Universidad Nacional Autónoma de México

FACULTAD DE CIENCIAS

UNA LINEA EVOLUTIVA DE LOS SISTEMAS OPERATIVOS

T E S I S

QUE PARA OBTENER EL TITULO DE

M A T E M A T I C O

P R E S E N T A

GABRIELA RAVELO SANCHEZ

MEXICO, D. F. 1990

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

4. SISTEMA OPERATIVO UNIX.

P R O L O G O

El objetivo de este trabajo es presentar una rama evolutiva de los sistemas operativos. Es muy cierto que el análisis de la evolución general de los distintos sistemas operativos es un tema muy interesante por desarrollar, sin embargo para esta tesis, se optó por situarse en un ángulo particular y desde ahí seguir tanto los problemas que se fueron presentando, como las soluciones propuestas a éstos.

En esta exposición se presentan cuatro sistemas operativos seleccionados con base en la importancia de sus características. Los sistemas analizados son: ATLAS, CTSS, MULTICS y UNIX.

La línea evolutiva que se escogió fue la que da origen al sistema operativo UNIX. Aquí surge la pregunta, Por qué UNIX y no otro sistema operativo?. La razón principal por la cuál se seleccionó UNIX es por la popularidad y aceptación creciente que éste sistema ha tenido en los últimos años. Han sido tanto el diseño de UNIX, así como el hecho de ser un sistema operativo escrito en un lenguaje de alto nivel, lo que lo han llevado a convertirse en un sistema operativo estándar de la industria de la computación.

Es preciso mencionar que de los cuatro sistemas que se presentan en este trabajo, el primero de ellos no tiene relación directa con UNIX. Sin embargo, ATLAS fue seleccionado por dos razones: la primera es por ser considerado el primer sistema operativo que operó como tal; la segunda razón es haber introducido el concepto de memoria a un solo nivel, lo que posteriormente dió origen al concepto de memoria virtual.

El segundo sistema operativo que se analiza es CTSS (Compatible Time Sharing System), el cual constituye el primer escalón dentro de la línea de evolución hacia UNIX. Este sistema operativo es contemporáneo del sistema ATLAS y fue uno de los primeros sistemas de Tiempo Compartido (Time Sharing).

El sistema CTSS dió origen al sistema operativo MULTICS (Multiplexed Information and Computing System), quien a su vez es el antecesor de UNIX. Tanto CTSS como MULTICS fueron desarrollados por el mismo grupo de trabajo del MIT (Massachusetts Institute of Technology).

El último de los cuatro sistemas que se presenta es UNIX, resaltando aquí la influencia obtenida de sus antecesores.

El trabajo está distribuido en cuatro capítulos, los cuales se organizan de la siguiente manera: el primer capítulo constituye una introducción a la problemática que existía antes de contar con un sistema operativo. El segundo capítulo se enfoca a la exposición de los sistemas operativos ATLAS y CTSS. El tercer capítulo trata el caso del sistema MULTICS . El cuarto y último capítulo expone el sistema operativo UNIX.

Cabe aquí hacer una aclaración. Debido a que un en principio los componentes de los sistemas operativos eran nombrados de distinta manera, no es posible seguir un análisis sistemático de las características de los mismos dentro del trabajo. Sin embargo, para poder facilitar el seguimiento de la exposición, se denominó con el mismo nombre a los componentes del sistema, aunque entre paréntesis se denota el nombre utilizado por sus autores.

1. INTRODUCCION

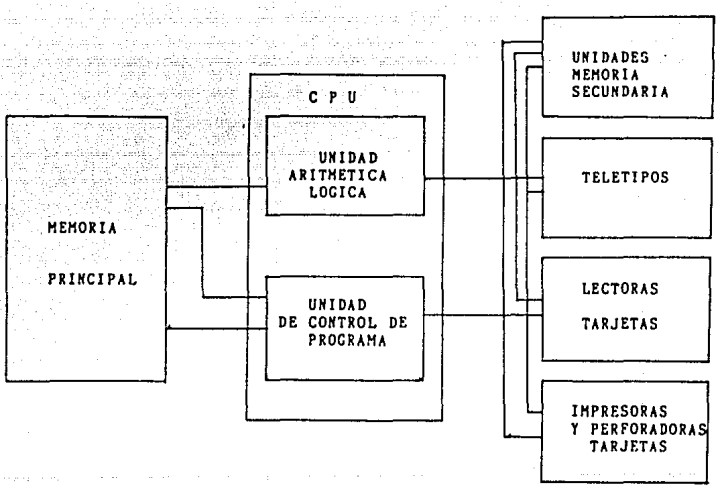
El objetivo de este trabajo, como ya se mencionó, es presentar la evolución de los sistemas operativos. No se pretende abarcar las distintas ramas evolutivas, sino situarse en un ángulo particular y desde ahí seguir tanto los problemas que fueron presentandose así como las soluciones propuestas a éstos.

Aquí surge la pregunta, Qué es y para qué sirve un sistema operativo?. En forma general, puede decirse que un sistema operativo tiene como objetivo transformar el "hardware" en una máquina dócil y tratable. Un sistema operativo es un conjunto de programas relacionados que se encargan de manejar y administrar el "hardware", controlan la ejecución de los programas usuarios y proporcionan los servicios que requieren dichos programas para hacer uso de la máquina.

En los inicios del procesamiento electrónico de datos no se contaba con sistemas operativos, por lo que el manejo de la información así como la operación misma de la máquina, ocurría de distinta manera de como ocurre ahora.

