UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

"CARACTERISTICAS Y CONCEPTOS FUNDAMENTALES DE SISTEMAS OPERATIVOS".

T E S I S

Que para obtener el Título de

M A T E M A T I C A

p r e s e n t a

DINORAH ELENA VALLADARES DIAZ

México, D. F. 1983

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

TESIS CON FALLA DE ORIGEN

PROLOGO

La importancia de los Sistemas Operativos puede entenderse en toda extensión, si pensamos que una misma máquina es capaz de comportarse como varias maquinas diferentes dependiendo del Sistema Operativo que este corriendo en un momento dado.

Cabe mencionar que la parte electrónica de una máquina no es de gran utilidad por sí sola ya que requiere de elementos que faciliten su manejo y optimizen el uso de todos los recursos ; tales elementos son los que conforman un Sistema Operativo.

En el presente trabajo se abordan algunas de las características y funciones básicas así como los problemas clásicos a resolver dentro de los sistemas operativos.

No pretende ser, con mucho, un estudio exnaustivo acerca de los Sistemas Operativos, sino proporcionar una guía para obtener una visión global sobre el tema, y fue desarrollado pensando basicamente en que podría servir como material de apoyo a un curso del mismo nombre, impartido en la Facultad de Ciencias.

Además de plantear la problemática general y posibles alternativas de solución, se describen brevemente tres sistemas particulares, donde se menciona su funcionamiento y características que los hacen de alguna forma representativos de un tipo de sistema. La idea de presentar primero los problemas generales de los sistemas y después ver como los resuelve cada uno de los ejemplos particulares.

UNIX

Es tomado como modelo, a su alrededor se han disen~ado muchos sistemas muy eficientes que siguen su estructura y filosofía.Su sistema de archivos es jerárquico y permite anexar directorios completos como una "hoja" de la estructura. Unix esta escrito casi en su totalidad en lenguaje C que es un lenguaje de alto nivel pensado para escribir Sistemas Operativos, lo cual lo hace muy portable y fácil de entender y modificar.

CP/M

Es un sistema muy popular en micros de 8 bits, existe una cantidad considerable de programas de aplicación desarrollados para él. Además, está considerado como el sistema estandard para micros y debido al gran impacto de este tipo de máquinas dentro de la computación actual, esta situación le confiere una gran importancia dentro de los sistemas operativos.

TENEX

En cuanto a TENEX, es una buena muestra de Sistemas de Tiempo compartido, con paginación bajo solicitud, diseñado para correr en una DEC PDP - 10, modificada para soportar el esquema de paginación planteado por la BBN.

TENEX tiene una interfase con el usuario llamada EXEC, sumamente eficiente, que realmente facilita el uso de la máquina, tiene código reentrante y tiene facilidades para que el usuario aprenda a manejarlo rápidamente.

Es el sistema que corre en la Foonly F-2 que se encuentra en el IIMAS. Esta máquina presta servicios, tanto al personal del Instituto, como a los alumnos de la Maestría en Ciencias de la Computación, por lo que su estudio resulta interesante.

Los algoritmos que estan descritos, se presentan como rutinas escritas en lenguaje C ya que es un lenguaje de alto nivel que permite desarrollar fácilmente Sistemas Operativos. Es un lenguaje portable y además UNIX está escrito casi totalmente en este lenguaje.

Como complemento se cuenta con un emulador del sistema operativo CP/M que podría ser utilizado como base para un laboratorio de Sistemas Operativos. En un apéndice se incluye la descripción de dicho emulador.

INTRODUCCION

1.- PRINCIPIOS BASICOS DE ORGANIZACION.

1.1 NOCIONES FUNDAMENTALES.

A pesar de que los detalles de organización varian de máquina a máquina , una vez que se han comprendido los conceptos basicos,es fácil aplicarlos a cada caso particular.

Primeramente se definirán algunos de estos conceptos básicos:

i) "SOFTWARE" .- Consiste en programas o rutinas que controlan y dirigen las operaciones de la máquina con objeto de facilitar al usuario el manejo de la computadora.

ii) "HARDWARE" .- Lo forman los dispositivos físicos y todo el equipo en general de que esta formado la computadora.

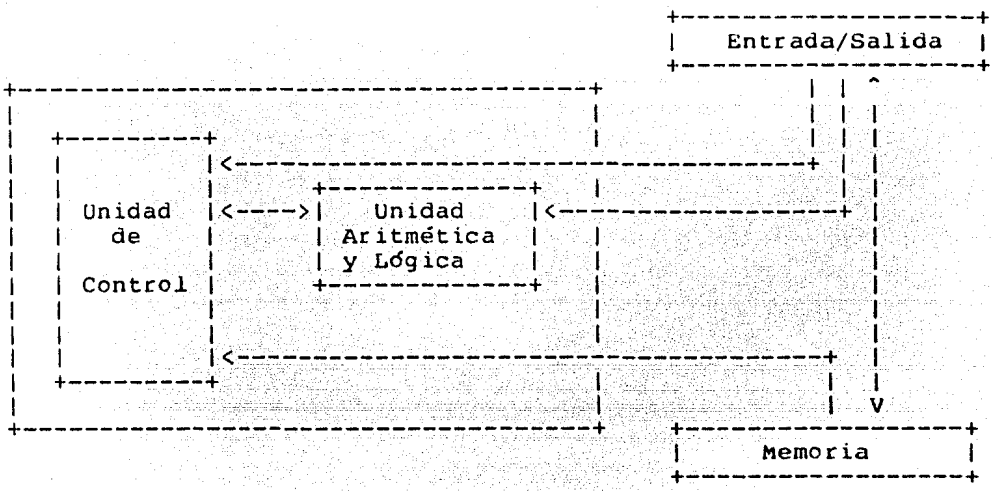
iii) SISTEMA .- Es una combinación de componentes de software y hardware relacionados de tal manera que al interactuar formen un todo organizado.

1.2 DESCRIPCION GENERAL DE UNA COMPUTADORA

Las computadoras son dispositivos diseñados para procesar información no importa de que tipo.Cada elemento de información puede ser representado como una combinación de componentes denominados BITS. Un bit es la unidad mínima de información y contiene valores 0 o 1.

A su vez, los bits se organizan en PALABRAS que es un grupo de bits que las máquinas manejan como UNA SOLA UNIDAD de información. El número de bits que conforman una palabra depende del diseño particular de cada computadora.En general,las máquinas pequeñas tienen palabras de entre 4 y 8 bits de longitud, las medianas de 12 a 32 y las grandes, entre 32 y 64 (algunas hasta más).

A continuación se presenta un Diseño General Simplificado que muestra las principales componentes de una computadora y resulta fácil adecuarlo para algún modelo en particular.



Las cuatro unidades fundamentales son:

1.2.1 UNIDAD DE CONTROL .- Interpreta y ejecuta las instrucciones en el orden especificado y genera señales de control y de contabilidad del tiempo para regular las demás actividades.

1.2.2. UNIDAD ARITMETICA Y LOGICA .- Realiza todas las operaciones aritméticas y lógicas.A la unión de las unidades aritmética y de control, se le denomina CPU o Unidad Central de Procesamiento.

1.2.3. MEMORIA .- Almacena los datos ,registros ,acumuladores y las instrucciones para poder ser utilizadas posteriormente.

1.2.4. UNIDAD DE ENTRADA/SALIDA .- Es el modo de comunicación entre la computadora y el medio ambiente externo.Esta conectada a uno o varios DISPOSITIVOS PERIFERICOS que son los encargados de convertir señales analógicas a digitales para que puedan ser procesados.

Un BUS es un canal de comunicación que permite el flujo de información entre las unidades de la computadora.También podemos pensar al bus como un cierto número de alambres paralelos que son capaces de transferir todos los bits de información en una palabra de máquina de manera simultanea.

En la mayoría de las máquinas tenemos 3 tipos de información:

a) DATOS
b) INSTRUCCIONES
c) SEÑALES DE CONTROL

En la mayoría de las máquinas tenemos 3 tipos de información:

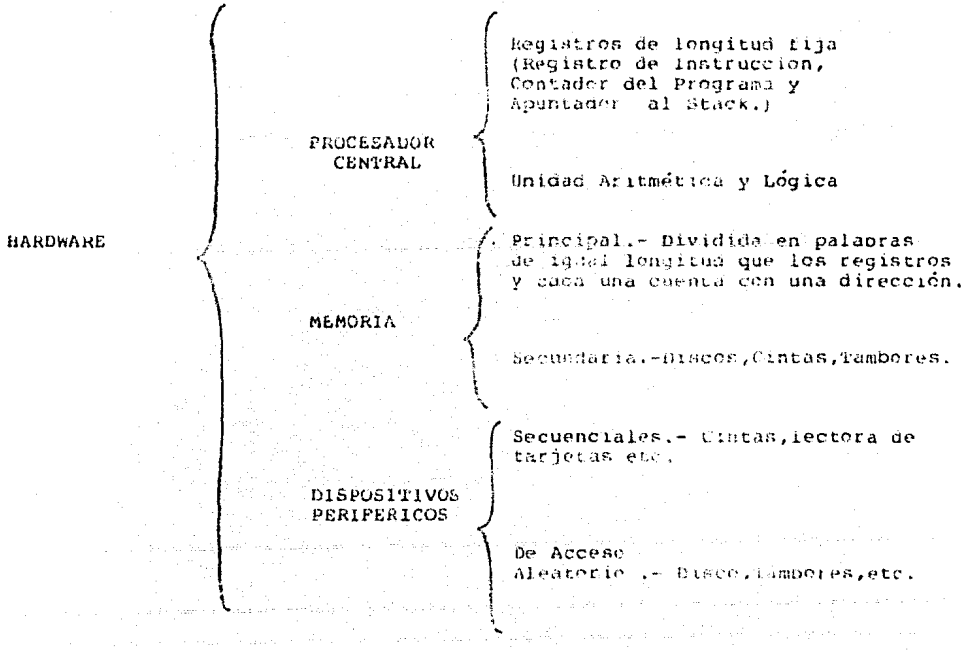
a) DATOS
b) INSTRUCCIONES
c) SEÑALES DE CONTROL

La información es manejada por medio de buses que dependiendo del tipo de información que lleven se clasifican en :

1.- Bus de Datos.
2.- Bus de Instrucciones.
3.- Bus de Control.

En la mayoría de las máquinas el bus de datos y el de instrucciones es el mismo y existen casos como la PDP 11 en que el Bus es único (concepto de Unibus) y a través de él se transfiere todo tipo de información.

Aquí tratamos de manera más detallada, el hardware con el que contamos en cualquier computadora:



Los dispositivos periféricos estan a veces unidos al CPU por medio de una computadora de proposito específico llamada CANAL.La comunicación entre el CPU y el Canal,se realiza a través de un DISPOSITIVO DE CONTROL,(Device Control Unit) que es un dispositivo electrónico que acepta comandos del canal y le hace ejecutar las acciones requeridas.

Un CANAL es un dispositivo capaz de controlar uno o más periféricos, realizando transferencia de control de datos entre ellos y la memoria, independientemente del procesador central.

Una INTERRUPCION es una señal que transfiere el control del CPU a una cierta localidad fija,mientras que guarda al mismo tiempo el contenido del contador del programa, para que en cuanto la interrupción sea atendida,pueda continuar la ejecución ene el punto donde fue suspendida.

El manejo de la entrada/salida se hace en base a INTERRUPCIONES.Una interrupción se dice que esta HABILITADA si al solicitarse la interrupción,ésta va a ser atendida.Cuando la interrupción se genera y esta habilitada,el CPU suspende la tarea que estaba ejecutando (salvando naturalmente toda la información necesaria para proseguirla después,o sea guarda el MEDIO AMBIENTE VOLATIL) y checa los REGISTROS DE INTERRUPCION para determinar el origen de la llamada.Los canales y el CPU trabajan de manera CONCURRENTE (ver capitulo 3).

1.3 COMPARTIBILIDAD DE RECURSOS

Un Recurso es cualquier objeto que puede ser colocado junto con el sistema.Se dice que un recurso es COMPARTIDO si a varios programas es posible otorgarles el completo control sobre ese recurso, por un período predeterminado de tiempo.

Existen dos formas de implementar la compartibilidad:

i) Centralizada .- Un programa se encarga de controlar el reparto de los recursos.

ii) Descentralizada .- Los mismos programas son los que manejan los recursos y los comparten entre sí.

La forma mas común es la CENTRALIZADA.Existen ademas facilidades que provee el software, como por ejemplo los BUFFERS o AREAS DE ALMACENAMIENTO de donde los periféricos toman o almacenan bloques de datos para optimizar el tiempo de CPU cada vez que éste quiera accesar datos.A esta técnica de manejo de información se le denomina BUFFERING.

1.4 MAQUINAS VIRTUALES

En la sección anterior se hizo una descripción general de las partes de una computadora,Sin embargo el tratar de resolver problemas utilizando exclusivamente las funciones básicas o elementales que otrece la máquina es algo sumamente complejo y eleborado además de que requiere un conocimiento preciso y detallado del manejo e implementación de dicnas funciones por parte del usuario.

Tratando de evitar este tipo de problemas surge el nuevo concepto de MAQUINA VIRTUAL.A partir de este punto entenderemos por Máquina Virtual "Aquella máquina cuyas características son diferentes de,pero más fáciles de manipular que, la mquina física en que se basa..." Además resulta ser una máquina mucho más poderosa, ya que al usuario le da muchas más facilidades de manejo y proceso de información.

Como ejemplos de máquinas virtuales,podemos mencionar aquellas que facilitan el manejo de las funciones de entrada/salida que de hacerse en base a las funciones basicas proporcionadas por el hardware,resultan muy laboriosas y altamente ineficientes.

Al lenguaje que nos permite dirigir las operaciones de la máquina durante la ejecución de un programa se llama LENGUAJE DE MAQUINA VIRTUAL y su necesidad salta a la vista ya que el usuario debe conocer las funciones con que cuenta la nueva máquina virtual,para poder utilizarlas.Es en base a este concepto de Máquina Virtual que definiremos el concepto central del presente trabajo: SISTEMA OPERATIVO.
======= =========

No existe una definición precisa de SISTEMA OPERATIVO en cuanto a que las definiciones estan basadas en la descripción de las funciones que un Sistema Operativo debe cumplir y que seran analizadas posteriormente en detalle.Sin embargo es posible definir un Sistema Operativo como un conjunto de programas que unidos proporcionan al usuario la máquina virtual que requieran sus necesidades.Basandonos en la estructura de diseño propuesta por Lister [1] podemos proponer la siguiente definición:

"El sistema Operativo es un NUCLEO que consta de las operaciones básicas del hardware y al cual se le van a ir añadiendo "capas" o "niveles" de programas que faciliten el uso de determinadas funciones, hasta obtener finalmente un todo que transforme las operaciones y las estructras de datos en una máquina ideal,inexistente desde el punto de vista físico ,pero con mucha mayor capacidad de acción que la máquina en la que se basa y de mucho mas fácil manejo."

****

2.1 QUE SON Y PARA QUE SIRVEN LOS SISTEMAS OPERATIVOS.

Para poder comprender la manera en que los sistemas operativos son construidos, es necesario en primer lugar saber QUE SON Y PARA QUE SIRVEN.Hablando de forma general, existen dos objetivos básicos que deben ser cubiertos cuando se tiene en mente disen~ar un sistema operativo:

i) El convertir el hardware de una máquina en una Maquina Virtual, de más fácil manejo para el usuario pero con mayor potencial.

ii) Asignar lo más eficientemente posible entre todos los usuarios, los recursos con que cuenta la máquina.

Naturalmente que los sitemas operativos están intímamente relacionados en cuanto a disen~o, con el propósito para el cual quiere desarrollarse la máquina virtual; sin embargo se ha caído en el error de querer crear sistemas tan generales, que fallan para cualquier aplicación particular.Además debe vigilarse de cerca el hecho de que el costo del sistema no se dispare a tal grado que resulte incosteable su realización.

Para facilitar el estudio de las funciones del Sistema Operativo es necesario clasificarlas en diferentes áreas de aplicación, como por ejemplo:

a) SISTEMAS DE TIEMPO REAL .- Son aquellos en los cuales se debe tener una respuesta a un estímulo externo dentro de los límites de un período predeterminado de tiempo.En general el tiempo de respuesta requerido es muy rápido.Ejemplos de este tipo de sistemas son:

i) Sistemas de reservaciones en Líneas Aereas, Hoteles, Boletrónico etc.
ii) Aplicaciones de Control de Equipo, como calderas, plantas procesadoras,Refinerías etc.
iii) Captura de Información y eventos en tiempo real.

Este tipo de proyectos lo constituyen por ejemplo, captura de información mandada desde un satélite o el proyecto RESMAC que se trata de una red de computadoras que registran movimientos telúricos y mandan la información a otra máquina para ser procesada.

El común denominador de estos sistemas es la RETROALIMENACION (feedback) y más importante que eso, LA RESPUESTA INSTANTANEA ya que su comportamiento se modifica en base a los estímulos externos que reciba el sistema.La principal función de un sistema de control es dar un máximo de confiabilidad con un mínimo de intervenciones del operador y "prever" cualquier situación inesperada o de falla y tener un patrón de conducta para estos casos.

Las Bases de Datos pueden entrar en este tipo de Sistemas ya que manejan un gran flujo de información que debe ser modificada constantemente (en ocasiones, varias modificaciones en un segundo) y por tanto debe presentar grandes facilidades para que los accesos sean rápidos y confiables.

b) SISTEMAS TIPO JOB-SHOP .- Son los que procesan una sucesión de tareas (tasks) sin ninguna restricción de tiempo en particular excepto el caso de exceso en el uso de recursos.Ejemplos de estos sistemas son las Nóminas, Contabilidades de Empresas, Inventarios o el acceso a las máquinas por tarjetas, donde desde un Remote Job Entry se dan los datos y el programa a la máquina y esta los procesa, mandando nuevamente los resultados al usuario.

Este tipo de sistemas resultan muy utiles sobre todo, cuando se desea manejar grandes flujos de información en forma de "paquetes" (JOBS) . Debido a esto y a la diversidad de necesidades que pueden presentarse, el sistema debe manejar gran cantidad de utilerías tales como compiladores de varios lenguajes, ensambladores,editores y un sistema para manejo de archivos , otro para el manejo de periféricos etc.

Podemos clasificar a este tipo de sistemas en 2 ramas:

SISTEMAS JOB-SHOP

- SISTEMAS BATCH: Entra un JOB y la comunicación entre el usuario y la máquina se interrumpe, en tanto no termine la ejecucion del JOB.
- SISTEMAS MULTIACCESO: Permite ejecutar un JOB por medio de comandos dados a través de una terminal, que facilita la comunica--ción máquina-usuario.

Realmente no está claramente establecida la diferencia entre un sistema Multiacceso y uno de tiempo real, excepto la velocidad en el tiempo de respuesta, que si es muy lento, en el primer caso no pasa de crear un malestar en el usuario, mientras que en el segundo podría tener consecuencias de gravedad.

2.2 FUNCIONES Y CARACTERISTICAS DE UN SISTEMA OPERATIVO

Si partimos del hecho de que contamos solamente con la máquina física sin ningun tipo de herramienta que nos ayude a manejarla, debemos considerar que para la ejecución de cualquier programa , no importa que tan simple sea, será necesario ejecutar una serie de pasos que requerirá la supervisión de alguna persona dedicada a este fin, como por ejemplo:

a) Poner las tarjetas en la lectora.
b) Iniciar el programa que las lea.
c) Iniciar el compilador o ensamblador.
d) Leer los datos que necesita.
e) Ejecutar el programa con los datos.
f) Recoger los resultados por la impresora.

Resulta evidente el gran desperdicio de tiempo de máquina, ya que su velocidad de proceso es muchisimo mayor que la velocidad del operador y las operaciones de ésta dependen de él.

De aquí es que surja la necesidad de tener un programa que controle todas las acciones necesarias para ejecutar un programa .Este tipo de programas de control, pueden considerarse como un primer intento de Sistema Operativo, por ejemplo, la Burrougns cuenta con WFL (además de su Sistema Operativo, claro).Partiendo ahora de que tenemos un lenguaje de control, analizemos las nuevas necesidades:

La máquina sigue desperdiciando recursos ya que ahora su actividad depende de la velocidad con que se lleve a cabo la entrada/salida de datos, información, etc. Para resolver este problema se planteó como posible solución,asignar esta tarea a una máquina más pequeña y de menor costo.En un principio, el medio que se utilizo para realizar la entrada/salida fue el uso de cintas magnéticas porque para ese tiempo eran el medio mas rápido y más económico.Sin embargo, presentan el inconveniente de que solo permiten aceso secuencial y no aleatorio a la información que se almacena.

En base a estos problemas surgen dos conceptos centrales dentro de los Sistemas Operativos:

i) Canales

ii) Interrupciones

Un CANAL es un dispositivo capaz de controlar uno o más periféricos, realizando transferencia de datos entre ellos y la memoria, independientemente del Procesador Central.

Una INTERRUPCION es una señal que transfiere el control del CPU a una cierta localidad fija, mientras guarda al mismo tiempo, el contenido del contador del programa (PC) para que en cuanto la interrupción sea atendida y se desee continuar la ejecución en el punto donde estaba, pueda lograrse.Es gracias a estas innovaciones que pudo manejarse el acceso a disco que lo hace de manera aleatoria.

La siguiente función que debe realizar un Sistema Operativo es tener una rutina que maneje las interrupciones y decida cual de todos los procesos es el que debe ejecutarse.Esta rutina se conoce como DESPACHADOR (Scheduler).Un sistema con estas características se utilizó a mediados de los años 60's y se denominaron sistemas "Single Stream Batch Monitor" y sólo eran capaces de ejecutar un programa a la vez, lo cual representa su principal desventaja: el dedicar TODOS LOS RECURSOS a la ejecución de una sola tarea.

Posteriormente surge la MULTIPROGRAMACION.Aquí la idea es tener varios programas cargados en la memoria y repartir entre ellos el uso del procesador central.Para redondear el sistema, se nace necesario proveer un sistema de protección entre las acciones de diferentes procesos.A estos sistemas se les denomina "Multi-Stream Batch Monitor".

Otra característica de los sistemas operativos es que deben ser fáciles de ejecutar y permitir la contabilidad de los recursos de la máquina.

Resumiendo, podemos enumerar las funciones básicas que debe realizar un Sistema Operativo:

1) Sucesión de actividades.
2) Interpretación de un lenguaje de control.
3) Manejo de errores.
4) Manejo de Entrada/Salida.
5) Manejo de Interrupciones.
6) Colas de Espera.
7) Control de Recursos.
8) Protección.
9) Multiacceso.
10) Contabilidad de Recursos.
11) Fácil manejo.

2.2.1 CARACTERISTICAS GENERALES

a) CONCURRENCIA .- Este es un concepto vital dentro del sistema y hablaremos de él con todo detalle en la siguiente sección.Por anora diremos que la concurrencia es la existencia de varias actividades en forma simultánea o paralela.

b) COMPARTIBILIDAD .- Ya que se requiere concurrencia, nabrá casos donde varios procesos necesiten compartir datos o bien, sea necesario eliminar redundancias como el necho de que cada usuario tenga una copia de un editor en memoria, en vez de compartirlo con otros usuarios.

c) Permitir ALMACENAMIENTO PROLONGADO de archivos.

d) CARACTER DETERMINISTICO Y NO-DETERMINISTICO .- Suena paradójico, pero un sistema debe ser determinístico en el sentido en que un programa que corre con ciertos datos y arroja ciertos resultados, debe mantenerse inalterable en esos resultados bajo las mismas condiciones y datos y debe ser no-determinístico porque no se sabe de antemano en que orden van a ejecutarse ciertos eventos y esto no debe influir en modo alguno en los resultados que se obtengan (por ejemplo, si estamos compilando, puede ser que la compilación se realice toda de una sola vez sin recibir interrupciones y si se vuelve a compilar y se recibe una interrupción que detenga el proceso y después continue a partir de donde se quedo, debe dar el mismo resultado que si lo hubiera hecho de una sola vez.)

e) EFICIENCIA .- Que puede verse desde varios puntos de vista:

i) Tiempo promedio para la ejecución de procesos.
ii) No desperdiciar tiempo de CPU.
iii) Tiempo de respuesta pequeño.
iv) Optimización en la utilización de recursos.

f) CONFIABILIDAD .- Por más que queramos hacer un sistema sin errores, todos los sistemas han fallado alguna vez pero lo que sí se puede es tratar de que esas fallas sean lo más esporádicas posibles.

g) MANTENIMIENTO .- Debe ser fácil de mantener, por lo que debe estar hecho de una manera estructurada,modular y contar con una documentación buena y clara.

h) PEQUEÑO .- Debe tener esta característica para que no ocupe demasiado lugar en la memoria y porque como todo programa si es demasiado grande,cuesta más trabajo encontrarle fallas o hacerle modificaciones.

i) PORTABLE .- Esta característica es muy deseable ya que así puede ser adoptado en varias máquinas sin muchos cambios. El sistema CP/M diseñado para micros debe buena parte de su popularidad al hecho de que, renaciendo solo un pequeño módulo, puede usarse en cualquier micro.(Ver sección donde se explica en detalle CP/M para mayor claridad.

2.3 DIFERENTES TIPOS DE SISTEMAS OPERATIVOS

Podemos hablar en general de los principios básicos de los Sistemas Operativos y las funciones que en general deben realizar, pero existen diferentes características que nos permiten agruparlos de varias maneras dependiendo de los aspectos que se consideren:

a) El número de usuarios que son capaces de atender.

b) El grado de experiencia profesional que se supone deben tener los usuarios para manejarlo.

c) El número y complejidad de funciones que provee al usuario.

Además los Sistemas Operativos difieren en cuanto a su estructura escencial (no es igual un sistema pensado para buscar y ejecutar programas de acuerdo a un evento externo, que uno que para funcionar debieron haber sido cargados previamente los programas y ejecutados mediante un comando dado por el usuario) y naturalmente en cuanto al hardware en que estan basados y la aplicación específica de cada uno.

Dependiendo de cuales sean las características que se analizan podemos obtener diferentes clasificaciones de los Sistemas Operativos.Tomemos por ejemplo como característica para clasificarlos, el hardware en que se basan.

En el caso de una máquina de gran tamaño donde el medio ambiente de trabajo se caracteriza por manejar una gran cantidad de procesos independientes a través de una gran variedad de fuentes, resulta natural pensar que las tareas del despachador y el manejador del sistema seran las más importantes.Un sistema operativo para una máquina grande resultara siempre sofisticado pero debe tenerse en cuenta que su principal función es la optimización del equipo con que se cuenta y cuyo precio es muy elevado.Por todo esto, un sistema de este tipo, se diseña como sistema de propósito general en el sentido más amplio posible del termino.

Por otra parte si se cuenta con una máquina pequeña, en general las funciones y los modos de acceso estan en cierta forma limitados.Sucede a menudo que los sistemas pequeños no pierden generalidad pero proporcionan menos ayuda a los usuarios .Por ejemplo la selección de dispositivos debe hacerse por los operadores o los programadores mismos en vez de hacerlos el sistema.

Viendolo desde este punto de vista, podríamos hacer nuestra clasificación, pero no resultaría muy acertada ya que ahora las máquinas pequeñas deben resolver problemas que antes solo se presentaban en las máquinas grandes.

Una característica utilizada a menudo para hacer la clasificación es el modo de acceso al sistema i.e. los medios que proporciona al usuario para interactuar con el sistema.De aní surgen conceptos como "Sistemas interactivos", "sistemas de tiempo-real" , "sistemas tipo batch", etc.

Tradicionalmente los sistemas han sido clasificados como:

1) Sistemas Operativos de TIEMPO REAL.

2) Sistemas Operativos tipo BATCH.

3) Sistemas Operativos de TIEMPO COMPARTIDO.

4) Sistemas Operativos de PROPOSITO MULTIPLE.

pero esta clasificación esta hecha en base a como se efectua el acceso a la máquina, sin considerar otras características. Debido a esto,, tenemos que dentro de los sistemas de tiempo compartido existen muchas diferencias y que incluso resulta difícil definir con exactitud lo que se entiende por "sistema de propósito multiple".

2.3.1 SISTEMAS OPERATIVOS EN TIEMPO REAL

Los Sistemas Operativos en Tiempo Real se caracterizan porque la actividad de procesamiento esta regida por la ocurrencia ALEATORIA de eventos externos.El procesamiento de un evento particular es acompañado de una sucesión de procesamientos de tareas cada una de las cuales debe completarse dentro de CONSTANTES RIGIDAS DE TIEMPO.

Los eventos externos pueden ser datos que se requieren, comandos del operador o bien, atender a los requerimentos de diferentes dispositivos.

Los sistemas de tiempo real, son sistemas de MONITOREO y CONTROL y que requieren un mínimo de intervención humana para su funcionamiento.Estas intervenciones están restringidas en general a casos donde se produce una violación en el sistema o bien para permitir la modificación de los parámetros con que se controla el sistema.Sus características principales son:

SON SISTEMAS DEDICADOS COMPLETAMENTE A LAS APLICACIONES DE CONTROL Y LO MAS IMPORTANTE ES GARANTIZAR UN TIEMPO DE RESPUESTA DEFINIDO DE ANTEMANO AUN CON CARGA DE TRABAJO MAXIMA.

Como aplicaciones de este tipo de sistemas citemos:

a) Monitoreo de lineas de producción (autos etc.)

b) Monitoreo de funciones clínicas de un paciente.

c) El manejo de los Semáforos.

d) Control de condiciones para experimentos en laboratorios.

e) Control de Plantas Energéticas, Refinerías etc.

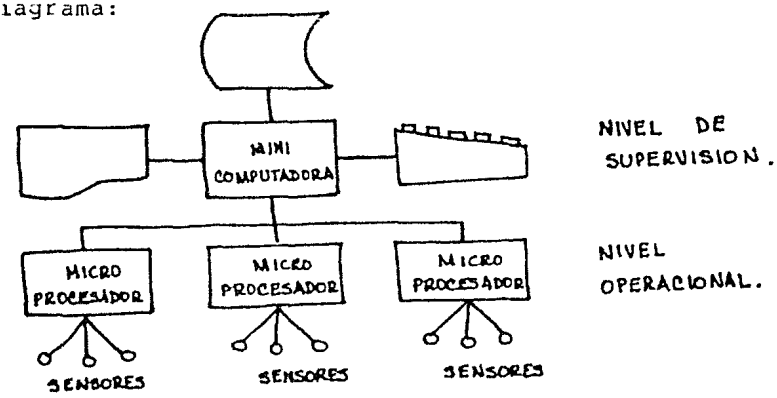
Los dispositivos ligados con estos sistemas son tan variados como las aplicaciones.Pueden ir desde convertidores analógico-digitales hasta satélites que manden señales a la computadora.El tamaño de la máquina no influye en mucho para determinar si un sistema es o no de tiempo real ya que actualmente se cuenta con algunos sistemas de este tipo basados en microprocesadores y son especialmente útiles cuando la aplicación requiere de poca interacción entre el operador y el sistema y este es capaz de correr,una vez echado a andar, durante largos períodos de tiempo sin requerir modificaciones.

Desde luego que procesadores de gran tamaño también se usan para Tiempo Real cuando el flujo de datos es muy grande y se requiere de gran capacidad de almacenamiento.Tal es el caso de las aplicaciones al ejército, donde se controlan las estrategias aéreas por medio de sistemas diseñados especialmente para este propósito.Como ejemplo mencionaremos a SACCS (Strategic Air Command Control System) y BMEWS (Ballistic Missile Early Warning System).

Considerando lo antes expuesto, surge la pregunta... cómo saber si un procesador servirá para soportar un sistema de tiempo real?

Su principal característica debe ser el tener un manejador de interrupciones sumamente eficiente y permitir determinación de intervalos de manera muy precisa.Es vital que el procesador sea capaz de responder rapidamente a interupciones externas y que arranque y/o suspenda procesos rapidamente.

El uso de multiples procesadores es muy común en Sistemas de Tiempo Real y se utiliza para aumentar la confiabilidad del sistema y/o para distribuir las funciones entre los diferentes procesadores.Como ejemplo tenemos el siguiente diagrama:



donde se muestran 3 microprocesadores conectados a sensores que comunican al sistema con el medio ambiente.Cada procesador manda sus señales a la minicomputadora la cual a ciertos intervalos de tiempo realizza reportes por escrito y almacena la información necesaria para el control y la supervisión.

SOFTWARE PARA TIEMPO REAL

El software para un sistema de tiempo real depende de la naturaleza de la aplicación,del procesador que se esta utilizando en un nodo determinado y de las relaciones entre esos nodos.

Sin embargo es posible nablar de características generales que estan siempre presentes .

Una de estas características fundamentales es la forma en que será orientado el diseño, ya sea formando colas de espera o bien en base a eventos.La idea es determinar de alguna manera cual será el siguiente programa a ejecutarse. Otra forma de diseño es en base a procesos y este patrón de ejecución se maneja en base al despachador y la optimización en la utilización de los recursos.Los sistemas de tiempo real cuentan con un manejador de colas que solicita y acepta mensajes del exterior,los identifica, analiza y ejecuta la acción correspondiente.

Otra característica común de los Sistemas de Tiempo Real es que la entrada de los datos es "en línea" y el procesamiento de programas también. Dado que es necesario que la respuesta sea rápida, la gran mayoría de los programas que se procesan deben estar residentes en memoria; de no ser posible, las técnicas del manejo de memoria (que se analizarán detalladamente en otra seccion) deben ser rapidas y simples de tal forma que permitan cargar y ejecutar los programas con un retraso mínimo.

Existe la tendencia de mantener el manejo del procesador igualmente simple, es decir, se permite que los programas corran hasta que terminan o bien durante un cierto período predeterminado de tiempo sin importar mucho si el uso del CPU o de la memoria se efectua de manera óptima.

2.3.2 SISTEMAS OPERATIVOS TIPO BATCH

El sistema Batch hace uso de de la capacidad de los centros donde se reciben los datos y los procesos para ser a comodados en colas, de donde seran posteriormente procesados según convenga al mejor uso del equipo. La velocidad de las maquinas de la segunda generación era tal que desperdiciaba mucho tiempo en iniciar y suspender tareas lo cual generaba largos períodos de inactividad para el procesador.

