizar funciones como wait_s() o signal_s(), etc.

- En segundo lugar tenemos el archivo fs.h, el cual es utilizado durante la ejecución de las funciones del sistema de archivos tales como la creación de archivos, lectura del directorio, etc.
- En tercer lugar tenemos el archivo omni.h, el cual es utilizado para acceder a las funciones más poderosas del sistema operativo, tales como lectura de bloques de control de los procesos, lectura de estructuras de datos privilegiadas del sistema operativo, etc., este archivo no se recomienda para programas de usuario a menos de que se necesiten realizar operaciones directas con las estructuras de datos internas del sistema operativo.

Posteriormente a la declaración de sentencias #include viene el cuerpo del programa en C, iniciando con la función main().

Ejemplo, programa que obtiene las tablas de multiplicar:

```
#include "OMNI Version Compatible"
void main(void)
{
int i,j;
printf("Programa que obtiene las tablas de multiplicar del 1 al 10\n");
for (i=1;i<11;i++)
        for (j=1;j<11;j++)
                printf("%d X %d = %d\n",i,j,i*j);
}
```

Programa de las tablas de multiplicar del 1 al 10

Para compilar este programa bastará salvarlo con un nombre como tablas.c para luego colocarlo en el directorio IF, posteriormente se procede a generar el código para migrarlo al disco con el sistema operativo con el siguiente mandato:

m tablas

con lo cual aparecerán los siguientes mensajes:

```
C:\OMNIVIF>m tablas

C:\OMNIVIF>bcc -ml -mt tablas.c
Turbo C++ Version 3.00  Copyright (c) 1987, 1998 Borland International
tablas.c:

        Available memory 3810508

C:\OMNIVIF>tlink /x /l c:\bc\lib c0t tablas omnif,tablas,,programs
Turbo Link  Version 7.0.  Copyright (c) 1987, 1998 Borland International
Warning: No stack

C:\OMNIVIF>copy tablas.exe tablas

C:\OMNIVIF>
```

Mensajes generados en una compilación exitosa

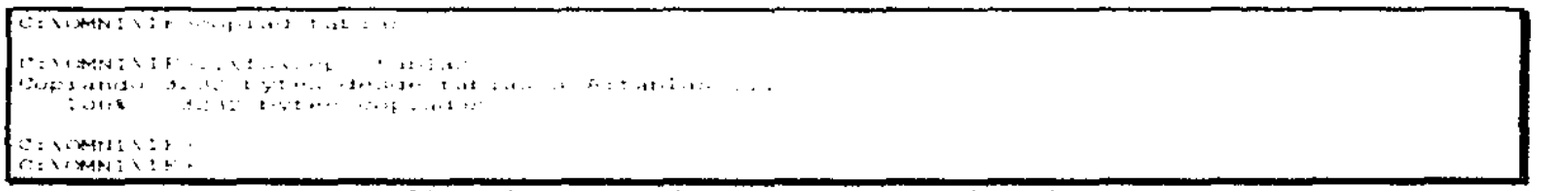
Con lo cual quedará el programa convertido en el formato ejecutable de OMNI, es importante recalcar que al momento de invocar la compilación en el nombre del

programa no se incluye el sufijo .c, es decir, si vamos a compilar el programa tablas.c solamente lo compilamos con el comando **m tablas**.

Ya que el programa no haya dado problemas al momento de la compilación es necesario migrarlo al disco del sistema operativo para probarlo, utilizaremos de ejemplo nuestro programa de las tablas de multiplicar. La migración se realizaria con el comando siguiente:

copiad tablas

Al igual que en el comando de compilación es necesario ejecutar este comando incluyendo solamente el nombre del programa a transportar, sin extensión, debido a que el programa objeto para OMNI fue generado sin extensión. El proceso de transportación al disco del sistema operativo OMNI genera los siguientes mensajes:



Mensajes generados por una migración exitosa

Recomendaciones

Es importante destacar que en este sistema operativo se deben seguir las siguientes recomendaciones para aprovechar el nuevo ambiente:

a) El tamaño máximo del programa es de 40 KB, dejando el espacio restante a la pila y al área de control del programa
b) Solo se puede utilizar el compilador de Borland y el ensamblador de Microsoft
c) No se soportan programas nativos de DOS
d) Solo se soportan programas en C, enlazados adecuadamente con las utilerías y bibliotecas proporcionadas
e) Para modificar el sistema operativo se debe comenzar con el shell, archivo sh.c, para posteriormente modificar los programas más complejos
f) Es importante que no se migren varios programas con el mismo nombre al sistema operativo ya que pueden ocurrir problemas de selección del programa al momento de su ejecución, es decir, se puede seleccionar el programa no adecuado para su ejecución

Con esta guía pueden crearse programas funcionales dentro del sistema operativo OMNI.

◆ CAPITULO VII.- CONCEPTOS AVANZADOS DE SISTEMAS OPERATIVOS

Sistemas operativos distribuidos

La tendencia actual la computación es la de dividir el trabajo en fases que permitan ejecutarse no en el procesador central sino en equipos conectados en forma remota, en computadoras con procesadores múltiples, éstas pueden estar o no en el mismo lugar de trabajo o contar con varios sistemas operativos o uno en común, ser ajenos unos con otros (sistemas híbridos), teniendo solo el protocolo de comunicación que sirve de puente para enlazarse Esta tendencia se hace con el fin de aprovechar ciclos de proceso desperdiciados en los equipos conectados a la red.
El caso de los sistemas operativos distribuidos tiene como fin utilizar las técnicas estándares que permitirán la interconexión con equipos y procesadores diferentes.

Actualmente se vislumbra un avance en teoría de desempeño, como se sabe es más difícil demostrar errores en un equipo de cómputo distribuido, pero los beneficios son mayores que en equipos centralizados. Cabe señalar que el software de desarrollo de los sistemas operativos distribuidos es muy diferente al de los sistemas centralizados, no así las aplicaciones que maneja dicho sistema, la teoría que describe estos sistemas es bastante compleja y requiere de mucha profundidad al implantarse.

Sin duda alguna como se mencionó anteriormente los protocolos y las teorías nuevas sobre este tema están marcando la apertura hacia la evolución de sistemas operativos más avanzados.

La Organización Internacional de Normas dentro de este contexto es esencial, ya que se encarga del modelo de interconexión de sistemas abiertos (OSI), todo aquel equipo que quiera trabajar en forma distribuida, deberá adaptarse a las normas propuestas por esta fundación.

Si bien la barrera de las diferencias ha sido superada, existe el superar la velocidad de transmisión que representa hoy en día más que un adelanto una necesidad. Los canales de comunicación como la línea telefónica o el cable coaxial requieren ser renovados por tecnología más avanzada. En realidad la fibra óptica cumple con las características aunque su costo es muy alto.
Con esta perspectiva la computación distribuida está alcanzando niveles extraordinarios y representa un reto para los desarrolladores de software tanto de sistemas como de aplicación.
Las siguientes puntos tocarán todo lo relacionado a los sistemas operativos distribuidos, la teoría, su desempeño y sus alcances

Definición de un Sistema Operativo Distribuido

Un sistema operativo distribuido es el sistema implementado dentro de un entorno de comunicación tipo red de computadoras autónomas comunicándose y compartiendo recursos, este sistema debe de controlar de manera transparente para el usuario las funciones normales de los sistemas distribuidos.
Como consecuencia de esta afirmación se reúnen dos características:

- Independencia en hardware: Ubicado físicamente en diferentes sitios
- Transparencia en software: El usuario ve al sistema como un único sistema monolítico, de un solo procesador; todo esto ocurre sin que sea advertido al usuario.

Las bases de diseño de un sistema operativo distribuido son las mismas que las de los sistemas operativos tradicionales, sincronización de procesos, bloqueos, planeación, sistemas de archivos, comunicación entre procesos, sin embargo, existen algunos rasgos dentro de estos sistemas distribuidos tales como: la carencia de memoria compartida, la carencia de un reloj global físico.
Además de que los retardos no predecibles en la comunicación hacen que el diseño de un sistema distribuido sea más difícil.
A veces se llega a confundir un sistema distribuido (controlado por un sistema operativo) con una red distribuida ¿cual es la diferencia?, esta diferencia radica en la manera que se accesa a los recursos y no a la manera en que está distribuido el hardware:

Mientras en un sistema operativo distribuido la existencia de múltiples computadoras autónomas y la ejecución de procesos en las mismas es transparente, (no existe la necesidad de enrutar archivos, ejecutar tareas en determinado procesador, buscar y encontrar archivos, etc., todo esto lo debe de hacer automáticamente el sistema) en una red distribuida, por ejemplo, una red de arquitectura cliente-servidor, el usuario explícitamente recurre a los servicios de una máquina determinada, y explícitamente mueve archivos a diferentes rutas, por lo general tendrá que encargarse de administrar de diversas formas la red adecuándola a sus necesidades. Esta es la sencilla definición que hace la gran diferencia; sin embargo, tanto uno como otra (sistema y red) se retroalimentan, teniéndose en consideración al ser diseñados, obedeciendo a principios de un estándar internacional. De aquí se obtiene entonces que es el software y no el hardware el que hace la diferencia entre lo que es una red y lo que es un sistema operativo distribuido.

Ejemplos de Sistemas Distribuidos

En el entorno de un banco con cientos de sucursales a nivel mundial, en donde cada sucursal tiene comunicación con las oficinas centrales y además cada

sucursal tiene una computadora maestra para las transacciones locales, en este caso cualquier cliente que haga una operación sin importar el lugar de la transacción o a que sucursal pertenece la cuenta que se afectará, y si el usuario no nota la diferencia entre este sistema nuevo y el anterior sistema centralizado, entonces se puede considerar un sistema distribuido
Otro caso sería el de una red de estaciones de trabajo en algún departamento de una Universidad, además de las estaciones de trabajo se cuenta con un cuarto de máquinas con una pila de procesadores que no están asignados a un usuario en especifico, sino que actúan de manera dinámica cuando se requieran, cuando un usuario ejecute algún comando este bien podría ser ejecutado en el procesador de la estación de trabajo en donde se encuentra el usuario o bien en algún otra estación de trabajo o en la pila de procesadores no asignados que se encuentran en el cuarto de máquinas, si este sistema se visualiza como un todo en donde existe el tiempo compartido clásico con un único procesador, entonces este sistema puede considerarse como distribuido

Ventajas de los Sistemas Operativos Distribuidos con respecto a los Centralizados

Existen grandes ventajas al utilizar los sistemas distribuidos en vez de los centralizados, consideremos los siguientes:

Economía.- El aspecto económico representa para cualquier usuario una gran influencia, es decir, influye en su decisión, resulta hoy más barato obtener un número adecuado de CPU's que adquirir un computador central.
Velocidad.- Además el computador central no podría superar en rendimiento a los microprocesadores distribuidos, pues al implantarse esta distribución la linealidad del rendimiento del sistema es superior a la del computador central.
Distribución.- La cooperación y la división del trabajo involucran a sistemas distribuidos, de esta manera al comparar la concentración de algún trabajo y la distribución del mismo, la ventaja tiende a irse del lado de la distribución pues existirán recursos compartidos como dispositivos, bases de datos comunicación en tiempo real, etc. y el equilibrio de cargas en el trabajo estará en óptimas condiciones.
Confiabilidad.- Al distribuir la carga, la falla en algún equipo detendrá solo a ese equipo y nada más, el resto seguirá intacto, afectando solo un pequeño porcentaje del total del sistema.
Crecimiento Integral.- Cuando haya necesidad de integrar nuevo hardware o software el sistema distribuido lo permitirá sin riesgo a tener que actualizar hardware o software en otros sentidos o tener problemas en las aplicaciones actuales, bastará con añadir los elementos y configurarlos en el sistema actual, haciendo por consecuencia directa que el sistema obtenga un mejor desempeño.

Desventajas de los Sistemas Operativos Distribuidos

A pesar de tener grandes ventajas los sistemas distribuidos como todos tienen algunas desventajas:

Software.- Esta desventaja radica en el hecho que las aplicaciones y desarrollos de los sistemas operativos distribuidos es un área que exige grandes descubrimientos aún y que la detección de errores en código por dar un ejemplo. resulta muy compleja.

Redes.- Cuando un sistema operativo distribuido se configura para una red existen los errores clásicos de saturación de líneas con la consecuente pérdida de información o retraso en el envío o recepción de la misma, si se desean evitar estos errores se tendría que equipar de buen cableado y otra buena estructura de red que costarían mucho siendo poco factibles sobre todo para entornos medianos y pequeños.

Seguridad.- Si bien se pueden compartir datos y recursos, en ocasiones esto podría sugerir intromisiones a información de tipo privado o de tipo confidencial, requiriéndose entonces mayor seguridad del sistema. Por lo pronto se dirá que para un sistema distribuido se requieren dos parámetros básicos: autenticidad y autorización, el primero se encargará de garantizar que una entidad cualquiera es realmente la que dice ser y el segundo parámetro se encargará de decidir que privilegios tendrá esa entidad y como y donde estarán disponibles.

Conocimiento global del estado del sistema.- El carecer de un reloj global, memoria compartida y además presentar retardos de comunicación no predecibles hacen muy difícil conocer información actual del sistema. Las técnicas usadas son complejas pero necesarias ya que por ejemplo, el conocimiento del orden de los eventos es fundamental para el diseño de sistemas operativos distribuidos.

Todavía con estos puntos en contra los sistemas distribuidos, para muchos, representan la mejor opción en cuanto al costo/beneficio, costo/rendimiento y quizá en los próximos años la implantación de sistemas centralizados sea menos utilizada.

Cuestiones importantes de diseño de sistemas operativos distribuidos

Dentro de las desventajas se examinaron algunos puntos relevantes en el uso de sistemas operativos y que por sus características son fundamentales en el diseño, a continuación se describirán otros puntos que puede resultar en beneficio de la implantación o no de un sistema operativo distribuido, viendo estas como ventajas o desventajas del mismo.

Nombres.- Los nombres son utilizados para hacer referencia a objetos. estos pueden ser computadoras, impresoras, servicios, utilerías, archivos, usuarios, etc.

Por ejemplo, cuando se desea un servicio o comando, se llama a éste con su nombre, este nombre se mapea a un dirección física de memoria haciendo uso de una tabla (directorio), o un algoritmo, o la combinación de ambos. En un sistema distribuido estos directorios pueden ser duplicados y almacenados en diferentes lugares evitando así posibles fallas y agilizando la disponibilidad de este servicio. Para poder tener este esquema de nombres de servicio duplicados se requiere de más capacidad de almacenamiento y requerimientos de sincronización para actualizar todos los directorios duplicados, lo cual puede ser muy difícil de implantar (por el reloj del sistema). Otra forma de implantar los nombres es hacer una partición de los directorios ubicándolos en diferentes lugares, pero el problema esta en el tiempo de búsqueda del nombre, con esto se puede ver que la duplicación de directorios es todavía más confiable que la partición
Todo esto involucra que el algoritmo o la tabla sean muy dependientes del nombre pero además se desea un método en donde el nombre del objeto para su localización no sea un requisito indispensable.