Para poder situarse mejor en esa época se revisará brevemente la arquitectura típica de las computadoras de aquel tiempo. Muchos autores nombran a estas máquinas como primera generación y es así como se hará referencia en adelante a ellas. La Fig 1.1 muestra la arquitectura de una máquina de primera generación. Sus componentes éran basicamente tres: la CPU, donde se efectuaban las operaciones, la memoria principal y secundaria que permitían almacenar la información, y los dispositivos periféricos para realizar la entrada/salida de los datos.

Fig. 1.1 Arquitectura de una máquina de la primera generación.
1945 - 1954



Entre las características principales de las computadoras de la primera generación están las siguientes:

- Eran computadoras de propósito general, lo que sin duda influyó a generalizar el uso de las máquinas.

- Tenían almacenados en una unidad de memoria tanto programas como datos, brindando de esta forma más facilidades a los programadores.

- La CPU contenía varios registros de alta velocidad usados como localidades de almacenamiento para operandos y resultados.

- Todo el control de la computadora estaba centralizado en la CPU, lo que resultaba poco eficiente y limitaba la velocidad del procesamiento, ya que el órden de ejecución de las instrucciones era rigurosamente secuencial.

- La programación desarrollada como soporte de operación era mínima lo que hacía difícil la ejecución de los programas.

- En un principio sólo se podía programar directamente en lenguaje de máquina y no fue sino hasta mediados de los años cincuenta que se fueron generalizando los lenguajes ensambladores y de alto nivel.

Cada vez que se requería ejecutar un trabajo, el programador se encargaba de operar en forma manual la computadora. Esta persona tenía que verificar direcciones, preparar los dispositivos que utilizaba, cargar paquetes de tarjetas, digitar instrucciones, esperar que su trabajo terminara de ejecutarse, recoger los resultados y por último comenzar nuevamente todas las actividades para poder ejecutar otro trabajo.

La corrección y depuración de programas bajo estas condiciones se hacía lenta y cansada ya que cada prueba requería parar la máquina y preparar el ambiente para la ejecución de los nuevos programas.

Por otra parte la transferencia de datos de entrada/salida era lenta. La velocidad de los periféricos no era comparable con la del procesador y si se toma en cuenta que la CPU intervenía directamente en cada operación del sistema, la transferencia de datos entre los periféricos y la máquina representaba pérdida de tiempo ya que la CPU permanecía dedicada a la operación de entrada/salida y no a la operación de trabajos.

Como respuesta a las limitaciones del "hardware" de éstas máquinas, surgieron innovaciones tecnológicas que dieron a su vez origen a la creación de "software" para manejo y control de los nuevos sistemas. Entre los desarrollos más importantes que se presentaron estan:

- El uso de los registros índice, así como el "hardware" necesario para aritmética de punto flotante. Con esto la máquina ampliaba el rango de cálculo de operaciones y se convertía en una máquina más rápida.

- La introducción de procesadores especiales para supervisar las operaciones de entrada/salida. De esta manera la CPU quedaba liberada de varias tareas que originaban un desperdicio de tiempo. Estos procesadores tenían acceso directo a la memoria principal y facilitaban la transferencia de información de los dispositivos a la memoria

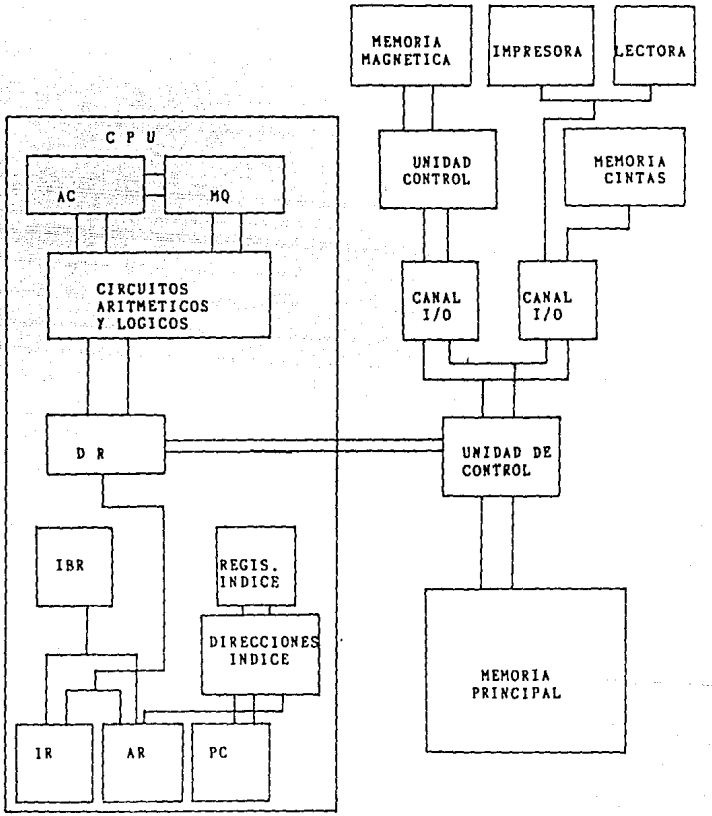
- Se introdujeron circuitos transistorizados, así como discos magnéticos, que revolucionaron la tecnología de las memorias.

A las computadoras desarrolladas con estas nuevas características se les conocen como máquinas de la segunda generación. Fig. 1.2. Como se aprecia en la figura los componentes básicos de estas máquinas siguieron siendo los mismos, pero más sofisticados. La CPU contaba con más registros internos que le permitían mejorar el manejo de las operaciones que realizaba. Las memorias principal y secundaria mantuvieron su función, pero contaban con unidades de control para el intercambio de la información entre ellas. Los periféricos eran más rápidos y apoyaban su función en la utilización de controladores de I/O o canales.