El objetivo de formar colas de jobs era incrementar la efectividad del sistema haciendo más continua la transición de tareas. En general, la idea es combinar la ejecución de jobs pequeños y grandes para que se minimize el tiempo dedicado a iniciar y suspender tareas.

Su principal característica consiste en lo elaborado de su despachador y asignador de recursos (scheduler y resource allocator).

Se pueden distinguir 3 subsistemas principales tipo Batch:

i) Sistemas de Un Solo Flujo.
ii) Multiprogramación Básica.
iii) Multiprogramación avanzada.

SISTEMAS DE UN SOLO FLUJO (SINGLE-STREAM SYSTEMS)

Este tipo de subsistemas es realmente antiguo pero su importancia radica en que fundaron las bases siguientes de los sistemas modernos:

a)Control de flujo.

b)Entrada-Salida a niveles superiores.

c)Relocalización de Programas.

d)Independencia de dispositivos.

El FMS (Fortran monitor System) fue el primer intento de organizar el lenguaje de programación,el manejo de Entrada-Salida y las rutinas de Utileria dentro de un todo compacto. Resulta interesante porque la estructura conceptual del sistema permitía a un programador que sabía FORTRAN, correr programas en FMS con haber leído someramente el manual.Excepción hecha por la limitación que implicaba el tener que usar fortran, este sistema mostraba la mayoría de las características de los sitemas Batch.

Daba la impresión al usuario que el proceso de "compile-load-go" (compila-carga-corre) era un proceso unificado. Aceptaba algunas directivas para hacer depuración de programas además de proveer un sistema rudimentario de control y manejo de OVERLAYS y que por supuesto estaba bajo la responsabilidad del programador.

Los elementos del sistema que permitían el concepto de "batch" estaban organizados en un monitor.El monitor leía la cuerda de jobs y cargaba al compilador cuando era necesario;al final de la compilación, el control pasaba de nuevo al monitor quién determinaba en base al siguiente job si el compilador iba a ser usado o no o si debía pasarse a la fase de carga y ejecución.

La entrada/salida en tiempo de ejecución estaba definida en modulos de biblioteca del compilador y no existía el concepto de programas que coexistieran sin estar ligados.Un programa en ejecución tenía el completo control sobre el hardware ya que no se manejaba el concepto de supervisor o de operaciones privilegiadas que forzaran al programa del usuario y a la e/s a estar separados por una interface formal.

2.3.3 SISTEMAS DE MULTIPROGRAMACION

Un fenomeno interesante en la industria es que los SISTEMAS BASICOS DE MULTIPROGRAMACION se desarrollaron casi simultáneamnete a los SISTEMAS AVANZADOS como respuesta a los diferentes tipos de máquinas.Algunos conceptos tuvieron que esperar a que las máquinas se desarrollaran lo suficiente para poder llevarlos a la práctica y esto se logró al arrivar las máquinas de tercera generación.

Un aspecto de particular relevancia en el desarrollo de Sistemas, es el surgimiento del DISCO a finales de los an~os 50´s y que afecto determinantemente en 3 aspectos:

a) Los tipos de dispositivos se hicieron más variados.

b) Había capacidad para hacer almacenamiento en línea.

c) La capacidad de almacenamiento directo representaba nuevas oportunidades para hacer interfaces entre programas y su E/S así como nuevos problemas para su acceso y manejo.

Además del almacenamiento en disco, hubo un incremento en el tipo de sistemas asíncronos (que no utilizan reloj para sincronizar sus funciones) y se desarrollan conceptos como "Interrupciones" ,"Privilegio" y "Protección" que surgen desarrollados en hardware.

SISTEMAS BASICOS DE MULTIPROGRAMACION TIPO BATCH

En general este tipo de sistemas requiere :

a) Gran cantidad de manejo de colas de trabajo almacenado previamente (off-line).

b) El manejo de memoria es conservado de tal forma que las particiones de memoria tienden a ser manejadas por el operador.

c) El manejo del CPU no se lleva a cabo pensando en su óptima utilización.

d) Se tiende a asignar el uso de los dispositivos durante TODA la vida del proceso y no solo el tiempo que realmente se requiere.

Un ejemplo típico es el sistema IBM DOS (Disk Operating System) tanto por sus caracetrísticas como por su desarrollo evolutivo.Es un sistema de Propósito General que fue pensado inicialmente para máquinas pequen~as.En sus versiones iniciales el sistema permitía compartir el sistema entre un número fijo y predeterminado de programas.La memoria estaba organizada en particiones decididas por el equipo encargado de hacer la instalación y esas particiones eran consideradas parte de la instalación.

Los ensambladores y compiladores traducían programas para particiones particulares y sólo corrían en ESAS particiones.

En realidad, EL SISTEMA DEFINIA MULTIPLES MAQUINAS INDEPENDIENTES QUE ERAN CONTROLADAS CADA UNA POR SU DESPACHADOR, EN FORMA INDEPENDIENTE DE LAS DEMAS, PERO PODIAN COMPARTIR EL USO DE DISPOSITIVOS Y EL CPU.

Un avance significativo sobre las máquinas de la segunda generación que soportaban sistemas de un solo flujo, es el hecho de manejar colas internas de trabajos.Anteriormente los sistemas ejecutaban los jobs en el orden en que el operador los había ordenado previamente y con el manejo interno de las colas, el sistema es capaz de ordenar y procesar los trabajos a medida que se van presentando.

Analizando esta nueva idea de las particiones,nos damos cuenta de que tiene varios problemas inherentes a esta estructura ya que al decidirse de alguna manera el tamaño de las particiones, se limita el procesamiento de trabajos grandes en estas áreas.

Por ejemplo, supongamos que se hacen dos particiones de 64 K y dos de 32 K.Para que se aprovechen al máximo, lo ideal sería que siempre corrieran programas de tamaños adecuados en cada una de las secciones;sin embargo puede darse el caso de que al compilar un programa de menor tamaño se le asigne un área de 64 K , lo que desperdicia tiempo y memoria.

El tener la memoria preparticionada, obligaba a los programadores a diseñar sus programas en forma tal que pudieran asignarseles alguna de las areas, lo cual creaba conflictos y pérdidas de tiempo.Este tipo de pequeños sistemas batch tuvieron avances como:

a) Permitir que el Operador REDEFINIERA las particiones de memoria desde la consola.

b) Hacer posible la compilación y ensamblaje RELOCALIZABLE (para que pudiera correr en cualquier partición).

c) Asignación simbólica de dispositivos.

d) Para el caso particular de DOS, se hace posible el permitir que cualquier programa de aplicación (de usuario) fuera organizado en una estructura de tareas.Esta estructura permitía que un programa en una partición particular pidiera al sistema operativo que controlara la ejecución de los programas asíncronos dentro de la misma partición.

e) Comienza a manejarse el concepto de MEMORIA VIRTUAL que son particiones de memoria que exceden el tamaño total de la amemoria física.(conceptos de segmentación y pagineo).

SISTEMAS AVANZADOS TIPO BATCH

Estos sistemas surgen a mediados de los años 60 y se distinguen de sus predecesores por el hecho de manejar los recursos del sistema.En su tiempo constituyeron un gran avance ya que realmente maximizaban el uso del equipo.De igual forma proporcionaban mayor variedad y flexibilidad en las funciones de soporte de programas, así como gran variedad de organización de datos y métodos para accesarlos.

Ejemplo de este tipo de sistemas son:OS/MVT, OS/MFT, GECOS, SCOPE y MCP que si bien difieren en cuanto a manejo de memoria , manejo de procesos y E/S tienen características que los unifica, tales como:

a) Manejo dinámico de memoria.

b) Capacidad para compartir código.

c) Elaboración de sistemas de Biblioteca (funciones para manejo de disco etc.)

d) Un eficiente manejador de trabajos.

e) El operador puede decidir el nivel de multiprogramación, cambiar las prioridades de los trabajos, o acomodarlos en una cola.

f) Los mecanismos del CPU van siendo cada vez más elaborados y completos para manejo de recursos e incluso la introducción de heurísticas que permitían mejores y más rápidas respuestas del sistema a patrones inesperados.

2.3.4 SISTEMAS DE TIEMPO COMPARTIDO

Decimos que un sistema es de tiempo compartido si su principal objetivo es distribuir los recursos de la computadora entre múltiples usuarios de forma tal que cada uno de ellos sienta que la máquina le atiende en forma particular. Los usuarios pueden ser programadores o bien usuarios en el sentido estricto de la palabra.La diferencia escencial entre los objetivos de un sistema de tiempo compartido y un sistema de tipo batch, es que el primero debe servir a los usuarios aunque no maximize la optimización del hardware, mientras que en el segundo, lo fundamental es la utilización de la máquina.

Los sistemas de tiempo compartido responden mejor si la carga de trabajo es más bien homogénea en cuanto a las funciones que va a realizar.Existe una gran variedad de sistemas de tiempo real que van desde los que manejan sólo un lenguaje de programación hasta los más complejos como MULTICS RSX11-M, TENEX Y UNIX.

En general las funciones básicas que debe proporcionar al usuario son:

1) Un conjunto de comandos de activación, pruebas y modificación de programas.

2) Todo un sistema de manejo de archivos.

3) Uno o más lenguajes de programación.

4) Una interface en tiempo real que pueda combinarse con comandos.

3. PROCESOS CONCURRENTES

3.1 DEFINICIONES DE PROCESO Y CONCURRENCIA

Consideremos al sistema operativo como un conjunto de actividades o funciones donde cada actividad consiste en la ejecución de uno o más programas que son activados en el momento que se requiere.A este tipo de actividades se les llama PROCESOS.

Un proceso puede verse como una sucesión de acciones,llevadas a cabo mediante la ejecución de una serie de instrucciones(programa) y cuyo resultado final es la ejecución de la función del sistema requerida.A la ejecución de un programa del usuario,también se le llama proceso.

La CONCURRENCIA puede ser vista como la activación de varios procesos al mismo tiempo.El aspecto no-determínistico de los Sistemas Operativos puede describirse facílmente en términos de los procesos,ya que estos pueden ser interrumpidos durante su ejecución en diferentes etapas y no podemos prredecir el orden en que estas interrupciones van a ocurrir y por tanto el orden en que las diferentes funciones van a ejecutarse.

En realidad la INTERRUPCION de un proceso puede verse como un interruptor que causa un evento impredecible que ocasiona que el proceso pierda el procesador(interrrumpa su ejecución).Es muy importante notar que un proceso es un conjunto de ACCIONES,suceptible de ser interrumpido,es decir,es DINAMICO,en tanto que un programa es una sucesión de INSTRUCCIONES,y es ESTATICO.

Los procesos dentro de un sistema de cómputo están siempre compitiendo por el uso de RECURSOS LIMITADOS (Procesador,memoria,archivos, etc) por lo que resalta la necesidad de que exista alguna manera de que los procesos se comuniquen entre sí.

Se tienen basicamente tres problemas donde la comunicación es muy importante:

Exclusión Mutua

Sincronización

Cerradura de la Muerte (Dead Lock)

La EXCLUSION MUTUA tiene como objetivo asegurar que los recursos NO-Compartibles sean accesados por un solo proceso a un tiempo.

Los recursos del sistema se dividen en dos tipos:

1.- COMPARTIBLES.- Lo que implica que pueden ser usados por varios procesos en forma concurrente.

2.- NO-COMPARTIBLES.- Su uso se restringe a un solo proceso a la vez.(ej: Impresora,lectora de tarjetas etc).

El aspecto de SINCRONIZACION debe observarse ya que debido al caracter impredecible en el que pueden ocurrir las interrupciones,la velocidad a la que se ejecutan los diferentes procesos es variable (de aní que se considere que los procesos corren asíncronamente unos con respecto a otros).Sin embargo,hay veces que es necesario asegurar la co-operación entre procesos y de alguna forma debe garantizarse la sincronía en la ejecución de sus actividades en algún o algunos puntos lo que constituye una de las funciones del Sistema Operativo.

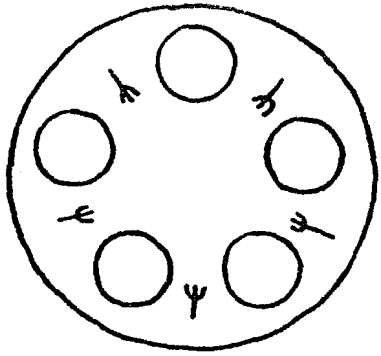
La CERRADURA DE LA MUERTE (Dead Lock) o ABRAZO DE LA MUERTE (Deadly Embrace) se presenta cuando hay varios procesos que compiten por recursos y en algún momento ninguno puede continuar su ejecución porque el recurso que necesita para continuar esta siendo utilizado por algún otro proceso,de tal forma que todos estan imposibilitados de terminar su ejecución o bien de liberar los recursos adquiridos.

3.2 EL PROBLEMA DE LOS 5 FILOSOFOS

Supongamos que varios procesos están continuamente solicitando, usando y liberando un conjunto de recursos compartidos. El mecanísmo debe ser tal que asegure que ningún proceso quedará bloqueado (i.e. que no pueda terminar su ejecución por falta de recursos.) o el que a un proceso le sea negado indefinidamente el uso de un recurso.

Para ejemplificar este problema, lo expresaremos en términos de un grupo de 5 filósofos cuyos únicos estados permitidos son comer y pensar.Cada filósofo está ocupando un lugar en una mesa circular.Para comer, necesita DOS tenedores necesariamente y solo hay cinco tenedores en la mesa,(uno entre cada filósofo).Los únicos tenedores que puede usar un filósofo son los que le quedan a la izquierda y a su derecha.

Todos los filósofos siguen el mismo patrón de comportamiento , es decir, comen y piensan alternadamente; ahora el problema es simular el comportamiento de los 5 filósofos evitando que un filósofo quede bloqueado, y evitando que una petición por el uso de un tenedor, le sea negada siempre.



Observación.- Cabe notar que dos filósofos adyacentes no pueden comer simultáneamente y que en un tiempo particular solo pueden estar comiendo dos concurrentemente lo que implica que esto es el máximo paralelismo.

Como una primera solución propondremos la siguiente: Un filósofo se adueña de un tenedor a la vez, por decir algo, el izquierdo primero y luego el derecho, haciendo llamadas a las rutinas adecuadas. Para regresar los tenedores sigue el proceso inverso, pero también uno a la vez.Esto queda expresado como:

```
filosofos()
{
        while(1)
        {
                tomatenedor(izq);
                tomatenedor(der);
                come();
                dejatenedor(izq);
                dejatenedor(der);
                piensa();
        }
}
```

Las entradas son parte del monitor que controla el acceso a los tenedores, que se manejan por medio de un arreglo en el que tenedores[i] indica si el tenedor i-esimo esta disponible (es decir tenedores[i]=1) o no (tenedores[i]=0).

Esta es una solución sencilla, pero que adolece de un gran defecto:Eventualmente nos puede llevar al "deadlock".

Supongamos que cada uno de los filósofos toma al mismo tiempo su tenedor izquierdo;al tratar de accesar el tenedor derecho, esta petición les será negada a todos. La Cerradura de la Muerte se dará porque los procesos no liberan recursos que permitirían a interesa dar una solución que evite la cerradura de la muerte .Cada filósofo sera representado por un proceso que repite indefinidamente el siguiente ciclo:

```
while (1)
{
        tomatenedores(i);
        come();
        dejatenedores(i);
        piensa();
}
```

Aquí el tomar y dejar los tenedores, implica hacerlo para los dos al mismo tiempo.El arreglo que denotaremos como numtenedores(i), sera manejado de manera diferente porque ahora tendrá 5 localidades, cada una representando a cada filósofo y contendrá el número de tenedores disponibles para cada uno en un momento dado (esto es, pueden ser 0,1 o 2).

La entrada a la rutina tomatenedores(i), se permitirá solo cuando numtenedores(i) sea igual a 2.De no darse esta condición,debe esperar a que su variable de control (listo (i)) se prenda.Cada vez que a un filósofo le es permitido el uso de los tenedores, debe decrementar en uno el número de tenedores disponibles a sus vecinos, en el arreglo numtenedores[i].

Llamaremos izq(i) y der(i) a los vecinos del filósofo i.(recordemos que la mesa es circular y por tanto el vecino izquierdo del filósofo 5 es el 1 etc.)Cuando un filósofo desocupa sus tenedores, debe incrementar los tenedores disponibles de sus vecinos y si al hacerlo, numtenedores[i-1]=2 o numtenedores[i+1]=2, debe encender sus respectivas variables de control (ready(i-1), ready(i+1)) para que éstos puedan accesarlos posteriormente.

Si actuamos de esta manera, nos hemos evitado el deadlock pero aún no resolvemos el caso en que a un filósofo le sea negado indefinidamente el uso de sus tenedores,cosa que podría pasar sí:

1) Supongamos que el filósofo 1 toma sus tenedores.

2) El filósofo 2 no puede entonces comer,pero el filósofo 3 sí.

3) El filósofo 3 toma sus tenedores y empieza a comer.

A partir de este momento, se suceden alternativamente los hechos donde el filósofo 1 y 3 estan comiendo,, de forma tal que el filósofo 2 tiene siempre ocupado al menos uno de sus tenedores, lo que le impide comer.

Una solución es llevar una estadística de que tan antigua es una petición,de manera que si un proceso lleva mucho tiempo solicitando recursos sin que le sean asignados, el sistema lo dtecte y se los asigne la siguiente vez que los solicite.

3.3 COMUNICACION ENTRE PROCESOS.

3.3.1 Para la comunicación entre procesos,Dijkstra,introdujo en 1965 el concepto de SEMAFOROS que no son otra cosa que un entero no-negativo el cual una vez asignado un valor inicial,este sólo se modifica por medio de las señales de WAIT y SIGNAL.

Estas operaciones actúan sobre el semáforo de la siguiente manera:

1.- SIGNAL(S)

Tiene el efecto de incrementar el valor del semáforo s en una unidad y este incremento debe verse como una operación INDIVISIBLE y no es equivalente a ejecutar "S:=S+1" ya que es posible que dos procesos alteren el valor del semáforo SIMULTANEAMENTE,ocasionando errores en la evaluación final.

2.- WAIT(S)

Su efecto es decrementar el valor del semáforo en una unidad siempre y cuando S sea no negativo y aquí nuevamente necesitamos que la operación sea hecha de manera indivisible y no es equivalente a tener "S:=S-1" ya que esto podría ocasionar incluso que el valor del semáforo pudiera volverse negativo en algún momento.

El valor de un semáforo está relacionado con su valor inicial y el número de operaciones WAIT y SIGNAL que se le hayan aplicado y esta determinado de la siguiente manera:

```
+-------------------------------+
| V(S) = C(S) + nS(S) - nW(S) |                    (1)
+-------------------------------+
```

donde

V(S) - Valor del semáforo
C(S) - Valor inicial.
nS - Número de operaciones SIGNAL que se le han aplicado.
nW - Número de operaciones WAIT que se le han aplicado.

además se tiene por definición que V(S) >= 0 siempre.

Tomando en cuenta (1) se tiene la siguiente relación:

```
+------------------------+
| nW(S) < = nS(S) + C(S) |                          (2)
+------------------------+
```

Estas relaciones nos servirán después para poder analizar (y en algunos casos garantizar) que durante la ejecución de procesos no se presentarán problemas de DEAD LOCK.

Volviendo al problema de la comunicación entre procesos veremos en que forma se aplicarían los semaforos como posibles soluciones a las situaciones planteadas con anterioridad:

3.3.2 EXCLUSION MUTUA

Aquí lo que nos interesa garantizar es que UN SOLO proceso tendrá el uso de un recurso no-compartible a un tiempo y con este objeto definiremos una REGION CRITICA que es la sección del programa que sólo puede ser ejecutada por un proceso a la vez.Es importante resaltar el hecho de que el uso de las regiones críticas presentan varias aplicaciones ya que dentro de ellas se incluye el código para la utilización de los recursos tales como Memoria,Impresora,Archivos etc.Por ejemplo en el caso de código reentrante (código puro que puede ser ejecutado por varios procesos simultáneamente),podemos tener definidas regiones críticas que podrán ser ejecutadas por tan solo uno de ellos a un tiempo.

Ahora el problema se transforma en garantizar la exclusión mutua en la ejecución de las regiones críticas,que puede lograrse mediante la utilización de WAIT y SIGNAL.

Observe que la "compartibilidad" de una región crítica se establece en base al valor inicial de los semáforos que la circundan,es decir,sí el valor inicial del semáforo es N,entonces N procesos pueden ejecutar ese código concurrentemente.En el caso particular donde N=1 sólo un proceso puede entrar a la región crítica a la vez.

La manera de definir la Región Crítica sería:

```
WAIT(S1)
        |
        |
    REGION   CRITICA
        |
        |
SIGNAL(S1)
```

dandole a S1 un valor inicial de 1 de tal forma que se cumpla que:

$$nw(S1) <= n(S1) + 1$$

lo que implica que a lo más un proceso ejecuta la región crítica a la vez.

3.3.3 SINCRONIZACION.

Para el problema de la sincronización debemos contar con algún medio para suspender la ejecución de un cierto proceso en un punto,en tanto otro proceso no alcance otro punto deseado.Para lograrlo basta con poner una operación de WAIT sobre un semáforo con un valor inicial de 0 en el lugar donde desea suspenderse la ejecución y una operación de SIGNAL del mismo semáforo en el punto donde deben sincronizarse ambos procesos.

Supongamos que tenemos dos procesos A y B y nos interesa por alguna razón que A se detenga en algún punto hasta estar seguros de que B ejecutó ciertas acciones,entonces podríamos sincronizarlos como sigue:

S2 es un semáforo con valor inicial 0,lo que implica que A suspende su ejecución en el punto P1 (esto es porque a tratar de ejecutar el WAIT se encuentra con que el semáforo tiene un valor de 0 y no puede decrementarlo) hasta que B ejecute el SIGNAL (lo que implicaría poner el semáforo en 1 y permitir de ese modo que A ejecute finalmente el WAIT y termine de correr).

Existe un ejemplo típico de la sincronización entre procesos y se conoce con el nombre de "Problema del Consumidor y el Productor", consiste de dos procesos A y B donde A es el productor y esta siempre en el ciclo de "produce un elemento y depositalo en la cola" en tanto que B es el consumidor que esta permanentemente en el ciclo "extrae un elemento de la cola y consúmelo" y la cola tiene una capacidad limitada a N elementos.

Aquí se requiere sincronización en ambos sentidos ya que el proceso A no puede seguir metiendo elementos en caso de que la cola este llena ,asi como el proceso B no puede extraer elementos de la cola si esta vacía.Lo que tenemos que asegurar es que se cumpla que:

$$0 <= D - E <= N \qquad (3)$$

donde

D = Número de elementos depositados
E = Número de elementos extraídos
N = Capacidad del buffer(cola)

La manera de implementar las regiones críticas es:

PRODUCTOR

```
wnile (true)
{
   produce un elemento;
   WAIT(espacio disponible)
        Deposita un elemento en la cola
   SIGNAL(manejo de buffer)
   SIGNAL(espacio disponible)
}
```

CONSUMIDOR

```
wnile (true)
{
   WAIT(elemento disponible)
   WAIT(manejo de buffer)
        Extrae un elemento de la cola
   SIGNAL(manejo de buffer)
   SIGNAL(elemento disponible)
   consume el elemento
}
```

La sincronización se asegura por medio de los semáforos ESPACIO DISPONIBLE y ELEMENTO DISPONIBLE cuyos valores iniciales son N y 0 respectivamente y la exclusión mutua se lleva a cabo con el semáforo MANEJO DE BUFFER que tiene un valor inicial de 1.

Utilizando las relaciones (1) y (2) vistas anteriormente podemos demostrar que sí se organizan de esta forma las regiones críticas los procesos podrán ejecutarse sin que se presente el problema del DEAD LOCK que en este caso significaría que el proceso A esperara eternamente a que el proceso B extrajera un elemento de la cola y éste no pudiera extraerlo en tanto A no meta un elemento en la misma.

La demostración es como sigue:

i) $0 <= nW(\text{espacio disponible}) <= nS(\text{espacio disponible}) + N$

ii) $0 <= nW(\text{elemento disponible}) <= nS(\text{elemento disponible})$

por el orden en que se ejecutan los WAIT y SIGNAL tenemos las siguientes relaciones:

iii) $nS(\text{elemento disponible}) <= D <= nW(\text{espacio disponible})$
iv) $nS(\text{espacio disponible}) <= E <= nW(\text{elemento disponible})$

entnces tenemos por un lado que:

$D <= nW(\text{espacio disponible})$ por iii

y

$D <= nS(\text{espacio disponible}) + N$ por i

por lo que tenemos

$$D <= E + N$$

de igual manera obtenemos...

$E <= nW(\text{elemento disponible})$ por iv

$E <= nS(\text{elemento disponible})$ por iii

y entonces se cumple que

$$E <= D$$

Finalmente tenemos la relación

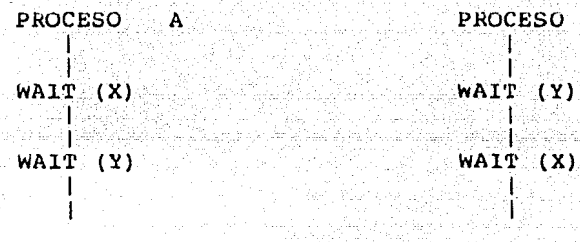
$$0 <= D - E <= N$$

que es lo que queriamos demostrar.

3.3.4 CERRADURA DE LA MUERTE (DEAD LOCK)

Esta situación se produce como ya dijimos porque los procesos estan esperando unos a otros a que terminen de ejecutar ciertas acciones o bien a que liberen los recursos que tienen ocupados.

Considerese la siguiente situación:



Suponiendo que el valor inicial de X y Y fuera 1,el primer WAIT en ambos procesos podría ejecutarse,más al llegar al segundo WAIT ,ni A ni B podrían continuar y se produciría el DEAD LOCK.

En el problema del "Productor y el Consumidor" (sección anterior) ,podría ocurrir el problema del punto muerto si se cambia el orden de ejecución de las señales de WAIT y SIGNAL.Si el buffer está vacío y el consumidor ejecuta el WAIT (manejo de buffer), al llegar al WAIT(elemento disponible) no puede seguir hasta que el productor deposite un elemento en el buffer y esto no puede ser porque el consumidor tiene bloqueado el manejo del buffer.Es claro entonces que es responsabilidad del programador el ordenar correctamente las operaciones de WAIT Y SIGNAL,ya que de ninguna manera su uso es garantía de que el Punto Muerto no va a presentarse.

Algunos Sistemas Operativos utilizan una técnica rudimentaria para evitar este problema y consiste en obligar a los procesos a definir de antemano TODOS los recursos que requieren para terminar su ejecución de tal forma que no comienzan a ejecutarse hasta estar seguros de que podrá terminar sin problemas.Naturalmente esto implica un desperdicio de recursos ya que estan ocupados más tiempo del requerido básicamente.

Es necesario que estas rutinas formen parte del núcleo del sistema operativo por varias razones entre ellas:

i) Deben ser accesibles a TODOS los procesos.
ii) Como la funcion WAIT puede ocasionar que un proceso cambie su status a "no-ejecutable",es necesario que este relacionado con el despachador.
iii) De igual forma la función SIGNAL puede modificar el status y marcarlo "Listo para correr" y debe estar relacionado con el manejador de interrupciones y con el despachador.

La primera forma de alterar el status es "BLOQUEANDO" o "DESBLOQUEANDO" procesos y para tal efecto,se maneja una COLA DEL SEMAFORO que funcione de la siguiente manera:

Cuando un proceso realize un WAIT sin éxito (es decir el valor del semáforo es cero),implica que ese proceso sea puesto en la cola del semáforo y sea marcado como NO-EJECUTABLE o BLOQUEADO.Análogamente cuando se realiza una operación SIGNAL en algún semáforo es posible que algún proceso se torne EJECUTABLE o DESBLOQUEADO y por tanto sea extraído de la cola del semáforo y sea marcado como listo para correr.

OBSERVACION.- Es diferente el hecho de marcar un proceso "Listo para Correr" a la función que realiza el despachador ya que éste escoge entre los procesos "listos para correr" al que tenga mayor prioridad y le asigna el procesador para hacerlo el proceso actual (que es el que tiene el procesador).

Para el manejo de esta cola bastaría con utilizar la política normal del manejo de cola (es decir, el primero en entrar es el primero en salir) y esto nos asegura de alguna forma que todos los procesos serán ejecutados ,pero también resulta conveniente el asignar PRIORIDADES dentro de la cola de manera que los procesos que requieren de rápida atención la obtenga rápidamente.

Con respecto a la asignación del procesador,el WAIT y SIGNAL actuan como EVENTOS SIGNIFICATIVOS ya que en caso de que ningún proceso resulte alterado en su status,el proceso actual continua su ejecución.La parte más importantre es la indivisibilidad de estos eventos y deberán ser implementados como procedimientos con algún mecanismo de protección.Con este objeto,introduciremos las operaciones de LOCK y UNLOCK.

En el caso de UN SOLO PROCESADOR,estas operaciones resultan triviales de implementar ya que basta con desnabilitar las interrupciones en el caso de LOCK y volverlas a habilitar para el caso del UNLOCK.En el caso de VARIOS PROCESADORES no resulta tan directo ya que al poder correr dos procesos en distintos procesadores,podrían entrar simultáneamente a la región crítica y causar problemas.Para evitar este conflito,,es necesario adecuar las funciones LOCK y UNLOCK para lo cual necesitamos una operación que cheque el valor de una localidad y modifique su contenido en una sola operación;a esta instrucción la denominamos TEST & SET y se implementa a nivel de hardware.La idea es usar una localidad como bandera que nos indique por medio de su contenido si se permite la entrada o no a los procesos a las rutinas de WAIT y SIGNAL.

Ahora la operación de LOCK se traduce en efectuar TEST & SET de la bandera,lo que causa que sea checada y puesta en cero al mismo tiempo.En el caso de que el valor de la bandera nubiera sido cero de antemano significa que algún otro proceso está ejecutando la región crítica y entonces ese proceso no puede continuar y se queda ejecutando un ciclo para intentar pasar.A este ciclo se le conoce como "BUSY WAIT" que se ejecuta hasta que el proceso que está dentro de la región crítica efectue el UNLOCK y permita la entrada de otros procesos.

Es muy importante notar que una solución de este tipo,nos lleva a desperdiciar el procesador en el caso de que un proceso no pueda pasar del LOCK y se quede ejecutando el busy wait.Asimismo es bueno enfatizar el hecho de que LOCK y UNLOCK NO SON SUBSTITUTOS DE WAIT Y SIGNAL y para clarificar este hecho pondremos la siguiente tabla:

	WAIT & SIGNAL	LOCK & UNLOCK
Propósito	Sincronización de procesos	Exclusión Mutua de las señales de WAIT y SIGNAL
Nivel de implementación	Software	Hardware
Mecanísmos de Espera	Manejo de Cola	Inhibición de Interrupciones o Busy wait

****

4. EL NUCLEO DEL SISTEMA

Ya hemos discutido que podríamos ver al Sistema Operativo como un conjnto de facilidades que nos proporcionan una máquina virtual mas accesible y útil para el usuario.Para llevar a cabo el diseño de dichas funciones,podemos verlas como capas o niveles que se superponen al hardware con que cuenta la máquina en sí.En la capa más baja se cuenta unicamente con las facilidades del hardware y el principal medio de comunicación entre el hardware básico y el Sistema Operativo esta dado por el NUCLEO DEL SISTEMA cuya función es PROPORCIONAR EL MEDIO AMBIENTE ADECUADO PARA QUE LOS PROCESOS PUEDAN EXISTIR.

Naturalmente para que surja este medio ambiente, debemos contar con algunas funciones básicas en el hardware tales como:

1) Mecanismos de interrupción.

2) Protección de Memoria.

3) Conjunto de Instrucciones Privilegiadas.

4) Reloj de Tiempo Real.

4.1 El MECANISMO DE INTERRUPCION es necesario para permitir que pueda interrumpirse la corrida de un proceso y este mecanismo debe ser tal que salve, al menos el valor del PC (Program Counter) del proceso interrumpido y transfiera el control a una localidad fija de memoria.Esta localidad es la entrada a una rutina llamada RUTINA DE MANEJO DE INTERRUPCIONES cuyo objetivo es determinar la causa de la interrupcion y tomar la acción correspondiente.

El mecanismo de PROTECCION DE MEMORIA es necesario para asegurar que los procesos no violen las areas asignadas a otros procesos mientras corren.

El conjunto de instrucciones privilegiadas debe contener instrucciones del tipo:

a) Permitir nabilitación y desnabilitacin de interrupciones.

b) Rotar el procesador entre todos los procesos.

c) Accesar registros usados por el hardware de protección de memoria .

d) Realizar la entrada/salida.

e) Detener el procesador central.

Es claro que no cualquier usuario tendrá acceso a ejecutar las instrucciones privilegiadas y por lo tanto se hace necesaria la existencia de dos modos de ejecución de procesos :

1) MODO SUPERVISOR

2) MODO USUARIO

El RELOJ DE TIEMPO REAL es esencial para el desarrollo de las políticas del despachador(scheduler) y para contabilizar los recursos consumidos por los usuarios.

Debido a las consideraciones anteriores, podemos concluir que el núcleo esta constituido por tres partes fundamentales :

a) MANEJADOR BASICO DE INTERRUPCIONES.

b) DESPACHADOR DE BAJO NIVEL.

c) LAS RUTINAS DE WAIT Y SIGNAL. (Ver procesos concurrentes).

Es interesante notar que debido a que el núcleo del sistema es la parte más directamente relacionada con el hardware, es la parte más dependiente de sus limitaciones y más cercana a sus facilidades y es quizá la única parte del sistema que necesite ser escrita en lenguaje ensamblador.