Escalabilidad.- Cuando se pretende implantar un sistema operativo distribuido es necesario pensar en el crecimiento que tendrá en el futuro, evitando que cuando éste crezca se pierda la disponibilidad de ciertos recursos o la degradación del rendimiento general del sistema.

Compatibilidad.- Otro punto de cuestión es el aspecto de la compatibilidad de sistemas operativos. Existen 3 niveles de compatibilidad en los sistemas operativos, nivel binario, nivel de ejecución y nivel de protocolo; el primer nivel es útil debido a que las rutinas se encuentran en lenguaje máquina por ser código binario se pueden ejecutar sin ningún contratiempo en otras máquinas siempre y cuando estas tengan la misma arquitectura con el fin de desarrollar mejoras en el rendimiento. El nivel de ejecución tiene mayor ventaja pues a nivel código fuente las rutinas pueden ser compiladas en máquinas diferentes. El nivel de protocolo es menos restrictivo y es preferible siempre y cuando no se afecte la interoperabilidad general del sistema, en este nivel varias computadoras corriendo diferentes sistemas operativos pueden trabajar con los mismos conjuntos de datos usando el nivel de protocolo; este nivel de compatibilidad es utilizado por ejemplo, en algunos sistemas como SUN para servicios como es el acceso a archivos de red (NFS).

Sincronización de procesos.- Dentro de los sistemas operativos distribuidos el concepto de sincronización se hace a un nivel lógico, ya que no existe memoria compartida; si no existe una técnica adecuada como un algoritmo que permita interactuar con los recursos compartidos el sistema presentará inestabilidad entrando en bloqueos mutuos provocados porque los procesos no fueron despachados en un determinado orden lógico.

Administración de recursos.- Este punto involucra el desempeño del sistema operativo al administrar los recursos el sistema debe de suministrar estos

recursos de manera fácil sin importar si son remotos o locales, para el usuario la ubicación es desconocida, dentro de estos recursos sobresalen los siguientes.

Migración de datos.- Este proceso deberá traer los datos a la máquina que los está solicitando, estos datos deberán estar almacenados (local o remotamente) en un archivo o bien contenidos en un área de memoria (de la misma máquina o de otra). Si estos datos son actualizados deberán también actualizarse en la localidad original
Cuando se habla de migrar datos de archivos, el sistema operativo distribuido lo hace bajo el concepto de *sistema de archivos distribuido* que es el componente que implementa un sistema de archivos común para las diferentes computadoras que existen en la red. La meta del sistema de archivos distribuido es proporcionar la funcionalidad y capacidad de acceso a archivos de manera homogénea para todas las máquinas, independientemente de su ubicación, al igual que lo hace un sistema de archivos de tiempo compartido de un sistema operativo centralizado; a esta propiedad se le denomina ***transparencia de la red***.
Por otro lado cuando se habla de migrar datos de la memoria física de otra computadora, entonces se hace bajo el concepto de administración de memoria compartida distribuida la cual presenta un esquema de un espacio virtual de direcciones compartido por todas las computadoras del sistema, esta implementación es la analogía de aquellos sistemas operativos que si tienen un memoria física compartida, además el sistema debe de minimizar los retardos en la comunicación y proporcionar consistencia en los datos compartidos.
Migración de datos computarizados.- La idea en este punto es la misma que el punto anterior, la diferencia radica en que en lugar de transportar los datos de una máquina a otra o accesar el espacio de direcciones virtual, se hace una petición enviando un mensaje hacia la computadora que se le quiere solicitar la información, esta computadora regresará el mensaje con los resultados solicitados sin necesidad de hacer uso de acceso a archivos o directorios. El mecanismo usado es el de llamadas a procedimientos remotos (RPC) que presenta ventajas pues es más eficiente y seguro que la migración de datos normal.
Planeación distribuida.- En este punto se ve como los procesos son distribuidos de una computadora a otra, es decir, son transportados a una computadora diferente a la que originó el proceso, esto es con el fin de suplir los recursos necesarios del proceso pues en ocasiones localmente la computadora carece de ellos (por ejemplo un co-procesador matemático), el sistema utilizará el concepto de transparencia de red y deberá ser capaz de identificar aquellos procesadores que cumplan con la petición de recursos del proceso para poder asignarlo de una manera directa y eficiente.

Tipos de Arquitectura de Sistemas Distribuidos

La arquitectura de los sistemas distribuidos se pueden clasificar básicamente en tres categorías:

Modelo de minicomputadora.- Consiste en varias minicomputadoras, donde cada computadora soporta múltiples usuarios y provee acceso a recursos remotos. La proporción entre el número de procesadores y el número de usuarios es siempre menor a 1.

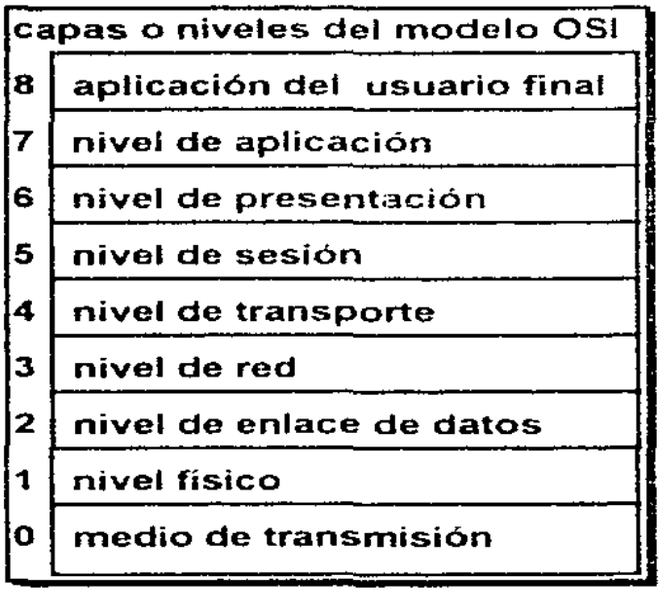
Modelo de estación de trabajo.- Este modelo se caracteriza por tener gran cantidad de estaciones de trabajo (hasta varios miles) Cada usuario tiene una estación de trabajo para su actividad, además con la ayuda de un sistema de archivos distribuido, los usuarios pueden accesar a recursos y archivos compartidos sin importar la localización física de estos. La proporción entre el número de usuarios y procesadores existentes es siempre 1.

Modelo de procesadores compartidos.- En este caso existen procesadores que se asignan a las tareas de los usuarios de manera que al terminar con esa tarea el procesador queda libre para realizar otra tarea con un diferente usuario, en este modelo la proporción entre número de procesadores y usuarios es mayor a 1 y es común ver el trabajo de estos sistemas en donde se requiere ejecutar algoritmos numéricos en paralelo.

Modelo OSI para la computación distribuida y arquitectura de redes

Este modelo es útil en sistemas distribuidos pues permite conocer la forma en que se realizará la comunicación entre equipos locales y remotos.

Este modelo es una descomposición en niveles o capas de manera jerárquica y arbitraria de las funciones de comunicación, cada nivel tiene asociado a sus vez diversos servicios y protocolos. El modelo presenta 7 capas, cada capa n utiliza los servicios de la capa n-1 y a su vez la capa n ofrece servicios a la capa n+1.



Las capas 1 y 2 se denominan capas inferiores, las capas 3 y 4 capas intermedias y las capas 5 a 7 son capas superiores; existen además fuera de la norma OSI dos niveles más: el nivel o capa 0 denominado *medio de transmisión* y el nivel o capa 8 denominado aplicación del usuario final. Los sistemas que se construyen bajo los protocolos OSI se denominan sistemas abiertos, los sistemas que utilizan todas las capas se llaman sistemas finales, los sistemas abiertos que utilizan solo las capas 1 a 3 para conectar sistemas finales se denominan sistemas intermedios.

En los sistemas operativos distribuidos los sistemas intermedios representan la interconexión de dos sistemas con interface de conectividad diferente.

Los niveles físico y de enlace de datos, suelen estar en circuitos integrados, el nivel de red y de transporte, se encuentra en procesadores especializados, los niveles superiores de sesión, presentación, y aplicación suelen estar en el software del sistema en cuestión.

Nivel de Aplicación (capa 7)

Este nivel como erróneamente se ha concebido no representa las aplicaciones finales como serían una hoja de cálculo, un programa de inventarios o un programa de desarrollo, sino más bien el nivel de aplicación es una interfaz con las funciones distribuidas del sistema operativo, como la capacidad de abrir un archivo remoto, o llamar a un procedimiento en otra terminal, etc. Este nivel está compuesto de elementos que no pertenecen al modelo OSI pero que si han tenido una gran aceptación, estos elementos se llaman servicios para aplicaciones específicas. Como ejemplo se tienen los MHS (sistemas para el manejo de

mensajes) utilizados en correos electrónicos de sistemas ampliamente distribuidos, los VTP (protocolo virtual de terminales) usados también en sistemas distribuidos para el soporte de las terminales conectadas a estos

Nivel de Presentación (capa 6)

Este nivel sirve para elegir la sintaxis de datos en el nivel de aplicación que se presentará en sistemas finales, es el encargado de establecer la notación de los datos en base a una sintaxis abstracta, la sintaxis abstracta no es más que un metalenguaje que sirve para acoplar un protocolo de comunicación y traducir caracteres como por ejemplo, ASCII en formatos de intercambio de información de datos binarios, o formatos de números de punto flotante.

Nivel de Sesion (capa 5)

Este nivel corresponde a la organización del flujo de datos y al procesamiento de aplicaciones, en este nivel se establecen puntos de sincronización, tipo de comunicación duplex o semiduplex, etc. sin importar el medio de comunicación, recuperación de errores, etc.; además este nivel como se cree erróneamente no se asocia con el proceso de entrar a una aplicación.

Nivel de Transporte (capa 4)

Esta capa se asocia a la asignación de direcciones en el sistema distribuido, el encaminamiento del mensaje y el control de flujo está asociado con el nivel de red. Además se encarga del control de errores.

Nivel de Red (capa 3)

Este nivel se encarga de hacer el ruteo en base a algoritmos que obtienen el camino óptimo, la dirección no resulta lo más importante sino más bien la interconexión entre los nodos del sistema.

Nivel de enlace de datos (capa 2)

Este nivel contiene el mecanismo para la detección de errores que puedan generarse en el nivel físico, los mensajes enviados en este nivel se subdividen en marcos, los cuales contienen métodos como el CRC que hacen una suma de verificación de los datos enviados poniendo bits especiales de inicio y final del marco, si la suma al llegar al transmisor no coincide se desecha el mensaje y se vuelve a pedir la transmisión.

Nivel físico (capa 1)

Este nivel es el encargado de mantener la estandarización de la parte física de los equipos; dentro de las conexiones más utilizadas para lograr esta estandarización se encuentra la norma EIA RS-232, en este nivel no se analiza el medio sino las interfaces de conexión y el significado de las señales transmitidas entre el transmisor y el receptor.

El nivel 0 y el nivel 8 están fuera del alcance de la norma OSI sin embargo son la representación baja y alta de un sistema final esto es la aplicación del usuario final y el medio de transmisión respectivamente

Sistemas multiprocesadores

Al hablar de computación distribuida podemos hablar también de la computación multiprocesamiento, en realidad existe una gran analogia de una y de otra; cuando se habla de sistemas débilmente acoplados se habla de sistemas distribuidos en donde la tasa de transmisión de los datos es baja y la transmisión entre equipos es lenta debido a que no existe una memoria compartida común, un sistema fuertemente acoplado es un sistema multiprocesamiento de forma distribuida también que presenta una tasa de transmisión alta y la transmisión de datos es a velocidad de memoria. En los sistemas multiprocesadores, se usa una sola memoria compartida en un solo espacio de direcciones común. Si se entiende esta diferencia se podrá ver entonces la división de sistemas multiprocesamiento y sistemas distribuidos o sistemas fuertemente acoplados y sistemas débilmente acoplados.

Definición de un sistema operativo multiprocesador

Un sistema operativo multiprocesador es aquel que opera sobre varios procesadores interconectados y que comparte una memoria común, pudiendo realizar múltiples tareas en forma concurrente y en tiempos más rápidos.

Motivaciones de los sistemas operativos multiprocesador

Dentro de las muchas motivaciones que existen para la realización de sistemas operativos multiprocesadores existen 2 principales: La tolerancia a fallas y la ejecución óptima; la primera motivación es muy valiosa pues en sistemas de un solo procesador la falla del mismo provoca pérdidas de toda índole, sin embargo en un sistema multiprocesamiento la falla de un procesador solo hará que el proceso en ejecución conmute hacia otro procesador libre evitando retrasos en tiempo y dinero, además el hecho de ejecutar tareas en forma concurrente aumenta la eficiencia del sistema logrando más tareas completadas por unidad de

tiempo, pudiendo si es el caso subdividir tareas y ejecutarse en paralelo en diferentes procesadores.

Estructuras de Sistemas Operativos Multiprocesador

Por las características de la estructura de control y por su misma organización, los sistemas operativos multiprocesamiento se dividen en diferentes clases, a continuación se describen tres clases básicas:

Configuracion de supervisor separado

En esta configuración los procesadores mantienen cada uno una copia del núcleo, del supervisor, y una estructura de datos. Utilizan también estructuras de datos compartidas para interactuar entre procesos, los cuales son protegidos por medio de mecanismos de sincronización como son los semáforos; existe para cada procesador un dispositivo de entrada y salida y un sistema de archivos. Hay muy poco acoplamiento entre los procesadores y cada procesador es autónomo como un sistema independiente. Esto provoca dificultad al dividir tareas en procesos paralelos concurrentes pues el problema de sincronización persiste al querer hacer la división de la tarea. Sin embargo en esta configuración se puede tomar ventaja del poco acoplamiento del sistema pues se podrán ejecutar mucho más tareas a la vez, en lugar de la división de una tarea.

Configuración Maestro/Esclavo

En esta configuración existe un procesador denominado maestro que administra a los demás procesadores asignando recursos y planeando las diferentes tareas que requiere el sistema. Esta configuración resulta menos compleja pues en realidad solo existe un procesador trabajando en el sistema, los demás procesadores solo ejecutarán los programas que el procesador maestro ordene. Esta configuración permite a diferencia de la configuración anterior dividir una tarea en varias subtareas y asignarlas a diferentes procesadores. La desventaja radica en que esta configuración es más susceptible a fallas pues un error en el procesador maestro vendría a ser un error generalizado en todos los procesadores del sistema.