Estas innovaciones ocasionaron que las computadoras aumentaran su velocidad de proceso, pero la operación aún seguía resultando lenta, ya que las máquinas eran más rápidas pero no se poseían todavía herramientas de programación que permitieran agilizar el proceso de carga de los trabajos; esto seguía haciendose en forma manual y estas interrupciones representaban un gran desperdicio de CPU.

Es en este contexto que surgen los primeros programas monitores, los cuales tenían como principal función, incrementar la eficiencia del proceso de operación. Estos monitores permitían al operador humano preparar todos los accesos a discos, cintas, programas de utilería e instrucciones de operación antes de ser necesitados. Los trabajos a ejecutar eran almacenados secuencialmente de manera anticipada en un dispositivo magnético y a la salida del proceso la información resultante se almacenaba también en un dispositivo magnético.

Fig 1.2 Arquitectura de una máquina de la segunda generación 1955 - 1964



A esta forma de proceso se le conoce historicamente como el procesamiento en lotes o "batch" y representó un gran cambio, ya que permitía ejecutar en forma continua los trabajos, aprovechando así mejor la CPU. Los procesamientos "batch" representaban también la ventaja de calendarizar el órden de ejecución. Sin embargo los monitores tenían la desventaja de ocupar bastante espacio de la memoria principal, reduciendo el espacio disponible para la ejecución.

Otra aportación importante que acompañó a estas máquinas de la segunda generación fué el desarrollo de lenguajes de programación de alto nivel, más generales y manejables que los lenguajes ensambladores, los cuales podían ser utilizados mediante la ayuda de un traductor conocido como compilador.

Con la introducción de estos lenguajes, la programación de las tareas resultaba ser mucho más sencilla, así como su depuración y corrección, sin embargo la comunicación del usuario con la computadora seguía siendo deficiente y el desarrollo de aplicaciones más sofisticadas, como lo eran los programas de juegos que empezaban a desarrollarse desde finales de los cincuenta, demostraciones de teoremas u otras tareas inteligentes, se veían frustradas por la falta de herramientas que permitieran establecer un diálogo ágil con el usuario.

Este es el punto de partida de este trabajo. Los avances tecnológicos se siguen dando y las máquinas aumentan cada vez más tanto su capacidad como su rapidez. Esto obligó al desarrollo de herramientas de programación que buscaran sacar el mejor provecho posible de los adelantos en "hardware", y facilitaran la interacción hombre-máquina, dando lugar al desarrollo de lo que hoy conocemos como sistemas operativos.

2. SISTEMAS OPERATIVOS ATLAS Y CTSS

El segundo capitulo de este trabajo contempla la exposición de dos de los primeros sistemas operativos, ATLAS y CTSS.

Se agruparon ambos sistemas en un capitulo porque representan una sola época en la cual, antes de que los dos surgieran como tales, no existian sistemas operativos. Al momento de ser diseñados, éstos sistemas debieron afrontar y resolver el problema que representaba contar con nuevas herramientas de "hardware" sin tener al alcance, en ese momento, posibilidad de aprovecharlas.

Ambos sistemas operativos fueron desarrollados para trabajar en máquinas de la segunda generación. El primero de ellos, ATLAS, es un sistema que centraba su atención en resolver el problema de capacidad de memoria. El segundo, CTSS, proporcionaba la solución al problema del desperdicio de la CPU con solamente un usuario en la máquina.

Ambos sistemas operativos son muy sencillos, sin embargo establecen las bases para el desarrollo posterior de otros sistemas más complejos.

2.1 SISTEMA OPERATIVO ATLAS.

El sistema de cómputo ATLAS fue un proyecto conjunto de la Universidad de Manchester y la compañía Ferranti LTD.

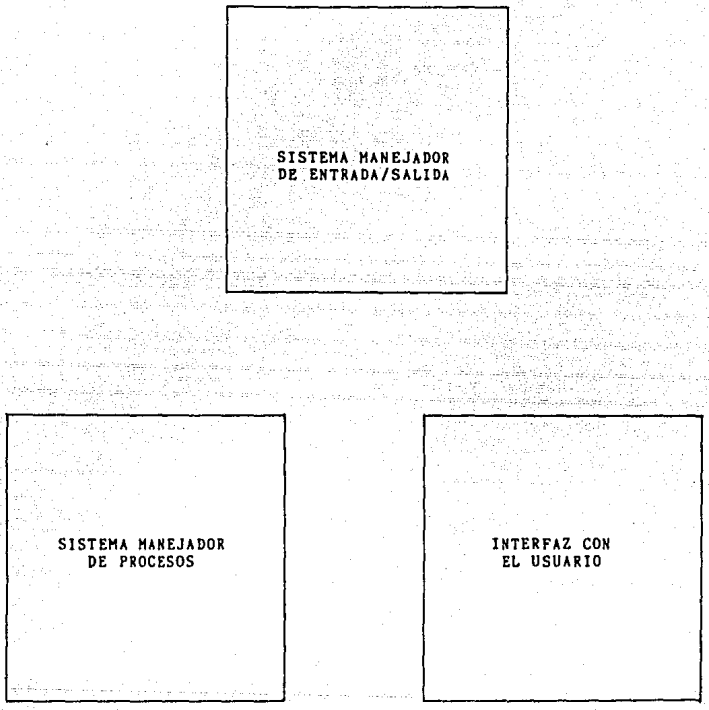
A pricipios del decenio de los cincuenta, la Universidad de Manchester comenzó la construcción de una serie de máquinas que introducían el concepto de memoria a un solo nivel. Esto significaba manejar la memoria primaria y la memoria secundaria como si fueran una sola, lo que daba la impresión de tener una memoria más amplia y con menos restricciones de capacidad. Dentro de las innovaciones de estas máquinas se encontraba un conjunto de registros llamados registros B, ahora conocidos como registros de índice y que se empleaban para acceder a los datos en memoria. La culminación de todo este desarrollo fue ATLAS, una computadora considerada como de la segunda generación.