4.2 DESCRIPTOR DEL PROCESO.

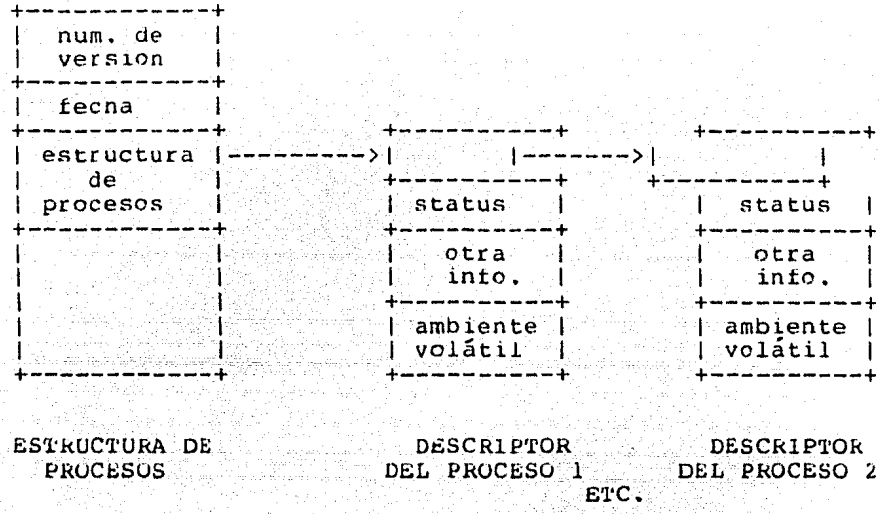
Ya mencionamos que es un proceso, ahora veremos como manejarlo dentro del sistema.Cada proceso puede ser representado por medio de un DESCRIPTOR DEL PROCESO (también conocido como bloque de control o vector de estados) que es un área de memoria que contiene toda la información relevante acerca de cada proceso.

Un proceso se encuentra en cualquiera de tres estados es decir, puede estar corriendo,listo para correr o no ejecutable (bloqueado).

El estado en que se encuentran los procesos es esencial para el despachador ya que es el el encargado de asignar el procesador entre los diferentes procesos.

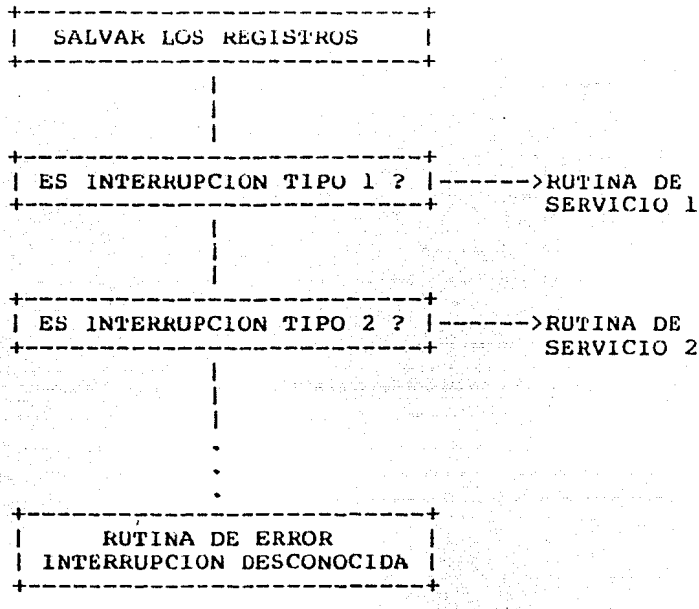
Dentro del DESCRIPTOR DEL PROCESO podemos incluir información del proceso que necesita ser salvada cuando por alguna razón le sea retirado el procesador pueda asegurarse que el proceso podrá seguir ejecutandose.

A está información se le denomina el MEDIO AMBIENTE VOLATIL e incluye datos como: valor de los registros de la máquina(pc,acumulador,registros de índice, registros de trabajo, archivos abiertos etc).El Descriptor del proceso de cada uno de ellos está ligado a los demás por medio de una ESTRUCTURA DE PROCESOS que actua como descriptor.También se contará con una TABLA CENTRAL DE PROCESOS que contiene información general acerca de los procesos (número de versión,tecna,apuntadores a los descriptores del proceso etc).



MANEJADOR BASICO DE INTERRUPCIONES

Es el encargado de responder a las sen˜ales del mundo externo (INTERRUPCIONES) y del propio sistema de cómputo (TRAPS). El objetivo del manejador es determinar la causa de la interrupción y atenderle.Se supone que el mecanismo de interrupción de la máquina debe al menos salvar el PC del proceso interrumpido,pero de no ser así, esta es la primera función el manejador.La facilidad con que puede determinarse la causa de la interrupción depende de las facilidads dadas por el hardware y en el caso de que sean escasas, se debe implementar una CADENA DE BRINCOS (Skip Chain) que se maneja cuando todas las interrupciones transfieren el control a una misma localidad y dependiendo del estado de las banderas de status , se brinca a diferentes subrutinas.



Las interrupciones al procesador central son inhibidas generalmente cuando se está trabajando en modoSUPERVISOR y esto asegura que los valores de los registros que deben salvarse cuando se entra a la rutina de manejo de interrupciones no sean superpuesto con otras interrupciones posteriores pero que ocurran antes de salir de la rutina.

Esto nos lleva a otro poblema:hay dispositivos que no pueden esperar demasiado a que las interrupciones sean atendidas, ya que podría perderse información etc.

Para evitarlo, se deben asignar PRIORIDADES a las interrupciones para que así pueda decidirse si la rutina de interrupciones permite o no ser interrumpida a su vez. Una interrupción a un cierto nivel, inhibe automáticamente otras interrupciones a un nivel menor o igual al suyo.

La segunada función de la rutina de interrupciones, es atenderlas dependiendo de quien produjo la señal.Finalmente resulta interesante subrayar que si ocurre una interrupción ya sea externa o interna altera con frecuencia el status de un proceso.Citemos por ejemplo el caso en que una operación WAIT, o un requerimento de entrada/salida se efectua;puede provocar que el proceso actual (el que tiene el procesador), sea suspendido y su status cambie de "proceso actual" a "proceso no ejecutable" en tanto la interrupción no se atendida.

4.3 EL DESPACHADOR

La función del DESPACHADOR es asignar el procesador central entre los diferentes procesos en el sistema.Su influencia es necesaria cuando el proceso actual no puede seguir corriendo o bien hay bases para suponer que el procesador deba ser asignado a otro proceso.

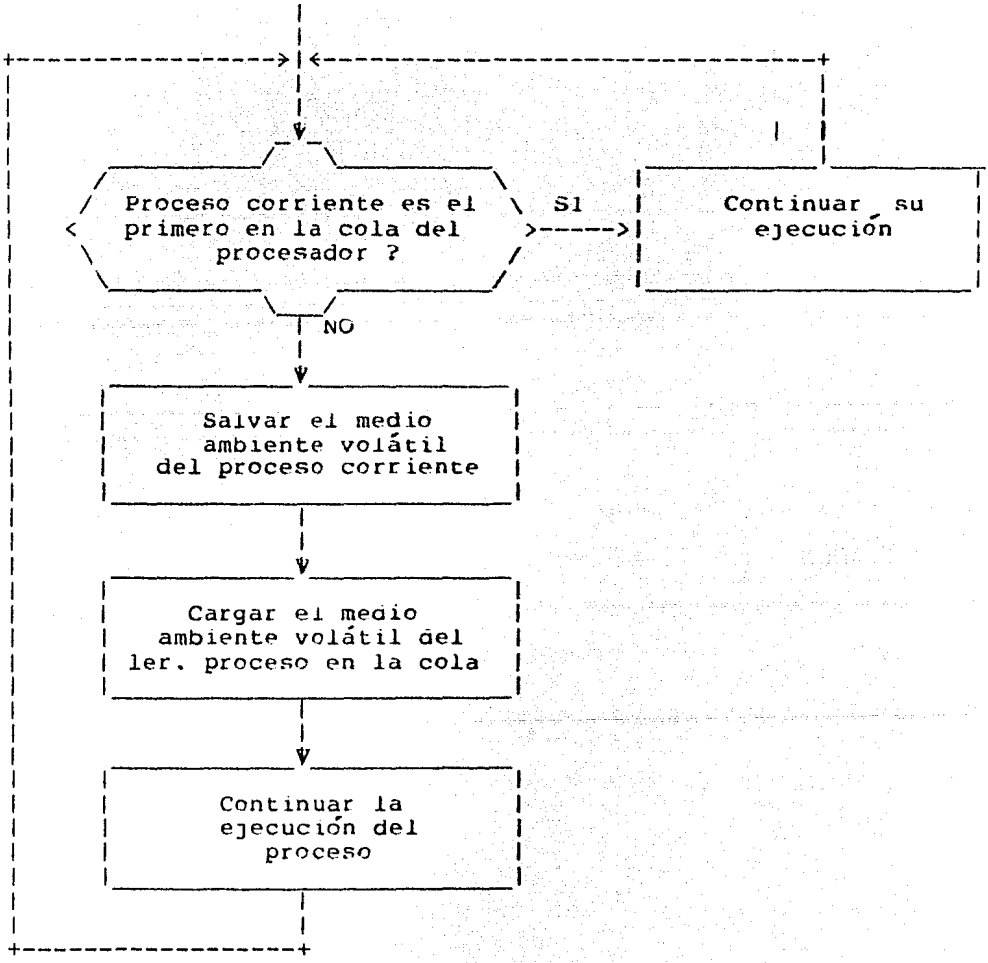
El tipo de situaciones que podrían presentarse son:
a) Que ocurra una interrupción externa que altere el satus del proceso.

b) Ocurra un WAIT que imposibilite al proceso para seguir corriendo .

c) Que ocurra una interrupción cualquiera y el proceso se suspende nasta que esta sea atendida.

Sin embargo, la asignación de prioridades a los procesos NO ES FUNCION DEL DESPACHADOR sino del MANEJADOR (scheduler) y en general para decidirlos se toma en cuenta la cantidad de recursos requeridos, el tiempo de procesador que lleva consumido, la prioridad asignada al usuario.

El despachador debe correr el primer proceso en la cola de los procesos "Listos para correr" y este puede ser o no el mismo proceso que estaba corriendo antes de invocar al despachador.



ALGORITMO DE FUNCIONAMIENTO DEL DESPACHADOR

Además podemos describir la secuencia que se sigue desde que se efectua una señal de interrupción hasta que interviene el despachador:

i) INTERRUPCION

Salva el PC.
Salva otros registros (opcional).
Entra a la rutina de manejo de interrupciones.

ii) MANEJADOR DE INTERRUPCIONES

Salva otros registros (si no lo hizo antes).
Identifica la causa de la interrupción.
Se entra a la rutina de servicio correspondiente.

iii) RUTINA DE SERVICIO

Se atiende la interrupción.Posiblemente se altere el status del proceso.

iv) DESPACHADOR

Se analiza cual es el nuevo Proceso Corriente,se carga su medio ambiente volátil y se le transfiere el control.

5. MANEJO DE MEMORIA

5.1 OBJETIVOS DEL MANEJADOR DE MEMORIA.

El manejador de memoria se encuentra dentro del siguiente nivel o capa de nuestro sistema operativo y sus principales tareas son:

1.- Permitir la RELOCALIZACION de programas.

2.- PROTECCION del área de ejecución de procesos.

3.- Permitir una buena ORGANIZACION LOGICA.

4.- Permitir una buena ORGANIZACION FISICA.

Cada una de ellas tiene gran importancia para el usuario ya que:

i) En el caso de la RELOCALIZACION el usuario no sabe de antemano,cuando está escribiendo su programa,en que localidades de memoria es que va a correr y por tanto resulta imposible el usar direcciones absolutas.

ii) Cuando se tienen varios procesos compartiendo la memoria,es necesario contar con un mecanismo que permita asegurarnos de que un proceso no alterará localidades asignadas a otro proceso.El mecanismo debe ser tal,que cheque dinámicamente cualquier referencia a memoria y asegurarse de que se está trabajando dentro de los límites asignados a ese proceso.

iii) Las computadoras en general,tienen un espacio de direccionamiento unidimensional o lineal,y es ahí donde los procesos son cargados.Esta forma de almacenamiento no necesariamente refleja el orden de ejecución de un programa,ya que la mayoría de los módulos o rutinas que afectan diferentes áreas de datos etc.

La DIVISION LOGICA de un programa repercute en una cierta SEGMENTACION del espacio direccionable(esto es,el programa se divide en módulos lógicos) lo que nos brinda las siguientes posibilidades:

a) Es posible codificar los segmentos de forma independiente de otros y el sistema se encarga de resolver las referencias externas.Esto se puede hacer al momento de ligado o en tiempo de ejecución.
b) Se puede ofrecer protección extra a los segmentos como solo lectura o solo ejecución con poco incremento de costo.
c) Es posible introducir mecanismos que permitan compartir segmentos entre procesos.

iv) Además de la organización lógica se tiene el problema de que cuando el programa es demasiado grande y no cabe en el espacio de direccionamiento,es necesario mover partes de el de memoria principal a memoria secundaria y viceversa.A esta técnica se le denomina TECNICA DE OVERLAYS o traslapamiento.Anteriormente el programador era el encargado de manejar los cambios por sí mismo,pero esto le implicaba un mayor trabajo y una gran pérdida de tiempo.Actualmente es el Sistema Operativo el que resuelve este conflicto que aparece transparente para el usuario.

5.2 MEMORIA VIRTUAL

Resulta mas o menos claro que los problemas planteados anteriormente son causados porque el espacio direccionable no es suficiente o no es manejado de manera adecuadapor lo que utilizaremos un mecanismo de traducción de direcciones llamado MAPA DE DIRECCIONES que se encarga de relacionar las direcciones usadas por el programador con las localidades físicas que se le asignen.Al rango de direcciones del programa se le denomina ESPACIO DE DIRECCIONES (ED) y al rango de las localidades de memoria , ESPACIO DE MEMORIA (EM) . Visto como función quedaría expresado como :

f: ED ---> EM

El Espacio de Memoria es lineal y su tamaño es igual a la cantidad de memoria incluida en la configuracion física del sistema de cómputo.Sin embargo el Espacio de Direcciones no siempre es lineal y su tamaño puede ser menor, igual o mayor que el espacio de memoria.

Existen varias formas de implementar Memoria Virtual:

i) Registros BASE y LIMITE
ii) Paginación
iii) Segmentación

5.2.1 REGISTROS BASE Y LIMITE

Usando los registros BASE y LIMITE tenemos que cuando un proceso es cargado a memoria,la dirección más baja se copia como valor del REGISTRO BASE y todas las direcciones del programa se manejan relativas a esa dirección base,cumpliendose entonces que:

f(dirección) = dirección + contenido del registro base

Esto asegura la relocalización y para asegurar la protección del espacio de memoria entre procesos, incluimos un REGISTRO LIMITE que contiene la dirección de la localidad más alta que le es permitido accesar a ese proceso en particular.Con esta implementación se cumple el siguiente patrón de comportamiento:

i) Si la dirección es menor que cero, implica que nay una violación de memoria.

ii) a´ = a + B

a´= Nueva dirección.
a = Dirección.
B = Contenido del registro base.

iii) Si a > límite entonces hay violación de memoria.

iv) a´ es la dirección requerida.

Resaltemos el hecho de que el espacio de direcciones mapeado en este esquema es lineal y su tamaño se obtiene haciendo la diferencia de los contenidos de los registros base y límite.Con objeto de que las operaciones de mapeo sean rápidas,es sumamente conveniente que los registros base y límite esten implementados en hardware rápido (generalmente semiconductores).

Una variante de esta implementación se presenta cuando se maneja la longitud del espacio de memoria en lugar del limite superior ya que así las condiciones de violación de memoria se revisan en paralelo antes de realizar la suma con el contenido del registro base para obtener la dirección relativa.

Como resulta económicamente incosteable el hecho de tener un par de registros para cada proceso , se tiene un par para cada procesador y estos registros forman parte del MEDIO AMBIENTE VOLATIL que se almacena cuando un proceso es suspendido.

En el caso de Código Reentrante,se mmanejan dos pares de registros donde un par delimita el tamaño del codigo y sus valores son comunes a TODOSlos procesos mientras que el otro par encierra al área de datos asociada a ese código para cada proceso que es claramente diferente en cada caso.

Cabe mencionar que en este esquema,el espacio direccionable es necesariamente menor o igual que el espacio de memoria.

$$ED <= EM$$

5.2.2 PAGINACION

Otra forma de implementar memoria virtual pero con un espacio de direccionamiento mayor al del espacio físico es logrado cuando se utiliza la técnica de PAGINACION.

El objetivo de ésta técnica es simular que no nay distinción entre tener partes de un programa en memoria principal o tenerlo en memoria secundaria.

La PAGINACION consiste básicamente en dividir el Espacio de Direccionamiento Virtual en PAGINAS DE IGUAL TAMAÑO al igual que la memoria principal.

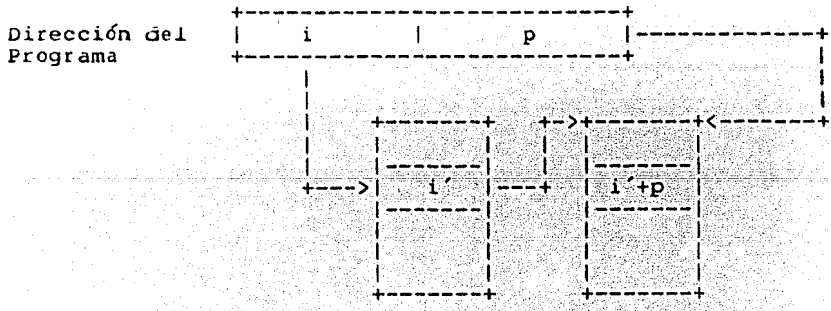
A las páginas residentes en memoria principal se le llaman PAGINAS ACTIVAS y a las que residen en memoria secundaria, PAGINAS INACTIVAS.El mecanismo de paginación debe cubrir dos funciones fundamentales :

a) Determinar cual programa se refiere a una cierta dirección y ver donde se localiza esa página.
b) Hacer las transferencias necesarias de memoria principal a secundaria siempre que se requiera y viceversa.

En el caso (a) se deben tomar los bits de más alto orden en la dirección y se interpretan como el número de página, mientrs que los bts de bajo orden se toman como el número de palabra dentro de esa página. Si el tamaño de la página es de 2 ** N los N bits menos significativos representan el número de palabra y el resto,el número de página.

Esta separación de la dirección en PAGINA y NUMERO DE PALABRA es una función que se realiza por HARDWARE y resulta transparente para el usuario,pero es necesario que el sistema operativo interprete cada uno de los campos y haga las transferencias necesarias.

Para esto, se maneja una TABLA DE PAGINAS cuyo tamaño es proporcional al tamaño del espacio direccionable y en la cual la i-ésima entrada de la tabla, corrresponde a la localidad i´ en memoria principal, que contiene el número de página i.Para mayor claridad ver el siguiente diagrama:



El número de palabra (p) se agrega a i´ para obtener la localidad requerida

donde:

d = dirección
i = numero de página
p = numero de palabra
z = tamaño de la página

y se relacionan de la siguiente manera:

i = d DIV z (parte entera de z)

p = d MOD z (residuo de z)

Como se puede detectar que se esta haciendo referencia a una página inactiva? Pues cuando al consultar la tabla de páginas, se encuentra con que su correspondiente entrada esta desocupada.En este caso se genera una interrupción de "Falta de Página" lo que obliga a suspender el proceso hasta que la página requerida sea transferida a memoria principal.

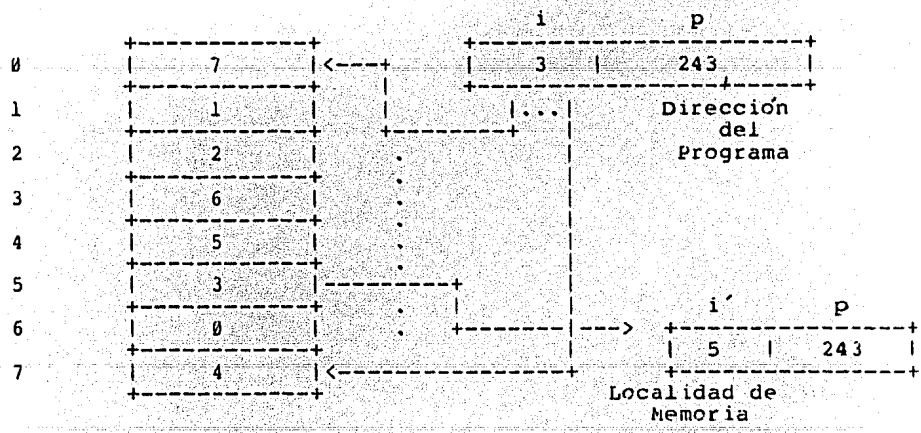
El algoritmo que debe seguirse para el reemplazo debe considerar aspectos tales como:

a) Número de veces que se ha referenciado una página.
b) El tiempo en que se hizo la última referencia a esa página.
c) Dónde esta escrita la página etc.

Estas políticas de reemplazo serán discutidas ampliamente dentro de las políticas de ASIGNCION DE MEMORIA.

Existen algunos casos donde se cuenta con REGISTROS DE DIRECCIONES DE PAGINAS (PAR Page Address Registers) cada uno de los cuales contiene el número de una página activa y tienen la propiedad de que pueden ser examinados SIMULTANEAMENTE.

ESQUEMA DEL MANEJO ASOCIATIVO



Para que sea posible direccionar todas las páginas activas,es necesario que haya igual número de registros que páginas en memoria principal .

Podría presentarse el problema de querer saber si una dirección pertenece o no al proceso actual para lo cual basta con agregar un bit a los PAR´s para indicar,mediante un 0 si pertenece o mediante un 1 si no pertenece a dicho proceso.Nuevamente se tiene la necesidad de mantener un conjunto de Registros de Direcciones de Páginas para cada procesador.A este mecanismo se le denomina LMACENAMIENTO ASOCIATIVO.

5.2.3 SEGMENTACION

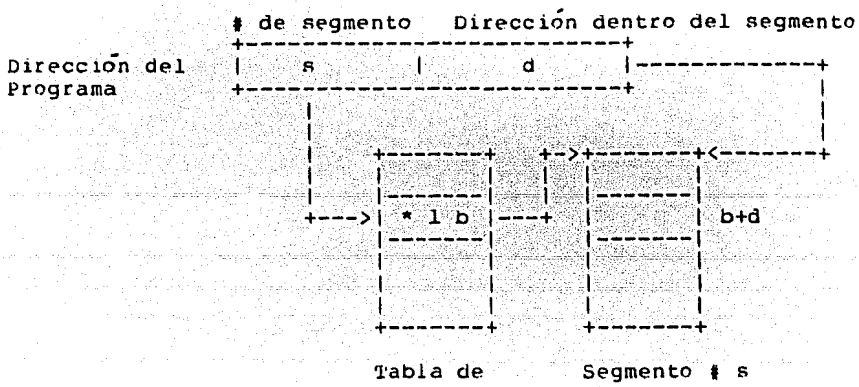
Además de la técnica de Paginación, se cuenta con la técnica de SEGMENTACION, donde la idea es dividir el Espacio de Direcciones en SEGMENTOS , cada uno de los cuales corresponda a procedimientos,módulos de programa o colección de datos.

LA DIFERENCIA FUNDAMENTAL ENTRE PAGINAS Y SEGMENTOS ES QUE LAS PAGINAS SON DE TAMAÑO FIJO Y LOS SEGMENTOS SON DE TAMAÑO VARIABLE.

Al igual que en el caso anterior, la segmentación puede plementarse por medio de una TABLA DE SEGMENTOS y las direcciones se ..viden en: NUMERO DE SEGMENTO y DIRECCION DENTRO DEL SEGMENTO.

El Mapa de Direcciones realizado en base a la Tabla de Segmentos obedece al siguiente comportamiento:

a) Extraer la dirección del programa (s,d)
b) Usar s como índice en la tabla de segmentos
c) Si d < 0 o d > longitud entonces hay violación de memoria
d) (b + d) es la localidad requerida de memoria



Segmentos

donde:

* = bits de protección
l = longitud
b = base

A manera de resumen tenemos la siguiente tabla comparativa:

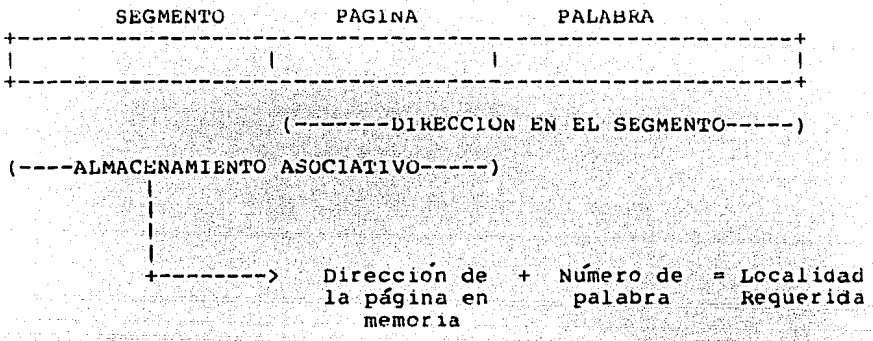
	PAGINACION	SEGMENTACION
Objetivo	Dividir la memoria FISICA	Hacer la división LOGICA del Espacio de Direccionamiento
Característica	Páginas de tamaño FIJO	Segmentos de tamaño VARIABLE
Nivel	Hardware Se checa overflow	Software No hay overflow,se detecta Violación de Memoria

5.2.4 SEGMENTACION CON PAGINACION

Existe además la técnica que combina ambas técnicas, donde cada segmento consiste generalmente de varias páginas y cada uno maneja sus propias tablas.

La forma de hacer el Mapeo de Direcciones es como sigue:

a) Se toma la dirección (s,a)
b) Se usa s como índice en la tabla de segmentos
c) Si su correspondiente entrada en la tabla se encuentra vacía, traer la tabla de páginas,si no, extraer la dirección de la tabla de páginas
d) Separar esa dirección en: Dirección de palabra = d y en Número de Página = p
e) Usar p como índice en la tabla de páginas
f) Sí su entrada correspondiente esta vacía, traer la página p a memoria principal,si no, tomar la dirección p'
g) Sumar p' a d para obtener la localidad requerida



Como ejemplo de máquinas que utilizan esta técnica tenemos a la Burroughs 6500 y la PDP 11/45.

5.3 POLITICAS DE ASIGNACION DE MEMORIA

Todas las técnicas expuestas anteriormente, contemplan solo el mecanismo para el manejo de memoria pero no especifican bajo que políticas van a realizarse las transferencias.Estas políticas pueden ser clasificadas como:

i) Políticas de REEMPLAZO
ii) Políticas de ALIMENTACION (Fetch)
iii) Políticas de ACOMODO

5.3.1 POLITICAS DE REEMPLAZO.-

Son las que determinan CUALES bloques de información deben ser transferidos a memoria secundaria cuando se necesita espacio para nuevos bloques en memoria principal.No siempre resulta fácil decidir ya que siempre se tiene el riesgo de transferir justamente el bloque que va a necesitarse enseguida, pero los tres algoritmos más utilizados son:

a) LEAST REACENTLY USED (LRU)
Este algoritmo reemplaza al bloque Menos Recientemente usado basandose en la suposición de que "si no se ha usado mucho ultímamente,probablemente siga sin usarse y por tanto se puede transferir a memoria secundaria".

b) LEAST FRECUENTLY USED (LFU)
Aquí se reemplaza el bloque Menos Frecuentemente usado, en base a un razonamiento análogo al caso anterior,pero hay que notar que aquí se debe considerar un costo adicional ya que hay que contabilizar los accesos hechos a cada bloque.

c) BLOQUE MAS ANTIGUO (FIFO First In - First Out)
Este algoritmo es el más simple y reemplaza al bloque que haya permanecido por más tiempo en memoria y resulta ser mucho menos eficiente que cualquiera de los otros dos.

5.3.2 POLITICAS DE ACOMODO.-
Determinan DONDE será colocada la información una vez traída a memoria principal.En un sistema con pagineo,la política de acomodo es realmente sencilla ya que basta con encontrar una página desocupada para hacer la transferencia y no hay problema en cuanto al taman~o ya que este es fijo;si el sistema no tiene pagineo es necesario tener un manejador más inteligente porque entonces no basta con encontrar un espacio libre,sino que además se debe checar que sea suficiente para el bloque que se desea transferir.

Es claro entonces que hay que mantener información de alguna manera acerca del taman~o y dirección de las localidades desocupadas. A esta lista se le llama "Hole - List" o "Lista de Huecos" y su objetivo es ayudar a decidir que región utilizar y actualizar esta información a medida que estas se ocupen.Lo más importante en estos casos es evitar la fragmentación de la memoria en pequen~as regiones que después no puedan ser utilizadas a menos que se hagan compactaciones de estos espacios para conformar regiones más amplias.

Existen numerosos algoritmos para asignación de regiones pero los ejemplos clásicos de los algoritmos más usados son:

i) Best Fit (El más aproximado en taman~o)
---- ---

Las regiones se ligan en orden creciente con respecto a su taman~o de tal forma que para colocar el segmento s se busca la región cuyo taman~ sea lo más aproximado al del bloque.Aquí se presenta el problema de la fragmentación ya que al escoger regiones que excedan en poco al taman~o del segmento,el espacio sobrante es poco probable que pueda ser utilizado posteriormente.

ii) Worst Fit (El menos aproximado en taman~o)
----- ---

Aquí las regiones se ligan en orden decreciente con res--pecto a su taman~o y el segmento s es colocado en la primera región ya que es la de taman~o máximo.El espacio que sobra, es ligado en su correspondiente lugar en la lista. Aunque parezca paradójico, este algoritmo lo único que pretende es asegurar que no habrá espacios pequen~os que no puedan ser utilizados después.

iii) First Fit (El primero donde quepa)
----- ---

Aquí la asignacion se hace en la primera región que se encuentre tal que sea lo suficientemente grande para albergar al segmento.Después de un tiempo, este algoritmo tiende a fragmentar demasiado la memoria nacia la cabeza de la lista y para evitarlo puede iniciarse la busqueda ciclí-camente un elemento mas abajo que el considerado la vez anterior.

iv) Buddy (Mellizos, Cuates)

Para este algoritmo, el taman~o de los segmentos debe ser potencia de dos

i.e. $s = 2^{**} i$ p.a. $i < = K$

donde K es un cierto taman~o máximo.

La lista se lleva en forma separada para los bloques de taman~o 2**1, otra para los de 2**2 etc.De esta manera cuando se ocupa una región de la lista de 2**i+1, el espacio sobrante se anexa a la lista de taman~o 2**i. (de ahí el nombre de "mellizos").Este algoritmo funciona recursivamente y una posible versión sería:

```
mellizos(i)
{
        if ( i== K+1)     error(); /*Si se excede al
                                   taman~o
                                   maximo,error*/
        else
        if(colavacia = sigelem(i)) /* Si no hay una
                                   region de
                                   taman~o i..*/
        {
                mellizos(i+1);     /* Llamada
                                   recursiva */
                divideregion();    /* Forma los
                                   mellizos */
                ordenalista();     /* Se acomoda en la
                                   lista corres-
                                   -pondiente */
        }
        tomaelem(i);               /* Toma el primer
                                   hueco de
                                   taman~o i */
}
```

5.3.3 POLITICAS DE ALIMENTACION.-

Estas son las que determinan CUANDO hacer el reemplazo de un bloque de memoria secundaria a memoria principal. Dentro de estas políticas se tienen dos clasificaciones:

a) De DEMANDA .- El cambio se hace cuando se necesita el bloque.

b) ANTICIPADA .- El cambio se hace ANTES de que se necesite el bloque.

En sistemas SIN pagineo (como Burrougns) se transtieren los bloques utilizando políticas de DEMANDA.Las políticas anticipadas,,se manejan en base a información acerca del comportamiento anterior de los programas.Esto se refleja en el postulado de DENNING que dice:

"Las referencias de los programas tienden a agruparse en pequeñas áreas dentro del espacio de direcciones y dichas localidades tienden a cambiar de manera intermitente".

****

6. Entrada Salida

Los mecanismos de entrada/salida resultan ser una de las partes mas importantes del Sistema Operativo ya que es por medio de ellos que los procesos pueden comunicarse con el medio ambiente externo; al mismo tiempo,resultan ser difícil y conflictivos debido a la gran variedad de dispositivos perifericos que se emplean. Estos dispositivos difieren generalmente en:

a) Velocidad.

b) Unidades de Transferencia.

c) Representación de Datos.

d) Operaciones Permisibles.

e) Condiciones de Error.

La velocidad puede ir desde 10 caracteres por segundo (caso del teletipo) hasta el caso del disco fijo que maneja 10**6 caracteres por segundo .

Las unidades en que se transfieren los datos pueden ser caracteres ,palabras enteras,bloques o registros;de la misma manera,estos datos pueden estar codificados de distintas formas.

Los perifericos tienen diversas opciones para su manejo y las operaciones permisibles dependen de la función particular . En caso de detectarse algun error,la manera en como lo maneja cada dispositivo,es muy diferente.

Es debido a todos estos factores que durante la etapa de diseño de los macanismos de entrada/salida deben considerarse los siguientes aspectos:

1) INDEPENDENCIA DEL CODIGO DE CARACTERES.

La forma en que se nace y maneja la codificación de los caracteres,debe ser transparente al usuario, y deben ser manejados de manera standard . Considerando estos aspectos se logran avances importantes, como el manejo de caracteres por medio de un CODIGO INTERNO DE CARACTERES que nace posible la uniformidad de las representaciones.

2) INDEPENDENCIA DE DISPOSITIVOS.

Es importante que los programas sean independientes del dispositivo en particular e inclusive del TIPO de dispositivo al que se hace referencia.

3) EFICIENCIA.

Como ya se mencionó, la E/S es una parte conflictiva presenta problemas que deben ser resueltos de la manera mas eficiente posible.

6.1 DISPOSITIVOS VIRTUALES (STREAMS)

Un concepto vital dentro de los Sistemas Operativos que se desprende de aquí, es el de DISPOSITIVOS VIRTUALES o STREAMS.Los dispositivos virtuales se usan en el programa sin hacer ninguna referencia a los dispositivos fisicos.La equivalencia entre los STREAMS y los diferentes dispositivos para un proceso en particular puede hacerse por medio de una lista apuntada desde su descriptor de proceso.

Otro punto que hay que destacar es que el sistema de Entrada/Salida deben ser de tal forma que las características del dispositivo esten claramente asociadas con los dispositivos mismos en vez de asociarse con las rutinas que los manejan y que se conocen como MANEJADORES DE DISPOSITIVOS (Device Handlers) .