Configuración Simetrica

En esta configuración todos los procesadores son tratados por igual en donde no hay ni jerarquías ni existen copias del núcleo ni de estructuras de datos, en realidad lo que se cuida en esta configuración es que el supervisor controle el orden de los procesos para mantener la integridad; todo el sistema simplemente se visualiza como una sección crítica permitiendo que solo un procesador ejecute el sistema operativo a la vez . Este método también es llamado *maestro flotante* pues se podría visualizar como una configuración maestro/esclavo solo que el

procesador maestro iría de un procesador a otro dependiendo quien es el que se apropia en determinado momento del sistema. Esta configuración requiere mecanismos no de sincronización sino de serialización por parte del supervisor que controlen accesos a estructuras de datos compartidas. además es considerada la mejor configuración dentro de los sistemas multiprocesamiento pues es más flexible en cuanto al paralelismo de tareas. es más confiable en cuanto a fallas y hace más eficientes las asignaciones de los recursos. sin embargo por lo mismo resulta más difícil de diseñar. de configurar y de implementar.

Cuestiones importantes de diseño de sistemas operativos multiprocesamiento

Al hablar de un sistema multiprocesamiento se deben de tener en cuenta determinadas cuestiones que son útiles en el diseño del mismo. En general el sistema es complicado pues debe de ser un sistema que soporte la ejecución de tareas concurrentes, además deberá tener la capacidad de explotar hábilmente el poder de los procesadores múltiples, deberá evitar fallas contraproducentes y deberá trabajar correctamente a pesar de la concurrencia física en la ejecución de procesos. Además de estas cuestiones genéricas tenemos las siguientes 5 cuestiones que deben considerarse como básicas para el buen desempeño del sistema operativo multiprocesamiento:

Hilos

La efectividad de la computación en paralelo depende en mayor medida de la ejecución de las primitivas que son usadas para el control del mismo paralelismo dentro de una aplicación. Se ha observado que el procesamiento tradicional impone demasiada sobrecarga en la conmutación entre procesos. A la luz de esto los hilos han sido ampliamente utilizados en sistemas recientes para correr aplicaciones concurrentes en múltiples procesadores.

Tradicionalmente un proceso cuenta con un simple espacio de direcciones y un solo hilo de control que ejecuta un programa dentro de ese mismo espacio, el proceso tiene que inicializar y mantener la información del estado que asume. La información de su estado típicamente comprende tablas de paginación, imágenes de intercambios, descriptores de archivos, peticiones importantes de entrada y salida, valores de registros salvados, etc. Esta información se mantiene por cada proceso y por cada programa dentro del sistema. El volumen de información del estado de los procesos se va incrementando, lo que hace que sea más difícil crear y mantener los procesos y por consiguiente conmutar entre ellos.
Con la llegada de la máquinas multiprocesadores de memoria compartida, fue imprescindible contar con la ventaja de la conmutación entre procesos y con la disponibilidad de concurrencia entre múltiples programas Como se dijo anteriormente, la efectividad de la computación paralela depende en gran medida de las primitivas que son usadas para control del mismo paralelismo dentro del

programa. Por ejemplo, en un sistema de red, los servidores (típicamente máquinas de un solo procesador) suministran servicios de sistema de archivos a todas las máquinas que están conectadas a la red y para cada uno de los usuarios en forma independiente y en solicitudes diversas. Para simplificar el diseño se han mantenido los procesos separados dentro del servidor y solo se conmuta entre ellos para poder servir a todos los usuarios activos en el sistema. este servicio requiere de una conmutación entre procesos muy eficiente.
Para manejar esta situación donde la frecuencia de ocurrencia para crear, mantener y conmutar procesos es alta se han propuesto los llamados *hilos* o *procesos livianos*
Un hilo es la unidad de ejecución más pequeña que se pueda realizar en determinado momento.
Un hilo ejecuta una porción de un programa, cooperando con otros hilos ejecutándose en forma concurrente en el mismo espacio de direcciones; varios hilos pueden formar una tarea. Cada hilo tiene un contador de programa por separado, un bloque de control y una pila de registros activos que describen el estado de la ejecución. El bloque de control contiene el estado de información necesario para la administración de hilos como puede ser poner un hilo en el estado listo y sincronizarlo con otros hilos. La mayoría de la información que guarda un proceso es común a todos los hilos ejecutándose en el mismo espacio de direcciones. La única *sobrecarga* del sistema por así decirlo en cuanto a compartir información común es la de crear y mantener esa información para todos los hilos, la cantidad de información que se necesita salvar al conmutar entre hilos del mismo programa se reduce considerablemente a diferencia de conmutar entre procesos.
Los hilos pueden soportar hilos a nivel usuario y también hilos a nivel de núcleo, a continuación veremos cuales son las diferencias.

Hilos a nivel usuario.- Son un conjunto de rutinas empaquetadas en bibliotecas de tiempo de ejecución que son ligadas al programa en ejecución sin la intervención del núcleo del sistema en la administración de las mismas. Estas rutinas son de bajo costo y son de excelente desempeño a diferencia de los hilos a nivel núcleo, además tienen las siguientes ventajas:

- No existe modificación en el núcleo del sistema operativo para poder soportar este tipo de hilos.
- Existe bastante flexibilidad para personalizar los hilos, cada usuario puede usar las rutinas que más le convengan para la realización de sus tareas, por ejemplo, por un lado se puede implementar una rutina de planeación de prioridad apropiable, mientras otra puede ser una rutina de planeación simple del tipo de primeras entradas -primeras salidas

Mientras existen estas ventajas existen también las siguientes desventajas.

- Estas rutinas están limitadas para aquellas aplicaciones que requieren utilizar el núcleo para determinada actividad, cuando el núcleo se involucra los hilos tratarán a un proceso como un procesador virtual ejecutándose bajo su control, esto no es cierto pues el núcleo es el que controla los procesadores, si llega a ocurrir una operación de entrada y salida por ejemplo, el hilo puede perder la noción de que existe un procesador virtual por lo tanto se generaría un error. En otras palabras la desventaja esta en el vacío que existe entre la planeación y la sincronización generando bloqueos mutuos en secciones críticas
- Los hilos a nivel usuario requieren que las llamadas al sistema sean no bloqueables, si por algún motivo un hilo se llega a bloquear, deberá también prevenir a otros hilos para no ejecutarse Debe tenerse en cuenta que existe una cantidad enorme de llamadas al sistema bajo el requerimiento de bloqueo como son cerrar, abrir, leer y escribir archivos por mencionar algunos.

Hilos a nivel núcleo.- Es el núcleo el que soporta múltiples hilos en ejecución por espacio de direcciones. La coordinación entre la sincronización y la planeación de hilos se lleva a cabo de una manera sencilla, ya que el núcleo mantiene toda la información del estado de todos los hilos.

Las ventajas son las siguientes:

- Existe una gran cantidad de eventos que interactúan con el núcleo siendo en un sentido estricto necesarios los hilos de este tipo.
- Al existir este tipo de eventos se incurre en menos sobrecarga a diferencia de procesamiento tradicional.

Las desventajas son las siguientes:

- La operación de administración de hilos a nivel núcleo causa demasiada sobrecarga a diferencia de los hilos a nivel de usuario. Cada operación involucra la apropiación del núcleo; igualmente cuando se conmuta entre hilos, se debe salvar el estado de los mismos checando parámetros y llevándolos al núcleo para asegurar su validación, haciendo demasiadas funciones y desperdiciando muchos ciclos de reloj.
- Ya que el núcleo es el único que puede realizar la operación de la administración de hilos, deberá ser capaz de suplir en forma razonable todas las necesidades de la aplicación. Esto significa que algunas aplicaciones no necesitarán de ciertos hilos particulares por

lo que el núcleo estará sobresaturado de hilos pasivos, los cuales no será necesarios ejecutarlos, sin embargo, deben de estar presentes, esto también reduce el desempeño general del sistema.

En síntesis se puede decir que los hilos a nivel núcleo son demasiado costosos y los hilos a nivel usuario pueden ofrecer un buen desempeño, pero involucran problemas de coordinación entre sincronización y planeación y también en bloqueos de llamadas al sistema, siendo estos los principales obstáculos para su realización. A pesar de esto los hilos a nivel de usuario son preferidos por los desarrolladores de sistemas multiprocesamiento quienes han generado ciertas soluciones como el definir hilos de primera clase para aquellos procesos que requieren ser tratados físicamente y no virtualmente, evitando que se pierda coordinación entre procesos, además se han implementado vectorizaciones de los procesos logrando que los bloqueos en llamadas al sistema por parte de los hilos sean manejados sin desestabilizar el propio sistema.

Sincronización de Procesos

La ejecución de un programa concurrente en un sistema multiprocesador requiere un robusto mecanismo que serialice el acceso a las estructuras de datos como también a la memoria compartidas, este es un clásico problema de exclusión mutua, sin embargo las soluciones propuestas para sistemas de un solo procesador no son prácticas en los sistemas multiprocesador, por ejemplo, el algoritmo de Dekker o el método de Peterson aplicado en un sistema multiprocesamiento hacen que el modo ocupado y el tiempo de espera consuman la mayor parte del ancho de banda de una interconexión en red y degradando por lo tanto el desempeño del sistema. Para evitar que esto suceda se utilizan ciertas instrucciones denominadas atómicas que leen y escriben en un área simple de memoria (en la memoria principal), haciendo que los datos compartidos elementales como podría ser el incrementar en 1 una variable se realicen poniendo la instrucción atómica en lenguaje máquina en la memoria principal, la exclusión mutua se lleva a cabo en el hardware y no en el software como se realizan en el procesamiento tradicional.
No obstante si algún acceso a datos está determinado por varias instrucciones la exclusión mutua necesitará ciertas primitivas de instrucción a nivel semáforo como son el lock y el unlock pero a nivel atómico en lenguaje máquina para proteger el dato de que sea utilizado por otro proceso, de esta manera la serialización de los procesos estará mejor determinada.
Para hacer que los procesos esperen, las instrucciones atómicas concurrentes a nivel lenguaje máquina se pueden implementar ciertos procesos de espera:

Ocupado esperando.- Esta implementación también llamada en inglés *spin lock* en forma continua ejecuta la instrucción atómica para checar el estado de la variable compartida desaprovechando ciclos del procesador y consumiendo demasiado ancho de banda en la conexión en la red, por lo cual se incrementa el

tráfico en la red, degradando el sistema haciendo que el acceso a memoria se incremente. Esta implementación resulta por lo tanto poco recomendable cuando existen demasiados procesos en el sistema.

Dormido bloqueado.- En esta implementación en vez de que constantemente se cheque el estado de la variable compartida si un proceso falla al obtener el bloqueo, entonces se suspende y un proceso suspendido anteriormente es activado para poder obtener el bloqueo cuando este sea liberado. En el estado suspendido, un proceso no abandona su procesador pero todas las interrupciones son deshabilitadas, excepto la interrupción interprocesadores. Cuando un proceso libera el bloqueo envía una señal por medio de la interrupción interprocesador a todos los procesos suspendidos, siendo uno y solo uno el que pueda modificar la variable compartida, esta implementación reduce el tráfico en la red pero consume aún demasiados ciclos de reloj.

Implementación de Colas.- En esta implementación un proceso en estado de espera se mantiene así hasta que un semáforo lo asigna a una cola global, otra función del semáforo hace que un proceso en la cola global salga de ella para entrar a modificar la variable compartida. Aunque esta implementación elimina tráfico en la red y elimina ciclos de reloj de manera considerable, introduce otro problema pues el algoritmo para las funciones de la cola requiere de varias instrucciones, que sobrecargan el sistema; también la cola debe de compartir una estructura de datos que a su vez debe de ser protegida de los accesos concurrentes de los procesos.
Cada implementación dentro de la sincronización de procesos debe de ser visto de manera general tratando de acoplar al sistema la mejor opción dentro de los requerimientos propios del equipo.

Planificación del Procesador

Para asegurar la eficiencia del hardware del sistema operativo multiprocesador, se deberán de ejecutar las tareas efectivamente en los procesadores del sistema, el sistema deberá de cooperar con el compilador para detectar y explotar lo mejor posible el paralelismo asociado a las tareas dentro de los procesadores con que cuenta el sistema.
Las tareas que se ejecutan pueden ser un simple programa o bien pueden involucrar una serie de diferentes programas. Ya que las diferentes tareas comparten variables o mensajes, en un sistema operativo multiprocesador se llega a dificultar la planeación de procesos, por lo que un planeador de procesos hecho de manera sencilla involucrará diferentes fallas y mal desempeño en todo el entorno del sistema
A continuación se presentan algunas estrategias de planeación del procesador

Co-planeación.- Este método fue presentado para un sistema operativo denominado Medusa por R. Ousterhout, en este método todas las tareas que

ccrren dentro de una aplicación son planeadas con anticipación simultáneamente en los procesadores. Así, cuando una tarea de una aplicación que esté ejecutándose necesite ser apropiable, todas las demás tareas que se desprendan de la misma aplicación también lo serán De esta manera la co-planeación se ejecuta en la porción de tiempo que le pertenece la aplicación y sus tareas asociadas, y en otra porción de tiempo se ejecuta otra aplicación con sus tareas, nótese que la conmutación se realiza a nivel aplicación y no a nivel tareas. La ventaja de este método es que se evitan gastos innecesarios de recursos en lo que se refiere a la sincronía de procesos pues no tiene que estarse constantemente checando variables y estructuras de datos para ver si fueron modificadas, pero si hay demasiada carga con respecto a la planeación general del sistema pues degrada al hacer en frecuentes momentos una depuración de las aplicaciones que han terminado o están en proceso de terminar, sobre todo interfiriendo en la memoria caché y pudiendo corromper parte de la memoria y de las aplicaciones ejecutándose en ese momento.

Planificación inteligente.- Esta estrategia fue propuesta por J. Zahorjan de la universidad de Seattle, en ella se presentan dos aspectos importantes. Primero: evita la apropiación de una tarea cuando esta se encuentra dentro de su sección crítica. Segundo, evita la replaneación de tareas que se encuentran en el modo de espera ocupada, liberándolas en el momento de su apropiación. Cuando una tarea entra dentro de su sección crítica, fija una bandera, el planificador no apropia una tarea si esta bandera está presente, al salir de su sección crítica, la tarea desactiva la bandera. La planeación inteligente elimina el desperdicio de recursos y provee de un rendimiento más sustentable en el sistema general.

Planificación de la ultracomputadora NYU.- Esta estrategia es propuesta por J. Edler de la universidad de Nueva York y combina la estrategia de las dos técnicas anteriores. En esta técnica, las tareas pueden ser formadas dentro de grupos y las tareas en un grupo pueden ser planificadas en cualquiera de las siguientes formas:

- Una tarea puede ser planificada o apropiada de manera normal
- Todas las tareas en un grupo son planificadas o apropiadas simultáneamente (estrategia de co-planeación)
- Las tareas en un grupo nunca son apropiables

Además, una tarea puede prevenir su apropiación sin considerar la política de planeación de su grupo, esto se puede hacer fácilmente implementando un método de espera ocupada. Esta técnica de planificación es flexible ya que permite la selección de variedades de políticas de planificación y estas técnicas pueden ser utilizadas por diferentes grupos de tareas. No obstante estas técnicas no reducen la sobrecarga dentro del contexto de la conmutación ni la degradación del desempeño debido a la corrupción de la memoria caché

Planificación basada en afinidad.- El método propuesto por E. Lazowska y M Squillante, es la primera política de planeación para la resolución del problema de corrupción de caché que se ha venido comentando. En esta política, una tarea es planeada en el procesador donde al final es ejecutada. El problema de la corrupción del caché se resuelve porque la mayor parte del trabajo que involucra la tarea se encuentra en la memoria caché del procesador cuando la tarea esta siendo re-planeada, evitando que haya demasiadas conmutaciones entre procesos para salvar datos, viéndose por lo tanto que en este método una tarea puede ahorrar una gran cantidad de tiempo que normalmente se gasta al conmutar entre tareas. Este método también decrementa el tráfico de los buses.
Sin embargo esta estrategia restringe el balance de tareas en los procesadores presentes, ya que una tarea no puede ser planificada en cualquier procesador sino solo en aquel al cual se asignó desde el principio, por esta razón el sistema cae en un desbalance ya que una tarea quizá tenga que esperar un procesador que está ocupado, mientras otro procesador está sin usarse. Los autores de este sistema presentan a su vez un análisis matemático para balancear el sistema y evitar el desperdicio de procesadores.