La Universidad de Manchester desarrolló también un sistema operativo complejo para la computadora ATLAS. Este sistema es considerado por algunos como el primer sistema operativo real, ya que introdujo facilidades y conceptos con los que cuentan hoy los sistemas operativos y que en su momento, elevaron la eficiencia de operación global del equipo. El sistema de cómputo ATLAS comenzó a trabajar como tal en el año de 1961.

ATLAS es un sistema operativo muy sencillo, compuesto básicamente de tres elementos; el sistema manejador de entrada/salida, el sistema manejador de procesos y la interfaz con el usuario. Cada uno de estos elementos se explicará con más detalle a lo largo del capítulo sin embargo para proporcionar una idea general los componentes se esquematizan en la Fig. 2.1.

Para entender mejor al sistema ATLAS, se expondrá primeramente la arquitectura de su máquina.

Fig 2.1 Componentes del sistema ATLAS.



2.1.1 Arquitectura del sistema.

El "hardware" del sistema de cómputo ATLAS estaba formado por los siguientes componentes: el procesador central, una memoria principal, una memoria fija, y almacenamiento en tambor magnético. Los dos últimos eran almacenamiento auxiliar y representaba un elemento importante en la facilidad que proporcionaba el sistema de almacenamiento a un solo nivel, ya que la información se transmitía en forma automática entre la memoria principal y, la memoria fija y del tambor magnético.

La capacidad de la memoria principal era de 32 bloques de 512 palabras cada uno, en donde cada palabra estaba formada por 48 "bits". Esto daba una capacidad de 16,384 palabras.

ATLAS contaba con una gran variedad y cantidad de equipo periférico por medio del cual efectuaba operaciones de entrada/salida al mismo tiempo que se utilizaba como ayuda para el manejo de la información.

La configuración con la que contaba el sistema ATLAS de la Universidad de Manchester era la siguiente:

- Un tambor magnético
- Ocho unidades de cintas magnéticas
- Cuatro lectoras de cintas de papel
- Cuatro perforadoras de cintas de papel
- Una impresora de línea
- Una lectora de tarjetas
- Una perforadora de tarjetas

El sistema estaba diseñado para poder operar con cualquier configuración, sin embargo la ausencia de cintas magnéticas reducía la eficiencia del sistema ya que, como se verá más adelante, las cintas constituían un elemento auxiliar muy importante en el manejo de la información.

La transferencia de información de y hacia la memoria principal estaba bajo el control de los programas contenidos en la memoria fija. La transferencia, en el caso de tambores y cintas magnéticas, se realizaba en bloques de 512 palabras, mientras que hacia los demás equipos periféricos la transmisión se realizaba por medio de un "buffer" de un solo caracter.

Entre las facilidades que introdujo ATLAS estaba la de permitir que mientras el procesador central ejecutaba alguna tarea, los programas en memoria fija controlaran la operación de los equipos periféricos. El procesador se vió liberado así del manejo de ciertas operaciones, lo que permitia que se realizaran procesamientos de múltiples tareas.

En el sistema ATLAS se diseñaron dos áreas de almacenamiento intermedio del tipo que hoy conocemos como "buffers" de entrada/salida. El procesador central tomaba información del "buffer" de entrada, la procesaba y la depositaba en el "buffer" de salida.

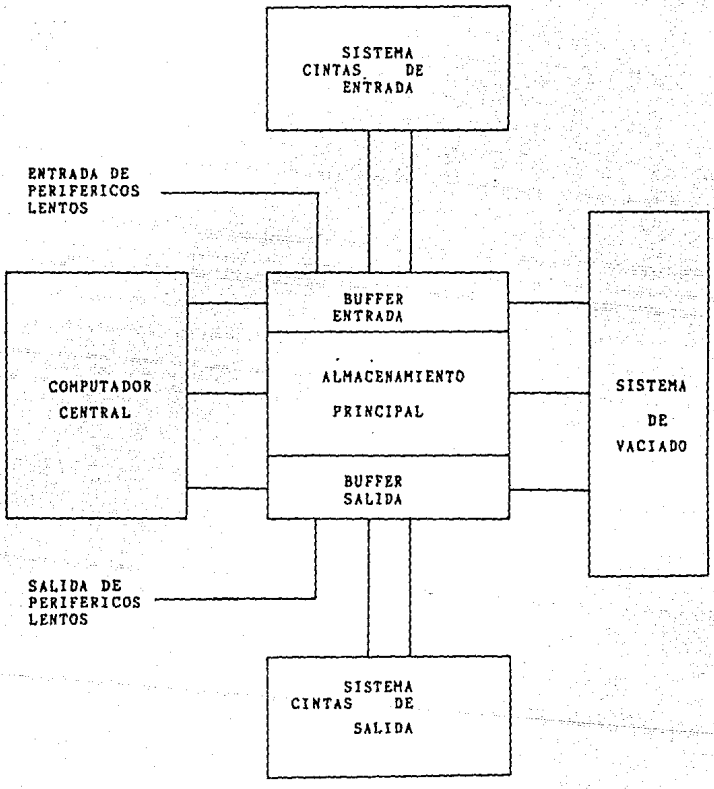
La velocidad a la que la información era procesada resultaba ser mucho más rápida que la velocidad a la que los periféricos manejaban dicha información. Con la utilización de estos "buffers" esta diferencia de velocidades se disminuía al mismo tiempo que se aumentaba la utilización de la unidad de procesamiento central.

La capacidad de los "buffers" se vió aumentada con la utilización de cintas magnéticas. Algunas de estas cintas eran utilizadas para almacenar información de entrada/salida, mientras que otras eran utilizadas como un sistema de vaciado para aquellas tareas que eran suspendidas por el programa supervisor.