Las características de los dispositivos pueden almacenarse dentro de un DESCRIPTOR DE DISPOSITIVOS, asociado a cada uno de ellos y usar este descriptor como fuente de información para el manejador de dispositivos.Dichas características pueden ser:

i) Identificación del Dispositivo.

ii) Instrucciones Permitidas.

iii) Apuntadores a las Tablas de Traducción.

iv) Status Actual del Dispositivo (Ocupado o no)

v) Proceso Actual del Usuario.

Todos los descriptores de dispositivos pueden ligarse en una ESTRUCTURA DE DISPOSITIVOS que es apuntada desde la tabla central.

6.2 PROCEDIMIENTO DE ENTRADA SALIDA.

Este procedimiento consta de código REENTRANTE de tal forma que puede ser usado por varios procesos al mismo tiempo.Es necesario pasarle varios parámetros:

i) Dispositivo Virtual.

ii) Modo de Operación.

iii) Cantidad de datos a ser transferidos.

iv) Destino.

v) Semáforo.

que son checados y si resultan válidos, se procede a iniciar el servicio requerido.

El proceso de E/S es el encargado de ensamblar los parámetros de la petición en el IORB (Input Output Request Block) BLOQUE DE PETICIONES DE ENTRADA/SALIDA que lo agrega a una cola de bloques similares para el mismo dispositivo.

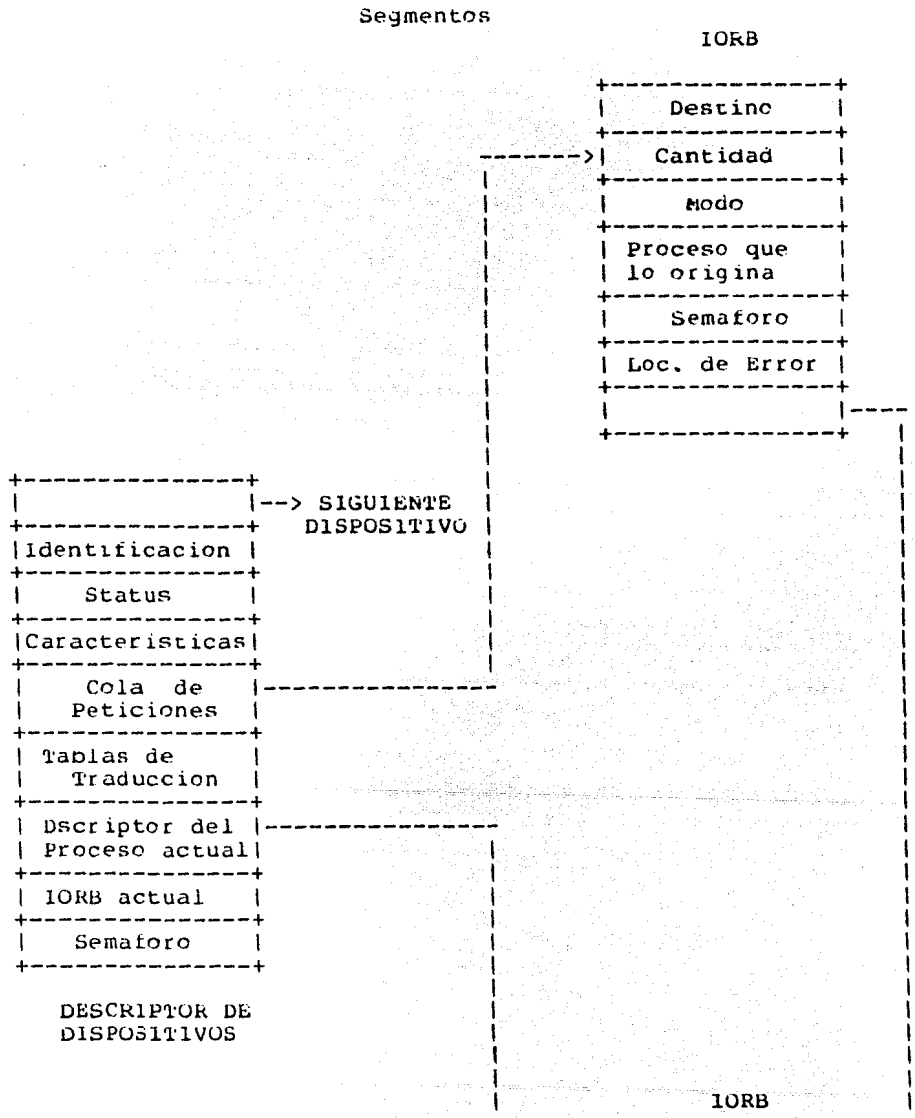
La COLA DE PETICIONES DE DISPOSITIVOS se liga al descriptor de dispositivos y es atendida por un proceso llamado manejador de dispositivos.

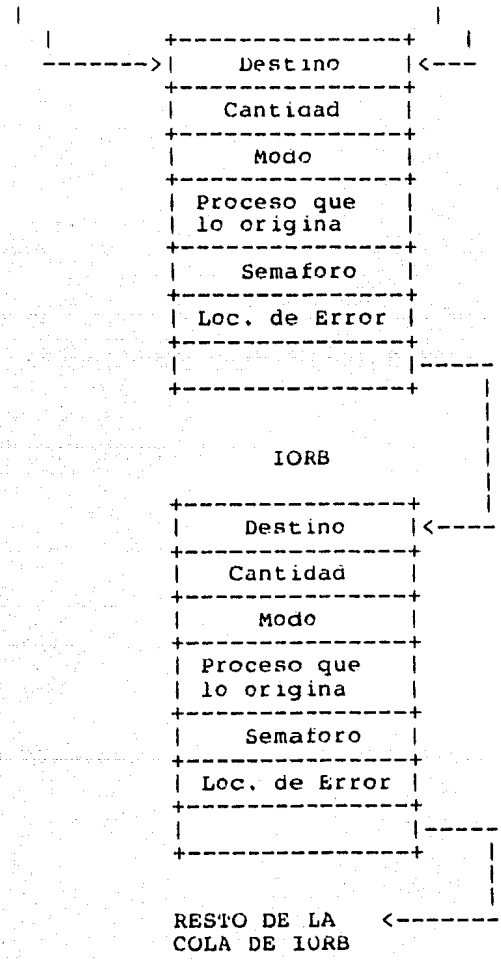
La rutina quedaría:

```
doio ( dispositivo,,modo,cantidad,destino,semaforo)
int dispositivo,modo,cantidad,destino,semaforo;
{
   checadisp(dispositivo);  /* busca el dispositivo en el
                               descriptor de procesos */
   checapar(modo,cantidad,destino,semaforo);
                        /* Checa los parámetros */
   if (error)  errores(bandera);
   ensambla();          /* Ensambla IORB */

   agrega();            /* Agrega el IORB a la cola del*/
                        /* dispositivo */

   signal(semaforo);    /* Avisa que fue atendido */
}
```


Segmentos
IORB
Destino
Cantidad
Modo
Proceso que lo origina
Semaforo
Loc. de Error
--> SIGUIENTE DISPOSITIVO
Identificacion
Status
Caracteristicas
Cola de Peticiones
Tablas de Traduccion
Descriptor del Proceso actual
IORB actual
Semaforo
DESCRIPTOR DE DISPOSITIVOS
IORB


Destino
Cantidad
Modo
Proceso que lo origina
Semaforo
Loc. de Error
IORB
Destino
Cantidad
Modo
Proceso que lo origina
Semaforo
Loc. de Error
RESTO DE LA COLA DE IORB

FIGURA

6.3 MANEJADORES DE DISPOSITIVOS.

El MANEJADOR DE DISPOSITIVOS (Device Handler) es el proceso responsable de atender las peticiones (que están en la cola de peticiones) hechas por un dispositivo e informarle cuando este haya finalizado.

El manejador funciona dentro de un ciclo continuo en el cual se extrae un IORB de la cola de peticiones, inicia las correspondientes operaciones de Entrada/Salida, espera a que se hayan ejecutado y lo notifica al proceso que lo solicitaba.

El ciclo seria algo como:

```
while(1)
{
   wait(petición);        /* Va a atender la petición */
   extrae(IORB);          /* Toma un IORB de la cola */
   ejecuta();             /* Ejecuta las operaciones de E/S */
   wait(terminado);       /* Espera hasta terminar de ejecutar */
                          /* las operaciones */
   if (error) then errores( bandera );
   else
   {
        traduce();        /* Hace las traducciones necesarias */
        manda();          /* Manda la información a su destino */
        signal(petición);       /* Avisa que termino de */
                                /* atender la petición */
   }
}
```

Observaciones:

1) El semáforo de peticiones es accesado por el procedimiento de E/S cada vez que anexa un IORB a la cola.

2) Si la cola esta vacia y peticiones=0, se suspende el manejador de dispositivos.

3) La forma en que se decide cual IORB atender,puede hacerse en base a diferentes algoritmos.

6.4 BUFFERING

Hasta este punto,cada requerimento de entrada salida de un proceso, ocasiona que se efectue una transferencia física al dispositivo apropiado.Esta situación crea conflictos y retrasos innecesarios,ya que cada vez que se hace una transferencia, el proceso suspende su ejecución hasta ser atendido. Para resolver éste problema,se maneja la técnica de BUFFERING que consiste en guardar los datos en un área de almacenamiento (buffer) antes de que la peticion sea hecha y asi agilizar la transferencia.

Las transferencias de Entrada son hechas por el sistema operativo dentro de un buffer toma de ahí los datos y solo debe interrumpir cuando el buffer se encuentra vacío. Análogamente, las transferencias de salida se hacen sobre el buffer, que es vaciado por el sistema operativo hasta que este lleno.

Esta técnica puede optimizarse si se utilizan 2 buffers:uno para entrada y otro paa salida; cuando se manejan varios buffers se tiene la tecnica de MULTIPLE BUFFERING y es importante aclarar que si bien agiliza elfuncionamiento del sistema y minimiza las suspensiones del proceso, ésto sucede hasta un cierto punto.Lo que realmemte limita es la capacidad de los dispositivos de E/S para operar.

6.5 DISPOSITIVOS DE ARCHIVOS

Cuando nos referimos a los parámetros necesarios dentro del proceso de E/S , mencionamos que debía especificarse el nombre del dispositivo periférico a utilizarse, pero esto puede resultar insuficiente en algunos casos. Esto quiere decir que no basta con identificar el dispositivo, sino que además es necesario aclarar donde esta la siguiente información que va a ser accesada.

Recordemos que hay dispositivos de acceso secuencial los cuales no presentan problema como la cinta magnética perc tambien hay casos como el disco que al permitir el acceso aleatorio, forzan a especificar que AREA DE DATOS dentro de ese dispositivo es la que se va a usar.

A cada area de datos se le denomina ARCHIVO (File) y en general posee un tamaño arbitrario.A un dispositivo capaz de manejar archivos se le llama DISPOSITIVO DE ARCHIVOS (File Device) y será tratado mas ampliamente después.

6.6 SPOOLING

Al igual que se presentan problemas para manejar cada unidad de datos en la E/S, también se presentan problemas cuando un dispositivo no-compartible se asigna a un proceso que lo requiere (el caso típico es la impresora).Este dispositivo es liberado hasta que el dispositivo virtual es cerrado o el proceso termina.Se utiliza el termino SPOOL para indicar que en lugar de realizarse las transferencias directas a los dispositivos asociados a un dispositivo virtual, el procedimiento de E/S se encarga de ejecutar la transferencia a un área intermedia (generalmente a disco).

La responsabilidad de mover los datos del disco al periférico, es de un proceso llamado SPOOLER que esta asociado a cada dispositivo.

Cada proceso que abre un dispositivo lógico para la impresora, en realidad lo manda a una impresora virtual. Cuando el dispositivo virtual se cierra, el archivo se manda a una cola de archivos similares, creados por otros procesos, para el mismo dispositivo.Naturalmente el tamaño de la cola dependerá de la velocidad de operación del dispositivo particular; el tiempo que los archivos permanezcan en ella es función del algoritmo que siga el SPOOLER.

```
spooler()
{
   while(1)
   {
        wait(semáforo); /* Espera a que haya algo para mandar */
        extrae(file);   /* Toma un archivo de a cola */
        abrearch();     /* Abre el archiivo */
        while( ! EOF)   /* Mientras no sea fin de archivo... */
        {
           doio(parametros de disco);   /* Llama al */
                                        /* procedimiento con */
                                        /* los parámetros */
                                        /* necesarios para el */
                                        /* manejo del disco */
           wait(semdisk);         /* Espera a ser atendido */
           doio(parametros de impresora);    /* Igual pero */
                                             /* ahora para */
                                             /* la */
                                             /* impresora */
            wait(semimpr);        /* Espera a terminar la */
                                  /* impresión  */
        }
   }

}
```

Algunas ventajas del spooling son:

1) Baja la presión sobre el manejo de periféricos usados continuamente .

2) Reduce la posibilidad del deadlock.

3) Es muy fácil producir copias de un mismo arcnivo si tener que correr nuevamente el proceso.

OBSERVACION: El spooler no es factible en un medio ambiente de tiempo real, porque ahi se requiere nacer la transferencia de manera inmediata.

****

7. SISTEMA DE ARCHIVOS

Cuando comenzo el uso generalizado de las computadoras, se pensaba en ellas como una nerramienta que ayudaba a realizar cálculos matematicos.Actualmente este concepto na quedado olvidado y ahora las funciones de la computadora van mucno mas allá.Se considera que es un dispositivo capaz de almacenar y manipular grandes cantidades de información de todo tipo, y por largos periodos de tiempo. Una de las funciones de los Sistemas Operativoos es la de facilitar el manejo y almacenamiento de esa información y para esto se necesita el SISTEMA DE ARCHIVOS, cuyos principales objetivos son:

1) Permitir un almacenamiento prolongado de los archivos.

2) Permitir compartir tanto datos como código.

3) Manejo de archivos de diversos tamaños.

4) Permitir al usuario la ejecución de funciones comunes sobre archivos como:

   i) Creación, Cambio de nombre, Borrar, Copiar etc.

   ii) Cnecar el modo de acceso a los archivos es decir , proporcionar protección entre usuarios.

   iii) Permitir el manejo de archivos atraves de nombres simbólicos mas fáciles de manejar por el usuario.

5) Permitir una asignación efectiva del almacenamiento secundario.

6) Flexibilidad y Versatilidad de Acceso.

7) Proporcionar seguridad sobre la información almacenada en los archivos.

El hecho de compartir la información, es realmente una de los aspectos mas importantes ya que evita el tener información redundante (copias innecesarias de archivos) para cada usuario.Ademas, aun en el caso de sistemas tipo JOB-SHOP es necesario el contar con una BIBLIOTECA de programas comunes a todos los usuarios, asi como un sistema que permita almacenar programas sin tener que cargarlos cada vez.

Los datos se organizan en unidades lógicas llamadas ARCHIVOS . Los elementos de un Archivo, generalmente llamados REGISTROS pueden ser accesados de diversas formas, dependiendo de como este estructurado el archivo.Sinembargo, de manera mas general, un archivo puede verse como una cuerda de bytes que pueden agruparse como registros.

Con respecto al taman~o variable de los acnivos, en general se encuentran divididos en BLOQUES de taman~o fijo y estos bloques se transfieren de memoria secundaria a memoria principal y visceversa, por medio del sistema de archivos.

Los taman~os mas comunes para los bloques son de 128 bytes hasta 1k o 2k palabras por bloque.El determinar un taman~o apropiado es muy importante, ya que depende de este hecho cuanto espacio puede desperdiciarse en terminos generales por un archivo.

Por otro lado hay que considerar el taman~o para definir cuantas entradas en el directorio deben tenerse, para que se accese el espacio disponibe para cada usuario.Los criterios considerados para determinar el taman~o pueden ser:

i) Espacio Promedio ocupado por un archivo.Mientras más pequen~o sea el bloque, más bloques se ocuparan, pero si es muy grande, es probable que deserdicie mucho espacio.

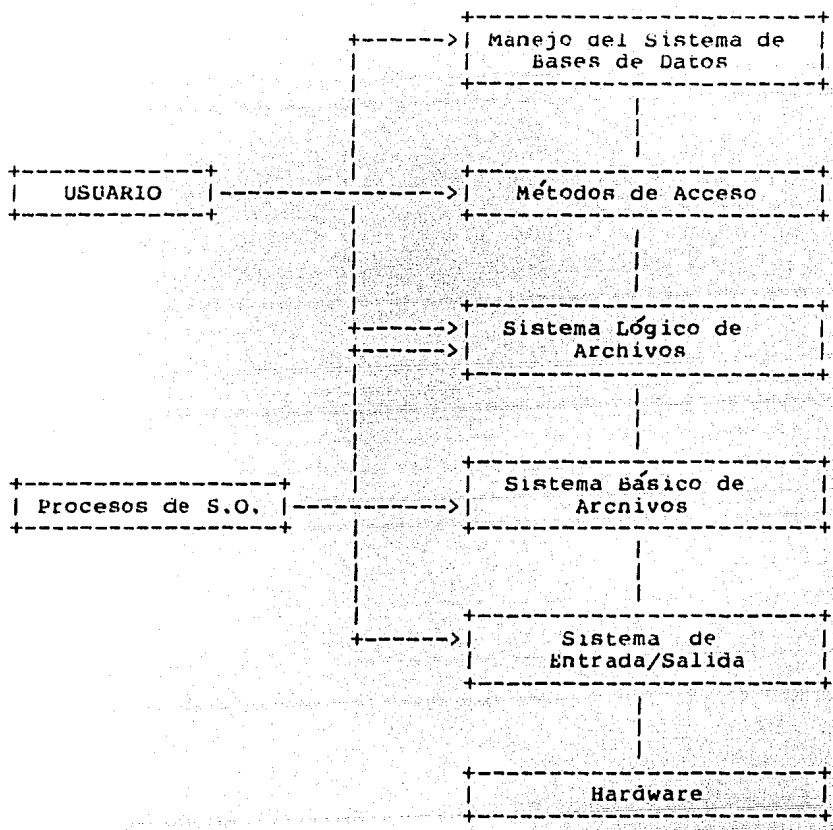
ii) Las unidades que manejan los dispositivos de almacenamiento,para hacer la transferencia de datos a memoria principal.

iii) La cantidad de memoria que necesitan las operaciones de "read" y "write".

7.1 ETAPAS DE DESARROLLO

Las funciones del sistema de archivos pueden ser descritas a diferentes niveles, que van desde funciones básicas de hardware hasta complejas rutinas de software.

Básicamente podemos considerar 5 niveles, donde los primeros niveles se relacionan con el ASPECTO FISICO del sistema y los últimos, con el ASPECTO LOGICO, orientado al punto de vista del usuario.



Explicaremos brevemente cada uno de los niveles :

7.1.1 HARDWARE

Consta de comandos de bajo nivel, orientados hacia dispositivos , como por ejemplo el oprimir un botón o emitir el comando para regresar la cinta etc.

7.1.2 SISTEMA DE ENTRADA/SALIDA

Las operaciones de los dispositivos fisicos son coordinadas a este nivel.Se manejan los dispositivos y se realizan las operaciones de Lectura y Escritura de bloques de información en las direcciones especificadas, ya sea en memoria principal o secundaria.El sistema de entrada salida permite al resto del sistema mandar y recibir datos de los dispositivos fisicos.Las direcciones en almacenamiento principal y de los dispositivos fisicos deben especificarse antes de que los datos puedan ser transferidos.

Para especificar tales direcciones, el sistema de archivos maneja DECRIPTORES DE ARCHIVOS (File Descriptor Block FDB) que es una pequeña estructura de datos que identifica al archivo y los bloques que ocupa físicamente, así como información de control que indica de que manera deben manejarse los siguientes descriptores (si es que los hay).

7.1.3 SISTEMA BASICO DE ARCHIVOS

A este nivel, el sistema convierte el IDENTIFICADOR UNICO de archivos en el DESCRIPTOR DE ARCHIVO, que proporciona la dirección fisica de éste, así como su longitud.Si el archivo contiene varios bloques, su descriptor apunta a ellos.

Con un sistema básico de archivos el sistema operativo puede manipular los descriptores de archivos de acuerdo acomandos de lenguaje de alto nivel.Algunas veces los usuarios de éste tipo de lenguajes necesitan poder accesar directamente los FDB y el dar ésta facilidad depende de como se diseñó el sistema i.e. si se usaron mnemónicos para describir los archivos (para lo que se requiere el sistema lógico de archivos) o si se usan descriptores físicos para los archivos.

7.1.4 SISTEMA LOGICO DE ARCHIVOS

Su función es DETERMINAR EL IDENTIFICADOR UNICO asociado con el nombre simbólico dado por el usuario.La función del sistema lógico de archivos no se relaciona con las propiedades físicas de los dispositivos en los cuales se almacenan los archivos, sino que trata de que sus funciones sean independientes de ellos.El sistema lógico proporciona comandos que permiten al usuario crear, leer y/o escribir en un archivo.Además es el encargado de "ABRIR" los archivos sobre el que desean ejecutarse los comandos, lo que implica localizar el archivo en el directorio, y posicionarse en el registro adecuado para ejecutar las operaciones deseadas y al finalizarlas, CERRARLO nuevamente con lo que se actualiza la información almacenada en los FDB del archivo.

7.1.5 METODOS DE ACCESO

El sistema proporciona métodos lógicos de acceso de datos que pueden diferir completamente del orden en que fueron almacenados físicamente.Los métodos de acceso seran descritos posteriormente con mayor amplitud.

7.1.6 SISTEMA DE MANEJO DE BASES DE DATOS

Aunque estrictamente nablando,las bases de datos no forman parte del sistema de archivos, pueden considerarse como una extensión lógica a ellos.Programas diferentes pueden requerir de una estructura lógica diferente a la de los archivos sobre los que actuan.Por otra parte resulta incosteable desde el punto de vista económico el mantener copias de archivos separadas para cada estructurra, si la información que contienen es la misma.Las Bases de Datos permiten compartir la misma información entre varios programas que lo requieran.

7.2 MANEJO DE DIRECTORIOS

El problema más importante a resolver por el Sistema de Archivos es el de nacer la correspondencia entre el nombre simbólico de un archivo y su localizacion física en memoria secundaria ; los usuarios prefieren llamar a sus archivos por medio de sus propios mnemónicos y dentro de este medio ambiente, un sólo nombre puede referirse a diferentes archivos y diferentes nombes pueden referirse a un solo archivo, por lo que es responsabilidad del sistema resolver todos los conflictos que pudieran surgir a causa de los nombres .

Los archivos son esencialmente segmentos de memoria y un nombre de archivo usado dentro de cierto contexto define una funcion que mapea las direcciones en elementos dentro del archivo ; para determinar ese contexto claramente, el sistema se ayuda de estructuras llamadas DIRECTORIOS.

El manejo del directorio varia de sistema en sistema pero en general debe proporcionar los mecanismos de protección entre los diferentes usuarios para evitar accesos no permitidos. Una medida al respecto es particionar el directorio en 2 areas:

1) MASTER FILE DIRECTORY (MFD) (Directorio Maestro).

2) USER FILE DIRECTORY (USD) (Directorio del Usuario).

Con este método,podemos controlar desde el MFD los archivos de cada usuario, manejando un apuntador a cada UFD particular. En realidad con ésto se evitan confusiones entre los nombres simbólicos de los archivos que, aunque cada sistema maneja sus propios campos, los mas comunes son:

DISPOSITIVO:<DIRECTORIO>NOMBRE.TIPO O EXTENSION;VERSION

donde:

DISPOSITIVO	Indica dónde esta almacenado el archivo.
DIRECTORIO	Identifica sobre qué área del sistema de archivos se está trabajando.
NOMBRE	Identifica al archivo dentro del directorio.
TIPO O EXTENSION	Indica que tipo de información guarda el archivo.
VERSION	Para distinguir diferentes versiones de un mismo archivo.

Ademas de ésta información,se guarda otra de tipo estadístico y de protección como:Fecha de Creación, Fecha del último acceso,Autor, Tipo de acceso permitido etc.

7.3 MECANISMOS DE PROTECCION

Ampliaré sobre la protección y acceso de archivos.Lo primero que es importante recalcar es el hecho de que cada archivo cuenta con una PALABRA DE PROTECCION, por medio de la cual se especifica el tipo de acceso permitido a los usuarios.En general, los modos de acceso son:

i) Leer
ii)Escribir
iii) Borrar
iv) Ejecutar
v) Agregar
vi) Actualizar

La manera de especificarlo es disponiendo de ciertos bits que indiquen dependiendo de si estan prendidos o apagados si se tiene o no cada uno de los privilegios.Además resulta útil clasficar los usuarios en 3 catergorias:

i) Autor o Duen~o
ii) Compan~eros de Grupo o Equipo
iii) Otros

de tal forma que dependiendo del grupo al que pertenece cada usuario se checa el acceso.Naturalmente el duen~o tiene todos los privilegios incluso, el poder cambiar la palabra de protección, en tanto que los compan~eros de grupo pueden leer y ejecutar pero no escribir y el resto solo ejecutar.

Una técnica más general y que no esta basada en las clases de usuarios es la que permite al usuario definir en su UFD una lista de usuarios con los privilegios que les va a conceder .Esta lista puede ser única para todos los archivos o cada uno manejar una lista separada.

Claro que esto puede llevar a ocupar demasiado espacio por lo que a veces, como en el caso de MULTICS y UNIX, se permiten LIGAS en los directorios.Si se usa éste método, cuando se borra un archivo, deben buscarse TODAS las ligas que llegan a él y borrarlas (caso de Multics) o bien, el archivo realmente no desaparece en tanto haya una liga que lo apunte. (caso de UNIX).

7.4 FORMAS DE ORGANIZACION DE DIRECTORIOS.

Existen varias formas de organizar los directorios y almacenar la informacion...

i) Bloques Ligados.

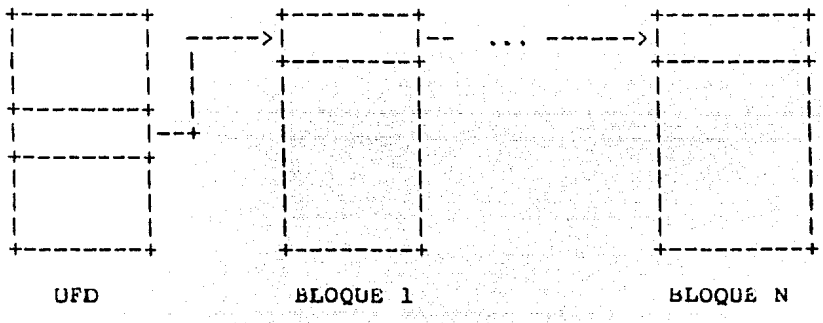
ii) Mapa de Archivos.

iii) Bloques Indexados.

iv) Bit Map (Mapa de Bits).

BLOQUES LIGADOS

En esta estructura se ocupa una palabra dentro de cada bloque para apuntar al siguiente bloque.El UFD tiene el apuntador al primer bloque.Su principal desventaja es el elevado numero de accesos a disco que deben nacerse para encontrar el fin de archivo o para borrar el archivo ademas de que se forza a que el tipo de acceso sea SECUENCIAL.

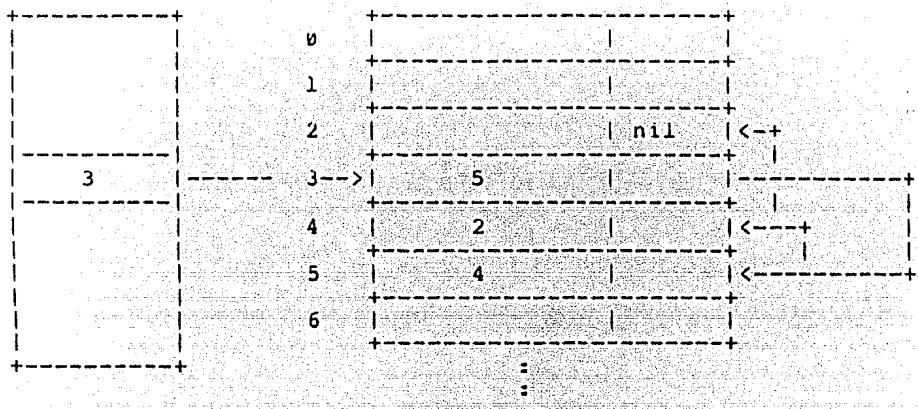


ESQUEMA DE BLOQUES LIGADOS

MAPA DE ARCHIVOS

Cada bloque del disco esta representado en una palabra de una tabla y el UFD apunta a la primera LOCALIDAD (donde se representa el primer bloque);esta a su vez apunta a la localidad que representa al segundo bloque etc.

Nuevamente el acceso es SECUENCIAL y como en general la tabla es grande, debe almacenarse en disco y con esto se hace necesario nacer N accesos extras a disco, para leer un archivo de N bloques.



ESQUEMA DE MAPA DE ARCHIVOS

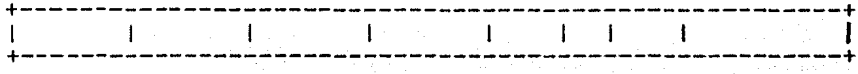
BLOQUES INDEXADOS

Los apuntadores para cada archivo se almacenan en bloques indexados en disco.Estos bloques son tablas cuyas entradas contienen apuntadores a los bloques del archivo.Para archivos grandes se necesitaran varias tablas, pero para archivos pequeños si se mantiene fija la longitud de la tabla de apuntadores, el numero de entradas que se desperdician puede ser significativo.El AUMENTAR o SUPRIMIR bloques implica reordenar los apuntadores, por lo que no se recomienda esta estructura para archivos que cambian frecuentemente . La ventaja es que el acceso NO ES NECESARIAMENTE SECUENCIAL.

Una alternativa es manejar las tablas de apuntadores tambien como una lista ligada para que no se desperdicie ninguna entrada.

MAPA DE BITS
---- -- ----

Aquí los bloques se representan en un área de memoria donde cada bit representa un bloque y si está prendido,ese bloque esta ocupado y si está apagado, está desocupado.



B0 B1 B2 B3 ..... Bn

Bi = 0 está desocupado

Bi = 1 está ocupado

7.5 METODOS DE ACCESO

Antes mencionamos que un archivo podía verse como un conjunto de bloques, sin considerar la estructura interna ni las características de la información que contiene. Cada usuario requiere estructurar su información de acuerdo a sus necesidades y a menudo necesita accesar un dato específico, sin importar el orden en que fué almacenado.

La posición de los registros requeridos puede darse directamente como una dirección o bien, indirectamente accesando ciertos campos de los registros.El sistema de archivos debe generar entonces los comandos de Entrada/Salida apropiados para que la transferencia física de los registros de almacenamiento secundario a memoria principal se lleve a cabo.

Hay muchas maneras de identificar un registro particular dentro de un archivo; La manera más simple es usar la posición del registro dentro del archivo.Si el archivo lógico tiene un orden implícito de los registros, entonces tiene sentido el direccionar el i-ésimo registro del archivo.(sin olvidar que el orden lógico de los registros es independiente de su localidad física en el dispositivo.

De la misma manera, si se tiene un apuntador al registro actual, tiene sentido preguntar por el "siguiente" registro o el "previo".

Una manera más flexible de accesar los registros es por medio de una LLAVE (Key) que es un campo dentro del registro.Naturalmente también pueden darse "Llaves Parciales" para accesar los registros.Estas llaves parciales se refieren a campos que pueden tener la misma información en varios registros, como sería por ejemplo el caso de los salarios en una empresa, donde varios empleados pueden tener la misma cantidad.

La manera de organizar el conjunto de datos, puede ser variado:

i) Una opción es organizar una busqueda secuencial de las llaves en todos los registros.Esto es altamente ineficiente porque puede implicar demasiados accesos a disco pero por otro lado es la solución más fácil de llevar a cabo.Se recomienda para archivos pequeños.

ii) La siguiente opción es manejar una TABLA DE INDICES que contenga apuntadores a todos los registros con una misma llave.

iii) Finalmente, podemos mantener una LISTA LIGADA con los registros que contienen una cierta llave.

Los métodos de acceso más comunes proporcionados por los Sistemas Operativos son: BASICO o EN COLA (queued) y puede ser SECUENCIAL o DIRECTO.BASICO se refiere a que el usuario debe proporcionar sus propios "buffers" para manejar los regitros y en COLA el sistema tiene un esquema standard de "buffering". El acceso SECUENCIAL se refiere a accesar los registros respetando el orden en que fueron almacenados, ,mientras que el DIRECTO permite accesarlos de manera aleatoria. Otras formas de acceso son MODO CONGELADO y MODO DERRETIDO. Esta clasificación es en otro sentido, ya que la se describe primero es con respecto al orden o secuencia de acceso y ésta última es más bien respecto a un esquema de protección/ consistencia y no es que sean menos comunes sino que constituyen una protección extra. El default en casi todos los sistemas es MODO CONGELADO.

7.6 RESPALDO Y RECUPERACION

El tiempo que se pierde en la búsqueda (SEEK) del siguiente bloque si estos se encuentran muy dispersos por el disco, puede minimizarse si se compactan periódicamente.

Por otro lado es vital que el Sistema de Archivos proporcione mecanismos eficientes de RESPALDO Y RECUPERACION (Back - Up y Recovery). Como ejemplo de estos mecanismos mencionaremos los siguientes:

i) Respaldo Masivo.

ii) Respaldo Incremental.

La diferencia estriba en que en el RESPALDO MASIVO, el contenido de TODOS LOS ARCHIVOS se guarda (generalmente en cinta) y en caso de necesitarse, se cuenta con las versiones que se tenian en el momento de nacer el Respaldo.En el RESPALDO INCREMENTAL, sólo se guarda la información que HA SUFRIDO MODIFICACIONES o HA SIDO CREADA desde la última vez que se hizo el Respaldo anterior.Es claro que debe llevarse una estadística por medio de banderas o palabras de control en cada entrada del directorio , que se enciende cuando el archivo se crea o modifica y se apaga automáticamente al momento de nacer el Respaldo.Incluso para evitar revisar todas las entradas, se puede tener una bandera o palabra de control a nivel de directorio.