Planeación en el sistema operativo MACH.- En el sistema conocido como MACH desarrollado en la Universidad Carnegie Mellon, se planean los procesadores del sistema en grupos independientes denominados *"conjuntos de procesadores"*, estos procesadores ejecutan las tareas que a su vez son divididas en hilos, en cada conjunto de procesadores existen hilos propios que tienen diferentes niveles de prioridad (de 0 a 31), y además tienen una cola de hilos que pertenece a otro conjunto de procesadores, dicha cola debe de estar lista para activar sus hilos cuando el procesador sea liberado, pero antes se tiene que dar prioridad a los hilos propios, generándose entonces otras porciones de tiempo de uso del procesador y aprovechando todos los procesadores y cada porción de tiempo de cada tarea dividida en hilos.
La base de esta estrategia es la de las *"sugerencias"*, una sugerencia es la información en forma de código que el sistema operativo reconoce de cada tarea en el sistema y que a su vez es suplida por el usuario, de esta forma la decisión más acertada para poner las tareas con mayor prioridad para ser ejecutadas en primer lugar recae sobre el tipo de "sugerencia que lleva asignada cada tarea". Esto trae como consecuencia una planeación más inteligente, mejor sincronización y ahorro de ciclos de procesadores que otros métodos no pueden realizar.

Administracion de la memoria

En un sistema operativo multiprocesador la administración de la memoria es primordial y no se deben escapar detalles en el diseño pues una manera de que el sistema sea más eficiente es la forma en que se comparten los datos y la memoria. La memoria se planea desde el aspecto virtual y los puntos que deben ser considerados con mayor atención son:

Portabilidad.- La portabilidad es la capacidad de un sistema operativo de correr en varias computadoras con diferentes arquitecturas. La memoria virtual es un componente del sistema operativo que basa principalmente su diseño en los fundamentos de la arquitectura de la computadora y esto pudiera en determinado momento, en un mal diseño o más aun en la implementación de un nuevo diseño un grave problema para la portabilidad. Por tanto un nuevo diseño debe de basarse en la independencia de arquitectura para evitar problemas posteriores

Datos compartidos.- En los sistemas operativos multiprocesadores, una aplicación es tipicamente ejecutada como una colección de procesos corriendo en diferentes procesadores. Estos procesadores comparten generalmente datos para comunicación y para sincronización, es por lo tanto indispensable proveer en la memoria virtual un soporte para la ejecución de programas en paralelo

Protección.- Cuando la memoria es compartida por varios procesos, la protección de memoria viene a ser un requisito indispensable. El sistema operativo deberá soportar mecanismos como un sistema de memoria virtual que pueda proteger objetos de memoria para evitar accesos no autorizados.

Eficiencia.- Un sistema de memoria virtual puede convertirse en un "cuello de botella" y limitar el desempeño del sistema operativo multiprocesador. Un sistema de memoria virtual deberá ser eficiente en la ejecución de intercambio de direcciones, tablas de paginación, reemplazamiento de páginas, etc. Más aún deberá correr en paralelo para aprovechar la ventaja de los múltiples procesadores.

Se puede ver que la disponibilidad de un buen administrador de memoria puede residir en el propio núcleo del sistema operativo, muchos sistemas en su diseño se han implementado de ese modo, ya que las primitivas del núcleo pueden ser más veloces y de mayor utilidad para desarrolladores y usuarios finales.
Otra de las características que encontramos, corresponde a los objetos de memoria, el objeto de memoria es un recuperador de datos indexado por byte que mapea direcciones virtuales en direcciones reales. El objeto memoria actúa como una memoria secundaria, los datos de éste son disponibles directamente en un espacio de direcciones físico cuando así se requiere, mientras no se requiera permanece en memoria secundaria. En este caso el núcleo actúa como un administrador de memoria caché (la memoria principal es tratada como tal) de objetos de memoria. Una referencia a los datos de un objeto memoria que no se encuentre en el espacio de direcciones físico causa una falla de página y es trasladada para que haga una petición al núcleo de la página solicitada.
Dentro del sistema operativo multiprocesador cada tarea es asignada en un simple espacio de direcciones con paginación, el tamaño del espacio de direcciones está limitado con las capacidades de direccionamiento del hardware. Las páginas varian de tamaño, es decir, una página de memoria de un programa

puede no ser del tamaño de la página que direcciona el hardware, o bien puede utilizar el mismo tamaño.
Una página en un espacio de direcciones de una tarea puede ser localizable o no localizable, una página no localizable no puede ser direccionable por los hilos de una tarea, mientras que una página localizable puede ser accesada directamente
Las páginas localizables no necesariamente consumen recursos del sistema, ya que las páginas en la memoria física (memoria principal) son no localizables hasta que la dirección virtual correspondiente es referenciada. Algunas de las páginas que han sido referenciadas no necesariamente están referenciadas en la memoria principal, ellas pueden ser almacenadas en memoria secundaria y pueden ser colocadas dentro de la memoria principal cuando son pedidas (petición por demanda).
La protección de memoria se puede implementar en el nivel de la página, cada página localizable va acompañada de dos códigos de protección asociados: *el código de protección actual*, el cual corresponde a la protección asociada con una página para referencias de memoria y el *código de protección máxima* el cual limita el valor de la protección actual. Una protección de página consiste de una combinación de permisos de lectura, escritura y ejecución; este tipo de permisos son mutuamente excluyentes. La protección actual solo puede incluir los permisos especificados en la protección máxima y la protección máxima solo puede ser de menor nivel a la establecida, es decir, se pueden borrar privilegios pero no se pueden adicionar nuevos permisos
Al hablar de la independencia de la máquina se puede hablar de un sistema implementado sin importar los diferentes tipos de hardware que se encuentran en el sistema, todo esto se realiza acoplando estructuras de datos que permiten mapear la memoria virtual de una manera común a todos las computadoras, es decir, la memoria virtual se planifica como algo completamente separado del resto del sistema y solo se hacen algunas pocas referencias a funciones del núcleo, de esta manera el sistema será fácilmente adaptable, y así la implementación del sistema de memoria virtual tendrá una interface más transparente que el resto del sistema.

La memoria compartida es muy importante para la ejecución de tareas en paralelo. Estas aplicaciones pueden usar memoria compartida para hacer eficiente la sincronización y comunicación entre procesos, sin estas facilidades las aplicaciones en paralelo resultarían demasiado costosas al tratar de implantar primitivas de sincronización, pues vendrían a degradar con sobrecargas el sistema general.

Otras consideraciones que se han tomado en cuenta para la eficiencia en los sistemas operativos es la implementación paralela, en esta implementación se explota el paralelismo inherente del sistema, todos los algoritmos y estructuras de datos deben de ser diseñados para correr en forma paralela.

Una de las filosofías de un buen sistema multiprocesador es el de usar algoritmos simples y estructuras de datos no complejas, pues de no ser así se desperdiciarían ciclos de CPU sin mejorar el desempeño propio del sistema.
Otro punto de eficiencia es la evaluación y el desempeño, en este caso se propone evaluar funciones de la memoria siguiendo el camino más difícil y viendo si en realidad esta función es o no necesaria. En algunos casos por ejemplo cuando se mapea una página virtual a su dirección física, se pospone esta función hasta que es requerida por una tarea actual, esto evitará que existan bloques de memoria inútiles residentes y que obligarían a gastar ciclos de procesador que no son necesarios

Confiabilidad y tolerancia a fallas

Un sistema operativo multiprocesador tiene redundancia inherente en los procesadores para hacerlo confiable y tolerante a fallas. No obstante, deberá contener esquemas de reconfiguración para reestructurar el sistema en caso de fallas o bien para asegurarse de que no caiga en degradación. Algunos aspectos que se deben de considerar son:

Aislamiento y detección de fallas.- Un sistema operativo multiprocesador deberá avisar cuando detecta una falla y más aún deberá de tomar medidas para aislar las fallas y poder corregirlas. Como se puede recordar, en multiprocesadores débilmente acoplados es bastante simple aislar la falla que ocurre en un procesador, ya que esta falla no influye sobre otros procesadores; eso no sucede con los sistemas fuertemente acoplados, pues las fallas pueden terminar en la pérdida de datos, corrupción de la memoria compartida y principal, y en el peor de los casos causar una falla general del sistema.

Recuperación de fallas.- Después de que la falla de un componente del sistema ha sido detectada, el sistema operativo deberá ser capaz de recuperar los procesos afectados por la falla. El sistema deberá ser capaz de restaurar los estados de los procesos en un estado consistente que permita continuar la ejecución normal

Eficiencia.- La detección de fallas y la recuperación de las mismas deberán de ser mecanismos que tengan un bajo uso de carga en el procesador. Un número de funciones deberá de delegarse a la arquitectura del hardware y el sistema operativo deberá trabajar junto con éste para alcanzar un alto desempeño, es decir, la recuperación debe de ir acompañada de un buen algoritmo que sirva para el hardware en el cual se origino el problema.

En algunos sistemas se pueden aprovechar los siguientes puntos:

Código de detección de error.- La memoria principal usa códigos de detección de errores y el hardware implementa que se particionen memoria, buses unidades

de entrada y salida para convertirlos en simples unidades de manera que cuando se produce una falla el error puede ser detectado en un simple componente, además se implementa paridad par y paridad impar pudiendo rastrear más rápido donde se localiza el error y por consiguiente haciéndolo altamente detectable, de esta manera se presta extrema atención a todas las fallas que se pudieran generar y en caso fortuito corregir.

Comparación de operaciones duplicadas.- El costo de hardware lógico para la generación de código de detección de errores es en algunos sistemas mucho mayor que el costo del componente mismo que se pone en revisión Para estos componentes resulta más barato usar hardware duplicado y compararlo para descubrir posibles fallas Cada componente es duplicado y probado por comparación, ambos componentes ejecutan independientemente sus operaciones comparándose luego las salidas para ver si existen discrepancias; de esta manera se asegura que el código de deteccion de errores es bueno y valdrá el costo de la implementación duplicada.

Protocolo de monitoreo.- Resulta a veces que los dos elementos anteriores no son suficientes para detectar todas las fallas del hardware, especialmente cuando se involucra el tiempo; el protocolo de monitoreo es usado para detectar errores que involucran el tiempo. Estos protocolos trabajan detectando violaciones en la secuencia y en el tiempo de comunicación entre dos componentes, si existe alguna desincronización el protocolo de monitoreo podrá enviar el mensaje de que ha existido un error y se podrá evitar que se generen errores en cascada

Para recuperar una falla en algún componente, se deberá tener donde y cuando ocurrió la falla para reconstruir un proceso consistente y evitar que procesos que estaban unidos al proceso que tuvo la falla no sean afectados y de esa manera puedan continuar con su ejecución.

Hay tres puntos importantes que se deben de tomar encuentra en la recuperación de errores:

Recuperación de fallas en los procesadores.- Un procesador puede fallar de diversas formas, la idea de la recuperación de un error en un procesador está dada en manejar una copia de respaldo del estado del procesador (llamada *shadow*), así cuando un procesador se encuentre con una falla existirá una copia que garantizará la recuperación del proceso a un estado consistente para continuar con su ejecución.

Recuperación de fallas en la memoria principal.- En este punto se considera el concepto de redundancia, es decir, existe una copia de la memoria principal para aquellas páginas o bloques de memoria que pueden escribirse, manteniendo una estructura que controla el núcleo del sistema para que de alguna manera si la memoria llega a fallar en ese bloque la estructura podrá actualizar la copia que

mantiene para poder eliminar el error. Si por algún motivo se quiere accesar una página siguiente del bloque que falló antes de haberlo corregido, se caerá en otra falla la cual esperará hasta que el primer error sea detectado y corregido y después seguirá con el uso normal de la memoria principal.

Recuperación de fallas de entrada y salida.- Las fallas de disco son manejadas usando discos espejo en forma par, en esta caso cuando se escribe en un disco se hará también en sus respectivos discos espejo. Las lecturas son hechas balanceando la mitad hacia los discos espejo, con el fin de evitar gastos de lectura, si un disco falla, el otro disco será usado en la parte donde haya fallado el primero; si un controlador de disco falla se utiliza una ruta alternativa hacia un disco espejo, si no hay ruta hacia un disco espejo se utiliza el segundo disco espejo, con esto se asegura la corrección de los errores de entrada y salida

El diseño de sistemas operativos multiprocesador es difícil, ya que cada sistema deberá ser capaz de soportar la ejecución de muchas tareas y aprovechar la ejecución paralela, además se debe de manejar eficientemente la planeación de tareas entre diferentes procesadores, se deberá de manejar con buen desempeño haciendo que el sistema no se degrade y evitar todas las posibles fallas, deberá de implementar primitivas para la sincronización y comunicación de los procesos con la memoria virtual. De acuerdo con lo que se vió el diseño de un sistema operativo multiprocesador suele ser de los más delicados y difíciles de realizar, existen variadas técnicas en la actualidad y se siguen creando nuevas teorías que pretenden enfocar mejor el uso de estos sistemas. En realidad el máximo aprovechamiento de un sistema operativo multiprocesador se logrará en la medida que se hayan planeado de una manera eficiente todos los componentes tratados en este estudio. Si se logra asegurar el buen funcionamiento de estos principios que se detallaron se puede decir que el sistema se puede implementar con confianza aprovechando y explotando todos los recursos con los que cuenta

Seguridad

Siempre en cualquier proyecto sobre sistemas, se asume una responsabilidad que está implícita y que requiere una atención especial. Dicha responsabilidad lleva características tanto de diseño como de propósitos éticos y el peso mayor tiende hacia la seguridad. Ya sea para salvaguardar el propio sistema o bien para proteger la información que en el se maneja, además es común encontrar sistemas que adquieren mayor responsabilidad pues son sistemas que involucran la seguridad física y la vida de las personas. Con todo esto es necesario pensar en mantener un control estricto cuando se accede a un sistema; si la seguridad es confiable el equipo estará a salvo, si no, entonces será necesario replantearla.