Bajo estas condiciones se podía realizar un traslape en cuanto a las operaciones, ya que mientras el procesador central se encontraba atendiendo una tarea, las operaciones de entrada/salida podían ser atendidas sin desviar por esto la atención del procesador central.

La Fig. 2.2 muestra la configuración del sistema.

Fig. 2.2 Configuración del sistema de cómputo ATLAS.



2.1.2 Sistema Manejador de Entrada/Salida

El sistema manejador de Entrada/Salida, ó Sistema de Entrada/Salida como se conocía en ATLAS, estaba formado de tres subsistemas. El sistema de entrada, el sistema de salida y el sistema de cintas magnéticas. Estos tres subsistemas constituían lo que conocemos hoy como manejador de Entrada/Salida. A continuación cada uno se analiza por separado.

Sistema de Entrada

El sistema de entrada de ATLAS era un sistema muy flexible que permitía que todos los periféricos lentos operaran en forma simultánea. Así mismo, este sistema aceptaba que un proceso complejo entrara completo con todos sus componentes para ser procesado.

Antes de seguir adelante es conveniente ver cómo se manejaba el término de proceso. Los diseñadores definieron a un proceso como: "Un conjunto de instrucciones o comandos que debía ejecutar la computadora en un órden específico".

Estas instrucciones especificaban por lo general, tanto la ejecución de programas como también detallaban información con respecto al conjunto de datos que se requería procesar, el espacio que necesitaba en memoria, tipos de dispositivos que utilizaban.

Los componentes de un proceso eran precisamente el conjunto de éstas instrucciones y podían estar agrupados de distintas formas. Era común que al proceso se le conociera con el nombre de "job".

Para poder controlar los procesos, el sistema ATLAS introdujo ciertas instrucciones de control que identificaban plenamente al "job" y sus componentes. A la vez cada componente del "job" estaba etiquetado con lo que se identificaba perfectamente a qué "job" pertenecía.

Este control permitía al programa supervisor organizar la información que entraba al sistema. Para esto el supervisor mantenía tanto listas de nombres o títulos de "jobs" como listas de "jobs" completos e incompletos. Sólo aquellos procesos que poseían todos sus componentes eran habilitados al procesador central.

El sistema operaba de la siguiente forma: recibía de algún periférico el nombre de un "job", el cual anexaba a la lista de "jobs" incompletos y le asignaba un número. Los títulos de los componentes de los "jobs" eran colocados en una lista, así cuando cualquier periférico introducía alguno de estos componentes, el sistema lo marcaba en su lista como presente y lo almacenaba junto con el nombre y número de "job".

Cuando todos los componentes se encontraban presentes el nombre del "job" se transfería a la lista de "jobs" completos. Si el procesador central requería un proceso para ejecución, extraía de la lista de "jobs" completos el nombre de uno así como también vaciaba los bloques apropiados del "buffer" de entrada.

La utilización del sistema de cintas de entrada no era indispensable, el sistema podía operar sin él ya que su utilización era una medida preventiva. El uso de este sistema evitaba que el "buffer" de entrada se saturara. Cuando el sistema de cintas se encontraba en uso, el "buffer" de entrada se dividía en dos partes: la primera de ellas era la encargada de recibir la información de los periféricos lentos y de transferir bloques completos de información al sistema de cintas de entrada, mientras que la segunda parte era la que alojaba "jobs" o procesos completos que eran transferidos al procesador central en cuanto se requerían.

Sistema de Salida

El sistema de salida ofrecía al igual que el sistema de entrada la facilidad de tener varios periféricos operando simultáneamente. Para este fin el "buffer" de salida estaba dividido entre los periféricos de salida en una razón aproximada a las velocidades a las que ellos transmitían la información.

Los problemas que originaba la baja velocidad a la que los periféricos vaciaban el "buffer" de salida eran solucionados en parte por el sistema de cintas de salida. Este sistema operaba en forma similar al sistema de entrada, el "buffer" de salida estaba dividido en dos partes: la primera era llenada por el procesador central y la segunda se llenaba con la información que se recibía del sistema de cintas de salida.

El sistema de cintas de salida estaba dividido en varias secciones, cada sección contenía un bloque para cada perforadora tanto de tarjetas como de cintas de papel y diez bloques para cada impresora de línea. La segunda parte del "buffer" de salida consistía de una o más de estas secciones. Para mayor eficiencia en la utilización de los periféricos se requería que en esta parte sólo se encontraran secciones completas.

El sistema supervisor era el encargado de asignar los canales de salida , transfiriendo solamente secciones completas a la segunda sección del "buffer" de salida. Además el supervisor determinaba en qué momento podía disponer de la primera parte del "buffer" de salida para asignarlo a un periférico en particular.

Sistema de Vaciado de Cintas

Como se mencionó antes este sistema era auxiliar. Los sistemas de entrada y salida se apoyaban en él cuando tenían excesos de información. El sistema de cintas de vaciado almacenaba en forma temporal la información que ya no podía ser manejada por los sistemas de entrada y salida. Adicionalmente, cuando se suspendía la ejecución de un programa, este último era transferido al sistema de vaciado, liberando de esta forma el espacio que ocupaba en memoria principal y permitiendo que otra tarea ocupara dicho espacio.

2.1.3 Sistema Manejador de Procesos ("Scheduling" o Despachador)

El "scheduler" o despachador formaba parte del programa supervisor y realizaba funciones muy semejantes a las que realizan en la actualidad los manejadores de procesos en los sistemas operativos contemporáneos.