Las desventajas en cada caso son:

Para el RESPALDO MASIVO el sistema debe ser suspendido (generalmente, aunque por ejemplo en el caso de TENEX no es asi...) o bien no se nace el respaldo de los archivos abiertos ademas de que como guarda TODA la información, no puede nacerse frecuentemente.El INCREMENTAL no necesita suspender el sistema pero el proceso de recuperacion resulta muy complejo.Si se da una falla, las cintas se ponen en orden cronologico inverso a como se grabaron y solo se van cargando los archivos que no tenian versiones mas recientes en las cintas.En ambos casos, es una buena oportunidad para COMPACTAR ARCHIVOS en bloques contiguos y facilitar posteriores accesos.

****

ASIGNACION DE RECURSOS Y SCHEDULING.

Para que un proceso pueda ser ejecutado se requiere que se le hayan asignado todos los recursos que necesita, sin embargo como un sistema de cómputo tiene un conjunto limitado de recursos es necesario diseñar políticas para compartir de una manera óptima los recursos del sistema entre todos los procesos. El objetivo de este trabajo es analizar las diferentes políticas de asignación de recursos, incluyendo aqui el despachador y las políticas de despacho.

En un ambiente en el cual los recursos son ilímitados, el método "pídalo cuando lo necesite" puede ser totalmente aceptable, sin embargo cuando hay demandas concurrentes de todos los procesos en el sistema, ésta política no podría satisfacerlas, por lo que las tecnicas deben considerar el compartir un numero limitado de recursos entre un gran número de procesos en competencia.

Los objetivos de estas técnicas son:

1) Exclusión mutua de procesos para recursos no compartibles.

2) Prevenir el caso de Cerradura de la Muerte.

3) Asegurar un alto nivel de utilización de los recursos.

4) Permitir a todos los procesos una oportunidad de adquirir los recursos que necesite en un tiempo "razonable".

La manera de evaluar las políticas de asignación de recursos y de despacho es observando la relación entre la satisfacción al usuario y un alto nivel de utilización de recursos, naturalmente lo que se persigue es el equilibrio. En sistemas de tiempo real debe garantizarse un tiempo de respuesta que haga que la balanza se ponga del lado del usuario, en un sistema de tipo batch es mas importante lograr un nivel más alto de utilización de los recursos.

La asignación de recursos debe examinarse bajo 2 perspectivas: Mecanismos y Políticas.

Los Mecanismos muestran aspectos de cómo es hecha la asignación, ésto incluye elementos como: estructuras de datos para describir el estado de los recursos, técnicas para asegurar el uso exclusivo de recursos no compartibles y por tanto formar colas de peticiones de recursos que no han sido satisfecnas inmediatamente.

Las Políticas gobiernan la forma en la que son aplicados los mecanismos, y pretenden asegurar la asignación de un recurso siempre que este se encuentre disponible, asegurando así la prevencion de "deadlocks" y manteniendo al sistema en un estado de balance ya que una asignación errónea puede llevar a una situación en la cual los procesos no pueden proceder o en la cual el sistema es sobrecargado con respecto a una clase especial de recursos.

8.1.ALGORITMOS DE ASIGNACION.

Los recursos de sistema pueden ser categorizados de la siguiente manera:

1) Procesador Central

Este es asignado por el despachador al primer proceso que no este corriendo de la cola del procesador, la estructura de datos que describe el procesador central puede ser visto como un descriptor el cual sería similar al descriptor de un dispositivo periférico. El descriptor contiene información como la siguiente:

a) identificador del procesador

b) estado actual (modo usuario o modo kernel).

c) un apuntador al descriptor del procesador del proceso actual.

En una configuración en donde los procesadores son diferentes el descriptor del procesador puede contener una indicación de las características individuales del procesador (por ejemplo si existe procesador de punto flotante), incluso en este caso cada procesador puede tener su propia cola apuntada por su descriptor.

2) Memoria

La memoria es asignada por un manejador de memoria (con o sin paginación), la estructura de datos que describe el estatus de la memoria son tablas de páginas, tablas de segmentos o listas de bloques de memoria disponible dependiendo de la arquitectura de la máquina.

Cuando un proceso espera por la transferencia de una nueva página o de un segmento de memoria secundaria se convierte en un proceso que no puede correr, es puesto en la cola de los procesos que no pueden correr, con un bit indicando que esperan por memoria, los requerimentos de páginas o segmentos son peticiones de entrada/salida y estos son manejados por el sistema de manejo de memoria y servidos por un manejador del dispositivo de memoria secundaria.

3) Periféricos.

La estructura de datos que describe un periferico es el descriptor del dispositivo, cada descriptor tiene asociada una cola en donde coloca los procesos que esperan asignación del periferico, la exclusión mutua en el uso de los periféricos se logra inicializando un semáforo con 1.

4)Almacenamiento Secundario o de Respaldo.

El almacenamiento de respaldo es usado para implementar memoria virtual, y es asignado por el manejador de memoria, y en el caso de que sea usada como espacio para archivos es asignada por el sistema de archivos.

La estructura de datos usada para almacenamiento de respaldo es una lista de bloques libres o un Bit Map. Los requerimentos de espacio de archivos son rechazados cuando un usuario individual se excedió en su cuota, o cuando la memoria de respaldo se agoto.

5) Archivos.

Un archivo individual puede ser visto como un recurso en el sentido de que varios procesos quieran compartirlo, si todos los procesos lo usan como ROM (Memoria de solo Lectura) el recurso es compartible, si un proceso desea escribir sobre el entonces el recurso se convierte en no compartible . Un archivo es asignado por el sistema de archivos a un proceso cuando éste lo abre y pierde esa asignación cuando cierra el archivo.La estructura de datos que describe los archivos es el directorio de archivos.

8.2. CERRADURA DE LA MUERTE.

La primer política de asignación de recursos a considerar es es aquella que contempla el problema del "Deadlock". Un "deadlock" ocurre siempre que un conjunto de procesos compiten por los recursos, y estos son asignados a los procesos solamente basándose en un criterio de disponibilidad.

En general las condiciones necesarias y suficientes para que ocurra un cerradura de la muerte son:

1) Los recursos involucrados son no compartibles.

2) Los procesos tienen asignados recursos mientras esperan por otros.

3) Los recursos no puden ser asignados mientras estén siendo usados.

4) Exista una lista circular de procesos tales que cada proceso tiene un recurso el cual está siendo requerido por el siguiente proceso en la lista.

La cerradura de la muerte puede ser resuelta adoptando una de las siguientes estrategias:

1) Prevenir una cerradura de la muerte asegurando todo el tiempo que al menos una de las 4 condiciones antes mencionadas no se de.

2) Detectar la cerradura de la muerte en el momento que ocurra y entonces tratar de recuperarlo.

3) Evitar la cerradura de la muerte.

Prevencion de la Cerradura de la muerte.

Para prevenir una cerradura de la muerte al menos una de las 4 condiciones antes señaladas debe evitarse, la primera condición es muy difícil de evitar ya que algunos recursos son de naturaleza no compartible. (el uso del "spooling" puede evitar posibles cerraduras de la muerte en periféricos no compartibles), la segunda condición puede ser evitada estableciendo cuales peticiones de recursos proceden y cuales no, sino hasta que todos los recursos pedidos sean asignados, el establecer esto tiene la desventaja de que los recursos que son usados por períodos muy cortos de tiempo son desaprovechados durante períodos largos de tiempo, la tercera condición es facil de evitar imponiendo la regla de que si a un proceso se le rechaza una petición de un recurso entonces debe liberar los recusos que tenga en ese momento, finalmente la cuarta condición puede ser evitada imponiendo un orden sobre los tipos de recursos, si a un proceso se le han asignado recursos del tipo k, entonces éste solo puede hacer peticiones de recursos de tipo que sigan al k en orden, lo cual conlleva a poner los recursos mas frecuentemente usados en orden menor.

Detección y recuperacion de la cerradura de la muerte.

Si las politicas de prevención de una cerradura de la muerte mencionadas anteriormente resultan muy restrictivas, entonces una alternativa que permita una cerradura de la muerte pero que la detecte y recupare puede resul tar razonable.

El estado del sistema en cualquier momento puede ser representado por una grafica dirigida, en donde cada nodo representa un recurso y el arco entre dos nodos implica que existe un proceso que tiene asignado el recurso del nodo origen y esta naciendo una petición del recurso del nodo destino; cuando ocurre una espera circular este se ve representado en la grafica como un ciclo.

El algoritmo trabaja sobre la grafica detectando los ciclos, i.e. detectando las cerraduras de la muerte, la inspección generalmente es hecha en intervalos fijos de tiempo, o bien puede ser hecha después de cada asignación de recursos, sin embargo como hay "simultaneidad" de asignaciones esta detección sería a cada momento.

La detección de cerraduras de la muerte resulta útil sólo si se puede recuperar de manera aceptable, la definición de "aceptable" puede ser enmarcada de acuerdo a la técnica que se elija para ello, de cualquiera de las siguientes:

a) Abortar todo proceso en cerradura de la muerte.

b) Reiniciar el proceso en cerradura de la muerte desde algún "checkpoint"si es que existe, si este metodo es aplicado ingenuamente puede llevar directamente a la cerradura de la muerte original, sin embargo lo no deterministico del sistema casi asegura que esto no pase.

c) Abortos sucesivos hasta que desaparezcan las cerraduras de la muerte,el orden de los abortos debe ser tal que minimice el costo de la perdida de lo invertido en los recursos ya usados, este proceso implica que despues de cada aborto, el algoritmo de detección debe ser llamado para que determine que ya no exite el cerradura de la muerte.

d) Los recursos que estaban asignados a los procesos en la cerradura de la muerte que fueron abortados son anora liberados.

Como en el inciso anterior debe asegurse un orden que minimice el costo de la función,tambien debe ser llamado el algoritmo de deteccion tras cualquier liberacion de recursos.

EVITANDO LA CERRADURA DE LA MUERTE.

Evitar una cerradura de la muerte significa usar un algoritmo el cual anticipe que va a ocurrir una cerradura de la muerte y el cual por tanto tenga la capacidad de rechazar una peticion de un recurso. Este concepto es diferente al de prevencion ya que asegura que una cerradura de la muerte no ocurrirá anulando una de las condiciones necesarias y suficientes.

Esquemáticamente tal algoritmo seria como sigue: antes de asignar un recurso, se debe cambiar la grafica de estado del sistema considerando el caso en que se asigno el recurso y llamar al algoritmo que detecta cerraduras de la muerte, si el resultado de dicha llamada es que no ocurrirá ninguna, entonces se asigna la petición, de otra manera se rechaza la petición y se deja la grafica de estado tal como estaba.

Desafortunadamente este algoritmo no funciona ya que se parte de la premisa de que de ocurrir una cerradura de la muerte, esta ocurrirá inmediatamente despues de que se otorge una petición de un recurso; sin embargo esto no ocurre asi, Dijkstra lo muestra usando un ejemplo en donde hay dos procesos, P1,P2 y dos recursos r1,r2 en el sistema, la ejecución de los procesos es mostrada en una grafica, en donde la componente vertical de la grafica representa la trayectoria del proceso P1 y la componente horizontal la trayectoria del proceso P2, en donde esas trayectorias se intersectan equivale a que los dos procesos estan corriendo simultaneamente, si la máquina huésped de los procesos es uniprocesador no ocurrirá esta intersección, la trayectoria de ejecución se ve restringida en la región de interseccion D, debido a que éste es un intervalo durante el cual al menos uno de los dos recursos r1, r2 es requerido por los dos procesos, en el momento en que los dos procesos entran en la region D el cerradura de la muerte es inevitable, ya que no hay trayectoria posible que pueda evitar la region D.

8.3. EL DESPACHADOR.

El término "scheduling" generalmente referencia al proceso que resuelve los problemas surgidos cuando un nuevo proceso es introducido dentro del sistema, y decide el orden con el cual los procesos seran ejecutados, éstas dos tareas estan íntimamente relacionadas con la tarea de asignación de recursos, y el proceso que tiene la responsabilidad de manejarlas es el "scheduler".

Las tareas del despachador (scheduler) puden ser descritas como:

1) Introducción de procesos nuevos.

Si el sistema es Batch, entonces las tareas que esperan su ejecución son guardadas en un almacen de tareas, el cual se encuentra en memoria secundaria, el despachador decide cual es la siguiente tarea a ejecutar dependiendo de los recursos que cada tarea requiera (ésto queda establecido en la descripción de la tarea), y del patrón de asignación de recursos en el sistema.

Con el objeto de lograr un alto "Throughput" (destogue, eficiencia), el despa chador debe iniciar un nuevo proceso cada vez que la capacidad de los recursos lo permita.

En un sistema multi-acceso los procesos son creados cuando el usuario entra al sistema, cada nuevo usua rio incrementa la demanda de recursos, un acceso puede ser rechazado cuando el número de usuarios en el sistema sea tal que se necesite aumentar el tiempo de respuesta del sistema para que sea aceptable.

2) Asignación de prioridades a procesos.

El orden en el cual son ejecutados los procesos se determina por el orden en la cola del procesador, o bien por el cual el despachador selecciona procesos de la cola, i.e por la prioridad que le asigne a cada proceso.

3) Implementación de politicas de asignacion de recursos.

Las políticas que serán revisadas son aquellas que evitan la cerradura de la muerte y mantienen el sistema balanceado, es decir aseguran que ningun tipo de recurso este sobre utilizado o subutilizado.

Debido a que el comportamiento del sistema se determina por las actividades del "scheduler", éste debe tener la más alta prioridad como proceso, de igual manera el despachador debe ser siempre un proceso cuyo estatus sea siempre el de "ejecutable", de ésta manera se asegurara que el sistema reaccione rápido de acuerdo a los cambios en la demanda.

Las circunstancias en las cuales el scheduler puede ser activado son:

1) Petición de un recurso

2) Liberación de un recurso

3) Llegada de una nueva tarea al sistema.

Puede hacerse una analogía entre las interrupciones y los eventos de "scheduling", ya que ambos ocurren en intervalos de tiempo impredecibles y ambos causan que se modifique el comportamiento del sistema.

En el caso de que ocurra un evento de "scheduling" las modificaciones afectan parametros globales del sistema como son las prioridades en los procesos, y el numero de procesos en el sistema, las modifica ciones debidas a las interrupciones se consideran de bajo nivel ya que sólo afectan el estatus de algunos procesos convirtiéndolos en ejecutables y la asignación del procesador central, las interrup ciones se espera ocurran en milisegundos a diferen cia de los eventos de scheduling que pueden o no ocurrir cada segundo, cuando el scheduler ha termi nado su trabajo este suspende su ejecución, reali zando una operacion de "wait" sobre un semáforo y será reanimado cuando el semáforo sufra una opera cion de "signal" es decir cuando ocurra otro evento de scheduling. En caso de que ocurra un cerradura de la muerte entonces no ocurrira tal evento, y por tanto debe preverse el despertar al scheduler de manera que detecte el cerradura de la muerte, esto es realizado ejecutando una operacion de "signal" sobre un semáforo con un reloj de interrupción, por ejemplo cada 10 seg.

La petición y la liberación de recursos puede puede ser implementada usando dos procedimientos:

a) Petición de Recurso(recurso,resultado)

b) Liberación de Recurso(recurso).

Estos procedimientos ponen la información necesaria sobre el recurso involucrado e identifican el proceso que lo solicita en una area de datos accesible al scheduler, y la operación de "signal" en el semáforo del sheduler. el segundo parametro es usado para transmitir el resultado de la petición y debe indicar la razón de por que fue rechazada. Las peticiones que son rechazadas deben quedar en la cola del "scheduler" hasta que el recurso este disponible.

8.4. ALGORITMOS DE SCHEDULING.

El objetivo general de los algoritmos de scheduling es el ordenar el patrón de trabajo llevado a cabo por el sistema de cómputo como una medida para maximizar la satisfaccion del usuario, esta medida difiere de sistema a sistema, por ejemplo en un ambiente batch este puede ser el desfogue total o el promedio de tiempo de respuesta para cierta clase de tareas; en un sitema multi acceso esta puede ser el tiempo promedio de respuesta del sistema ofrecida al usuario, o el tiempo de respuesta ofrecido a cierta clase de interacción.

1) La tarea de menor tiempo.

La cola del procesador es ordenada de acuerdo al al tiempo de ejecución requerido por cada proceso, éste algoritmo sólo es recomendable para los siste mas batch, donde puede obtenerse de la descripción del proceso una estimación del tiempo requerido para ejecución, su objetivo es minimizar el tiempo de "tunround" de tareas pequeñas, en su forma más sim ple el algoritmo permite a todos los procesos que su ejecución sea terminada, una modificación del algoritmo es el permitir que un proceso se apropie del procesador si el tiempo asignado al proceso actual es menor que el que requiere para terminar su ejecución, este nuevo algoritmo se llama: La tarea de menor tiempo se apropia del procesador, otra modi ficación asegura que las tareas largas no se retra sen indefinidamente, gradualmente incrementa la prio ridad del proceso de acuerdo al tiempo que tiene en la cola.

2) "Round Robin:"

Este algoritmo fue desarrollado para dar respuesta rapida a peticiones puequeñas de procesador, cuando no se puede estimar de antemano el tiempo de ejecución, cada proceso en el sistema recibe un quantum fijo de tiempo de procesador y es regresado al final de la cola, la cola es circular, y es ordenada de acuerdo a la cantidad de tiempo desde el último servicio, si el proceso requiere una gran cantidad de tiempo dara varias vueltas en la cola antes de terminar su ejecución, si el proceso requiere menos tiempo en la primera vuelta terminara su ejecución.

Este algoritmo fue usado en los primeros sistemas de tiempo compartido, y posteriormente ha sido implementado con algunas variaciones en diferentes sistemas operativos.

La mayoría de variantes intentan reducir la tendencia a que el sistema sufra caidas por estar sobre cargado, el análisis y la experiencia han mostrado que si el sistema esta muy cargado dado un tamaño fijo de quantum, entonces su funcionamiento repentinamente se ve degradado, una forma de contrarrestar ese efecto es aumentando el tamaño del quantum en la medida de que la cantidad de procesos aumente, de tal manera que aumente la probabilidad de que un proceso termine su ejecucion dentro del quantum y por tanto tambien se reduce el "overhead" de cambio de proceso.

3) Cola de 2 niveles:

Una variación del algotitmo de "Round Robin" el cual asume el contrarrestar la degradación en el funcionamiento del sistema, es el algoritmo de cola de 2 niveles, en donde los procesos los cuales no terminan su ejecucion en un número fijo de quantums, son sacados de la cola y puestos en otra llamada 'background', la cual es servida sólo si no hay procesos de terminal.

Este algoritmo puede ser facilmente de generalizar a un sistema multi-nivel, como en el caso de la DEC-10 en dónde existen 3 colas"round robin"con quantums 0.2sec, 0.25 sec y 2 sec respectivamente. Los parámetros óptimos para el algoritmo de"round robin" y sus variaciones, referentes al tamaño del quantum asi como el número de quantums antes de mover un proceso a las colas"background" son muy difícil de determinar analíticamente, aunque se han encontrado aproximaciones usando simulación y ajustándolas despues de experimentar con ellas en condiciones reales.

Los algoritmos antes mencionados forman la base de las políticas mas populares de "scheduling", existen otros algoritmos que pueden encontrarse en los trabajos de investigacion de Coffman y Kleinrock. Una variación particular es la que permite especificar la prioridad de un proceso externamente el efecto que tiene es que un proceso con prioridad externa muy alta, recibe un mejor servicio que en caso de que el algoritmo sea el "round robin" puro, este algoritmo es llamado"round robin sesgado"y asigna un quantum diferente a los procesos dependiendo de su prioridad externa.

En los sistemas el procesador central no es el único recurso importante, debido a ese hecho todos los procesos tienen uno de los siguientes estatus: 'listo', i.e. el proceso esta listo para ejecución en caso de que le sea asignado el procesador, 'bloqueado', i.e. son los procesos que deben esperar por una petición de transferencia de I/O, o de memoria,finalmente 'ejecutandose', i.e. es el proceso que tiene asignado el procesador en ese momento. En el sistema existen varias colas, la del procesador maneja los procesos 'listos', y existen además colas asociadas con varios semáforos los cuales causan que un proceso sea bloqueado, los procesos cambian de bloqueado a listo despues de que ocurre un evento tal como la operación de 'signal' en un semáforo, la cual puede presentarse en intervalos impredecibles, el movimiento de procesos entre colas puede ser estudiado para diferentes politicas de"scheduling", por análisis formal o por simulación.

Debido a la complejidad del sistema esas técnicas son difíciles de aplicar, uno de los objetivos de los algoritmos de"schedling"es minimizar el número de transiciones de estado bloqueado a listo en los procesos. Debido a que las operciones de I/O estan fuera de su control los algoritmos deben minimizar el numero de peticiones de recursos que son recnazadas y por tanto introducir al sistema sólo aquellos procesos cuyas demandas de recursos son satisfechas con los recursos disponibles, esta política es facil de implementar en sistemas batch donde la demanda máxima de recursos se conoce de antemano, en sistemas multi-acceso las demandas son virtualmente impredecibles, un ejemplo de una implementación batch es el sistema operativo Atlas en el cual las tareas son divididas en tres categorias: 1) trabajos cortos, 2) tareas de cinta magnética y 3) trabajos largos. 4) " Highest Priority First." (HPF)

El método mas simple para scneduling en la cola de listos es asignar el procesador al proceso que tenga la prioridad mas alta, éste algoritmo puede funcionar en dos casos, en el caso de que sea válido la no apropiación del procesador, el proceso con prioridad mayor se ejecuta nasta terminar o bien nasta que sea bloqueado, si llega un proceso con mayor prioridad que la que tiene el proceso que se esta ejecutando éste debe esperar nasta que termine el primero, en el otro caso se vale la apropiación del procesador, y en ese caso si llega un proceso con prioridad mayor que el que se está ejecutando, éste se apropia del procesador y manda a la cola de los listos al que se estaba ejecutando. En esta disciplina de sneduling los parámetros, de prioridad, el uso de apropiacion o no del procesador, asi como la organicion de la cola de listos son parametros libres en la implementación. En muchos sistemas las reglas de sneduling usan elincremento lineal de la prioridad, a cada proceso le es asignado una prioridad al entrar al sistema, la prioridad se va aumentando con una tasa 'a' mientras el proceso este esperando en la cola de listos y con una tasa 'b' mientras esta en procesamiento, asi tambien la prioridad puede ser incrementada dependiendo del la cantidad de operaciones de entrada salida del proceso, dependiendo de como se tomen los parámetros 'a' y 'b' existen reglas diferentes, si $0 < a <= b$ entonces se convierte en primero en llegar primero en servir (FCFS), 'a' y 'b' pueden basarse en una prioridad externa o bien pueden ser funciones no lineales.

8.5. JERARQUIA DE PROCESOS.

Como se mencionó antes el scneduler es responsable de iniciar nuevos procesos, de tal manera que el scneduler es el padre de todos los procesos introducidos en el sistema, y es el respon sable del bienestar de sus hijos con las politicas que gobier nan la asignacion de recursos e inluyen en el orden en el cuál los procesos son seleccionados por el despacnador.

Este papel de padre no es necesariamente restringido al scheduler, éste puede ser extendido a otros procesos a los cuales se les da capacidad para:

1) Crear subprocesos.
2) asignar a sus subprocesos un subconjunto de sus propios recursos.
3) Determinar la prioridad relativa de sus subprocesos.

El efecto de esto es habilitar la creación de jerarquias entre procesos con el scheduler a la cabeza, la forma natural de la estructura de procesos es la estructura de árbol y este es parcialmente anadido a la cola del procesador, el cual liga todos los procesos listos.
Las ventajas de una estructura jerarquica son dos: primero, permite a un proceso el cual nace varias tareas independientes crear procesos subsidiarios para cada tarea y por tanto que las tareas puedan ejecutarse en paralelo, si la máquina es uni procesador solo se logra una concurrencia aparente, pero si la maquina es multiprocesador su ejecución sí es en paralelo. segundo, esto permite varias versiones de un sistema operativo cada una orientada a diferentes aplicaciones, para corre simultaneamente y en paralelo. Las diferentes versiones, o subsiste mas se basan en las facilidades provistas por el nucleo del sistema, pero pueden usar algoritmos distintos para implementar las otras funciones que no son parte del nucleo.

8.6. CONTROL Y CONTABILIDAD DE RECURSOS

En la presente sección, se tratran los aspectos de la asignación de recursos, considerando su tiempo estimado de uso asi como analizando un cierto patrón de demanda que debe observarse de tal manera que se asegure que a todos los usuarios se les asigne una cantidad equitativa de los recursos de la máquina.

Aunque estrictamente nablando, esta no es una función del disen~ador del sistema, es necesario que este proporcione las nerramientas de medición necesarias para que el encargado de disen~ar el control pueda hacerlo realmente.

Podemos considerar que trabajamos siempre con dos tipos de instalaciones: las de SERVICIO y las de CONSUMO INTERNO. El trato es diferente en cada caso ya que en las instalaciones de servicio se trata de dar las mayores facilidades al mayor número de usuarios siempre y cuando estos paguen la cuota fijada y aqui el control se limita a llevarr la contabilidad de los recursos utilizados por cada uno de los usuarios y cargarselos a sus respectivas cuentas.

En el caso de las instalaciones de consumo interno, como seria por ejemplo el caso de las universidades etc, es comun que el numero de usuarios excedan las capacidades de la instalacion; a pesar de esto, se trata de que los usuarios tengan las mayores facilidades en cuanto a la utilizacion de recursos y para esto se hace necesario que se hagan distinciones entre usuarios y contar con un metodo que permita checar que ninguno exceda los limites que le fueron impuestos.

Como realmente los mecanismos de control para una instalacion de servicio se pueden considerar como un subconjunto de los de consumo interno, ampliae sobre los segundos.

RESTRICCIONES DE ACCESO.

Las facilidades de acceso que involucren gran utilizacion de recursos pueden ser anuladas por completo para ciertos usuarios (como programadores con poca experiencia o estudiantes) y mas aun, puede limitarse el uso de los recursos alos usuarios que trabajen en tiempo compartido mientras que al los que trabajen en batch, se le puede permitir una mayor cantidad de recursos, que sera asignados cuando estos no tengan demanda.

RACIONALIZACION

Las politicas de racionalizacion pueden ser cortas o largas. Las cortas se refieren por ejemplo al uso del procesador, o la asignacion de memoria que puede ser usada por un proceso.Las largas se refieren a la cantidad de recursos que son utilizados por un usuario en una semana o un mes.

En el caso de que algun usuario quiera sobrepasar el limite de los recursos a que tiene derecho, el sistema suspendera la ejecucion del proceso o simplemente negara la asignacion de dicno recurso. En ambos casos los mecanismos de control pueden implementarse por medio de un arcnivo de contabilidad que contenga la siguiente informacion:

1) Identificacion del usuario o cuenta.

2) Password del usuario.

3) Un "Vector de Facilidades" que indica a que tipo de recursos y que cantidad de ellos tiene acceso ese usuario.

4) Un limite de recursos por job.

5) La cantidad de recursos utilizada bajo esa cuenta especifica.

6) La cantidad de recursos que le quedan disponibles todavia.

La manera de controlar la cantidad de recursos varia dependiendo del tipo de sistema que se trate, por ejemplo si es un sistema tipo batch, se hace un balance previo a la ejecucion del job, de tal forma que si no tiene recursos suficientes, no es atendido; en sistema de multiprogramacion, el balance de recursos se hace cada determinado tiempo para checar que ningun usuario se exceda.

Generalmente se tienen una serie de parametros con un cierto peso, que se utilizan para integrarlos en una formula que da el balance global de los revursos utilizados. Como ejemplo podemos citar el de ESSEX University DEC System :

Numero de Unidades Usadas = 133p + 6c + 13.3k + 5.5w

donde

p = Tiempo en segundos del procesador central.

c = Tiempo de coneccion en la terminal.

k = Memoria ocupada.

w = Numero de transferenncias a disco.

Es importante notar que el peso que se asigne a los diferentes parametros puede impactar de alguna manera la utilizacion que hagan los programadores del sistema, ya que si por ejemplo, se cobra muy alto el hecho de utilizar por largos periodos de tiempo la terminal, podria repercutir en que en un futuro se utilizaran mas las ventajas de programar en batch.

Existen algunos mecanismos mas sofiticados que manejan basicamente la misma idea de los parametros pero con diferentes pesos dependiendo de la carga del sistema a una determinada hora.

Un reloj de tiempo real resulta indispensable y la contabilidad de cada proceso puede incluirse como parte de la informacion del descriptor de procesos en tanto que las estadisticas globales pueden almacenarse en un archivo comun.

El peso que debe asignarse a cada parametro depende de la aplicacion particular, del tipo de instalacion que se tenga y de la opinion particular del diseñador del sistema y es una parte importante en el desarrollo del mismo.

****

UNIX

Unix es un sistema operativo que realmente ha tenido una gran demanda y ha marcado en cierta forma un nuevo esquema a seguir por los nuevos sistemas que continuamente se estan desarrollando.Su influencia ha sido tal que existen actualmente muchísimos sistemas "Unix-Like" (parecidos a Unix) y que según sus autores, superan al Unix original en muchos aspectos; como ejemplos podemos citar a XENIX,TUNIX,IDRIS etc.

UNIX es un sistema operativo de propósito general,interactivo y multiusuario y fue diseñado para correr en las computadoras PDP-11 de la Digital Equipment Corporation y la Interdata 8/32.

La mayoría de las aplicaciones de UNIX son sobre enseñanza por computadora,manejo y formato de textos y documentos, recolección y procesamiento de datos que pueden llegar desde distintas máquinas, además por supuesto, de contabilizar y administrar las cuentas de los servicios telefónicos, ya que este sistema fue desarrollado en los Bell Telephone Laboratories.

Los diseñadores de UNIX argumentan que para hacerlo no fue necesario invertir ni mucho dinero en equipo, ni mucho tiempo para desarrollarlo completamente y que tiene características que lo hacen superior a otros sistemas, como son: Simplicidad,Elegancia y Facilidad de Uso.Con respecto a este punto, los detractores de UNIX alegan que no es cierto que sea un sistema de fácil manejo, sino que por el contrario, resulta hostil a los usuarios por la gran cantidad de comandos y la forma críptica que presenta el sistema para utilizarlos.

UNIX proporciona entre otras facilidades:

a) Compilador de C.

b) Editor de Textos.

c) Ensamblador.

d) Cargador-Ligador.

e) Depurador Simbolico.

f) Varios Lenguajes de Programación, como: ALGOL 68, PASCAL, BASIC, SNOBOL, APL etc.

UNIX fue instalado en una computadora PDP 11/70 cuya longitud de palabra es de 16 bits y 768 kb (con bytes de 8 bits) de memoria principal. El Kernel del sistema ocupa aproximadamente 90 kb divididos entre las tablas de datos y el código.El sistema es capaz de manejar una gran cantidad de manejadores de dispositivos (device handlers) y proporciona un gran espacio para los buffers del sistema de E/S.

La PDP de los Laboratorios Bell tiene dos discos de cabeza movible de 200 Mg para el sistema de archivos, interfaces para la comunicación entre máquinas, una impresora, una graficadora y un sintetizador de voz, una red digital de switches y una máquina de ajedrez.UNIX fue escrito inicialmente en ensamblador y posteriormente fue reescrito en C para facilitar su comprensión y modificación, así como para permitir nuevas técnicas de multiprogramación y código reentrante.

PROCESOS E IMAGENES

IMAGEN.- Es un ambiente de ejecución que incluye:Imagen en memoria,Registros, Archivos abiertos, Directorio Actual.

PROCESO.- Es la ejecución de una imagen en memoria.Cuando un proceso se esta ejecutando, las imagenes de los otros procesos estan estáticas, ya sea en disco o en memoria, dependiendo de su tamaño.

En UNIX todo es un proceso.Cuando SHELL (el núcleo de UNIX) recibe un comando, crea un proceso para ejecutarlo y no ejecuta otro comando hasta que los n procesos que activó el proceso previo hayan terminado.Las llamadas a SHELL son como las llamadas a subrutinas, como las llamadas a BDOS en CP/M, pero a diferencia de CP/M se utiliza otro stack por razones de seguridad y privilegios.

La imagen en memoria se divide en tres segmentos lógicos, contiguos o no:

1) Segmentos de Texto (Código puro).

2) Segmento de Datos.

3) Segmento de Datos (del Stack).

El Segmento de Texto comienza siempre en la dirección Ø del espacio virtual.Este segmento es Read/Only. Con ésta facilidad es posible que varios procesos compartan el código, es decir, el código es reentrante.Hay en el sistema una tabla de Segmentos de código, que representa a los segmentos activos y que contiene entre otras cosas el número de procesos que lo comparten.

Acerca de los segmentos de datos del usuario tenemos como ya dijimos, segmentos de datos y segmentos del stack. Existe además un segmento denominado DATOS DEL SISTEMA que contiene el resto de la imagen (registros, archivos abiertos, y el directorio actual, además del proceso y un área auxiliar de variables.) Como esta parte no es formalmente una parte del área de datos del usuario, no puede ser modificada.Hay un segmento de datos del sistema por proceso.

Todas las direcciones son virtuales y en ejecución son relativas y esto es con objeto de que el proceso pueda ser guardado en disco y al cargarse de nuevo pueda ejecutarse independientemente del lugar que le toque.

CREAR UN PROCESO IMPLICA CONSTRUIR SU IMAGEN Y EJECUTARLO ES EJECUTAR SU IMAGEN.

COMUNICACION ENTRE PROCESOS

La comunicación entre procesos se hace através de diferentes llamadas como por ejemplo pid = fork() divide el procesos en dos procesos idénticos que son el proceso padre y el proceso hijo y corren de manera independiente.Comparten el mismo segmento de código, y los archivos abiertos.Al momento de ejecución se crean dos áreas de segmentos de usuario que cada uno modifica de manera independiente.Para identificar a cual de los dos procesos nos referimos, fork() regresa Ø si se trata del hijo y el número del proceso si se trata de padre.

Existen en UNIX unos elementos llamados PIPES que literalmente quieren decir "Tuberías"aunque no es en este sentido que vamos a usarlos aquí. PIPE significa un canal de comunicación entre procesos.Este canal, así como los demás archivos abiertos se pasan de procesos padres a procesos hijos por medio de la llamada fork. Si ponemos fd = pipe() funciona como un apuntador a un archivo abierto que es una dirección en la tabla de archivos abiertos. Este método permite la comunicación entre procesos pero el PIPE debe ser generado por un ancestro común a los procesos que queremos comunicar.