Hoy en día se gastan millones de dólares para proteger los datos, los sistemas y la seguridad personal, y cada vez más la responsabilidad recae en las computadoras y el personal que las diseña y administra.

El lugar que ocupa el sistema operativo en seguridad es de gran importancia, sea cual sea el sistema y su objetivo. Si bien el sistema operativo solo es una pequeña parte del total del software, es la base que requieren todas las aplicaciones para compartir recursos e información.
Hoy en día el incremento de sistemas intercomunicados en red en forma distribuida a través de equipos remotos u otro tipo de comunicaciones da lugar a la creación de controles más complejos para evitar la entrada a intrusos, sin embargo, pareciera que todo esto contradice el principio actual de construir sistemas amigables con el usuario y fáciles de usar, sistemas abiertos que por su misma naturaleza pierden confiabilidad; pero es necesario desmentir que un sistema sencillo es fácil de penetrar y un sistema complejo es difícil de usurpar. los sistemas sencillos tienen un control más definido de verificar el acceso, a diferencia de los sistemas complejos que muy frecuentemente dejan "huecos" de seguridad por lo que un experto no comprometido con la integridad de dicho sistema fácilmente lo hace vulnerable a ataques. A continuación se hará un análisis de lo que se debe considerar en cuestión de la seguridad de un sistema

Debilidades en los sistemas operativos

Cuando se habla de seguridad en un sistema no se puede visualizar como una seguridad completa, un sistema tiene determinado grado de debilidad en realidad no existe el sistema perfecto incapaz de ser penetrado pero existen variadas técnicas que lo hacen capaz de soportar las más duras pruebas de intromisión. Las personas que realizan las intromisiones ya sea por gusto y vanidad o bien por mala intención se les ha denominado *hackers* y *crackers* respectivamente. Estas intromisiones van desde invertir los bits del estado de la máquina, para que pase de estado problema a estado supervisor para lograr los mayores privilegios de un usuario, asignación de mayor tiempo gratis para ellos mismos, control sobre otros usuarios y su información, etc., así como también controlar el área de entradas y salidas donde los dispositivos de almacenamiento, memoria y discos podrán sufrir graves alteraciones. Un método común que se hace para proteger las debilidades de un sistema, es estimar lo que se denomina *factor de labor de penetración*, es decir, cuanto esfuerzo y recursos se tienen que invertir para que el intruso pueda obtener el acceso no autorizado al sistema. Si el factor de labor es grande, se puede pensar que el intruso desistirá, pero no se asegura que se mantenga protegido el sistema. Los siguientes son puntos de ataque que sufren los sistemas y que es necesario proteger.

Asincronía.- Cuando procesos asincronos requieren modificar parámetros, estos procesos deben de verificar la validez de dichos parámetros antes de ejecutar el proceso, de otra manera no se asegura si los parámetros están bien pasados o si existió una intromisión que provoque una falla posterior. De alguna manera si los procesos presentan asincronía, los eventos de verificación deben de estar bien sistematizados para evitar caer en trampas.

Hojeo.- Sucede cuando un usuario constantemente revisa el estado del sistema intentando localizar información con privilegios superiores

Entre lineas.- Este ataque interviene una linea de comunicación empleada por un usuario inactivo, la linea la controla el intruso sin que el administrador se de cuenta.

Código no legitimo.- Este ataque pretende engañar a los usuarios introduciendo código que simula corregir errores en el sistema operativo, dicho código genera "puertas" que posteriormente permiten la entrada no autorizada

Acceso no autorizado.- En este ataque se prohibe el acceso a usuarios legitimos manteniendo el sistema en ciclos infinitos, o utilizando procesos fantasma, con lo cual el sistema se cae. El fin es evitar que los usuarios entren al sistema.

Sincronia.- Aqui se utiliza la sincronización de procesos para pasar información falsa entre los mismos.

Desconexión de linea.- Similar al ataque entre líneas solo que el intruso queda conectado antes de que el sistema reconozca la desconexión cuando el usuario la solicita.

Disfraz.- El más común de los ataques, es el de obtener claves de acceso autorizado de usuarios comunes por medios ilegitimos y clandestinos

Ataque de reconocimiento negativo (NAK).- Cuando un usuario interrumpe un proceso para realizar otra tarea, el intruso puede penetrar al sistema en un modo no protegido apropiándose del control y generando graves resultados.

Engaño del operador.- Muy usual en los casos que el usuario es inexperto, en este caso, el operador del sistema instruye al usuario inexperto en generar procesos que pongan en peligro al sistema completo.

Parásito.- En este ataque el intruso interviene entre el usuario y el procesador por medio de una terminal, interceptando y modificando los mensajes.

Caballo de troya.- Este ataque muy conocido, deja código en el sistema que permitirá un acceso posterior no autorizado, borrando después cualquier indicio de su existencia en el sistema.

Parámetros inesperados.- El intruso dará valores inesperados en llamadas al supervisor que generarán errores en la verificación de legitimidad del sistema

Se han hecho estudios de penetración que proporcionan datos que se deben de considerar para tocar los puntos débiles en la protección de los sistemas. Muchas veces estos puntos se dejan como intranscendentes por lo que el sistema se verá seriamente afectado en su seguridad. A continuación se dará detalles de estos:

Verificación de autenticidad.- Muchos sistemas no controlan si el equipo o los programas son en realidad los que deberían de ser, no hay un control estricto en cuanto a esto, permitiendo que programas o dispositivos falsos puedan dar acceso no autorizado a los intrusos.

Encriptamiento.- Muchos sistemas no encriptan la tabla de contraseñas o los datos en el sistema, siendo presa fácil de intrusos.

Revisión exhaustiva de las rutinas.- En muchos casos los programadores asumen una confianza extrema en su código sin siquiera probar los parámetros que se involucran.

Diseño Vs Realización.- Muchas veces un buen diseño de seguridad no es llevado a su realización o bien, es puesto en marcha en forma inadecuada.

Compartimiento implícito.- En algunos casos el sistema pudiera dejar información privilegiada de su estado sin haberse dado cuenta en el espacio de direcciones del usuario.

Comunicación entre procesos.- Si el sistema no está bien implantado, algún intruso obtendrá a prueba y error información importante del sistema por medio de la comunicación de procesos en donde el sistema envía un mensaje reconociendo sin verificar al usuario y dejando accesar al sistema.

Comprobación de la autenticidad.- Como se mencionó en el punto anterior, la verificación en muchos sistemas no es demasiado segura.

Desconexión de línea.- En este caso, el intruso se basa en algún momento cuando algún usuario pierde la conexión de la línea y el sistema no es capaz de finalizar la sesión, por lo que el usuario queda "activo" permitiendo al intruso usar los recursos que no le pertenecen.

Descuido del operador.- Uno de los puntos más atacados es cuando el operador casi siempre por descuido y en ocasiones obligado, provoca que el intruso tenga privilegios y asignaciones que no le corresponden.

Paso de parámetros por referencia en lugar de por valor.- Cuando se pasan por valor los parámetros resulta más fiable la seguridad que cuando se apunta a direcciones de memoria que pueden en algún momento modificarse, muchos sistemas no toman en cuenta este sencillo consejo.

Contraseñas.- En ocasiones, algunos sistemas permiten contraseñas sencillas que se pueden accesar por intentos repetidos.

Trampas para el intruso.- Casi ningún sistema contiene un mecanismo de trampa para intrusos que quieren penetrar al sistema.

Privilegios.- Muchos programas contienen mayores privilegios de los que debieran tener, pudiendo ser puertas de acceso al sistema para intrusos.

Aislamiento de programas.- En ocasiones los usuarios se prestan programas que pudieran esconder caballos de troya para acceder al sistema, este ataque es muy común y se requiere un estricto control por parte de los administradores del sistema para evitar préstamos de programas de origen desconocido.

Prohibición.- En ocasiones se prohibe a usuarios a utilizar ciertas rutinas pero dichas rutinas pueden no ser bien controladas siendo accesibles por cualquier usuario desde cualquier terminal.

Obtención de información de desecho.- Cuando se depositan datos impresos en la basura, es necesario hacer una revisión de lo que contienen dichos papeles, por lo regular no se hace y las consecuencias son de que el intruso adquiere la información simplemente hurgando el cesto de basura. Muy pocas empresas tienen cuidado de depositar información confidencial en trituradores comerciales.

Blindaje.- Cuando existen comunicaciones vía cable, este debería ser blindado ya que el campo magnético que generan puede servir para accesar la transmisión sin hacer contacto físico. Por ser un poco más costoso este método, los sistemas son implantados sin esta característica con sus posibles riesgos.

Valores límite sobre intentos de entrada.- Cuando una contraseña no es aceptada un considerado número de veces (3 o 4 máximo) el sistema deberá ser bloqueado notificando a la terminal del administrador, muchos sistemas no cuentan con esta estrategia.

Como se aprecia, existen infinidad de debilidades de un sistema, las características varían dependiendo el diseño, pero cuando se plantea la seguridad es necesario analizarla por medio de una tabla de posibles debilidades e ir llenando los huecos en aquellas que pudieran ser más atacadas.

Requisitos de seguridad

La seguridad debe ser vista como una suma total de varias estrategias, es decir, debe de implicar varios niveles de protección como son el nivel de seguridad externo, nivel de seguridad de la interface con el usuario y nivel de seguridad

interno. Estos niveles pueden subdividirse en otros pero en general son los más importantes.

Seguridad externa

Comprende la seguridad del equipo contra ataques físicos que pudieran sufrir, ya sea por intrusos o por algún desastre natural. Es necesario colocar detectores y sensores de humo y de temperatura, así como también restringir el paso de personal y considerar tarjetas electrónicas de identificación, más aún en sistemas más avanzados, el reconocimiento de voz o huellas dactilares es un buen método. El personal que labora en vigilancia debe de ser seleccionado con mucho detalle, evitando riesgos de complicidad y los administradores del sistema deben de realizar funciones divididas que eviten el conocimiento pleno del funcionamiento o en aquellos aspectos del sistema que pudieran ser definitivos para penetrar en el mismo sin autorización.

Seguridad de la interfaz con el usuario

En este apartado es necesario implantar mecanismos que ayuden al reconocimiento pleno de la persona que acceso la seguridad externa y que se encuentra lista a utilizar el sistema, los métodos más comunes son usar contraseñas con límite de acceso. Es también muy útil implantar además de contraseñas, una bitácora con fines estadísticos y de auditoria: a pesar que el usuario no visualiza en el sistema dichas condiciones, el sistema debe de ser capaz de mantenerlos y actualizarlos en intervalos no muy grandes.
También el sistema deberá encriptar las claves y toda la información vital del sistema de manera que no pueda ser accesada por cualquier usuario. Entre más este protegida esta información, el factor de labor de penetración será mayor y el intruso desistirá.

Seguridad interna

La seguridad interna compete por la protección de los datos almacenados, por los cambios que pudiera sufrir el sistema operativo mismo y por controlar las estructuras de datos de los programas. Cuando el usuario obtiene el acceso del sistema vía la interface con el usuario, es necesario generar otros tipos de seguridad para evitar la manipulación de la información general del sistema.
Para esto se ocupan técnicas como la llamada *supervisión de amenazas*, en donde el usuario no obtiene acceso directo a los recursos del sistema, sino que existe un *programa de vigilancia* que se activa para obtener el acceso real al recurso y después pasar los resultados al usuario; si el usuario intenta apropiarse del recurso o del programa de vigilancia, este deberá notificar al administrador para tomar las medidas necesarias.
También es necesario hacer una clasificación de los derechos de acceso con que cuenta el usuario. Dicha clasificación se puede hacer como una matriz en donde

las columnas corresponden a los elementos del sistema u objetos y las filas corresponden a los usuarios, la matriz se llena con los diferentes privilegios que cada usuario tiene sobre los objetos como puede ser la lectura, la escritura, la ejecución, etc Dicha matriz debe de ser una de las entidades más protegidas del sistema.
Cuando se habla de objetos y elementos del sistema estos pueden ser datos almacenados, estructuras de datos, núcleo del sistema, recursos, dispositivos, etc.
Cada una de las diferentes partes que lo conforman deben de tener protección especifica. Ya se menciono la matriz de control de accesos, también existe protección por hardware para dispositivos, en donde se implantan funciones de supervisión en el equipo a nivel hardware, dichas funciones no se pueden modificar y son más rápidas y seguras que el nivel de funciones software. La protección del núcleo del sistema operativo involucra asegurar las funciones de control de accesos, entrada al sistema, supervisión, administración del almacenamiento virtual y real, sistema de archivos, sistema de entrada y salida, etc.
Existen hoy en día estrategias como los núcleos en capas que se hablaron anteriormente, sin embargo es necesario desarrollar nuevos métodos que involucren núcleos pequeños con rutinas apropiadas para salvaguardar la seguridad del sistema.
El manejo de información en un sistema requiere ser a menudo encriptado. El estudio de la criptografía ya no es solo interés militar y de gobierno, sino va más allá, a niveles de espionaje comercial y de empresas y transacciones que involucren dinero. La criptografía es toda una metodología que requiere un estudio exhaustivo, sin embargo podemos hablar de ella como métodos de funciones irreversibles que son construidas con facilidad y para encontrar su función inversa, es decir, reconstruirlas, es necesario conocer cierto parámetro, si no se conoce el parámetro es casi imposible resolverla. Los números primos juegan un papel importante, así como la descomposición factorial, cuando se multiplican dos números primos, el resultado no es otro número primo, pero si se tiene el resultado (es decir, la función inversa) y se desean conocer cuales fueron los dos números primos que lo generaron, resulta más difícil encontrar cuales fueron aquellos dos números que lo generaron, sobre todo si son números de 10 dígitos o más, esta es una función irreversible.
A mediados de los 70, un grupo de investigadores encabezados por Ron Rivest, Ad Shamir y Leonard Adelman, descubrieron la noción de utilizar la descomposición factorial en la encriptación e inventaron el criptosistema RSA (por las iniciales de sus apellidos). Estos científicos retaron al mundo a descomponer un número primo de 130 dígitos y encontrar las dos raíces que lo generaron, ellos asumían que la solución tardaría millones de años, sin embargo en 1993 un grupo de 600 académicos alrededor del mundo y con las ventajas de la nueva tecnología y de la red internet, lograron en menos de un año con sistemas distribuidos y multiprocesamiento desencriptar el número en sus dos raíces, un número primo de 64 dígitos y el otro de 65 y leer el mensaje contenido en él. A

pesar de que se solucionó el reto, se puede ver que el factor de labor es grande y por lo tanto poco factible para casos prácticos Por lo que el criptosistema RSA es aceptable.