El programa despachador en ATLAS tenía la función de determinar el órden en el cual los "jobs" que estaban esperando ejecución debían ser ensamblados en memoria principal, compilados y ejecutados. Para determinar este órden, el programa despachador podía acceder a la información que describía al "job" en espera de ejecución.

Al contar ATLAS con un "scheduler", tanto el nivel de utilización como el rendimiento del sistema de cómputo en general se elevaba, haciendo de ATLAS un sistema novedoso. El programa "scheduler" llevaba control absoluto sobre las ejecuciones a la vez que administraba el equipo periférico.

Es importante señalar que el "scheduler" de ATLAS no hubiera sido tan elegante y sofisticado si el "hardware" del sistema no hubiese contado con características especiales de protección e interrupción de programas en ejecución.

Tarea del "Scheduler"

El sistema operativo ATLAS tenía como principal objetivo mantener actividad el mayor tiempo posible en aquellas partes del sistema de cómputo que podían funcionar en forma simultánea, por esto los períodos de inactividad causados por la transferencia de información, espera de periféricos así como la espera del procesador central, eran eliminados hasta cierto punto.

El "scheduler" se encargaba de transferir los "jobs" desde la lista de "jobs" disponibles hacia la lista de "jobs" en curso de ejecución. El sistema supervisor de ATLAS, permitía compartir el procesador central entre varios programas en curso de ejecución.

La lista de ejecución podía contener más de un "job" . Sin embargo para obtener mayor eficiencia el número de estos "jobs" no debía ser excesivo porque provocaba contratiempos:

- Intercambio de control constante, ya que a pesar de que el supervisor estaba diseñado para realizar intercambios de control rápidos entre un programa objeto y él, al aumentar la cantidad de programas aumentaban también los intercambios que debía realizar el supervisor.

- Utilización inadecuada de gran cantidad de memoria principal, ya que las partes operativas de las tareas debían encontrarse en memoria principal para asegurar un intercambio rápido de controles, sin embargo solo una parte de esa información era utilizada.

- Tener demasiados "jobs" que utilizaban cintas magnéticas disminuía el nivel de eficiencia, ya que se perdía tiempo en lo que el operador montaba las cintas.

El manejo de esta lista de ejecución era delicado y estaba sujeto a varias consideraciones. El "scheduler" era el encargado de proveer trabajos a la lista de ejecución, era activado cuando aparecía una vacante en la lista de ejecución y trataba de arreglar tan rápido como fuera posible la entrada de otro "job" a dicha lista. El "scheduler" se activaba también cuando entraba a través del equipo periférico alguna tarea o cuando algún equipo periférico activo de salida estaba en posición de ser desocupado.

"Streams"

Uno de los problemas que se presentaban era la selección del tipo de tarea o "job" que debía ensamblarse y ejecutarse. El tiempo necesario para realizar la búsqueda del "job" dentro de la lista de disponibles así como la posibilidad de que alguno de los "jobs" nunca fuera escogido para ejecución eran dos limitaciones que debía superar un sistema "scheduler".

El sistema ATLAS introdujo un nuevo concepto, utilizar una serie de grupos o clases de trabajo, a las que los diseñadores de ATLAS bautizarón como "streams". En estas clases se iban colocando los "jobs" conforme entraban a la computadora y de ahí se escogía el "job" próximo a ser habilitado. Con esta innovación se resolvió el problema de la elección de los "jobs".

Para poder llevar a cabo esta clasificación se debían considerar varios factores. La primera división consistía en separar los "jobs" o tareas en tres grupos, las tareas que sólo utilizaban el procesador, las que utilizaban unidades de cintas y aquellas que estaban limitadas por los periféricos.

Se consideraban en forma especial aquellos "jobs" que duraban poco tiempo (un minuto) ya que no era posible determinar de qué tipo eran. También había que considerar aquellos "jobs" que no traían descripción propia y tomaban los valores de "default" del sistema.

No era conveniente que "jobs" que duraban mucho tiempo estuvieran en la misma clase de "jobs" cuya duración era corta, ya que la ejecución de los primeros podía originar acumulación de "jobs" cortos.

Debido a que la salida de los "jobs" ocurre en períodos de tiempo distintos, no era posible relevar un trabajo largo para mantener constante el uso del equipo periférico.

Era aconsejable que los "jobs" que utilizaban periféricos fueran ejecutados en series para asegurar que todos los periféricos tuvieran salida en forma regular.

Para poder llevar un buen control sobre las unidades de cinta, era necesario contar con una rutina que se encargara de este manejo. Era más fácil implementar esta rutina si todos los "jobs" que utilizaban cintas se encontraban en una sola clase de trabajo.

El sistema hacia una segunda clasificación de los "jobs" en tres grupos principalmente y los colocaba en tres clases de trabajo. La primera clase contenia a los "jobs" de corta duración que no utilizaban cintas. Estos "jobs" eran agrupados bajo una nueva clase, la cual correspondia al tipo de periféricos que utilizaban con mayor frecuencia.

La segunda clase contenia a los "jobs" de corta duración que utilizaban cintas, también contenia a todos los "jobs" largos cuyo tiempo de uso de cintas era mayor a su tiempo de cómputo.

La tercera clase contenia a todos los "jobs" largos los cuales tenían limitación en cintas.

El manejo de estos segundos grupos no solo reducia el tiempo empleado en la busqueda, sino también eliminaba la posibilidad de que un "job" no fuera escogido nunca.

Los "jobs" eran clasificados dentro de las clases de trabajo conforme entraban a la computadora. Cuando se requeria un tipo particular de "job" en la lista de ejecución se consultaba la clase correspondiente y se escogía el "job" colocado en primer lugar.

El "scheduler" tenia como responsabilidad consultar las clases de trabajo y decidir que "job" sería habilitado. Para poder tomar decisiones el "scheduler" debia conocer tanto el estado de los "jobs" que se encontraban en ejecución, como el estado de los equipos periféricos.