EL NUCLEO DE UNIX

El núcleo o Kernel de Unix esta constituido por 10,000 líneas de código escritas en C que es un lenguaje de alto nivel, y de aproximadamente 1000 líneas en lenguaje ensamblador. Este código representa entre el 5 y 10 por ciento del sistema operativo y es el Kernel la única parte de Unix que debe ser sustituida cada vez que desea implementarse Unix en otra máquina.

El núcleo es un programa residente permanentemente y proporciona un medio ambiente básico para que puedan correr los demás programas. En él es donde se encuentran implementadas las llamadas al sistema, y arregla que todos los programas compartan los recursos de la máquina en forma armoniosa.

Para la mayoría de la gente, un sistema operativo constituye un buen ejemplo de lo que es un programa poco portable, sin embargo, la mayor parte de Unix resuta ser bastante portable. El núcleo puede ser dividido en tres partes que difieren en su grado de portabilidad.

La primera parte es la que contiene LAS PRIMITIVAS EN LENGUAJE ENSAMBLADOR que son las de más bajo nivel y de hecho son interfase con el hardware. Son aproximadamente 800 líneas de código en el ensamblador de la Interdata 8/32 que fue donde se implemento Unix, después de la PDP-11. Algunas de estas rutinas pueden llamarse directamente, otras necesitan ser llamadas por programa. Como ejemplo del tipo de rutinas que se encuentran a este nivel podemos citar:

a) Habilitar y Deshabilitar interrupciones.

b) Invocar operaciones básicas de Entrada Salida.

c) Pasar el control de un procesos a otro.

d) Transmitir información entre el espacio de direccionamiento del usuario y el del sistema mismo.

Cada vez que desea implementarse Unix en una nueva máquina es ésta la parte que debe reescribirse por completo, y no sólo es a nivel de traducción de un ensamblador a otro, sino que el problema va más allá , ya que deben efectuarse ajustes a cada caso particular de mapeo de memoria, manejo de interrupciones, protección etc. que varía muchísimo de máquina a máquina.

La segunda parte está constituida por EL MANEJADOR DE DISPOSITIVOS que es el que proporciona el manejo de las interrupciones,procesamineto de comandos de entrada salida y la recuperación en caso de errores para todos los dispositivos conectados a la máquina.En el caso de la Interdata el tamaño total de los manejadores de disco,cinta magnética, consola y terminales es de aproximadamente 1100 líneas de código en C. Naturalmente estos son programas dependientes de la máquina en la que van a correr por la naturaleza misma de los dispositivos.

La parte restante es la más grande, aproximadamente 7000 líneas en C. En esta parte se puso especial interés en mantenerla lo más independientemente posible de la máquina en la que fuera a correr, para que fuera sumamamente portable,por lo que se parametrizaron todas las constantes y se se fue muy cuidadoso en cómo se declaraban las variables.

De las 7000 líneas de que consta esta parte,solo 350 difieren entre la versión que corre en la PDP-11 y la que corre en la Interdata 8/32 lo que implica que son iguales en el 95% del código. Los aspectos donde se requirieron las modificaciones fueron:

a) Manejo de llamadas a subrutinas. (Los satcks en cada caso crecían en direcciones opuestas).

b) Manejo de Memoria.

c) Rutina que maneja los errores e interrupciones al procesador.

MANEJO DE MEMORIA

El área de datos total asociada a cada proceso y que está formada por el segmento de datos del usuario, el segmento de datos del sistema y el segmento de texto es intercambiada de memoria principal a secundaria conforme se necesite. El segmento de datos del usuario y el del sistema se almacenan en partes contiguas de memoria principal, para disminuir el tiempo de latencia.

La asignación tanto de memoria primaria a secundaria se hace en base al algoritmo del "primero que encuentre".Cuando un proceso crece, se le asigna una nueva área de memoria y el contenido del área vieja se copia en el área nueva, libera el área vieja y actualiza las tablas. Si no hay suficiente espacio en memoria principal, el procesos se pasa a memoria secundaria y es intercambiado de nuevo con su nuevo tamaño en cuanto sea posible.

Un proceso separado en el kernel,el proceso de intercambios, es el encargado de intercambiar a otros procesos dentro y fuera de la memoria principal. Esto lo lleva a cabo examinando los procesos listos para correr y le asigna memoria principal y trae sus segmentos, para que ese proceso pueda competir con otros procesos que ya estan cargados, por el procesador central.

Si no hay memoria principal disponible, el proceso que intercambia checa en una tabla cuáles procesos podrían ser sacados para intercambiarlos por el nuevo. Hay dos algoritmos para hacer el intercambio de los procesos ; para elegir cual de todos los procesos que estan afuera es el que va a entrar, se toma en cuenta el tiempo que llevan en almacenamiento secundario y entra primero el que lleve más tiempo.

Para decidir cual procesos de los que están cargados es el que debe ser sacado, se toma en consideración el tipo de eventos que están esperando y se sacan primero aquellos cuyos eventos son más tardados en ocurrir como por ejemplo,los de entrada salida o simplemente los procesos que no están corriendo.

LLAMADAS DE ENTRADA/SALIDA

El sistema de llamadas para Entrada/Salida fue pensado paa eliminar dentro de lo posible, las diferencias entre los dispositivos y las formas de acceso.

No hay distinción entre acceso random y secuencial;no existen registros de tamaño fijo impuestos por el sistema.El tamaño de un archivo se determina por la cantidad de bytes que ocupa.Se maneja un DESCRIPTOR DE ARCHIVOS que es un entero que se usa para identificar el archivo en operaciones subsecuentes de lectura y escritura.

Ya mencionamos que las entradas de los directorios contienen un nombre asociado a cada archivo y un apuntador al archivo mismo.Este apuntador es el NUMERO DE INDICE (i-number) del archivo. Cuando un archivo es accesado, su i-number se usa como índice en una tabla del sistema llamada i-list, que esta almacenada en el dispositivo donde se encuentra el directorio.

La entrada correspondiente al archivo, se denomina por i-node , tiene una longitud de 64 bytes y contiene la siguiente información:

1) ID del usuario y grupo al que pertenece.

2) Bits de protección.

3) Direcciones físicas en disco o cinta, del archivo. tiene espacio para 13 direcciones.

4) Tamaño del archivo.

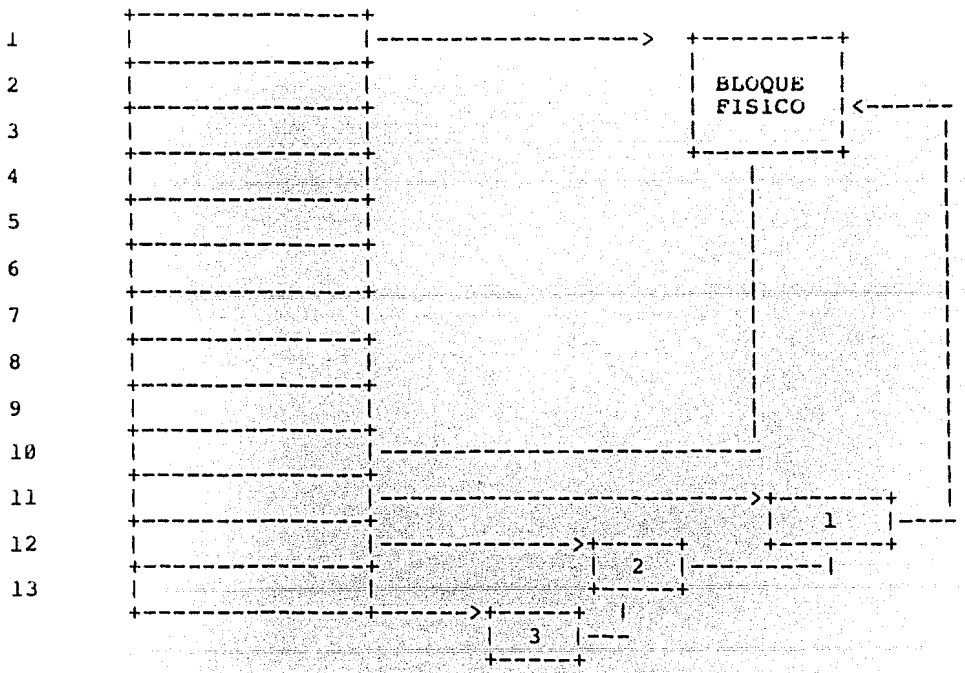
5) Fecha de creación, ultimo acceso, modificación.

6) Número de Ligas.

7) Clave que indica el tipo de archivo de que se trata.

De las 13 direcciones en disco o cinta,las primeras 10 apuntan a los primeros 10 bloques del archivo.Cada bloque es de 512 bytes.Si el archivo es mayor de 10 bloques, la dirección 11 apunta a un bloque indirecto que contiene 12818 direcciones para bloques adicionales al archivo.Si aún no es suficiente, la dirección 12 apunta a 128 bloques indirectos, cada uno apuntando a 128 bloques del archivo;análogamente, la dirección 13 proporciona un triple indireccionamiento.

Así vemos que un archivo puede crecer hasta un tamaño de (10 + 128 + 128**2 +128**3) . 512) = 1,082,201,088 bytes y dependiendo de su tamaño, sera el número de accesos a discos necesarios para leer o escribir en él.



La configuración del disco es como sigue:

1) Boot del sistema.

2) Super-Block (Contiene los datos del disco).

3) i-list (lista de nodos,cada uno de 64 bytes, direccionados por un i-number)

4) <bloques> (Todos están ligados en una lista de espacio disponible, mientras no sean ocupados).

Cuando se ejecuta una llamada de OPEN y CREATE, el sistema debe transformar la trayetoria del nombre específicado por el usuario, en el i-number.Una vez que el archivo está abierto, su dispositivo, sui-number y su apuntador para las operaciones de lectura y escritura se colocan en una tabla del sistema, de tal forma que las siguientes operaciones al archivo encuentren su descriptor facílmente.

Un beneficio de usar i-number es que no se manejan los nombres completos de los archivos.El costo de usar un archivo se reparte equitativamente entre todos los que tengan ligas a él y no cabe duda que esto no representa un grave problema ya que UNIX se pensó para usarse dentro de un ambiente científico.

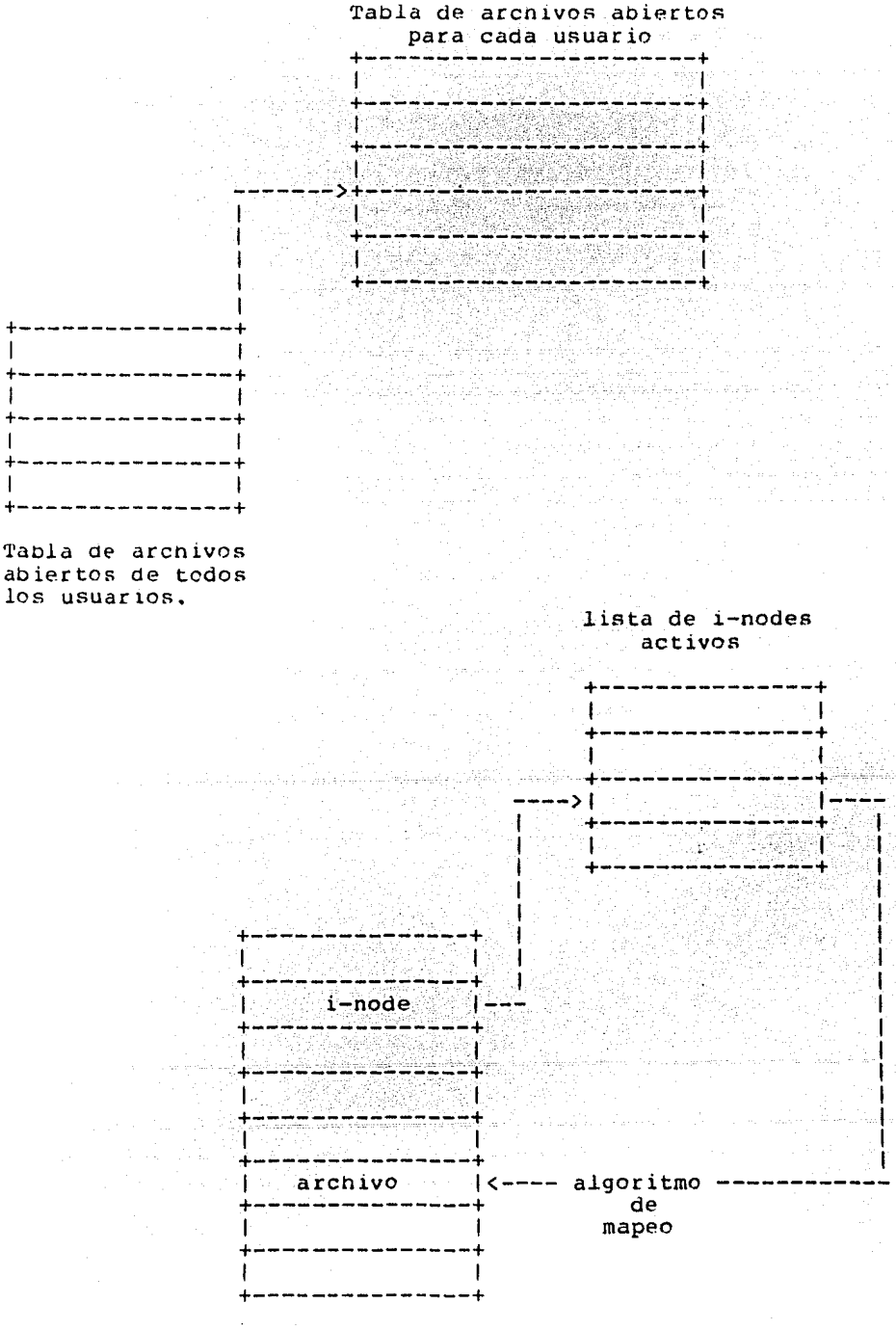


Tabla de archivos abiertos
para cada usuario
Tabla de archivos
abiertos de todos
los usuarios.
lista de i-nodes
activos
i-node
archivo
algoritmo
de
mapeo

EL SISTEMA DE ARCHIVOS

Como ya vimos una de las funciones más importantes dentro de un Sistema es el manejo del SISTEMA DE ARCHIVOS. Desde el punto de vista del usuario hay tres tipos de archivos:

i) Ordinarios (archivos en Disco).

ii) Directorios.

iii) Archivos Especiales.

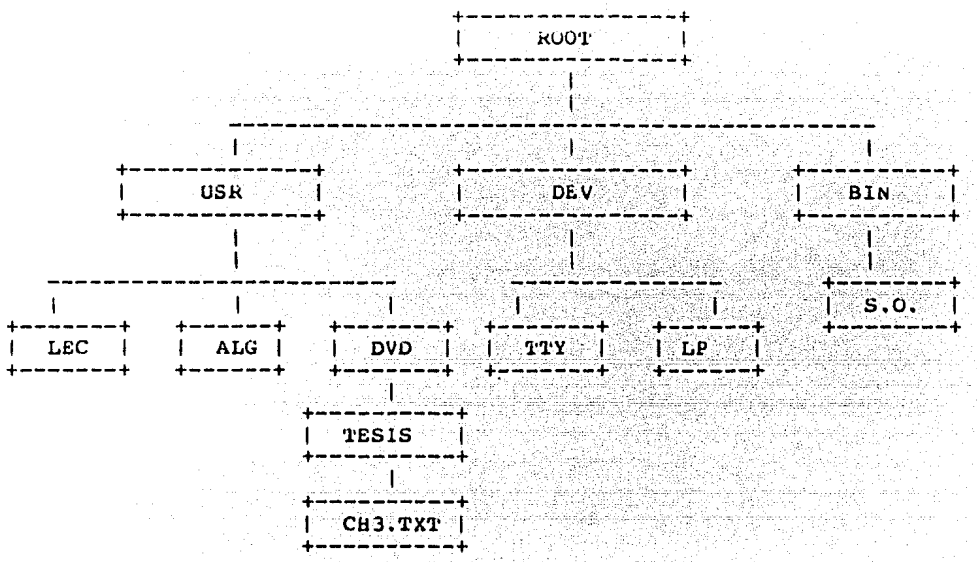
Los Archivos Especiales pueden contener cualquier tipo de información y no requieren de una estructura específica, ya que en UNIX, los archivos son cuerdas de caracteres, donde cada línea se delimita con un caracter de New Line.

Los Directorios son los archivos que hacen la correspondencia entre los nombres de los archivos y los archivos físicos.

Cada usuario tiene su propio directorio que a su vez puede contener subdirectorios de archivos agrupados por temas o tópicos, esto es, los directorios mantienen una estructura Jerárquica.El sistema maneja varios directorios para su propio uso y uno de ellos es ROOT.

Cualquier archivo dentro del sistema pede ser accesado siguiendo trayectorias através de los distintos directorios, hasta encontrar el archivo deseado.Los comandos e encuentran en otro directorio donde estan todos los programas de uso general.Los Directorios se comportan exactamente como cualquier otro tipo de archivos excepto que solo los usuarios privilegiados(Super-User) pueden modificarlos o crearlos.

Los nombres de los archivos pueden contener nasta 14 caracteres y para localizarlos dentro del directorio, puede especificarse una trayectoria que es una sucesión de nombres de directorios separados por diagonales ("/") y al final, el nombre de un archivo.



ESTRUCTURA JERARQUICA DEL DIRECTORIO

Si el nombre comienza con "/", la busquéda comienza en ROOT y si no , en el directorio del usuario.En UNIX se tiene la posibilidad de tener un mismo archivo bajo diferentes nombres e incluso bajo diferentes directorios. A esta facilidad se le conoce como "LINKING", es decir, una entrada de un directorio puede tener un apuntador o liga a un archivo.Todas las ligas tienen el mismo status.

Una entrada del Directorio, consiste en el nombre del archivo y un apuntador a donde se encuentra el archivo.Un archivo no desaparece mientras haya una liga apuntandolo, no importa desde que directorio.

Todos los directorios tienen archivos de default:

a) El identificador "." se refiere al mismo directorio. (apunta a sí mismo)

b) El identificador ".." se refiere al directorio que lo creó. (apunta al nivel inmediato superior)

Los Arcnivos Especiales constituyen la característica más importante de UNIX. CADA DISPOSITIVO DE ENTRADA SALIDA ESTA ASOCIADA CON AL MENOS UNO DE ESTOS ARCHIVOS.Los archivos especiales pueden ser leídos y escritos como cualquier archivo ordinario en disco y estas operaciones implican la activación del dispositivo requerido.

Las entradas de los archivos especiales están bajo el directorio "/dev" y se pueden ligar como cualquier otro archivo, ejemplo:

Para usar la terminal: /dev/tty3

Para usar la cinta magnética: /dev/mt

Se obtienen varias ventajas del manejo de archivos especiales, entre las que destacan:

a) Los dispositivos de E/S se manejan lo más homogéneamente posible.

b) Los nombres de archivos y dispositivos tienen el mismo significado, de tal forma que los dispositivos pueden ser pasados como parámetros.

c) Los archivos especiales se sujetan a los mismos mecanísmos de protección que los demás archivos.

SISTEMA REMOVIBLE DE ARCHIVOS

UNIX permite guardar el sistema de archivos en diferentes dispositivos de tal manera que no es necesario que la estructura completa resida en uno solo de estos dispositivos.Por medio del comando "MOUNT" es posible anexar estructuras independientes al sistema de archivos, con su propia estructura de directorio.

El efecto del MOUNT es substituir una hoja (archivo ordinario) dentro del sistema de archivos, por la estructura nueva.De hecho, cuando se carga el sistema, se hace un MOUNT de ROOT con el resto de los directorios, que se encuentran almacenados en otros dispositivos.Una vez hecho el MOUNT, no hay diferencias para referirse a los archivos de una u otra estructura, excepto que no puede haber ligas entre un directorio y otro.Esto es con objeto de facilitar el subir nuevamente la información y evitar tablas con informaciones adicionales.

PROTECCION

El esquema de protección es sencillo.El sistema asigna a cada usuario un número uníco que lo identitica del resto de los usuarios.Cada vez que se crea un nuevo archivo, se le asigna la identificación del usuario (llamada ID) y le proporciona 10 bits de protección:

3 bits para lectura
3 bits para escritura
3 bits para ejecución

Si el decímo bit está prendido,indica al sistema que puede cambiar temporalmente el ID del usuario actual por el del que genero el programa y esto solo vale durante la ejecución del mismo. Esta característica proporciona privilegios a programas que de otro modo no podrían accesar algunos archivos, como por ejemplo, una Base de Datos, que no permitiera ser leída ni modificada excepto por la misma Base de Datos.Los mecanismos de protección son pasados por alto para los usuarios privilegiados.

DESPACHADOR Y SINCRONIZACION

La sincronización de procesos se lleva a cabo mediante procesos que esperan por eventos, que son representados por enteros arbitrarios.Por convención, los eventos tienen asignadas direcciones en tablas, asociadas con dichos eventos es decir, sí por ejemplo, un fork está esperando que termine uno de sus "hijos", esperará por un evento que es la dirección de su propia tabla de descripción de proceso. Cuando un proceso termina, sen˜ala el evento representado en el descriptor de procesos de su padre.Si nadie esperaba por ese evento no afecta en nada, pero sí varios procesos esperaban por ese evento, todos se despiertan al mismo tiempo.

Este esquema difiere considerablemente del que propone Dijkstra para la sincronización de procesos ya que no existen localidades de memoria asociadas a los eventos; no nay necesidad de asignar eventos antes de usarlo, sino que simplemente existen cuando son usados.

Es importante resaltar también las desventajas que representa el manejar los eventos de esta manera ya que no existe la nocion de "que cantidad" asociada con el mecanismo de eventos, por ejemplo: Los procesos que esperan por memoria, se activan simultáneamente tan pronto haya memoria disponible y se produzca el evento asociado con esta operación, sin importar el tamaño de la memoria disponible, por lo que todos compiten por este recurso aunque para algunos resulte insuficiente el recurso liberado.

Además, puede suceder que el evento ocurra mientras el proceso se está poniendo en estado de wait, y esto implica que el proceso se puede quedar esperando indefinidamente por un proceso que ya ocurrió y que puede no volver a ocurrir. Esta situación se presenta porque no hay un lugar en memoria que indique que el evento ya ocurrió; la manera en como altera un evento a los procesos es marcarlos de estado de espera a estado de listos para correr.Este problema se resuelve en gran parte por el hecho de que el cambio de proceso solo ocurre en el kernel por medio de una llamada explícita al mecanismo de eventos. Si el evento es mandado por otro proceso, no hay problema, pero si es resultado es por una interrupción de hardware se debe tener especial cuidado. Estos problemas de sincronización, reperesentan una de las mayores dificultades para adaptar UNIX a configuraciones con múltiples procesadores.

Ya mencionamos que la acción de los eventos es marcar a varios procesos como listos para correr, ahora bien, cual de todos es el que va a elegirse? UNIX asocia a cada proceso una prioridad. Las prioridades de los procesos del sistema se asignan de acuerdo al código correspondiente a un evento que indica la velocidad de respuesta que se espera tenga ese evento , por ejemplo, los eventos de disco tienen una alta prioridad, los teletipos son más lentos, y algunas utilerías no muy importantes tienen una prioridad aún más baja.

Las prioridades de los procesos de los usuarios son menores que las más bajas de las prioridades del sistema, y son asignadas mediante un algoritmo que se basa en la razon que se obtiene de dividir el tiempo de procesador entre el tiempo real transcurrido ;esta razón es calculada cada segundo. Un proceso que ocupe mucho tiempo de procesador en poco tiempo real tendrá una prioridad baja, mientras que los procesos interactivos que en general no llevan mucho tiempo de proceso, se les asigna una prioridad alta, y con esto se aseguar un rápido tiempo de respuesta al usuario.

El algoritmo del despacnador simplemente toma el proceso con la más alta prioridad, tomando por tanto todos los procesos del sistema primero y el proceso del usuario con más alta prioridad después.

Para todos los procesos con igual prioridad o procesos que no terminaron su ejecución en un cuantum de 1 segundo, se manejan por medio del "Round Robin". El algoritmo del despachador checa que si algún proceso con alta prioridad se "apodera" por demasiado tiempo de la máquina, automáticamente le baja la prioridad y si por el contrario, un procesos a estado por mucho tiempo esperando a causa de una prioridad baja, se le aumenta.

SEGURIDAD

Seguridad significa la habilidad de protegerse contra accesos indeseados o destrucción de datos por otros usuarios. UNIX fue pensado para correr en un medio ambiente científico por lo que no se hizo mucho énfasis en la parte de protección. La mayoría de las versiones carece de métodos que chequen el sobreconsumo de recursos como espacio en disco, número de archivos o número de procesos por usuario. En general cuando un recurso se acaba, es fácil detectar cual usuario fue el causante y se soluciona la situación, lo realmente peligroso es cuando ocurre un error accidental en el sistema.

En el aspecto de archivos ya se describió la protección en una sección previa.

EL SHELL DE UNIX

Para la gran mayoría de los usuarios, la comunicación se lleva a cabo por medio del programa SHELL, que es el núcleo del sistema. El SHELL es un interprete de líneas de comandos, que recibe un comando y considera como una petición para ejecutar otro programa.Una línea de comandos presenta el siguiente formato general:

COMANDO arg1, arg2,..., argn

Shell se encarga de separar el comando de los argumentos y "COMANDO" se busca en los directorios de programas ejecutables y cuando lo encuentra, le proporciona la lista de argumentos.

Si no lo encuentra, cambia el pretijo de la trayectoría por el de "/bin" e inicia otra vez la búsqueda.La trayectoria puede modificarse al gusto del usuario.

Todos los archivos necesitan de un descriptor de arcnivo que es asignado en el momento de ser abiertos o creados por un programa, sinembargo, los programas creados por SHELL inician con tres archivos abiertos cuyos descriptores de archivos son 0, 1, y 2.Cuando el programa inicia su ejecución, el arcnivo 1 es abierto para escritura y se denomina Archivo Standard de Salida y en general es siempre la terminal del usuario. Análogamente, el archivo 0 denota el Archivo Standard de Lectura y los programas que requieran leer mensajes tecleados por el ususario, leen este archivo.

Por otra parte, es posible "redireccionar" las entradas o salidas por medio de ">", lo cual le indica que por la duracion de ese comando el medio se modifica al que sucede al simbolo ">" por ejemplo: el comando "ls" normalmente manda a la impresora los nombres de los archivos en el directorio actual, pero puede ser modificado a que cree un archivo llamado ej1 y mande aní su salida por mediio del comando:

```
ls > ej1
```

o bien podemos lograr que el editor, que usualmente toma su entrada de la terminal, tome ahora su entrada del archivo ej2 por medio del comando

```
ed < ej2
```

El archivo 2 se mantiene asociado a la terminal, ya que en caso de hacerse el redireccionamiento, pueden ocurrir condiciones especiales o de error, que el sistema notificaría pero que no serían atendidos por el usuario porque irían todos al otro arcnivo de salida.

FILTROS

Una extensión de la noción de entrada /salida se usa para dirigir la salida de un comando, como entrada de otro. Una sucesión de comandos separados por barras verticales ocasiona que SHELL ejecute todos los comandos simultáneamente y que haga las redirecciones especificadas por el comando.

Hay varios tipos de filtros:

tr	Traduce una tabla de caracteres a otra.
sort	Hace un sort de líneas.
uniq	Cancela las líneas duplicadas.
comm	Hace una comparación entre arcnivos y escribe las diferencias en la pantalla.

utilizando estos filtros, podemos hacer fácilmente un corrector de sintaxis mediante la siguiente línea de comandos:

```
cat arcn | tr [' ']|<cr>] | tr | , : ;]|] | sort | uniq|comm
```

Otra característica de UNIX es que los comandos no requieren ser dados en líneas separadas para poder ser ejecuyados, sino que basta separarlos con ";" para que el SHELL ejecute uno primero y luego otro. Sinembargo se tiene otra opción mas interesante porque permite tener MULTITAREAS por medio del comando "&". Esto forza al SHELL a contestar con su prompt inmediatamente, aún antes de haber concluído la ejecución del comando anterior por ejemplo:

```
as fuente > salida &  ls > archivos &
```

indica que "fuente" será ensamblado y su salida será mandada a "salida" mientras que por otra parte se manda el listado de los archivos en el directorio, a "archivos".

IMPLEMENTACION DEL SHELL

La mayor parte del tiempo, SHELL está esperando a que el ususario le teclee comandos. En cuanto se da cuenta de que se completó un comando con sus argumentos, este analiza el comando, acomodando los argumentos en forma apropiada para el "execute" . Posteriormente se hace una llamada "fork" que es la encargada de crear dos procesos idénticos: el padre y el hijo. El proceso hijo, trata de realizar el "execute" con los argumentos apropiados y de lograrlo, el proceso padre espera (mediante un wait) a que el hijo se muera, cuando esto sucede, el SHELL se entera que el comando se terminó de ejecutar y manda su prompt para esperar por otro comando.

Dado este esquema de trabajo, el implementar MULTITAREAS resulta más o menos fácil, ya que siempre que la línea de comandos contenga un "&", indica a SHELL que no debe esperar a que el hijo avise si terminó o no de ejecutar el comando. Cuando un proceso es creado por medio de "fork", hereda no solo la imagen de su padre en memoria, sino también todos los archivos abiertos incluyendo los de descriptor 0, 1, y 2. Cuando se tiene "<" o ">" en la línea de comandos, se hace la redirección antes de entrar a ejecutar, cerrando los archivos standard y abriendo los nuevos archivos específicados.

EVOLUCION DEL SISTEMA UNIX

Como ya mencionamos, la popularidad de UNIX se debe a las características que ha desarrollado como son: Un Sistema de Archivos Jerárquico, Un manejo compatible para archivos, dispositivos y entrada/salida entre procesos. Las versiones más extendidas de los Sistemas basados en UNIX son System III y System V de XENIX (de la Western Electronics) y Coherent e Idris , todos basados en la versión 7 de UNIX.

Los Laboratorios Bell transportaron UNIX a una computadora VAX 11/780, de 32 bits con hardware para manejo de memoria virtual y en 1979, otro grupo de la Universidad de Berkeley, le añadió el manejador de memoria virtual. Este proyecto incluye además otras mejoras como son: el control de jobs a nivel foreground y background, reiniciar el sistema automátcamente en caso de una caída y un poderoso depurador simbólico.

A este sistema desarrollado en Berkeley se le conoce como UNIX 4.2 BSD y permite la ejecución de procesos de longitud considerable. La clave esta en utilizar un algoritmo de reemplazo de páginas global, que ayude a minimizar la antidad de memoria utilizada por cada proceso reteniendo el conjunto de páginas que han sido usadas más recientemente; por lo tanto es posible la ejecución concurrente de varios procesos grandes, cuyo tamaño es mayor al de la memoria principal disponible. Más aún, es posible correr procesos de 16 megabytes en una VAX/11 (superminicomputadora) con 4.2 BSD o bien en una estación de trabajo llamada SUN y que esta basada en un microcomputador 68010.

El agregar las facilidades del manejo de memoria virtual, incrementa considerablemente el número de usuarios que puede sopotar un sistema de tiempo compartido, porque en vez de cargar todo el programa completo en memoria, las páginas se van trayendo a medida que son requeridas.

UNIX maneja su sistema de archivos con un disco con capacidad de transferencia de un megabyte por segundo, pero en el caso de ser un archivo grande, el promedio baja a 50 kilobytes, debido a la dispersión de los bloques. En la nueva versión, se utilizan nuevos algoritmos y nuevas estructuras de datos, que agilizan el acceso en disco.

Antes explicamos que en cada acceso, UNIX trae un bloque de 512 bytes en una sola transferencia y para una cantidad mayor, se requieren de dos o mas accesos.A medida que pasa el tiempo, los archivos que son creados y borrados, revuelven demasiado la lista de bloques libres, de manera que los bloques de un mismo archivo quedan muy dispersos por el disco y esto retarda sus acceso.La nueva versión e UNIX, nace un estudio profundo de la geometría del disco, y coloca los bloques de un archivo lo más contiguos posibles uno de otro, ya sea en un mismo track o cilindro del disco para evitar el tiempo de seek (búsqueda) del siguiente bloque.

Con respecto a la comunicación entre máquinas, hay que recordar que UNIX se desarrollo antes de que las redes de computadoras fueran una práctica común, por lo que no preveía este tipo de requerimentos.Las facilidades que ofrecen los pipes son para procesos en la misma mquina y no contempla el esquema de comunicación con otras computadoras. En lugar de definir una arquitedura de redes nueva, el sistema 4.2 BSD introduce un subsistema que soporta protocolos para redes de computadoras.

Este subsistema permite manejar diferentes tipos de protocolos, de la misma manera que los drivers pueden manejar diferentes tipos de discos.El subsistema de comunicación abstrae los conceptos comunes a los sitemas de comunicaciones y las proporciona al usuario, evitando los detalles.Por lo que este sistema además, es muchomás flexible que otros que definen su propio protocolo de comunicación y lo mantiene abierto para manejar los protocolos ya existentes o cualesquiera otros que aparezcan en el futuro.

La práctica ha demostrado que las facilidades proporcionadas por la red local de comunicación del 4.2 BSD es realmente excelente. En el procesador de la VAX 11/750 o en el MC68000 se tiene un overnead en el protocolo de menos de un milisegundo por paquete y que el promedio de transferencia de datagramas es alrededor de 1000 por segundo. Estos promedios son de dos a cinco veces mas rápidos que los que se logran usando el Transmission Data Protocol/Internet Protocol (TCP/IP) en procesadores similares.

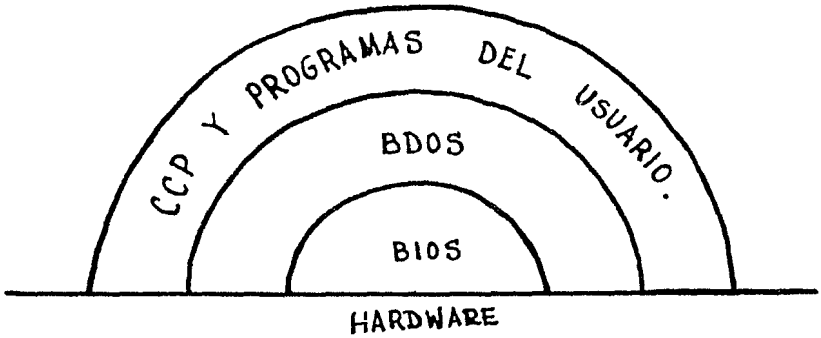
CP/M

CP/M es un sistema operativo diseñado para correr en un procesador de 8080 pero permite al usuario efectuar ciertas modificaciones para que sea utilizado bajo distintas configuraciones de hardware.