Poco después y basados en los conceptos anteriores en 1977 Whitfield Diffe y Martin Hellamn inventaron la encriptación mediante clave pública, donde los datos pueden ser encriptados con una clave pública (user name) conocida por todos y una clave privada (password) que solo la persona que la tiene puede desencriptar los datos. Este método de encriptación es actualmente utilizado en la red internet con buenos resultados
Se han clasificado diferentes tipos de encriptación haciéndolos seguros o poco seguros, una de las mejores clasificaciones está dada por el departamento de defensa de Estados Unidos y ha sido utilizado como recurso de la mayoria de los diseñadores de sistemas alrededor del mundo, sin embargo faltan métodos que permitan asegurar todavía más los datos, soportando la nueva tecnologia que se abre paso de manera ilimitada y que también constituye un factor de riesgo

Gusanos y Virus como ataque a sistemas altamente seguros

A pesar de contar con diferentes métodos de seguridad, los sistemas hoy en dia son vulnerables a un reciente ataque masivo de programas que se reproducen a si mismos y que a diferencia de los penetradores e intrusos afectan de manera más directa y en ocasiones sin hacerse notar a los sistemas, siendo capaces de alterar al sistema tanto en su desempeño como en su información, dichos programas se denominan gusanos y virus. La diferencia entre un virus y un gusano radica en que el primero infecta a los archivos modificándolos o provocando su destrucción, los virus están presentes en sistemas distribuidos o centralizados, por el contrario el gusano, también se reproduce pero solo en sistemas distribuidos, consumiendo tiempo de procesos y saturando recursos.

Como antecedentes históricos de los virus y gusanos resalta el informe de John Von Newman en 1942 titulado "Teoria y Organización de Autómatas Complicados", en donde por primera vez se habla de los programas que se reproducen asi mismos. Cabe señalar que los programas en ese entonces se "alambraban" pues las primeras computadoras utilizaban las conexiones fisicas para trabajar. En la década de los 50 en los laboratorios Bell, tres cientificos H. Douglas McIlroy, Victor Vysottsky y Robert Morris desarrollaron un juego llamado "core wars" que consistia en hacer "programas" que pudieran atacar una computadora ficticia, estos programas deberian de utilizar técnicas como ocultamiento, reproducción muy semejantes a los virus actuales. Más adelante en 1972 en la empresa AT&T en su revista "Software: Practice and Experience" describe otrc juego denominado "darwin" que es un programa polimórfico capaz de evolucionar, y en ese mismo periodo John Shoch y Jon Hupp, investigadores de Xerox PARC generan programas de diagnóstico en sistemas distribuidos que permitían ver si los equipos trabajaban normalmente o tenian problemas, estos

programas se duplicaban en los diferentes equipos, este es el antecesor del gusano de ARPANET que apareció un par de años después. Este gusano llamado "creeper" desarrollado por otro investigador llamado Bob Thomas Este gusano era benigno y solo emitia un mensaje divertido, para ese entonces también apareció lo que podría ser la primera "vacuna", un programa llamado "reaper" que eliminaba al gusano. En 1981 apareció un programa para Apple II llamado "elk cloner" que se duplicaba cada vez que se escribía un verso, al siguiente año apareció otro programa para la misma plataforma desarrollado por Jim Hauser y llamado por el mismo "electronic hitchhicker", un programa que se adheria a otro sin ser detectado. 1983 marca el año en el cual este tipo de programas adquieren el nombre de "virus". Ken Thompson ganador del premio A.M. Turing of Association of Computing Machinery insta a experimentar en este tipo de programas y Fred Cohen utilizando una minicomputadora VAX 11/750 crea un programa capaz de no ser detectado, modificar otro programa y posteriormente duplicarse, era el primer virus reconocido. En 1984 en la revista Scientific American A.K. Dewdney difunde la descripción del juego "core wars", dando mayores expectativas para realizar experimentos. En 1986 aparece lo que se denomina el primer virus de una PC con MS-DOS, a este virus se le denominó "brain" y no era dañino, posteriormente fueron apareciendo más virus y con ello surgen investigaciones como la de Ralf Burger acerca de las posibilidades de infectar archivos propios del sistema y de comandos Todos estos son hechos importantes, pero quizá el más sobresaliente resulte el ocurrido el 2 de noviembre de 1988 en Estados Unidos, donde Robert Morris Jr., hijo de aquel científico de los laboratorios Bell, estudiante graduado de la Universidad de Cornell con apenas 23 años, aprovecha ciertos "agujeros" en la seguridad de la red INTERNET, afectando cerca de 6000 equipos VAX y SUN de corporaciones de investigación y militares, es sin duda un hecho trascendente porque resulta ser el primer caso juzgado por el gobierno con una pena de tres años de libertad condicional, 10,000 dólares y 400 horas de servicio. El gusano no causó daño a los archivos pero consumió gran cantidad de recursos y horas/hombre durante un lapso de 36 horas. Cuando por fin se detectó el código binario y se analizó, los expertos consideraron que era un programa no muy bien escrito y que no afectaba los defectos de seguridad más sutiles de Unix

Todo esto dió fundamentos para pensar en el desarrollo de leyes que sirvieran para proteger los sistemas de ataques de virus y gusanos, sin lugar a dudas la difusión de programas sin licencia, instan a la generación de tales ataques, pero también pueden ser causa de venganzas o hasta de descuido de programadores. En nuestro país poco se ha difundido en cuestiones de organización y leyes acerca de esta clase especial de ataques, aún en el programa de desarrollo informático 1995-2000 son nulos los puntos que tocan esta cuestión. Sin lugar a dudas los virus que destruyen y manipulan los recursos del sistema de manera drástica son peligrosos, pero más lo son los virus que dañan la información sin dejar rastros para poder detectarlos. No es necesario repetir que la seguridad de un sistema debe de ser uno de los puntos de diseño y que se debe de considerar

a fondo el ataque de virus de manera que los sistemas estén respaldados tanto físicamente como legalmente

Desempeño

Cuando se habla del desempeño en un sistema operativo se entran en un análisis que puede ser tan profundo o tan superficial como se quiera considerar, sin embargo un buen análisis resulta demasiado difícil porque es casi imposible analizar cada una de las partes que integran el sistema tanto en hardware como software y estudiar las diferentes técnicas que emplean los diseñadores para mejorar el desempeño

En este caso salen a resaltar dos puntos.

- Se requieren diferentes unidades de medida para conocer el desempeño del sistema.
- Se requiere la determinación de conceptos estándar entre los diferentes sistemas para poder compararlos y medir su desempeño

De hecho el desempeño de un sistema viene a ser un atributo necesario para poder implementarlo, más aún para adquirirlo o no.

Definición de desempeño

El desempeño o rendimiento de un sistema operativo es la capacidad de respuesta que brinda a las diferentes tareas que se van presentando y la manera como el sistema cumple con su objetivo; el desempeño se puede medir de varias maneras. En ocasiones las mediciones resultan subjetivas y es difícil expresar el desempeño para determinados parámetros. Un elemento importante que se destaca al medir el desempeño es el tiempo; cuando se desea que un sistema reduzca *el tiempo de respuesta* se habla del tiempo que lleva el sistema entre el principio y fin de la tarea, llamado en ocasiones *tiempo de ejecución*. Otro aspecto que se considera es la *productividad* del sistema, es decir, la cantidad de trabajo realizado en determinado tiempo. Todos estos parámetros se denominan medidas de desempeño, las cuales se pueden agrupar en dos categorías: las enfocadas al usuario (tiempo de ejecución) y las enfocadas al sistema (productividad).

Ya que todas las variables que involucran la medida de desempeño son probabilísticas, es necesario utilizar técnicas apropiadas que permitan acercarse más a la realidad antes que esperar una respuesta definitiva. Otras medidas de desempeño en un sistema son las siguientes:

Carga de trabajo.- Es la medición que se hace del trabajo introducido en el sistema el cual debe de ser procesado en condiciones normales de manera que el sistema se considere aceptable.

Capacidad.- Es una medida de producción utilizada para asegurarse que el sistema esta preparado a recibir trabajos inmediatamente después que finaliza otros.

Utilización.- Corresponde a la medida de la fracción de tiempo en la cual está en uso algún recurso. En ocasiones está medida puede resultar por si sola engañosa, pues no se puede definir si el recurso utilizado es en realidad aprovechado para la terminación del trabajo o tarea general o bien si solo está consumiendo ciclos de reloj inútiles

Técnicas de evaluación del desempeño

Tiempos.- El parámetro de tiempo resulta importante sobre todo cuando se desea hacer una comparación rápida entre dos o más sistemas. Hoy en día se han clasificado máquinas de acuerdo a su velocidad de procesamiento y se han denominado sistemas MIPS o BIPS y hasta TIPS, pero en realidad el tiempo no suele ser suficiente para evaluar de una manera general el sistema.

Mezcla de instrucciones.- Esta técnica evalúa un promedio de instrucciones que pueden ser las de uso más frecuente en el sistema y no necesariamente para una misma tarea; se pondera un valor más o menos significativo y se obtiene un resultado que en la mayoría de las veces es solo uno de muchos resultados que influye para definir el rendimiento del sistema en general.

Programas Núcleo.- Las dos técnicas anteriores carecen de análisis más completos sobre el sistema, sin embargo esta técnica es mejor pues abarca un rango mayor de componentes. En esta técnica se utiliza un programa llamado programa núcleo (no tiene nada que ver el núcleo del sistema operativo) que representa una serie de instrucciones que el fabricante puede implementar al momento de instalar el sistema. Esta técnica es más efectiva que las dos anteriores pero requiere de mayor esfuerzo y tiempo para programar las rutinas que servirán de programa núcleo. Aunque esta técnica hace más claro el desempeño del sistema, el resultado del mismo obtenido en las pruebas resulta menor cuando se corren las aplicaciones reales.

Modelos analíticos.- En este método se representa el sistema a examinar como un modelo matemático como puede ser el de teoría de líneas de espera o el de cadenas de Markov. Sin embargo presenta ciertas desventajas pues se requiere de evaluadores con amplia experiencia matemática, además para sistemas sencillos resulta una solución bastante aceptable, pero hoy en día cuando más complejos son los sistemas, los modelos analíticos muchas veces caen en simplificaciones con tal de evitar verdaderos problemas de análisis perdiendo con esto un buen resultado del desempeño del sistema.

Con todo esto los evaluadores deben de contar con bastantes técnicas y habilidades matemáticas que los lleven a realizar distintos análisis, teniendo todos estos en forma conjunta algún refuerzo y validez que los lleve a ser aceptable.

Programas de prueba.- Este es uno de los métodos más difundidos para medir el desempeño de un sistema. Consiste en instalar programas de aplicación reales en diferentes equipos y de una manera más bien subjetiva se decide cual tiene el mejor desempeño. Los evaluadores deben de ser personas que conozcan bien el programa que servirá de prueba y deben de conocer las diferentes funciones reales que realiza el sistema para tener un resultado real, por lo tanto es muy común que los evaluadores sean los usuarios finales que a fin de cuentas manejarán el equipo que resulte seleccionado. Una ventaja de estos programas de prueba (también llamados Benchmarks) es que están disponibles en el mercado y que solo basta hacer una buena selección de que programas servirán de prueba. Además son útiles para evaluar tanto hardware como software. Quizá una desventaja radica en que dichos programas evalúan el desempeño presente y no puede evaluar o predecir cuando hay cambios al sistema, que en estos días resultan muy comunes. Estas pruebas tienen su aceptación sobre todo en la comparación de hardware y software y se realizan a menudo en un entorno de guerra comercial.

Programas sintéticos.- Esta es una combinación de las técnicas de los programas núcleo y de los programas de prueba, en esta técnica se utilizan funciones intensivas que se extraen de un programa real y se utilizan como prueba de evaluación, estos programas son muy eficientes pues arrojan mejores resultados que los demás, sin embargo son difíciles de implementar, pues en ocasiones no se tiene el tiempo y el recurso suficientes para aislar, codificar y depurar las rutinas que serán utilizadas. Es por eso que en muchas ocasiones se buscan simples programas de prueba existentes lo más parecido a los sintéticos con resultados aceptables. Dentro de los programas sintéticos más conocidos destacan dos: *el Whetstone y el Dhrystone*.
El Whetstone fue un programa sintético implementado a finales de los 70 cuando los procesadores se empezaron a hacer más sofisticados y se requería evaluar el desempeño de programas científicos hechos en Algol 60, este programa se realizó en Fortran y posteriormente fue traducido a otros lenguajes, las medidas empleadas eran Whetstone/segundo, es decir, el número de ejecuciones de una interacción del programa por segundo. El Dhrystone es un método más reciente basado en el mismo principio pero para procesadores más avanzados.

Simulación.- Otra técnica de medida del desempeño es la simulación, en esta técnica se utiliza un modelo computarizado del sistema, haciendo que el comportamiento de dicho modelo refleje el sistema real. La simulación es frecuentemente utilizada para evaluar sistemas que no existen o que están en diseño, evitando con esto la realización de un diseño pobre o bien encontrando

soluciones que mejoren el desempeño. Los evaluadores deben de ser de un nivel superior, pues los resultados obtenidos por la prueba en la mayoria de los casos no son pocos y requieren un análisis exhaustivo, además deben de ser capaces de implantar un simulador y validar su modelo en relación con el real. En base a su desempeño los simuladores se clasifican en dos tipos.

- Simuladores manejados por eventos.- En donde existe un control del mismo por todos los eventos que son producidos de acuerdo a una distribución de probabilidad dada.
- Simuladores manejados por libreto.- Estos se controlan por datos empiricos y subjetivos que reflejan en mucho el comportamiento esperado del sistema

Los simuladores son más bien utilizados cuando hay en juego bastante responsabilidad o dinero de por medio como puede ser un proyecto espacial o una puesta en marcha de un sistema de control financiero.

Existen diferentes maneras de probar la eficiencia de un sistema. El desempeño es una característica que se debe de analizar con cuidado, cada uno de los métodos que se utiliza prueba sobre diferentes parámetros la mayoria de ellos relacionados con el tiempo como pueden ser los tiempos de respuesta, de retorno de reacción del sistema; o bien con la producción, con la capacidad, con la utilización. Existen gran cantidad de programas software y hardware para medir todos estos parámetros, más aún existen técnicas especificas para lograr la mejora de un sistema en base a su desempeño; una de las técnicas utilizadas para la mejora es la segmentación de un sistema.

Al igual que esta, hoy en día existen más técnicas de mejora y se desarrollan cada vez más técnicas complejas para medir el desempeño. Es por eso que, cuando se analiza un sistema desde su raiz, es decir, el sistema operativo, se debe tener en cuenta el desempeño como una caracteristica de diseño ligada a todo el proyecto que le rodea.

◆ CONCLUSIONES

El diseño y la implantación de un sistema operativo es una tarea compleja que requiere de teoría suficiente en conjunción con las herramientas de desarrollo adecuadas para su realización.

En las empresas dedicadas a venta masiva de aplicaciones, tales como Microsoft e IBM, los equipos de desarrollo del área de sistemas operativos trabajan duramente a lo largo de meses para lograr su objetivo final, **un sistema operativo con las características deseadas por el usuario**. Este sistema operativo aun a pesar de ser entregado con retraso presenta defectos los cuales son corregidos conforme aparecen dentro del mismo.