De esta manera el "scheduler" preparaba para ejecución el "job" que, debido a las circunstancias, era el indicado para correr. Sin embargo, en muchas ocasiones no era posible seguir el órden natural de los "jobs".

Las clases de trabajo contenían dentro de sus entradas información sobre los "jobs", cómo el nombre del "job", referencias a su posición en cinta, tiempo de ejecución, requerimientos de espacio, cantidad y tipo de salida, compilador utilizado, así como alguna prioridad externa. Esta información era almacenada en 120 "bits" y era de gran valor para las rutinas de asignación tanto de espacio como de unidades de cinta.

Para que el "scheduler" cumpliera su función de alimentar de tareas al procesador central en forma continua y, para evitar periodos improductivos, siempre se mantenían dos "jobs" en la lista de ejecución, no importando en dado caso que fueran extraidos de la misma clase.

Ensamble de "jobs"

Una de las tareas que debía realizar el "scheduler" era ensamblar los "jobs" previamente. La información necesaria para el ensamble de los "jobs" se tomaba de la descripción del "job" que realizaba el programador y se registraba dentro de una clase.

El ensamble de los "jobs" se realizaba antes de entrar en la lista de ejecución. El "scheduler" al escogerlos de las clases los depositaba en una lista que contenía "jobs" a ser ensamblados, llamada lista de activos. Cuando se requería un "job" para ejecución solo aquellos que se encontraban en esta lista y cuyo ensamble había terminado eran considerados.

El proceso de ensamble consistía en recolectar toda la información requerida para correr una tarea. Esta información se leía del sistema de cintas de entrada, de cintas privadas, de documentos o de archivos. El proceso también contemplaba cargar el compilador en memoria, así como dar instrucciones al operador para montar las cintas magnéticas y verificar los títulos de todas las cintas.

Una vez que se había leído la información y cargado el compilador, el "job" era incluído en la lista de activos. Es importante resaltar que el ensamble de un "job" podía extenderse por un periodo de tiempo prolongado ya que intervenían varios factores, entre ellos, la intervención del operador así como rutinas de asignación de memoria y asignación de unidades de cinta. Sin embargo, la lógica tanto del "scheduler" como de las rutinas de ensamble tendían a minimizar estas demoras.

Interacción con el operador

El sistema "scheduling" podía ser independiente de cualquier intervención del operador, aun así existían dos formas de comunicación entre el "scheduler" y el operador:

- Cuando se requería la intervención del operador, como en el caso de montar una cinta, el "scheduler" enviaba un mensaje con este requerimiento.

- Si se deseaba alterar la prioridad interna de un "job", el operador se lo hacía saber al "scheduler" mediante un comando.

Para ambos casos el sistema de "scheduling" debía considerar ciertos detalles. En el primer caso, los mensajes de requerimientos eran enviados con la suficiente anticipación como para compensar la velocidad de operación humana. En el segundo caso, el sistema se protegía de una asignación errónea de prioridad por parte del operador, mediante la confirmación de la misma. Esta alteración podía ocasionar choques con el sistema "scheduling", el cual siempre consideraba la mejor acción a seguir según el estado del sistema en esos momentos.

El sistema ATLAS manejaba cuatro tipos de prioridades:

- Prioridad superior.- El "job" era ejecutado y los resultados eran obtenidos tan pronto como fuera posible.

- Prioridad alta.- El "job" era colocado a la cabeza de la clase de trabajo que le correspondía.

- Prioridad normal.- El "job" era ejecutado en el órden natural que le correspondía.

- Prioridad baja.- El "job" era tratado en forma normal hasta que llegaba a la lista de ejecución en donde era colocado en el fondo de la misma.

Rutinas de asignación de cintas

El sistema operativo ATLAS requería normalmente para su operación de tres unidades de cinta, una para cintas de entrada, una para cintas de salida y otra para cintas de vaciado aunque, como se mencionó anteriormente, era posible que el sistema trabajara con menos recursos, pero con un rendimiento bajo.

El "scheduler" afectaba en cierto grado las decisiones que debían tomar las rutinas de asignación de unidades de cinta. De igual manera, cualquier acción tomada por estas rutinas repercutían en las decisiones del "scheduler".

El sistema establecía a nivel general una serie de prioridades para la asignación de unidades de cinta:

- Cintas para el supervisor.
- "Jobs" que utilizaban cintas durante su ejecución.
- "Jobs" que requerían cintas privadas o documentos de los sistemas de cintas, para entrar a la lista de ejecución.
- "Jobs" que se encontraban cerca del tope de sus clases de trabajo y requerían cintas privadas o de los sistemas de cintas.

Es importante señalar que el sistema de cintas de vaciado era un auxiliar muy importante que ayudaba a que el sistema mantuviera un buen nivel de respuesta, ya que no sólo actuaba como una extensión de la memoria principal sino que, además, recibía los errores en monitoreo, programas suspendidos e información que no era posible almacenar en los otros sistemas de cintas.

Rutinas de asignación de espacio

La política de las rutinas de asignación de espacio consistía en formar una jerarquía de usuarios del espacio en memoria.

El sistema consideraba seis grupos de categorías para la asignación:

- Espacio libre
- "Buffers" de entrada/salida
- "Jobs" que estaban preparándose para ejecución
- "Jobs" que estaban en ejecución
- Compiladores
- El supervisor

La asignación de prioridad era en forma dinámica dependiendo del estado de la máquina en ese momento. La mecánica de asignación de espacio podía verse en forma sencilla. Existía un espacio libre que podía ser ocupado por cualquier petición siempre y cuando el espacio fuera suficiente para él. De no ser así, el sistema podía tomar dos acciones: la primera consistía en colocar el requerimiento en una cola de requerimientos de espacio y esperar ahí la liberación del espacio suficiente para él; la segunda consistía en mandar a la rutina que solicitó el espacio un mensaje de que su requerimiento fué rechazado.