El CP/M standard (version 2.0) esta diseñado para manejar discos de densidad sencilla, pero es posible alterar algunos campos para que se hagan las adaptaciones para el manejo de otros subsistemas de disco.

CP/M es un sistema pequeño,que emplea técnicas basadas en tablas lo que permite una gran flexibilidad para reconfigurarlo para una extensa variedad de capacidades de disco.

Para proporcionar una mayor independencia de los dispositivos, CP/M se dividió en tres modulos independientes cuya estructura puede situarse por capas como en el esquema que plantea Lister:



BIOS

(Basic I/O system) Es el Sistema Basico de Entrada Salida y es el único módulo dependiente del medio ambiente (hardware) en que corre el sistema.

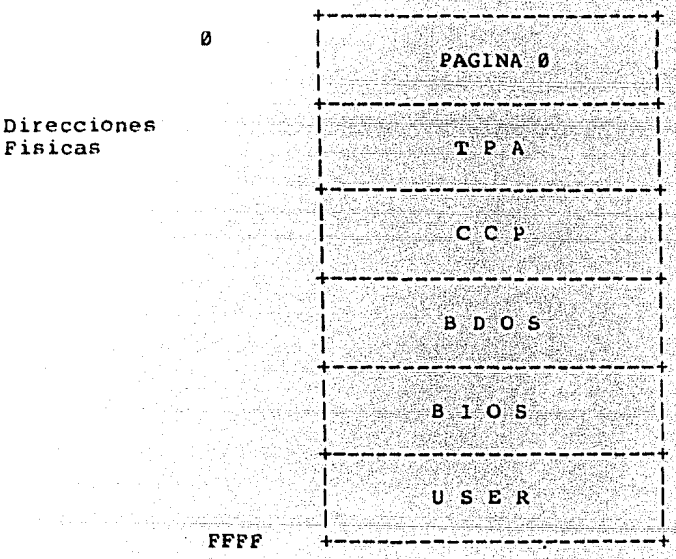
BDOS

(Basic Disk Operating System) Es el sistema de manejo basico del disco y es independiente de la configuración del hardware. Todas las funciones para las operaciones en disco se encuentran aqui.

CCP

(Console Command Processor) Es el Procesador de comandos de la consola y es el módulo que permite la interacción del usuario con la máquina.

Un esquema de como distribuye CP/M la memoria central es:



CARACTERISTICAS DE TRANSPORTABILIDAD Y MODIFICACION

CP/M debe gran parte de su popularidad a características que lo hacen realmente atractivo, como el hecho de necesitar solo 12k para correr, y la facilidad para reconfigurarlo a un nuevo tamaño de memoria. Ninguna de las partes lógicas en que esta dividido esta forzosamente ligada a localidades físicas en particular, sino que ocupan posiciones relativas que tienen como base la parte alta de memoria.

En caso de que un programa del usuario requiera de un espacio mayor al que proporciona el sistema para este efecto, se puede traslapar sobre el área ocupada por el CCP. Naturalmente en caso de que ésto sucediera ,es responsabilidad del usuario no hacer llamadas al CCP antes de haberlo cargado nuevamente, por medio de un "warm boot".

Para manejar a los diversos dispositivos con sus diferentes capacidades, CP/M maneja todo por medio de tablas que contienen los parámetros necesarios para controlarlos. Cada vez que se adapta el sistema a un nuevo hardware, se llenan las tablas con los parámetros adecuados y esto permite a CP/M un trato uniforme de los dispositivos.

El hacer modificaciones al BDOS y al CCP resulta complicado, debido a que para empezar esta escrito en lenguaje ensamblador de 8080 y por razones de optimización de espacio, el codigo es sumamente compacto y es compartido por el mayor numero de rutinas posibles, lo que lo hace difícil de comprender y bastante obscuro.

TPA

Transient Program Area. Es un área reservada para que corran los programas del usuario (programas transientes). Es posible que al estar ejecutando un programa, se ocupen localidades asignadas al CCP, por lo que al terminar de procesarse debe hacerse un "warm boot", es decir, traer nuevamente el CCP del disco.Es posible que un programa invada incluso la zona del BDOS y mas aún, del BIOS, pero en ese caso el programador es responsable de efectuar un brinco a la localidad 0 al terminar la ejecución de su programa, para que el sistema sea cargado nuevamente.Ademas es obvio que no se deben hacer llamadas del BDOS en este caso, porque ya no estarán en sus localidades asignadas en las tablas.

BDOS

Basic Disk Operating System.Es la parte que contiene las funciones de manejo de disco y pueden dividirse en tres grupos:

a) Manejo de disco.

b) Manejo de Dispositivos.

c) Manejo de Tablas del Sistema.

d) Varios.

Además, para que el BDOS funcione independientemente del tamaño de la memoria del sistema, maneja el acceso a las funciones por medio de saltos a la localidad 5H y de ahí se identifica el número de función que desea ejecutarse ya que este número es pasado como parámetro generalmente en los registros DE, y el resultado de la función es regresado generalmente en el registro A.

A continuación daremos una breve descripción de algunas de las funciones de BDOS.

Función 0:

SYSTEM RESET

La rutina SYSTEM RESET regresa el control al CCP. Este reinicializa subsistema de disco,seleccionando el drive A como drive de default

Parámetros de Entrada: Registro C .- 00H

Función 1:

CONSOLE INPUT

La rutina CONSOLE INPUT lee un caracter de la consola,lo pone en el registro "A". A los caracteres imprimibles se les da eco, asi como al <CR>, <LF>, <back-space> y <TAB>.

Parámetros de Entrada: Registro C .- 01H

Valores que regresa: Registro A .- Caracter ASCII

Función 2:

CONSOLE OUTPUT

La rutina CONSOLE OUTPUT escribe el caracter que estaba en el registro E. Es similar a la función CONIN

Parámetros de Entrada: Registro C .- 02H
Registro E .- Caracter ASCII

Función 3:

READER INPUT

La función READER INPUT lee el siguiente caracter de la lectora especificada en el IOBYTE. El control no regresa hasta que el caracter haya sido leído

Parámetros de Entrada: Registro C .- 03H

Valores que regresa: Registro A .- Caracter ASCII

Función 4:

PUNCH OUTPUT

La función PUNCH OUTPUT manda el caracter del registro E a la perforadora

Parámetros de Entrada: Registro C .- 04H
Registro E .- Caracter ASCII

Función 5:

LIST OUTPUT

Esta rutina manda el caracter que se encuentra en el registro E al dispositivo lógico para listar

Parámetros de Entrada: Registro C .- 05H
Registro E .- Caracter ASCII

Función 10:

READ CONSOLE BUFFER

Esta rutina lee una linea y la deposita en un buffer especial donde se guardan los caracteres leídos de la consola (BUFFCON). El primer campo contine el tamaño del buffer, el segundo campo, el total de caracteres leídos.

Parámetros de Entrada: Registro C .- 0AH
Registro DE.- Dirección del Buffer

Valores que regresa: Caracteres en la consola

Función 15:

OPENFILE

Esta rutina abre un arcnivo que ya este creado, la dirección del FCB esta apuntada por DE

Parámetros de Entrada: Registro C .- ØFH
Registro DE.- Direccion del FCB

Valores que regresa: Registro A .- Código de Directorio

Función 2Ø:

READ SEQUENTIAL

Esta funcion hace la lectura secuencial de un archivo direccionado por DE y que debio haber sido abierto con OPEN o MAKE, los registros se van colocando en el DMA

Parámetros de Entrada: Registro C .-14H
Registro DE.- Direccion del FCB

Valores que regresa: Registro A .- Código de Directorio

Función 21:

WRITE SEQUENTIAL

Esta función es la que ejecuta la escritura secuencial en un arcnivo cuyo FCB esta apuntado por DE y que debio haber sido abierto con OPEN o MAKE

Parámetros de Entrada: Registro C .- 15H
Registro DE.- Dirreccion del FCB

Valores que regresa: Registro A .- Código de Directorio

Función 22:

MAKE FILE

Sirve para crear un arcnivo. Si ya existe regresa el valor "FF" en el registro A, si no existe regresa Ø y crea el FCB correspondiente

Parámetros de Entrada: Registro C .- 16H
Registro DE.- Direccion del FCB

Valores que regresa: Registro A .- Código del Directorio

BIOS

Basic I/O System. Aquí se encuentran las rutinas de entrada salida básicas para accesar los manejadores (drives) del disco, y establecer la interface con los dispositivos periféricos. Hace distinción entre los dispositivos, dependiendo del tipo de entrada-salida que realizan, como por ejemplo los discos que hacen acceso por bloques o la consola que lo hace por caracteres. Para estandarizar la entrada salida de tipo caracter,se definen cuatro dispositivos lógicos y a cada uno de estos se les puede asignar hasta cuatro dispositivos fisicos uno a la vez.Esta información se encuentra en el IOBYTE y esto da una gran ventaja al usuario, ya que le permite redireccionar la entrada salida.

Además de la tabla con los parámetros del los dispositivos, BIOS tiene un Bit Map dela asignación de memoria cuya información es mantenida por BDOS e indica el espacio ocupado y disponible para cada dispositivo.

USER

Es un área reservada al usuario para que efectue ahí las modificaciones necesarias a las rutinas del BIOS, por ejemplo, si necesita modificar CONIN (que lee el siguiente caracter de la consola y lo deposita en el registro A),se hace la nueva rutina y se coloca en esta área y después en las tablas del sietema, se le indica que la rutina está en la nueva direccion.

CCP Console Command Processor

El CCP es el procesador de comandos de la consola y es el que permite la interacción de la máquina y el usuario.Cuenta con dos tipos de comandos:

a) Comandos Intrínsecos.

b) Comandos Transientes.

Una de las ventajas que presenta es que permite hacer referencia a grupos de archivos mediante el uso de "*" y "?" en el nombre de los archivos.Todos los archivos, sin distinción, se identifican por medio de un FCB lo que da uniformidad al sistema.

Otra ventaja es que se le pueden definir archivos de comandos indirectos. Resulta muy interesante la forma en que ejecuta los programas del usuario ya que primero checa si es un comando del CCP y de no ser asi, busca el identificador de archivo que recibió, entre los archivos ejecutables y si lo encuentra, lo ejecuta inmediatamente.

A partir de la version 2.0 se permite la edición de líneas de comandos, y el conjunto básico de estos son:

^H Back Space. Recorre el cursor un lugar hacia atras.

^C Da un "re-boot" al sistema, es decir, lo carga nuevamente.

^R Reescribe la linea de comandos como esta actualmente.

^E Concatena las líneas de comandos,es decir,es como dar <LF> pero el comando no se transmite hasta dar <CR>.

^M Equivale a <CR>.

^J Equivale a <LF>.

^X Borra la linea completa.

^S Detiene la salida de un archivo por el dispositivo en tanto no se de otro ^S.

^P Hace que la salida sea mandada a la impresora en tanto no se de otro ^P.

Nota: La señal "^" significa apretar la tecla de "control" simultaneamente con la tecla deseada.

ESPECIFICACIONES DE ARCHIVOS

Un archivo en CP/M queda especificado mediante los siguientes campos:

Disco : Nomarch.ext

Disco — Esta parte es opcional, en caso de no especificarse, se considera el disco activo como el disco de default,es decir el disco sobre el cual se efectuan las operaciones.

Nomarch — Es una cadena de caracteres ascii hasta de 9 caracteres.

Extensión — Es una cadena de caracteres ascii, de hasta tres caracteres.

Si en alguno de estos casos se utiliza un "*", se consideran todas las posibilidades dentro de ese campo.

Los comandos intrinsecos del CCP son:

a) ERA

b) DIR

c) REN

d) SAVE

e) TYPE

ERA (Erase)

Por medio de este comando, le es permitido borrar al usuario archivos del disco qque se encuentre activo en ese momento.Su formato general es:

ERA Nomarcn.ext <cr>

DIR (Directory)

Este comando despliega los nombres de los archivois especificados o bien, si sólo se da DIR y <cr>, se listan todos los archivos del disco activo.Su formato es:

DIR Nomarcn.ext <cr>

REN (Rename)

Este comando permite al usuario cambiar el nombre de los archivos en el disco activo.Su formato es:

REN Nomnuevo.ext = Nomviejo.ext <cr>

SAVE

Con este comando es posible salvar o guardar n páginas de 256 bytes cada una; estas páginas se encuentran en memoria, denro del área de programas transientes (TPA) y lo hace a partir de la dirección 100H.Su formato es:

SAVE Nomarch.ext

TYPE

Este comando despliega un archivo de caracteres ascii por la consola. Su formato es:

TYPE Nomarch.ext

NOTA: En el caso de los comandos ERA y DIR es valido dejar el archivo sin especificar completamente, es decir, se permite el uso de identificadores de archivos con "*" en alguno de sus campos.

COMANDOS TRANSIENTES

Este otro tipo de comandos son proporcionados por CP/M como parte de las utilerías del sistema y son básicamente rutinas que nos dan estadísticas acerca estatus del disco (STAT) , o bien proporciona un programa de intercambio periférico (PIP) , un editor de textos (ED), un ensamblador (ASM) o un programa de depuración (DEBUGG). etc

MANEJO DE ARCHIVOS

El manejo de archivos se hace en CP/M por medio de una tabla de FCB´s que es la tabla de descripcion de archivos.Cada archivo abierto tiene uno o varios FCB´s asociado a el ya que cada uno de estos descriptores corresponde a una entrada en el directorio . El hecho de que a un solo archivo se le asignen varios FCB´s, esta en función directa del tamaño que ocupa.

La longitud de un FCB depende de si estta en disco o en memoria ya que en disco tiene una longitud de 33 bytes y en memoria es de 36 bytes (para el manejo de accceso aleatorio). En general su formato es el siguiente:

```
+-----------------------------------------------------------------+
FCB |dr|n|o|m|b|a|r|c|h|e|x|t|ex|s1|s2|rc|d1|d2| ... |d15|cr|r0|r1|r2|
+-----------------------------------------------------------------+
```

dónde cada campo significa:

dr Es el drive en el que se encuentran los archivos y tiene un valor entre 0 y 16

0 es para el drive de default.
1 es para el drive A.
:
:
16 para el drive P.

nomarch Es el nombre del archivo en caracteres ascii,mayúsculas y co el bit mas significativo apagado.

ext Es la extensión del archivo,en caracteres ascii, mayúsculas y con el bit mas significativo apagado. sirve ademas para determinar el satatus del archivo:

Dependiendo de si esta prendido o no el MSB de cada uno de los caracteres que forman la extension:

- e Archivo de sólo lectura
- x Archivo del sistema, no se lista en el directorio.
- t No se usa.

ex Es el número correspondiente a la extensión que se esta accesando, como ya se mencionó, es posible que un sólo archivo tenga varios FCB dependiendo de su longitud. Esta información es necesaria para saber manejar los bloques del archivo.

s1, s2 Son reservados para el sistema.Tienen 0 al iniciar.

rc Es el Record Count o Contador de Registros y es el que nos dice el número de registros que tiene ocupados un archivo dentro de ese FCB.Su rango de valores es entre 0 y 128.

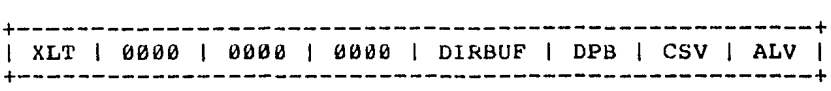
d0...d15 Es el Block Allocation Map y tiene una longitud de 16 bytes e indica el número de bloques que ocupa un archivo determinado. Dependiendo del valor del extent y del extent mask es como se toman los bloques para hacer el mapeo.

cr Es el Current Record o Registro Actual y nos indica el registro sobre el cuál se van a efectuar las operaciones de lectura y escritura.

r0,r1,r2 Se utilizan para el acceso random.

TABLAS DE PARAMETROS DEL DISCO.

De manera general, cada drive del disco tiene asociado un encabezado con los paramentros del disco (Disk Parameter Header DPH) y proporciona un area auxiliar para las operaciones del sistema. El DPH tiene una longitud de 16 BYTES y su formato es el siguiente:



DONDE CADA CAMPO TIENE UNA LONGITUD DE 16 BITS.

XLT Es la dirección del vector de traslación lógica a física de los sectores de ese drive particular.Si el orden lógico o físico es el mismo, en esta localidad se encuentra un 0000H . Los discos con iguales características , comparten la misma tabla de traducción.

0000 Valores auxiliares que se usan dentro del BDOS. De hacho su valor inicial no es relevante.

DIRBUF Dirección de un área auxiliar de 128 bytes para las operaciones de directorio dentro del BDOS. Este apuntador es el mismo para todos los DPH's.

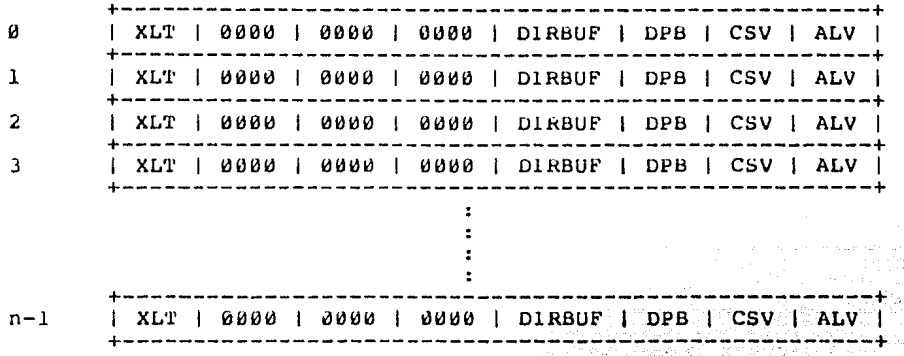
DPB Es el apuntador al Disk Parameter Block, que es la tabla que contiene los parámetros para ese drive particular . LOS DRIVES CON IDENTICAS CARACTERISTICAS, APUNTAN AL MISMO BLOQUE DE PARAMETROS.

CSV (Checksum vector) Apuntador a un area auxiliar utilizada para detectar cuando se efectua un cambio de disco. Cuando el sistema transfiere un sector, se le calcula su checksum con lo que se obtiene un entero, y cada vez que se transfiere un sector, se calcula este número nuevamente, y si no coinciden el sistema detecta un acmbio de disco.

ALV Es un apuntador a un area auxiliar utilizada por el BDOS para almacenar la información del espacio asignado en disco. Es diferente para cada DPH.

Si se cuenta con n drives, los DPH se acomodan en una matriz, cuyo primer renglon de 16 bytes corresponde al drive 0 y asi sucesivamente nasta el drive n-1.

DPB:



donde la etiqueta DPBASE define la dirección base de ésta tabla de DPH's.

DISK PARAMETER BLOCK

El Bloque de Parametros de Disco es más complejo.Como ya mencionamos, un mismo DPB puede estar apuntado desde varios DPH y su formato general es:

SPT	BSH	BLM	EXM	DSM	DRM	AL0	AL1	CKS	OFF

SPT — Es el número total de sectores por track.

BSH — Es el factor de corrimientos de asignación de datos (Block Snift Factor) y esta determinado por el taman~o del bloque.

BLM — Es el número de sectores por bloque -1.

EXM — Es la máscara de la extensión (Extent Mask) determinada por el taman~o del bloque y la capacidad fisica del disco.

$$EXM = \begin{cases} \dfrac{BLS}{2^{10}} - 1 & \text{si DSM} > 255 \\ \dfrac{BLS}{2^{11}} - 1 & \text{si DSM} < 256 \end{cases}$$

pudiéndose obtener por tanto , la siguiente tabla:

BLS	DSM < 256	DSM > 255
1024	0	NO SE PERMITE
2048	1	0
4096	3	1
8192	7	3
16384	15	7

DSM — Determina la capacidad total de almacenamiento en disco.

DRM — Es el número de entradas del directorio (FCB's) ocupadas en el disco.

CKS — Longitud del vector para checar el directorio.

OFF — Numero de tracks reservados para el sistema, al principio logico del disco.

Los valores de BSH y BLM determinan implícitamente el tamaño del bloque BLS y que no es una entrada propiamente dicha dentro de la tabla de paramentros. Existen varias relaciones entre los diferentes parametros y todas son usadas por el sistema al efecturase las funciones del Bdos.

(1) $$BLM = 2^{BSH} - 1$$

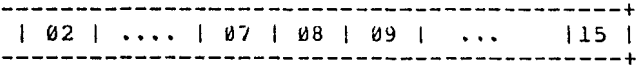
(2) $$BLS = (BLM + 1) * 2^{7}$$

De donde podriamos obtener la siguiente tabla:

BSH	BLM	------->	BLS
3	7		1024 = 2 ** 10
4	15		2048 = 2 ** 11
5	31		4096 = 2 ** 12
6	63		8192 = 2 ** 13
7	127		16384 = 2 ** 14

El valor de DSM es el número total de bloques con que cuenta un disco en particular, medido en unidades BLS. El producto de BLS por (DSM + 1) proporciona el numero total de bytes en el disco, y por supuesto estos parámetros deben ser tales que no excedan la capacidad fisica del dispositivo.

Los valores de AL0 y AL1 estan determinados por el DRM.



Cada bit denota un bloque del directorio y si por ejemplo se necesitan 4 bloques para el directorio, se deben encender los primeros 4 bits de ALØ que son los de bajo orden etc.

Cada FCB, es decir cada entrada del directorio requiere de 32 bytes en disco (porque en memoria ocupa 35) para guardar la información por lo tanto, tenemos las siguientes relaciones:

DRM = NUMERO TOTAL DE ENTRADAS AL DIRECTORIO = # FCB'S

DRM * 32 = # TOTAL DE BYTES EN EL DISCO.

$$\frac{DRM * 32}{BLS} = \text{\# TOTAL DE BLOQUES RESERVADOS PARA EL DIRECTORIO.}$$

$$BLS = \text{\# DE FCB'S POR BLOQUE}$$

$$\frac{}{32}$$

El valor del CKS se determina de la sguiente manera:Si el disco es removible el CKS = (DRM + 1) / 4 y si el disco es fijo, el CKS = Ø es decir, los registros del directorio no se cnecan en este caso.

Es importante subrayar el necho de que varios DPH's pueden apuntar al mismo DPB siempre y cuando sus características sean idénticas.Mas aún, el DPB puede ser alterado dinámicamente cuando un drive nuevo es accesado, cambiando simplemente su apuntador respectivo en el DPH, ya que el BDOS copia su valor en un area local cada vez que se ejecuta una función de SELDSK (Select Disk).

TENEX

TENEX es un sistema operativo que presenta varias características interesantes de diseño desde diferentes puntos de vista.La MAQUINA VIRTUAL que emula, proporciona Espacio de Direccionamiento Virtual con pagineo, asi como mecanismos de protección y facilidades para compartir recursos.

Podemos mencionar como sus características más relevantes, las siguientes:

1) MULTIPROCESAMIENTO con facilidades de comunicación.

2) El Sistema de Archivos tiene estructura de directorio simbólico de multiniveles, con sistemas de protección y acceso consistentes a los dispositivos de entrada/salida, lo que proporciona una mayor homogeneidad en las funciones del sistema.

3) Proporciona un conjunto extendido de instrucciones.

Una característica relevante de TENEX es su capacidad para almacenar y recuperar el medio ambiente de un proceso en ejecucion por medio de pseudointerrupciones que se controlan por medio de "^c" para suspender y el comando CONTINUE para seguir la ejecución a partir del punto donde se quedó. La longitud de palabra que usa es de 36 bits y el mecanismo de protección de directorios es muy parecido al que usa UNIX.

Una limitación grande de TENEX es que fué escrito en ensamblador de la PDP-10 lo que lo hace poco portable hacia otras máquinas ademas de dificultar su comprensión.

Este sistema posee un Interprete del Lenguaje de comandos, llamado EXEC y es por medio de el que se lleva a cabo la interaccion usuario-maquina. El EXEC esta pensado para dar grandes facilidades al usuario para usar el sistema, ya que cuenta con una gran variedad de comandos y estos pueden especificarse de varias formas, como explicare mas adelante.

En cuanto a su DESARROLLO, MANTENIMIENTO y MODIFICACION, TENEX posee software modular, lo que facilita las correcciones o agragados que deseen hacerse Cuenta con medios que permiten detección de errores y modificaciones en tiempo real (Real Time Debbuger) asi como chequeo redundante.
Permite ajustes dinamicos de servicio y tambien hacer reconfiguraciones extensas sisn tener que reensamblar todo el sistema.

ESTRUCTURA DE LA MEMORIA VIRTUAL.

TENEX tiene un espacio de direccionamiento lineal de 256 Kw, divididas en paginas de 512 palabras cada una. Su contenido es especificado en un mapa de 512 entradas o "ranuras" que el usuario puede escribir y leer. Aquie es donde se especifica el tipo de acceso que se va a permitir, ademas de su posición en el espacio de direccionamiento virtual.

Cada una de las entradas puede tener diferente tipo de información como puede ser:

a) Un apuntador a una pagina privada.

b) Apuntadores indirectos, a este u otros procesos.

c) Apuntador a una página de un archivo, en el sistema de archivos.

DISPOSITIVO DE PAGINEO

Para el manejo de direccionamiento virtual, se cuenta con un PAGINADOR que es una interface entre el procesador PDP-10 y el bus de memoria.

Proporciona un mapeo (relocalización) de cada página, que tiene 512 palabras cada una, que pertenezca ya sea al usuario o al espacio de direcciones delmonitor, usando mapas separados para cada caso.

El paginador utiliza REGISTROS ASOCIATIVOS y tablas de memoria para almacenar la información del mapeo.En cada petición de memoria del procesador, los 9 bits de alto orden de la dirección y el nivel de la petición (lectura, escritura, ejecución), se comparan paralelamente con el contenido de cada registro asociativo.

Si checa con alguno,ese registro contiene los 11 bits de alto orden de la dirección, que permite direccionar nasta 1,048,576 palabras en memoria fisica o sea, 1 megapalabra.

Si ninguno checa, se accesa una TABLA DE PAGINAS de 512 palabras, en memoria física. La manera de seleccionar la palabra dentro de la tabla, esta basada en los 9 bits de alto orden de la dirección especificada. Aquí pueden suceder varios casos:

Caso 1) Es una Página Privada EN MEMORIA.

En éste caso, los 11 bits de la dirección y la protección, etstan en esa palabra y son cargados automáticamente en un registro asociativo, por el paginador.

Caso 2) La Página NO ESTA EN MEMORIA, tiene Protección a ese tipo de acceso o simplemente no existe.

En el caso de que la página que desea acesarse no está en memoria, ocurre un error y se interrumpe el proceso nasta que la página sea traída a memoria. Si la página es una página compartida, el mapa contiene un "apuntador compartido" a una tabla del sistema que contiene información de la localización de la página; si la página pertenece a otro proceso el mapa tendra un "apuntador indirecto" a un elemento de otro mapa, el cual contiene la información.

Las maneras en que repercute dentro del funcionamiento del sistema son:

a) Permite compartir programas, tanto código como datos.

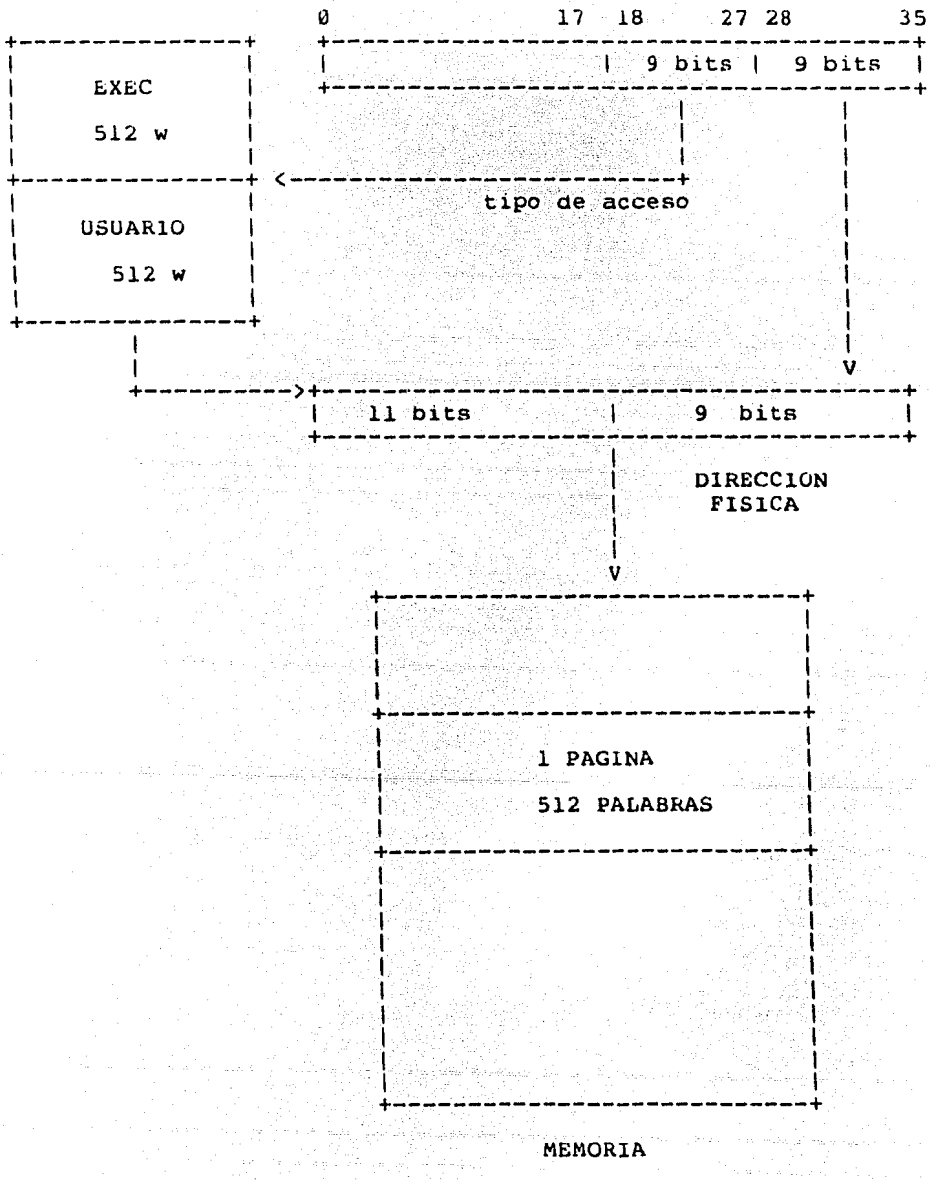
b) Páginas activamente compartidas, i.e representadas en más de un espacio de direccionamiento y lo mas importante, ocupando el mismo espacio en memoria.

c) Se tienen tablas de control y estructuras simples, manejadas sin mucna burocracia.

d) Para la comunicación entre procesos se cuenta con páginas compartidas para operaciones de lectura y escritura.

Se facilita al usuario el compartir grandes porciones del espacio de direcciones y obtener copia sólo de aquellas páginas que son diferentes. Con este objeto, se maneja un BIT DE STATUS por pagina, accesible al usuario, que en caso de querer hacer nacer una escritura, el sistema le crea automáticamente una copia privada para el.

Una característica muy especial, es que el paginador llava el registro de la actividad de cada página en memoria en una TABLA DE STATUS y sabe cuando una página ha sido referenciada, que proceso la ha usado y si se le ha escrito o no a la pagina. Eta información es usada por el manejador de memoria para el reemplazo de páginas.


0
17
18
27
28
35
9 bits
9 bits
EXEC
512 w
USUARIO
512 w
tipo de acceso
11 bits
9 bits
DIRECCION
FISICA
1 PAGINA
512 PALABRAS
MEMORIA

Facilita el codigo reentrante porque maneja las direcciones de regreso en una localidad aparte dentro del vector de transferencias. Existen dos contextos: USUARIO y MONITOR; JSYS actua en ambos contextos y una de las modificaciones que se hicieron al procesador fue AGREGAR UN BIT AL PSW (Process Status Word) con efecto de detectar en cual de los dos medios ambientes se llevo a cabo la llamada.

ESTRUCTURA DE LOS JOBS

Un JOB es un conjunto de uno o más procesos, relacionados jerárquicamente y que tienen los siguientes atributos:

1) Nombre del usuario.

2) Cuenta.

3) Archivos abiertos.

4) Una jerarquía de procesos activos o suspendidos.

puede además tener una o más terminales u otro tipo de dispositivos asociados o conectados (attached). Como una observacion mencionaremos que en TENEX los procesos se denominan FORKS.

JERARQUIA DE PROCESOS.

TENEX permite tener multiples procesos corriendo simultaneamente. La relación entre ellos se define como una estructura de árbol donde cada proceso tiene EXACTAMENTE un predecesor o padre y este puede tener uno o mas hijos. Dos procesos corren en paralelo si su padre es el mismo.

Existen varias formas en que los procesos pueden comunicarse:

1) Compartiendo Memoria.

2) Por Control Directo

3) Usando Pseudo Interrupciones.

Entre las ventajas del multiproceso podemos mencionar el hecho de que EXEC es capaz de correr los programas del usuario, atienda los errores y las paticiones de interrupción, y permita tener un programa depurador a salvo de errores del programa que se esta probando.

EL EXEC

El EXEC es el programa EJECUTIVO de TENEX y es el intérprete de comandos que sirve como medio de comunicación entre el usuario y el sistema. Es un programa reentrante, compartido, que corre en modo usuario, usualmente en el tope de la estructura de los procesos. Es fácil de aprender y fácil de usar ya que da facilidad de completar comandos por medio del caracter <esc>.

Hay tres formas de especificar comandos:

a) Completo.

b) Abreviado.

c) Completarlo con <esc>

SISTEMA DE PSEUDO INTERRUPCIONES.