La construcción y el diseño de un sistema operativo es una tarea realizada por múltiples personas dentro de un proyecto, aquí se trato de simplificar al máximo el sistema operativo, sin sacrificar funcionalidad ni componentes, para ofrecer al estudiante de la materia de sistemas operativos un medio balanceado de teoría y practica.

Durante esta implantación se adquirieron conocimientos generales de la PC y su arquitectura, se trabajo directamente con dispositivos para poder leer caracteres desde el teclado, escribir estos mismos caracteres en la pantalla, y para poder finalmente guardarlos en disco utilizando un sistema de archivos.

Se investigó acerca de las tendencias de los sistemas operativos modernos, también se obtuvo información de los proyectos de las grandes compañías en cuanto a su estrategia actual de sistemas operativos. Finalmente se probaron múltiples sistemas operativos de PC para obtener mayor información de los mismos.

Es importante mencionar que la bibliografía y documentación obtenida ofrecía detalles incompletos de los componentes de los sistemas operativos así como también de su implantación por lo cual se tuvieron que integrar múltiples ideas de diferentes autores para consolidar una respuesta final satisfactoria a cada uno de los problemas con los que nos encontramos en el desarrollo de esta tesis.

En cuanto a la documentación utilizada algunos autores definían los componentes de una manera muy clara, pero su deficiencia en la practica era notoria, como en el caso de Silberchatz-Peterson, y Deitel, mientras que otros ofrecían el código completo de un sistema operativo grande, pero casi sin documentación de la implantación, como en el caso de Tanembaum con el Minix, y de los sistemas operativos Linux y BSD.

Por otra parte los detalles de la manipulación directa de los dispositivos no se encuentran bien detallados en un solo manual de hardware, por lo que al igual que con la teoría general e implantación se tuvo que consultar un gran numero de estos manuales.

En la parte de la implantación se usaron lenguaje C y ensamblador, ensamblador para manejar la parte mas interna del sistema operativo

La documentación encontrada de C y ensamblador fue igualmente integrada, debido principalmente a que la mayor parte de la misma no cubria en un solo volumen la información requerida para esta tesis

Finalmente se pudo realizar la integración de todos los componentes de la teoria, es decir, se mezclaron todos los conocimientos posibles para entregar un proyecto completo que tuviera utilidad teórica y practica en el área de sistemas operativos.

Aplicación en el ambiente educativo

Este fue un proyecto muy interesante, pero muy complejo, en este proyecto se agrupan 40 años de desarrollos en las áreas del software y del hardware, todo esto con el propósito de integrar en un solo lugar lo necesario para que el alumno pueda estudiar sistemas operativos y practicar con un sistema pequeño pero completo.

Es importante destacar que se evaluaron varios sistemas operativos para incluirlos dentro de esta tesis, pero algunos eran muy pequeños y no incluian todos los componentes de un sistema operativo moderno, tal es el caso del KMOS de Milan Milenkovic, que es pequeño, alrededor de 2000 lineas de código en C, pero no incluye el administrador de archivos debido que esta orientado a sistemas controladores de tiempo real, aun a pesar de eso es un sistema operativo muy claro que puede ser utilizado como referencia. Otro caso es el del Minix, sistema operativo diseñado por Andrew S. Tanembaum, este es un sistema operativo completo, pero es mucho mayor que el KMOS, el Minix en su versión mas actualizada tiene alrededor de 24,000 lineas entre código en ensamblador y código C, adicionalmente Tanembaum no detalla muy claramente su implantación. Finalmente se evaluaron dos sistemas operativos mas a los cuales se tiene el acceso de todo el código fuente, uno es el Linux, sistema operativo desarrollado por Linus Torvalds y múltiples colaboradores de Internet, y el BSD, desarrollado por la Universidad de Berkeley en California, ambos sistemas operativos tienen todos los componentes de los sistemas operativos modernos, ambos se encuentran muy detallados en la documentación existente, pero su estudio tiene dos problemas fundamentales, el primero es que el código de estos sistemas es de mas de 120,000 lineas, lo cual es demasiado para un proyecto de un semestre, el segundo, que quizá no sea tan critico como el del código es que en estos

sistemas operativos se agregan un gran numero de componentes adicionales que complican su estudio e implantacion, tales como los de los dispositivos virtuales, que si bien su uso es muy practico incrementa en mucho el código de los sistemas.

Por la complejidad o simpleza de los sistemas operativos encontrados se decidió que era mejor trabajar con un sistema operativo intermedio que no fuera tan simple como el KMOS pero tampoco fuera tan grande como el Linux, esta fue una de las motivaciones para construir un sistema operativo mediano con una finalidad académica.

Nuestro sistema operativo lo bautizamos como OMNI, que significa todo en latin, porque integra todo lo necesario para el estudio de un sistema operativo moderno, posee un cargador del sistema operativo, un administrador de procesos, un administrador de memoria, varios administradores de dispositivos, y un administrador de archivos, adicionalmente se incluye todo el código fuente para su estudio y modificación, el cual es de 7300 lineas incluyendo comentarios.

Este sistema puede ser utilizado para impartir la cátedra de sistemas operativos a nivel licenciatura, en esta tesis se adjunta la teoría necesaria para revisar y modificar cada uno de sus componentes, adicionalmente se proporciona una bibliografía extensa para estudiar a fondo algunos temas de interés, así como también investigar temas no explicados con detalle, como el de comunicaciones. Esta tesis pretende ser el documento central de la enseñanza de sistemas operativos a nivel licenciatura.

Aplicación en otros ambientes

La aplicación inicial para la cual fue diseñado el sistema operativo descrito en esta tesis fue el área educativa, por lo que no se recomienda para áreas productivas, principalmente debido a que no es un sistema totalmente probado, por lo cual podría fallar bajo alguna circunstancia pudiendo causar algún daño a la información procesada o utilizada dentro de este sistema operativo.

Aplicación en materias universitarias

Este sistema operativo puede ser utilizado en distintas materias de la carrera de Ingeniería en Computación, tales como:

- Programación de sistemas
- Programación estructurada
- Estructuras de datos
- Ingeniería de programación
- Sistemas operativos

En programación de sistemas pueden ser utilizados los módulos en ensamblador para ver de una manera practica la interacción del procesador con diferentes dispositivos periféricos, adicionalmente se puede aprender la administración de la programación segmentada del la plataforma Intel.
En programación estructurada se pueden estudiar los programas en C que fueron diseñados utilizando esta disciplina, ya que se diseñaron funciones especificas para cada una de las tareas necesarias para la administración de este sistema operativo
En estructuras de datos se pueden utilizar las estructuras de datos complejas como archivos para observar como se administran dentro de la computadora, es decir, como se manipulan y almacenan diferentes tipos de estructuras de datos dentro de la computadora.
En ingeniería de programación se puede aplicar la metodología para el desarrollo del sistema operativo, o ver a este sistema operativo como un sistema complejo en el cual se aplican los conocimientos de la materia de ingeniería de programación.
Y finalmente la podemos aplicar en sistemas operativos ya que es la dirección inicial de esta tesis

La aportación principal de este sistema es en el área académica ya que proporciona todos los elementos necesarios para el estudio de un sistema operativo moderno.

En lo que se refiere a utilización de código es importante destacar que donde se pudo utilizar código existente se utilizo, tal es el caso de las funciones primitivas de manejo de video, teclado y disco, las cuales fueron escritas apoyándose lo mas posible en el BIOS de la PC, adicionalmente algunos otros componentes de este sistema no fueron escritos desde cero, en su lugar se utilizo parte del código existente en la bibliografía, ya sea de los libros de sistemas operativos o de los manuales de lenguajes o de los libros de estructuras de datos.

Se utilizo como modelo de diseño el sistema operativo Unix debido a que es de gran popularidad en el ambiente académico, es decir, se incluyeron funciones similares a las proporcionadas por el sistema operativo Unix, tales como sus llamadas al sistema, su sistema de archivos, etc

El sistema OMNI en conjunción con su fundamento teórico cumplió con los objetivos iniciales pues proporciona todos los elementos necesarios para el estudio de sistemas operativos, así como también la parte práctica siempre necesaria en estas áreas de la ingeniería.

Este sistema operativo puede ser extendido para aumentar su funcionalidad y su alcance.
Algunas de las modificaciones o adiciones sugeridas a este sistema operativo es el soporte al modo protegido del procesador 386 o superior, soporte a

comunicaciones, soporte a un mayor numero de procesos y terminales virtuales, manejo de memoria virtual, separación de los servicios en procesos de usuario, inclusión de seguridad, inclusión de manejo de directorios en el sistema de archivos. Estas son solo unas de las modificaciones que pueden ser realizadas en este sistema operativo.

En lo referente a la migración a otras plataformas cabe destacar que es una tarea muy compleja, esto es debido principalmente a que este sistema operativo utiliza lo mas posible el BIOS, el cual es único a la familia de computadoras basadas en Intel, otra parte que dificulta la migración a otras plataformas es que gran parte del sistema fue escrito en ensamblador, aunque ya existen interpretes de ensamblador Intel para otras maquinas

Comparándolo con otros sistemas operativos OMNI es un sistema operativo muy pequeño, puede parecerse en cierto sentido al viejo DOS pero con capacidad multitareas limitada, como se menciono anteriormente se utilizaron algunas llamadas al sistema similares a las del sistema operativo Unix, adicionalmente se utilizo también un sistema operativo parecido al de Unix aunque el sistema de archivos de OMNI no soporta la facilidad de directorios.
No tiene una interfaz gráfica como los sistema operativos actuales, tales como OS/2, Windows 95 o Windows NT, pero cumple con la capacidad multitarea de una manera similar a Windows NT y OS/2, aunque las aplicaciones quedan protegidas solo se permiten aplicaciones pequeñas en OMNI.
En pocas palabras OMNI cumple con las características principales de los sistemas operativos modernos.

Finalmente menciono que la tendencia de los sistemas operativos modernos es la integración al ambiente de red, es decir, nadie quedara incomunicado, implantación de seguridad avanzada, es decir, nadie manipulara u obtendrá datos sin autorización, capacidad de tolerancia a fallas, es decir, si algún componente físico o lógico falla el sistema completo no fallara, interfaz intuitiva probablemente visual, es decir, ya no será necesario el teclado o el ratón para la interacción con la computadora, estructura de procesamiento cliente-servidor, es decir, todas las solicitudes de usuario pueden ser atendidas por diferentes maquinas lo que redundara en un desempeño muy alto de los sistemas y manejo transaccional avanzado, es decir, se podrán realizar múltiples operaciones sin error con la confianza hasta ahora solo proporcionada por los grandes equipos centrales, con estas características tendrán que diseñarse los sistemas operativos del futuro, aunque algunos sistemas actuales ya poseen estas características todavía existe mucho por depurar en las técnicas actuales de implantación de las mismas.
Por ahora la compañía que mas ha avanzado en las características anteriormente mencionadas es Microsoft con sus sistema operativo Windows NT.

◆ BIBLIOGRAFIA

Libros

[1] Silberschatz Abraham; Peterson James L.; Galvin Peter B., 1994, *Sistemas operativos, conceptos fundamentales*, Wilmington, Delaware: Addison Wesley

[2] Milenkovic Milan, 1994, *Sistemas operativos, conceptos y diseño*, Aravaca, Madrid; McGraw Hill

[3] Deitel Harvey M., 1993, *Introducción a los sistemas operativos*, Wilmington, Delaware; Addison Wesley

[4] Tanenbaum Andrew S., 1992, *Sistemas operativos modernos*, Naucalpan de Juárez, Edo. de México; Prentice Hall

[5] Madnick Stuart E.; Donovan John J., 1986, *Operating Systems*, Singapur, McGraw Hill

[6] Salas Parrilla Jesús, 1992, *Sistemas operativos y compiladores*, Madrid España; McGraw Hill

[7] Singhal Mukesh; Shivaratri Niranjan G., 1994, *Advanced concepts in operating systems*, Singapur; McGraw Hill

[8] King Adrian, 1994, *Windows 95*, México; McGraw Hill

[9] Gates Bill, 1995, *Camino al futuro*, México; McGraw Hill

[10] Barry B. Brey, 1994, *Los microprocesadores Intel, arquitectura, programación e interfaces*, Naucalpan de Juárez, Edo. de México; Prentice Hall

[11] Norton Peter, Aitken Peter, Wilton Richard, 1993, *The Peter Norton PC Programmer's Bible*, U.S.A.; Microsoft Press

[12] Tanenbaum Andrew S., 1996, *Sistemas operativos distribuidos*, Naucalpan de Juárez, Edo. de México; Prentice Hall Hispanoamericana S.A.

[13] Tackett, Jack.; Gunter, David. Lance, Brown., 1996, *Linux, Edición especial*, Naucalpan de Juárez, Edo. de México; Prentice Hall Hispanoamericana S.A.

[14] Wirth, Niklaus, 1985, *Algoritmos + Estructura de datos = Programas*, Villaviciosa de Odón a Pinto, Madrid; Ediciones del Castillo S.A.

[15] Johnson, Robert, 1993, *MVS, Manual para programadores*, Aravaca, Madrid; McGraw Hill

[16] Aitken Peter, Jones Bradley, 1994, *Aprendiendo C en 21 días*, Naucalpan de Juárez, Edo. de México; Prentice Hall Hispanoamericana S.A.

[17] Perry Greg, 1993, *C by example*, Indianapolis, IN, United States of America; Que Corporation

[18] Morgan Christopher, Waite Mitchell, 1985, *Introducción al microprocesador 8086/8088 (16 bits)*, México; Byte Books/McGraw Hill

[19] Wyatt Allen, *Lenguaje ensamblador Manual de bolsillo*, México; Que Corporation

[20] Schildt Herbert, 1990, *Lenguaje C Programación avanzada*, Humanes, Madrid; Osborne/McGraw Hill

[21] Dorfman Leri, 1990, *Object-Oriented Assembly Language*, United States of America; Windcrest/McGraw Hill

[22] Kernighan Brian W., Ritchie Dennis M., 1991, *El lenguaje de programación C*, Naucalpan de Juárez, Edo. de México; Prentice Hall Hispanoamericana S.A.

[23] Godfrey Terry, 1991, *Lenguaje ensamblador para microcomputadoras IBM para principiantes y avanzados*, Naucalpan de Juárez, Edo. de México; Prentice Hall Hispanoamericana S.A.