TENEX maneja un sistema de pseudo interrupciones que son equivalentes a interrupciones simuladas por software osea, son interrupciones a la maquina virtual.

Esto proporciona facilidad para que un proceso reciba señales asíncronas de otros procesos, para fines de comunicación. La manera de provocar una pseudointerrupcion es apretar el caracter específico en el teclado (^c).

Las pseudointerrupciones pueden usrase para detectar condiciones no usuales como:

1) Referencias ilegales a memoria.

2) Condiciones en el procesador como overflow etc.

3) Fin de archivo, Errores en Datos etc.

SISTEMA DE ARCHIVOS

TENEX proporciona un manejo de nombres simbolicos para archivos. Esto se maneja primero, traduciendo el nombre simbolico a un apuntador interno al Bloque Descriptor de Archivos asociado a ese nombre. En segundo lugar se deben checar cosas como el STATUS (si existe ese archivo, que protección tiene, que modos de acceso permite etc), y despues checar si un proceso particular tiene derecho a accesarlo en ese modo.

Los archivos son la unidad fundamental de almacenamiento y el tamaño del byte no es fijo, su rango va desde 1 a 36 bits. NO EXISTE EL CONCEPTO DE REGISTRO.

Los archivos son cadenas de bytes y se pueden accesar byte por byte, secuencialmente o random. Tambien se permite nacer transferencia de cuerdas (multiples bytes).

Las especificaciones de archivos se nacen mediante 5 campos que son:

Dispositivo:<Directorio> nombre.extension;version

y maneja opciones de default en todos los campos y de acuerdo a la operacion.

El directorio tiene una estructura jerarquica de dos niveles y el mecanismo de proteccion funciona tanto a nivel de archivo o de directorio.El formato de la palabra de protección es :

Dueño	Grupo	Otros

Cada campo tiene una longitud de 6 bits y el significado para cada campo es:

bit 0 Lectura

bit 1 Escritura

bit 2 Ejecución

bit 3 Ampliación

bit 4 Por Pagina

bit 5 No se usa

EL DESPACHADOR

Los objetivos del despacnador son entre otros, el proporcionar una distribucion justa del servicio de CPU; Hace ademas una identificación de los procesos interactivos para darles una atención rapida ya que esta es una de las políticas que sigue TENEX: dar una rápida respuesta al usuario. Se trata de tener un uso efeciente de la memoria central para maximizar el uso del CPU.

Maneja un denominado CONJUNTO BALANCEADO (Balset) que usa el principio del Conjunto de Trabajo (working set) dado por Denning. Se nace ademas un monitoreo periódico de todos los procesos listos para correr, con objeto de detectar cambios en las prioridades y en el tamaño del conjunto de trabajo para ajustar el BALSET.

El algoritmo de reemplazo que se utiliza es el de LRU (Least Recently Used) y la asignación de prioridades maneja tres colas de procesos listos para correr. Dentro de cada cola se maneja el algoritmo de Round Robin.

****

CONCLUSIONES

Ahora que hemos visto las características y objetivos generales de los sistemas operativos cabe preguntarnos hacia adonde van el desarrollo de este tipo de sistemas, es decir, si en el futuro seguirán siendo validos los conceptos y definiciones manejados hasta ahora.

Es conveniente cuestionar cosas como: Es posible que a nivel del usuario se manejen las rutinas básicas de un Sistema Operativo o es necesaria la intervención de un operador que se encargue de "optimizar" los recursos? Más aún, seguirán siendo los mismos objetivos los que se consideren básicos para ser cubiertos por un sistema operativo convencional?

Lo que podemos mencionar es que existe una tendencia a desarrollar sistemas que sean más faciles de manejar desde el punto de vista del usuario , así como poseer un alto grado de portabilidad.

Es válido pensar que los sistemas de multiprogramación serán cosa del pasado y que serán desplazados por microcomputadoras que ofrezcan "medios ambientes" para desarrollar diferentes funciones como escritura de texto, almacenamiento de datos etc.

El concepto de medio ambiente no es todavía muy claro pero algunos sistemas que empiezan a utilizar esta nueva filisofía podrían dar un poco de luz sobre la materia. Como ejemplos podemos citar a VisiOn que es desarrollado por la Visicorp, San José California o LISA que es de la Apple Corporation.

A manera de resumen mencionaremos las principales aportaciones de cada uno de los sistemas analizados aquí, así como sus ventajas y desventajas.

UNIX

APORTACIONES.- La entrada/salida se hace independiente de las características de los dispositivos y hace la equivalencia entre archivos y dispositivos.El uso de los "pipes" permite desarrollar en forma independiente programas que luego pueden ser instalados en el sistema sin perder funcionalidad y sin tener que crear archivos intermedios. Además, gracias a la estrecha relación entre UNIX y el lenguaje C, este se desarrolló grandemente, lo que constituye una aportación importante al campo de los sistemas operativos, ya que C proporciona las ventajas de un lenguaje de bajo nivel (accesar bits, corrimientos etc.) con la comodidad y facilidades de un lenguaje de alto nivel.

VENTAJAS.- Una ventaja a mi modo de ver es el manejo que hace del directorio, la forma en que es capaz de anexar directorios enteros como una "hoja" más dentro del árbol me parece realmente útil, porque permite una gran flexibilidad así como ahorro de espacio en un momento dado. Además ahorra espacio porque en vez de duplicar archivos, hace referencia a ellos sin necesidad de crear copias nuevas.No se presentan cerraduras de la muerte, porque en cada momento solo se asigna un recurso a cada proceso lo cual hace disminuir el trabajo extra en el sistema. En mi opinión es un sistema sumamente accesible desde el punto de vista del usuario, pero por lo mismo, como no pone muchas restricciones puede resultar peligroso en algún momento.

DESVENTAJAS.- No permite "pipes" entre procesos que no tengan un ancestro común, aunque incluye comunicación entre procesos. El uso de memoria cache, implica que se tenga que hacer una purga periódica de los buffers de salida por lo que resulta fácil perder información en caso de que se cayera la máquina. No maneja semáforos porque los diseñadores consideraron suficiente tener "pipes" y comunicación por medio de ancestros comunes. A todo esto podemos considerar como desventaja el caracter críptico de los comandos que si bien es cierto que una vez aprendidos resultan más o menos fáciles de usar, representan al principio un medio hostil al usuario.

CP/M

APORTACIONES.- Su principal contribución consiste en la separación lógica en modulos independientes que redundan en que este sistema sea sumamente portable y homogéneo por el manejo de tablas con parámetros para el control de dispositivos, o el manejo de los archivos.

VENTAJAS.- La idea de mantener tanto el BDOS como el CCP lo más independientes de la máquina que sea posible, es lo que en gran parte hace tan popular a CP/M , ya qu solo es necesario escribir el BIOS y lo demás se queda como esta. Además permite hacer un mapeo fácil entre los dispositivos lógicos y físicos. Una característica de este sistema que puede considerarse lo mismo como ventaja que como desventaja, es el hecho de que el usuario puede en un momento dado, accesar las tablas de información que maneja el BDOS lo cual puede resultar peligroso en el caso de un usuario poco experimentado.

DESVENTAJAS.- Todos los dispositivos son residentes en BIOS se usen o no, por lo que nace falta un mecanismo que permita reconfigurarlos. El hecho de que en algún momento no se encuentre presente el CCP y que el sistema no sea capaz de detectarlo y avisar al usuario, me parece poco adecuado; yo propondría que al requerir el usuario un comando del CCP y este no estuviera presente a causa de un traslape, el sistema debería ser capaz de cargarlo nuevamente sin que el usuario tuviera que enterarse siquiera.

TENEX

APORTACIONES.- Así como UNIX favoreció el desarrollo de C, TENEX impulsó el desarrollo de otro lenguaje bastante poderoso llamado SAIL que es muy similar en características a C y desde el cual es fácil hacer llamadas al sistema, o utilizar el leguaje ensamblador de la PDP-10 (macro).Permite compilaciones condicionales permitiendo con esto el desarrollo de útiles nerramientas para el usuario. Además de esto, TENEX permite al usuario desarrollarse en un medio ambiente de trabajo bastante agradable, ya que le facilita el manejo y aprendizaje de los comandos y el tiempo de respuesta es siempre lo mas rápido posible.Fue diseñado pensando siempre en darle al usuario el máximo de facilidades y comodidad.

VENTAJAS.- Como ya mencioné, la principal ventaja es la serie de facilidades que proporciona al usuario, así como la eficiente política de scneduling que tiene gracias a la cual y conjuntamente con el cálculo dinámico de prioridades, es posible la velocidad de respuesta excelente y una justa distribución de los recursos entre los jobs. Por otra parte, permite tener una estrustura jerárquica de procesos y tener corriendo varios de ellos y da facilidades para suspender su ejecución o continuarla a partir de donde fue interrumpido.

DESVENTAJAS.- Podría tener un manejo de archivos más eficiente, ya que éste no es jerárquico.El tamaño de la palabra no es muy común, lo que respresenta un problema en caso de querer transportarlo a otra máquina. El esquema de paginación bajo solicitud causa problemas al sistemas por una sobrecarga de trabajo y no está totalemnte solucionado por TENEX.

A continuación esta una tabla que incluye todas las posibles facilidades de los sistemas como se consideran actualmente, y estan marcadas con una "x" aquellas funciones que son proporcionadas por cada sistema en particular.

ASPECTO Y TEMA	CP/M	UNIX	TENEX
1. manejo de procesos			
-concurrencia		x	x
-deadlocks		x	x
-multiproceso		x	x
-multiprogramación		x	x
-exclusión mutua		x	x
-despacho ("scheduling")		x	x
-control de tráfico		x	x
-sincronización		x	x
-modos de ejecución(superv/usuario)		x	x
2. manejo de memoria			
-estrategias de asignación/desasignación		x	x
-distribución de memoria principal	x	x	x
-memorias distribuidas			
-dispositivos de almacenamiento secundario	x	x	x
-segmentación/paginación		x	x
-jerarquías de almacenamiento			
-"swapping"	overlay	x	x
-memoria virtual			x
3. manejo del sistema de archivos			
-métodos de acceso	x	x	x
-estructuras de directorio	x	x	x
-organización de archivos	x	x	x
-sistemas de archivos distribuidos		x	
-mantenimiento	x	x	x
4. manejo de comunicaciones			
-"buffering"		x	x
-e/s	x	x	x
-envío de mensajes		x	x
-comunicación con redes	x	x	x
-manejo de terminales	x	x	x
-manejo de interrupciones		x	x
-interfase al operador/usuario	x	x	x
5. confiabilidad			
-procedimientos de respaldo		x	x
-verificación	x	x	x
-manejo de errores	x	x	x
6. protección			
-autenticación	x	x	x
-controles criptográficos			
-control de flujo de información	x	x	x

7. generación del sistema			
-generación inicial	x	x	x
-configuración/reconfiguración	x	x	x
-"coldboot / warmboot"	x	x	x
8. contabilización de recursos		x	x

Uno de los principales problemas que enfrenta el usuario hoy día, es la gran variedad de software que se maneja para algunos sistemas, por lo que resulta incompatible en el caso de algunas aplicaciones. UNIX es hoy por hoy uno de los sistemas más populares y resulta sorprendente ver la gran cantidad de sistemas que han basado su diseño en él.Ejemplos de estos sistemas son XENIX,IDRIS,ONIX, etc.

XENIX es una derivación de UNIX diseñada por Microsoft que es una compañía que ha demostrado interés en diseñar sistemas consistentes.Tiene además, el sistema MS-DOS que es similar al CP/M (otro de los sistemas operativos de moda) .MS-DOS esta pensado para correr en un microprocesador 8088 de un solo usuario y que pretende ser un punto intermedio entre los sistemas monousuario y sistemas más complejos ya que la idea de Microsoft es facilitar al usuario la transición entre un sistema y otro.

Las principales ventajas que presenta MS-DOS sobre anteriores versiones son:

a) Se proporciona un manejador de terminales, lo que apoya la transportabilidad del software de gráficas.

b) Permite instalar manejadores de dispositivos dentro del "boot" del sistema lo que permite dar de alta dispositivos sin tener que generar todo el sistema.

c) Permite redireccionar la Entrada/Salida, filtros y canales de comunicación.

d) Selección dinámica de sectores en disco, lo que permite incrementar la velocidad de las operaciones en disco.

e) Presenta una estructuta jerárquica de archivos como lo maneja XENIX.

f) Permite efectuar tareas "background", es decir permite la ejecución de tareas que se llevan cabo durante las interrupciones al procesados como el manejo de correo electrónico o el manejo de spooling.

Sin embargo, MS-DOS presenta algunas desventajas:

a) La serie de comandos que debe manejar el usuario, son muy grandes y complicados, lo que impide en un momento dado que se exploten al máximo las ventajas que ofrece.

b) En cuanto a los dispositivos que pueden agragarse dinámicamente, Microsoft no proporciona las rutinas necesarias para hacerlo y deben ser diseñadas por el usuario.

XENIX , como ya mencionamos, es un derivado de UNIX y en la version 3.0 combina los beneficios de un sistema desarrollado para funcionar en un ambiente científico, con un sistema práctico para negocios.

El sistema consta de tres puntos:

1) Una porción de tiempo compartido.

2) El desarrollo del sistema.

3) El sistema de procesamiento de texto.

Como una ventaja adicional a UNIX proporciona posibilidad de compartir áreas de datos mediante una nueva llamada del sistema.

A pesar de que UNIX es muy popular, muchos esgrimen razones en su contra y éstas son de muy diversa índole. Una de las principales es la naturaleza hostilque presenta al usuario porque maneja un gran número de comandos y todos son muy crípticos.

Otra desventaja que presenta es carencia de elementos para aplicaciones de elementos para aplicaciones en tiempo real y las acciones que toma con los arcnivos en una situacion de crisis.

Además, UNIX no tiene un standard por lo que resulta difícil que el sottware sea completamente portable.

Masscomp Inc. es una de las compañías que, viendo alguna de las deficiencias, tratan de sacar provecho de esto y desarrollan sistemas que lo superan en algunos aspectos.

Masscomp es un sistema basado en dos procesadores 68000 lo que permite incrementar la velocidad en la captura de datos. La mayoría de las modificaciones se nicieron sobre la entrada salida y las tareas del despachador del sistema.

Este tipo de modificaciones son por ejemplo, el proporcionar un conjunto de prioridades mayor que el de UNIX de tal forma que les sea garantizado un quantum de procesador sin aceptar interrupciones.

La versión de Masscomp proporciona una asignación contigua de los archivos en disco lo que repercute en una mayor velocidad de las operaciones que sobre él se realizan, así como un tiempo de respuesta menor.

Por último las interrupciones por software (traps) se nacen asíncronas. La Entrada/Salida se maneja asíncronamente y cocmo resultado se puede nacer manejo de disco y captura de datos simultaneamente.

Los procesadores más usados para soportar las versiones de UNIX son:

1) el 68000 con el 80 %

2) el 8086 con el 13 %

3) el z8000 con el 7 %

Cabe recordar además, que aunque el hardware de algunas máquinas sea el mismo, lo que hace la diferencia en su modo de operacion es el sistema operativo que las controla.

En el presente trabajo se han examinado las funciones que deben realizar los sistemas, para optimizar el uso de los recursos y como habrá podido observarse, mucnas funciones de chequeo llevarían demasiado tiempo si se quisieran ejecutar por software.Es por esta razón que algunos fabricantes intentan desarrollar por hardware (sobre todo para aplicaciones en tiempo real) partes del sistema con suficiente inteligencia para manejar el procesador o un monitor con varios puntos de entrada. Es ya algo común ver que el ejecutivo o núcleo, se encuentra grabado junto con rutinas escritas por el usuario en un solo chip de ROM.

IDRIS es otro sistema operativo derivado de UNIX que fue desarrollado por Whitesmitn LTD. su presidente, Plaugert opina que su objetivo no es competir con UNIX sino coexistir con él dentro del mercado, en máquinas con poca capacidad .

Existe otro sistema operativo, llamado COHERENT que está escrito en lenguaje C y proporciona mucho software para diseño y desarrollo de nuevos sistemas. Es compatible con UNIX tanto a nivel de código (ya que está escrito en C) como a nivel decomandos del usuario (shell). COHERENT proporciona soporte para lenguajes de alto nivel, como C, Pascal estandard y Basic. Esto le da cierta ventaja sobre sistemas como PICK que es derivado de UNIX y que aunque es más fácil de manejar que XENIX, solo soporta BASIC, lo que limita en mucho sus aplicaciones.

A continuación se muestra una tabla con algunos de los sistemas operativos más usados para micros de 16 bits. Por qué es que nos interesa mencionarlos? porque en lo personal creo que la tendencia es que cada usuario tenga su propia máquina y que los usuarios se comuniquen entre sí por medio de redes de computadoras, por lo que el concepto de sistema operativo tendrá que evolucionar para cubrir las nuevas necesidades.

Sistema Operativo	Procesador que lo soporta	# de usuarios
CP/M	8086,8088	1
MP/M	8086,8088	16
CP/M concurrente	8086,8088	1
XENIX	8086,68000,Z8000	depende del hardware
UNIX	68000,16032	-
IDRIS	68000	-
COHERENT	68000,Z8000,8088	-
MS-DOS	8086,8088	1

Como podemos ver, existe una amplia variedad de sistemas que podrían utilizarse en los diferentes equipos de cómputo, y la decisión de qué sistema es el que nos conviene usar en un momento dado, depende completamente del tipo de aplicación de que se trate. Posiblemente el decidirnos por CP/M o por otro sistema para el cual podamos conseguir bastante software desarrollado, cuando la aplicación que tenemos en mente es bastante específica, por ejemplo un sistema de control en tiempo real, sería mejor utilizar un sistema menos "universal" y usar otro mas específico.

****

GLOSARIO

El siguiente es un glosario de los términos mas usados dentro de la presente tesis. La idea es proporcionar a los lectores una fuente de información rápida y condensada donde pueda consultar en un momento dado los conceptos mas comunes dentro de los Sistemas Operativos.

COMPUTADORAS Y SISTEMAS OPERATIVOS.

a) DATOS .- Fenómeno fisico escogido por convencion para representar ciertos aspectos y conceptos del mundo real. El significado asociado a los datos se llama INFORMACION. Los datos se utilizan para transmitir informacion entre los humanos, para almacenar la informacion para uso futuro o para generar nueva informacion en base a la ejecucion de reglas formales (operaciones).

b) OPERACION .- Es una regla para generar un conjunto finito de datos llamados SALIDA, a partir de otro conjunto de datos llamado ENTRADA. Una vez iniciada una operacion, esta concluye en un periodo finito de tiempo. Una operacion siempre genera la misma salida, cuando actua sobre el mismo conjunto de datos de entrada, independientemente del tiempo que le lleve su ejecucion.

c) CALCULO .- (Computation) Conjunto finito de operaciones que se aplican sobre un conjunto de datos para resolver un problema especifico. Si un calculo resuelve un problema, es llamadao tambien ALGORITMO.

d) IMAGEN .- Es un ambiente de ejecucion que incluye la imagen en memoria, Registros (tanto de proposito general como Program Counter, Stack Pointer etc), los arcnivos abiertos y el directorio actual.

e) PROCESO .- Es la ejecucion de una imagen en memoria.Cuando un proceso se esta ejecutando, las imagenes de los otros procesos estan estaticas, ya sea en disco o en memoria, dependiendo de su tamaño.

f) PROGRAMA .- Es la descripcion de un algoritmo en donde las operaciones estas expresadas en un LENGUAJE DE ALTO NIVEL.

g) COMPUTADORA .- Es un sistema fisico capaz de ejecutar algoritmos por medio de la interpretación de programas.

h) MAQUINA VIRTUAL .- Es una computadora simulada en parte por programa, y en general resulta más fácil de manejar y mas poderosa que la computadora sobre la que fue desarrollada.

i) SISTEMA OPERATIVO .- Conjunto de procesos que permiten a un usuario o un grupo de usuarios, compartir la computadora eficientemente .Entre sus principales funciones destacan: proporcionar un medio de almacenamiento prolongado, protección contra los otros usuarios, y llevar la contabilidad de los recursos utilizados por cada usuario. En el caso de los sistemas para un solo usuario, se pretende que el sistema de facilidades para el manejo de la máquina y proporcione multiples funciones que ayuden al usuario a sacar el máximo provecno de su equipo.

j) SISTEMA INTERACTIVO .- Es un sistema que permite por medio de la retroalimentacion, modificar las operaciones del sistema.

PROCESOS CONCURRENTES

a) PROCESOS CONCURRENTES .- Son procesos que se traslapan en el tiempo. Procesos concurrentes se llaman DISJUNTOS si cada uno de ellos nace referncia sólo a datos privados y se llaman INTERACTIVOS si nacen referencia a datos en común.

b) MULTIPROGRAMACION .- Son las técnicas de programación usadas para controlar los procesos concurrentes.

c) REGION CRITICA .- Es un conjunto de operaciones dentro de una estructura de datos común que se excluye una aotra en el tiempo.

d) SINCRONIZACION .- Es un término general para denotar a cualquier restricción que se imponga al orden en que las operaciones deban de ser ejecutadas. Una regla de sincronización puede por ejemplo, especificar la precedencia, prioridad o exclusión mutua de las operaciones en el tiempo.

e) SEMAFORO .- Es un entero no negativo el cual una vez asignado un valor inicial, sólo puede ser modificado por las operaciones de WAIT y SIGNAL y es utilizado para intercambiar señales de tiempo entre procesos concurrentes. El valor de un semáforo esta relacionado con su valor inicial.

f) DEADLOCK .- Es una situación en donde dos o más procesos estan esperando indefinidamente por eventos que no van a ocurrir nunca.

DESPACHADOR

a) JOB .- Es un conjunto de uno o más procesos que tienen los siguientes atributos:

Nombre del Usuario.
Cuenta.
Archivos abiertos.
Una estructura de procesos abiertos o suspendidos.

b) RECURSO .- Termino general para cualquier objeto (procesador, memoria, datos, programas etc.) compartidos por los procesos. Existen recursos COMPARTIBLES si dos o más procesos pueden actuar sobre ellos al mismo tiempo o recursos NO COMPARTIBLES si solo un procesos puede usarlo a un tiempo.

c) ALGORITMO DEL DESPACHADOR .- Es el algoritmo que determina el orden en que serán ejecutados los procesos y como es que se les asignarán los recursos.

d) PROMEDIO DE LLEGADAS.- Número promedio de requerimentos de jobs por unidad de tiempo.

e) PROMEDIO DE SERVICIO .- Número promedio de jobs terminados por unidad de tiempo.

f) TIEMPO DE ESPERA .- Tiempo que transcurre hasta que un job es atendido.

g) TIEMPO DE RESPUESTA .- Es el intervalo de tiempo que transcurre desde que un proceso hace su petición de ser atendido hasta que termina su ejecución y regresa los resultados al usuario. El tiempo de respuesta es la suma del tiempo de espera y el tiempo de servicio.

h) PRIORIDAD .- Es un número usado para establecer un orden en la precedencia entre los procesos que compiten por recursos. Las prioridades puden ser fijas o bien pueden ser modificadas dinámicamente .

i)"TIME SLICE".- Es un intervalo de tiempo durante el cual un proceso utiliza un recurso sin que otro pueda arrebatárselo.

j)"ROUND ROBIN".- Asignacion cíclica de recursos entre varios procesos, con rebanadas de tiempo (time slice) fijas.

MANEJO DEL PROCESADOR Y ALMACENAMIENTO

a) DESPACHADOR A CORTO PLAZO .- Es la parte del algoritmo del despachador que asigna el procesador y el área de almacenamiento a los procesos tan pronto como se encuentren disponibles y permitan una utilización eficiente de la computadora. Es a éste nivel de programación donde se implementan las rutinas de sincronización que son las que permiten la interacción de los procesos.

b) DESPACHADOR DE MEDIANO PLAZO .- Es la parte del algoritmo del despachador que inicia o termina un proceso, de acuerdo con la política de utilización de la computadora para los usuarios. Este nivel de programación establece la identidad y prioridades de cada usuario, analiza sus peticiones y lleva la cuenta de los recursos consumidos.

c) INTERRUPCION .- Es una señal de tiempo que causa que el procesador suspenda la ejecución del proceso actual e identifique el origen de la señal para determinar la acción a seguir.

d) LOCALIDAD DE ALMACENAMIENTO .- Una componenete del área de almacenamiento que puede representar cualquier conjunto finito de datos.

e) TIEMPO DE ACCESO .- Promedio de tiempo requerido para obtener o registrar el valor de una localidad de almacenamiento. En el caso de cintas, tambores y discos, el tiempo de acceso es el tiempo de latencia, más el el búsqueda y transferencia que consume pasar o depositar información a una localidad de almacenamiento.

f) ALMECENAMIENTO INTERNO .- Es un área de almacenamiento de capacidad moderada y Rapido Acceso, usado para guardar datos o programas durante la ejecución.

g) BACKING STORE .- Area de almacenamiento de gran capacidad y Lento Acceso para guardar datos y programas hasta que necesiten ser transferidos al almacenamiento interno.

n) DIRECCION .- Número que se usa para identificar una localidad de almacenamiento. Una dirección REAL es UNICA dentro del area de almacenamiento entera, mientras que una dirección virtual es única sólo dentro de una parte de esa area. La relación entre las direcciones virtuales y las direcciones fisicas esta dada por el Mapeo de Direcciones.

i) ASIGNACION DE AREAS DE ALMACENAMIENTO .- La asignación de las localidades de memoria a datos y programas puede hacerse de diferentes maneras, como por ejemplo:

a) En Tiempo de Compilación (Asignación Fija)

b) Antes de la Ejecución (Asignación Dinamica)

c) Durante la Ejecución (Relocalización Dinamica)

j) CONJUNTO DE TRABAJO (WORKING SET) .- Es la cantidad mínima de almacenamiento interno que se nesecita por un proceso para poder ejecutarse.

k) ALIMENTACION POR DEMANDA .- Una forma de almacenamiento multiplexado en la cual los segmentos se pasan a almacenamiento secundario y son transferidos a almacenamiento interno sólo cuando se hace referncia a ellos.

l)"TRASHING".- Es el tiempo que pierde el Procesador mientras se efectua la transferencia de segmentos de almacenamiento principal a secundario.

DOCUMENTACION DEL EMULADOR DE CPM

La siguiente documentación tiene por objeto facilitar el manejo del emulador CPM.SAV que es un programa que al ser corrido manda el prompt de el drive de default (A.) y espera le sean dados comandos del CCP o bien el nombre de algún programa cuya extension sea .SAV y por tanto pueda ser ejecutado.

El Sistema CP/M esta dividido lógicamente en cuatro partes,que son: el BIOS (Basic I/O System), el BDOS (Basic Disk Operating System), el CCP (Console Command Procesor) y finalmente el TPA (Transient Program Area) Éste emulador consta de varias partes que son:

BDOS .- Conjunto de rutinas del BDOS de CPM.

CCP .- Conjunto de comandos ejecutables del CCP de CPM (Esta implementación cuenta con comandos extra como BYE para terminar la sesion o CARGA...)

TPA .- La parte de los Transientes es un simulador de la Intel 8080. y los programas a ser ejecutados deben ser cargados de la dirección 100 de memoria en adelante.

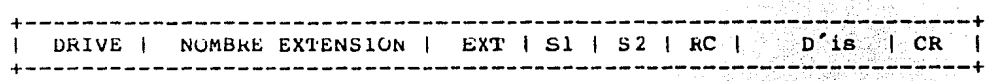
Rutinas de Utilería.- Rutinas auxiliares para el emulador.

El emulador es ejecutado al teclear el comando CPM y entonces este manda el prompt en señal de que esta listo para recibir comandos.Estos comandos pueden tomar varias formas dependiendo de si son funciones como DIR o TYPE o son nombres de programas transientes.En el primer caso,el comando es ejecutado de forma inmediata en tanto que en el segundo,el CCP busca en el disco actual un archivo con ese nombre y que pueda ser ejecutado.Si el archivo es encontrado se supone esta cargado en memoria en la parte de TPA y es ejecutado inmediatamente.

En esta implementación se cuenta con dos drives (A y B) Existe un archivo llamado "DRIVE.A" y otro llamado "DRIVE.B" para cada uno de ellos.Dentro de cada drive se encuentra la información del DPB y del DPH.Cada drive cuenta con 96 entradas (FCB's) para el manejo del directorio.

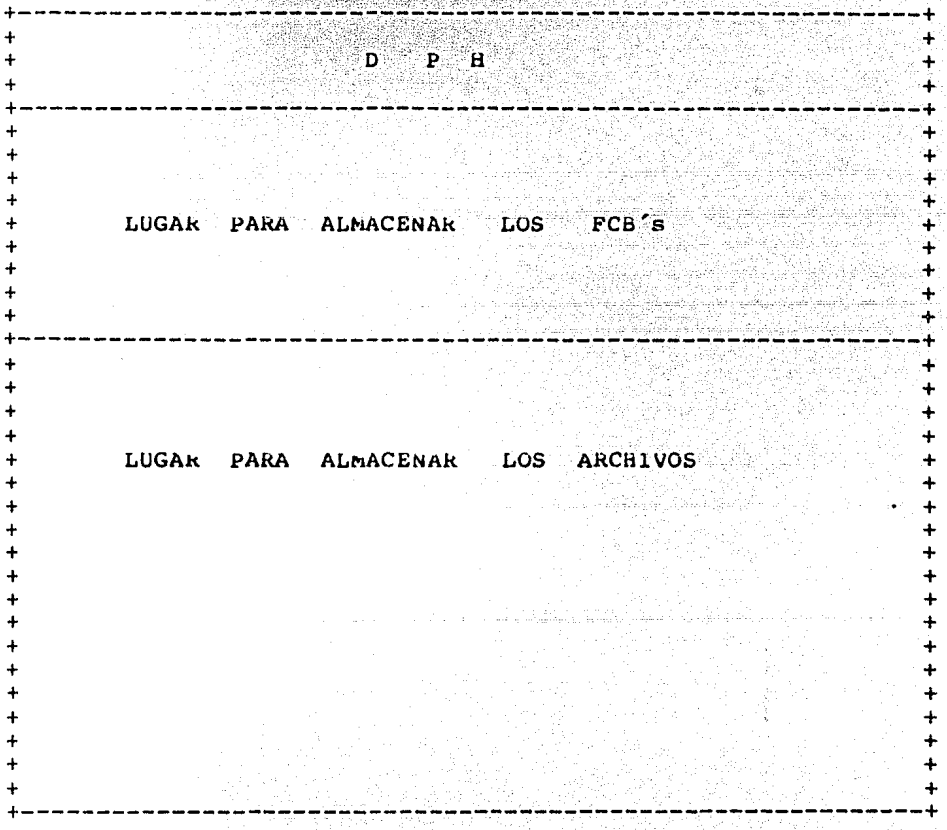
Cada File Control Block consiste de 33 bytes para acceso secuencial o bien de 36 bytes para acceso random.

La estructura general de un FCB es:



Cada drive cuenta con 256 bloques par almacenar información.

La estructura general de cada drive es commo sigue:



BIBLIOGRAFIA

1.- Fundamentals of Operating Systems.
A.M. Lister

2.- Operating Systems Principles.
Brinch Hansen
Prentice Hall

3.- Operating Systems.
H. Lorin H.M. Deitel
Addison wesley

4.- Introduction to Operating Systems.
A.N. Habermann
Science Research Associates, Inc.

5.- Operating Systems.
Madnick Donovan
McGraw Hill

6.- Operating Systems.
D. Tsichritzis
Academic Press.

7.- The Bell System Technical Journal.
"Unix Time Sharing System"
Julio-Agosto 1978
Vol 57, numero 6 parte 2.
pags. 1897 - 2021

8.- Modularizacion y Jerarquias en familias de Sistemas Operativos.
A.N. Habermann, Lawrence,Cooprider.
Communications of the ACM
Mayo 1976, vol 19 # 15.

9.- Medio Ambiente de Programación de UNIX.
Kernighan, Masney
Computer IEEE Computer Society
Abril 1981, vol 14 # 4.

10.- Seminario de Microprocesadores CP/M Microcomputadoras.
G. Becerril, M. Diaz, J.C. Lopez, A. Medina-Mora,
A. Outon.
Agosto 1983. (Notas)

BIBLIOGRAFIA

1.- Fundamentals of Operating Systems.
A.M. Lister

2.- Operating Systems Principles.
Brinch Hansen
Prentice Hall

3.- Operating Systems.
H. Lorin H.M. Deitel
Addison Wesley

4.- Introduction to Operating Systems.
A.N. Habermann
Science Research Associates, Inc.

5.- Operating Systems.
Madnick Donovan
McGraw Hill

6.- Operating Systems.
D. Tsichritzis
Academic Press.

7.- The Bell System Technical Journal.
"Unix Time Sharing System"
Julio-Agosto 1978
Vol 57, numero 6 parte 2.
pags. 1897 - 2021

8.- Modularizacion y Jerarquias en familias de Sistemas Operativos.
A.N. Habermann, Lawrence,Cooprider.
Communications of the ACM
Mayo 1976, vol 19 # 15.

9.- Medio Ambiente de Programación de UNIX.
Kernignan, Masney
Computer IEEE Computer Society
Abril 1981, vol 14 # 4.

10.- Seminario de Microprocesadores CP/M Microcomputadoras.
G. Becerril, M. Diaz, J.C. Lopez, A. Medina-Mora,
A. Outon.
Agosto 1983. (Notas)

11.- CP/M User Guide.
Tom Hogran
Osborne.

12.- Manuales de CP/M.

13.- Manuales de TENEX.

14.- Studies in Operating Systems.
R.M. McKeag, Wilson, Huxtable
Academic Press.

15.- "Bell Lab's UNIX Operating System spreads powerfull new wings".
S. Evanczuck
Electronics
Julio 1983.

16.- "Lisa's Alternative Operating System"
B. Daniels
Computer Design
Agosto 1983.

17.- "The C Language: Key to Portability".
Rifkin, Williams
Computer Design.
Agosto 1983.

18.- Comparación de tres Sistemas Operativos.
A. Saboya V.
Octubre 1983. (Notas)

19.- Estudio Comparativo de Sistemas Operativos.
J. Becerra B.
Octubre 1983. (Notas)

20.- Introducción a Microcumputadoras.
Sloan.
Prentice Hall.