[24] Norton Peter, Socha John, 1992, *Assembly Language for the PC*, New York, NY, United States of America; Brady Publishing

[25] Sheldon Tom, 1992, *Windows 3.1 the complete reference*, Berkeley, California; Byte Books/McGraw Hill

[26] Maguire Steve, 1993, *Código sin errores*, México; Microsoft Press/McGraw Hill

[27] Zack William H., 1992, *The OS/2 handbook -> Applications, Integration, And Optimization*, United States of America; Van Nostrand Reinhold

[28] Hallberg Bruce, Ivens Kathy, 1995, *OS/2 Certification Handbook*, United States of America; New Riders Publishing

[29] Tyson Herb, 1992, *10 Minute guide to OS/2 2.0*, United States of America; Alpha Books

Revistas

[1] Allison Charles B., *A simple Real-Time Executive*, The C users Journal, 1992, Volume 10, Number 11, pp 33-43

[2] Varhol Peter D., *Trends in Operating System Design*, Dr. Dobb's Journal, November of 1994, pp 18-27

[3] Burgess Richard, *MMURTL. Your Own 32-Bit Operating System*, Dr. Dobb's Journal, November of 1994, pp 38-45

[4] Borlotti J. F., Bernard P., Bouchet E., *RTMK: A Real-Time Microkernel*, Dr. Dobb's Journal, November of 1994, pp 70-76

[5] Yager Tom, *Linux Matters*, Byte, Volume21, Number 2, pp 123-128

[6] Shemitz Joe, *Multitasking state machines*, The C users Journal, 1992, Volume 10, Number 11, pp 23-32

[7] Palmer Stewart, A RISC OS for All Seasons, Byte, Volume 21, Number 12, pp 49-50

CD-ROMS

[1] C/C++ Users Group Library, Volume 445, July 1996, United States of America, Walnut Creek

[2] Dr. Dobbs Journal/CD Release 3, 1995, United States of America, Miller Freeman

[3] 4.4BSD lite, 1994, United States of America, Walnut Creek

[4] Plug and Play Linux Yggdrasil, 1994, United States of America, Walnut Creek

[5] Linux programs, 1995, United States of America, PowerSource Multimedia

[6] Linux Release 2.3 Slackware, 1992, United States of America, Linux System Labs

◆ GLOSARIO

Adaptador de video. Tarjeta que controla el flujo de datos desde la computadora hacia el monitor.
Algoritmo. Metodología para la resolución de un problema dado.
Altair. Primera computadora personal.
API. Interfaz de programación de aplicaciones
Aplicación. Programa creado por una compañía que cumple con alguna necesidad de los usuarios.
Aplicaciones de tiempo real. Aplicaciones que tiene límites de tiempo críticos para concluir determinadas tareas.
Apuntador. Variable cuyo contenido es una dirección de memoria.
Archivo. Información almacenada en un dispositivo con características específicas que puede ser invocada por su nombre para su utilización.
Arquitectura. Tipo específico de hardware.
ASCII. American Standard Code for Information Interchange, código utilizado para representar caracteres tanto en teclado como en video.
ATM. Modo de transferencia asíncrono, método de comunicación que cada vez se vuelve más popular.
Atributo. Característica de un archivo.
Base de datos. Repositorio de datos el cual tiene elementos que permiten la fácil explotación de su contenido.
BASIC. Lenguaje popular de computadoras. Beginners All Purpose Symbolic Instruction Code.
Bell Laboratories. Subsidiaria de AT&T, creadora del Unix.
Bibliotecas. Conjunto de programas de uso común disponibles para los programas de usuario.
BIOS. Sistema básico de entrada salida, software de las PC's que interactúa directamente con el hardware proporcionando los servicios básicos de entrada y salida.
Bloque de control del proceso. Estructura de datos que tiene el estado más actualizado de un proceso.
Bloqueo. Situación existente cuando un proceso está en espera de un recurso y este no se encuentra disponible.
BSD. Desarrollo de software de la Universidad de Berkeley California.
Buffer. Memoria auxiliar que permite que las operaciones de entrada y salida sean más rápidas.
Cache. Memoria de alta velocidad que ayuda a agilizar las operaciones con dispositivos lentos como el disco.
Cambio de contexto. Cuando se salva el estado de un proceso en ejecución y se restaura el estado de otro proceso para que continúe con su ejecución.
Canal de comunicaciones. Medio por el cual se conectan múltiples computadoras.

Candado. Cuando un proceso espera por un recurso que otro proceso tiene y este otro proceso espera por un recurso que tiene el primer proceso
Cargador. Programa que lee un programa ejecutable del disco y lo coloca en memoria principal para su ejecución.
CD-ROM. Dispositivo de gran capacidad de solo lectura, permite guardar hasta 650 MB de información, pero la información almacenada no es modificable.
Cliente servidor. Método de procesamiento en el cual se distribuye el trabajo entre varios procesadores, pudiendo éstos ser máquinas distintas
Compactación. Movimiento de todos los bloques de memoria utilizados hacia la parte más baja
Compilador. Programa que transforma un texto en lenguaje de alto nivel en código entendible por la maquina destino.
Computadora. Máquina que realiza operaciones repetitivas de una manera eficiente.
Concurrencia. Ejecución de múltiples programas a la vez
Contador de programa. Registro de control interno de la CPU que guarda la dirección de la última instrucción ejecutada.
Controlador. Componente de un dispositivo periférico que opera directamente el mismo, es decir, es el circuito que maneja al dispositivo
CP/M. Sistema operativo de Digital Research para computadoras basadas en el microprocesador Z80 y 8080.
CPU. Unidad central de proceso. Procesador.
Cracker. Persona que se dedica a destruir sistemas de cómputo.
CRC. Cyclic Redundancy Check, es utilizado para validar la consistencia de datos.
Dato. Unidad elemental de información.
Desempeño. Medida de rendimiento de un equipo de cómputo.
Despachador. Parte del sistema operativo que le entrega los recursos disponibles a un proceso.
Directorio. Estructura de datos que maneja varios nombres de archivo así como también sus atributos.
Disco. Unidad de almacenamiento secundario que generalmente contiene grandes cantidades de información.
Encriptación. Manera de proteger la información mediante manipulación aritmética de la misma.
Emulación. Simulación de una característica no nativa.
Ensamblador. Lenguaje de bajo nivel muy similar al lenguaje de máquina, se utiliza cuando se debe interactuar directamente con el hardware o cuando se requiere velocidad para realizar alguna operación dentro de la computadora.
Entrada salida. Operación que se realiza entre la computadora y algún dispositivo cuando se está transmitiendo información de o hacia el dispositivo.
Esclavo. Dispositivo que no tiene el control de la operaciones, sus operaciones son controladas por un dispositivo maestro.
Espacio de direcciones. Área de memoria donde se ejecutan los programas.
FPU. Unidad de punto flotante.

Fragmentación. Existencia de múltiples bloques de memoria inutilizables debido a su tamaño pequeño.
GUI. Interfaz gráfica de usuario. Tipo de interfaz que utiliza iconos para representar objetos de un escritorio.
Hacker. Persona especialista en software que se dedica a descifrar la complejidad de los programas.
Hardware. Parte física de la computadora. Ejemplo, teclado, monitor, procesador, etc.
Hilo. Unidad mínima de despacho, bloque de control cooperativo con los procesos de su mismo espacio de direcciones.
HTML. Hyper Text Markup Language, lenguaje utilizado para la creación de hojas dentro de Internet.
IBM. Industries Business Machines, empresa creadora de la PC.
IEEE. Instituto de Ingeniería Eléctrica y Electrónica.
Inanición. Situación existente cuando un proceso que tiene muy baja prioridad no es ejecutado por existir siempre procesos con prioridad más alta.
Ingeniería del software. Ingeniería que se dedica al estudio de los programas y las técnicas para su implantación.
INT. Acrónimo o abreviatura de interrupción.
Intel. Compañía manufacturera de circuitos electrónicos, creadora de la serie X86 de procesadores de computadoras personales.
Interactivo. Que responde rápidamente a las peticiones del usuario.
Interfaz. Elemento que permite la comunicación entre la computadora y el usuario, o la computadora y algún dispositivo periférico.
Internet. Red de redes, inicialmente desarrollada por un proyecto de la Secretaría de Defensa Americana, ARPA.
Intérprete. Programa que lee instrucciones en lenguaje de alto nivel y las ejecuta sin convertirlas en código entendible por la máquina.
Interrupción. Método con el cual se llama la atención del procesador para procesar una operación de más alta prioridad o que tiene prioridad crítica.
LAN. Red de área local.
Lenguaje de programación. Lenguaje que permite la ejecución de tareas útiles dentro de la computadora mediante la creación de procedimientos.
Ligador. Programa que enlaza un programa compilado con su biblioteca de funciones.
Linux. Clon de Unix desarrollado por Linus Torvalds.
Llamada al sistema, al supervisor, o al núcleo. Invocación de un procedimiento del sistema operativo.
Maestro. Dispositivo que tiene el control de las operaciones o de otros dispositivos.
Manejador. Software que generalmente interactúa directamente con el núcleo del sistema operativo que es utilizado para la administración de algún recurso.
Máquina virtual. Aquella que entrega a cada usuario una copia virtual del hardware como si fuera físicamente el dispositivo.

Mecanismo. Implantación de los lineamientos para la administración de un recurso.
Memoria compartida. Área de memoria utilizada por varios procesos a la vez.
Memoria virtual. Memoria utilizable por un proceso, combinación de memoria real y memoria secundaria.
Memoria. Lugar donde se almacena la información a ser procesada en un tiempo corto.
Microsoft. Compañía creadora del sistema operativo DOS para PC.
Modo núcleo o supervisor. Modo del procesador en el que se permite la ejecución de todas las instrucciones, incluyendo las instrucciones privilegiadas de entrada salida.
Modo usuario. Modo del procesador en el que no se permite la ejecución de todas las instrucciones.
Monitor. Dispositivo de salida de video, en el cual aparecen todos los resultados de los cómputos realizados.
Monocromático. Monitor que solamente puede desplegar dos colores, color negro y otro color.
Monolítico. Programa sin estructura. Conjunto de programas unidos en un solo código los cuales pueden invocarse a sí mismos o a los demás si proporcionan algún servicio útil a éstos.
MS-DOS. Primer sistema operativo de Microsoft.
Multihilos. Capacidad de un sistema operativo que permite a un programa que tiene múltiples componentes que se ejecuten varios de sus componentes a la vez, cada uno en un hilo.
Multiplexión. Intercambio de control del procesador entre múltiples procesos.
Multiprocesador. Equipo que tiene más de un procesador.
Multiprogramación. Práctica de computación en la cual aparentemente se están ejecutando varios programas a la vez en un solo procesador.
Multitarea apropiativa. Cuando el sistema operativo asigna tiempos para la ejecución de tareas, y en caso de que no se cumplan los tiempos entrega el control al siguiente proceso listo.
Multitarea cooperativa. Cuando los programas en ejecución ceden el control al sistema operativo cuando han realizado alguna actividad útil.
Multitarea. Capacidad de los sistemas operativos que permite atender a múltiples tareas a la vez.
Multiusuario. Capacidad de los sistemas operativos que permite atender a múltiples usuarios a la vez.
MVS. Almacenaje virtual múltiple, sistema operativo de IBM utilizado en toda la industria.
Núcleo o kernel. Parte principal del sistema operativo en la cual se llevan a cabo las operaciones de más bajo nivel, es decir, es la parte del sistema operativo que está directamente sobre el hardware.
Paginación. Método de administración de memoria en la que los programas que no están en uso se envían temporalmente a un área mapeada del disco.
Paradigma. Verdad o regla actual conocida.

Paralelismo. Capacidad de ejecutar múltiples operaciones a la vez, esto se apoya básicamente en sistemas multiprocesadores los cuales procesan cada uno una pequeña parte de un problema mayor.
Parámetro. Dato que se envia a un programa para que sea procesado o se utilice como referencia.
PC. Computadora personal.
Periférico. Dispositivo que no es parte principal de la computadora, aditamento con funciones específicas que se le agrega a la computadora .
Pila. Estructura de datos cuya principal característica es que los primeros datos que ingresan a ella son los últimos en salir.
Planificador. Parte del sistema operativo que decide que proceso se debe de ejecutar.
Politica. Lineamientos que se deben de seguir para administrar algún recurso.
POSIX. Normatividad que deben cumplir todos los sistemas Unix para que sean etiquetados como sistemas abiertos
Prioridad. Valor que se le da a un proceso para atenderlo preferencialmente
Procesador. Parte de la computadora donde se realizan todos los cálculos, parte central de la computadora.
Procesamiento por lotes. Método de procesamiento en el cual se procesaban múltiples trabajos del mismo tipo en paquetes denominados lotes.
Proceso secuencial. Proceso en el cual no se ejecutan varias partes del código a la vez, es decir, solo se pueden procesar en un orden iniciando desde el principio.
Programa. Secuencia de instrucciones que se ejecutan en una computadora para realizar alguna tarea específica.
Programación de eventos. Es un sistema que espera que suceda algo específico para iniciar un proceso.
Protocolo. Lenguaje de comunicación de redes
Quantum. Cantidad de tiempo asignada a un proceso para su ejecución.
RAM. Memoria de acceso aleatorio de lectura y escritura.
Ratón. Dispositivo de entrada que permite seleccionar objetos en la pantalla.
Recurso. Elemento del sistema de cómputo a administrar. Ejemplo, procesador, memoria, disco, etc.
Registro límite. Utilizado para marcar inicio y fin de áreas de memoria protegidas.
Registros. Áreas de memoria dentro del procesador que se utilizan para hacer operaciones.
Rendimiento. Balance entre la carga de trabajo y el tiempo en el cual se concluye dicha carga, se procura que este balance sea positivo, es decir, se concluya la carga de trabajo en el menor tiempo posible.
ROM. Memoria de solo lectura.
RPC. Llamada a procedimiento remoto.
Rutina. Pieza de un programa que tiene un propósito especifico.
Segmento. Área de memoria principal de tamaño determinado.
Semáforo. Estructura de datos que controla el acceso a una variable compartida.

Servicio. Implantación de alguna operación útil para el usuario por parte del sistema operativo.
Shell. interprete de comandos del sistema operativo
Sincronización de procesos. Cuando dos procesos están comunicándose se alternan la utilización del medio de comunicación
Sistema centralizado. Es aquel que utiliza un gran procesador central para cumplir con los trabajos solicitados por los usuarios.
Sistema distribuido. Es aquel que utiliza múltiples procesadores remotos para cumplir con los trabajos solicitados por los usuarios de dicho sistema.
Sistema operativo. Conjunto de programas que facilitan la utilización de un equipo de cómputo.
Software. Parte lógica de la computadora. Programas
SPOOL. Simultaneous Peripherical Operation Online, proceso que se utiliza para enviar archivos de impresión a disco mientras son atendidos
Tambor. Dispositivo de almacenamiento secundario predecesor de los actuales discos
Teclado. Dispositivo de entrada de la computadora, desde éste se ingresa información a la computadora.
Tiempo compartido. Método de procesamiento en el cual se les asigna a los usuarios la misma cantidad de tiempo de procesador.
Trampa. Implantación de una interfaz para comunicación directa con sistema operativo.
Transacción. Operación atómica la cual es totalmente exitosa o fallida, y en caso de ser fallida todos los pasos intermedios son cambiados a su estado original, antes de iniciar la operación.
Trayectoria. Camino lógico para localización de archivos.
TTY. Teletipo, equipo con el cual se comunicaban los operadores con los antiguos sistemas de cómputo.
Unix. Sistema operativo desarrollado por AT&T.
Video. Dispositivo de salida de la computadora, despliega información al usuario.
Virtual. Que no es real, simulado.
Xenix. Sistema operativo compatible con Unix lanzado por Microsoft.