Dentro de la cola de requerimientos era importante cuidar la prioridad de dichos requerimientos, ya que no era conveniente que requerimientos con baja prioridad y mucho espacio solicitado bloquearan a requerimientos de alta prioridad. Sin embargo la prioridad de los requerimientos podía ir cambiando en forma proporcional al tiempo que habían permanecido en la cola.

La memoria principal poseía dos tipos de información: una de ellas era la que no poseía copias en cintas magnéticas y la otra era la que poseía un duplicado en cintas y podía en determinado momento liberar el espacio que ocupaba.

Sin embargo, no era fácil llevar un control sobre la información que había sido removida de memoria principal. Para estos fines se implementó un directorio de sistema, que permitía obtener nuevamente ésta información así como también una serie de rutinas.

Estas rutinas se encontraban almacenadas en dos partes: en memoria fija para llevar control de la mayoría de los requerimientos de espacio y en memoria principal en donde contenían la mayoría de las rutinas para realizar el vaciado.

2.1.4 Interfaz con el usuario (Descripción de usuarios)

El sistema de cómputo ATLAS fue desarrollado con varias finalidades. Una de ellas era proporcionar servicio a distintos departamentos de la Universidad de Manchester.

Para garantizar la fluidez y rapidez del trabajo se diseñaron una serie de instrucciones de control. Estas instrucciones de control fueron el antecedente de lo que se conoce como lenguajes de control y constituyen la manera en que el programador se comunica con el sistema operativo. El objetivo principal que tenían estas instrucciones era liberar al operador del mayor trabajo posible, así como ofrecer al usuario un buen nivel de servicio.

Con el uso de las instrucciones de control la intervención humana se minimizaba al mismo tiempo que el programador se veía involucrado en forma directa en el control ya que proporcionaba él mismo las instrucciones a realizar para la ejecución de su tarea.

A continuación, sin pretender entrar en muchos tecnicismos, se expone la descripción de estas instrucciones de control.

Títulos y nombres.

Para el sistema ATLAS un "job" estaba formado de varias secciones de información, cada una precedida por un título que la identificaba. El título de estas secciones de información iba precedido por un nombre que indicaba el tipo de información del que se trataba. Dichos nombres podían ser:

COMPILER (x) x es un lenguaje de programación
DATA
JOB

si tiene lo siguiente

COMPILER FORTRAN
ANALISIS-REGRESION

La información agrupada bajo el titulo de ANALISIS-REGRESION, era un programa fuente en lenguaje FORTRAN. En este caso la información respetaba la sintaxis de FORTRAN, e incluía tanto datos como instrucciones. Aquí cabe mencionar que en esa época no se habían desarrollado muchos lenguajes de programación de alto nivel y FORTRAN era sin duda el más utilizado.

DATA
ANALISIS-REGRESION

La información agrupada bajo el titulo de ANALISIS-REGRESION, consistía en datos que eran leídos por algún programa durante ejecución.

JOB
ANALISIS-REGRESION

La información agrupada bajo el titulo de ANALISIS-REGRESION, constituía la descripción del "job". Es información opcional que podía incluir entre otras cosas:

- cintas de entrada, de programas y de datos
- equipo de salida usado
- cintas magnéticas
- almacenamiento requerido
- tiempo de cómputo

Es importante señalar que la tarjeta de JOB se requería para que cualquier proceso se iniciara, es decir, la lectura de tarjetas como COMPILER o DATA no iniciaban su desarrollo a menos que ya se hubiera leído la tarjeta de JOB.

Cuando se leían juntas las tarjetas de JOB, COMPILER y DATA el título se podía omitir.

```
JOB
ANALISIS-REGRESION
COMPILER FORTRAN
DATA
```

La descripción de un "job" constituía el medio por el cual el usuario especificaba sus requerimientos al sistema. Esta descripción estaba constituida por cuatro secciones:

- Descripción de entrada
- Descripción de salidas
- Descripción de cintas
- Miscelánea

Descripción de entrada

Esta sección comenzaba con la palabra INPUT seguida por una lista de los títulos de las cintas usadas en este "job", cada una iba precedida por el número de programador.

```
INPUT
1 ANALISIS-1
2 ANALISIS-2
```

En donde ANALISIS-1 y ANALISIS-2 correspondían a dos cintas de datos con los números de programador 1 y 2.

Cuando el número de programador era n=0, indicaba que se leerían datos que seguían al programa como parte de la misma cinta.

```
JOB
ANALISIS-REGRESION
INPUT
1-ANALISIS-1
0 ANALISIS-0
```

En donde ANALISIS-REGRESION correspondia al titulo de la información, ANALISIS-1 correspondia a una cinta de datos y ANALISIS-0 correspondia a la cinta del programa. En este caso el nombre del compilador a usarse debía escribirse al principio de la cinta de programa.

Cuando el encabezado JOB se encontraba en el principio de la cinta de datos se debía incluir la instrucción:

```
SELF = CMD
```

En donde N representaba el número de programador bajo el cual este dato era conocido dentro del programa.

Si se omitía la sección de INPUT, se tomaba como si estuviera incluído y el siguiente programa era compilado y ejecutado. El "default" era:

```
INPUT
SELF = 0
```

Descripción de salida

En esta sección el programador especificaba el equipo de salida que iba a requerir. Se especificaba al principio la palabra OUTPUT seguida por una lista de mecanismos de salida cada uno con su número de programador.

```
OUTPUT
1 LINEPRINTER  CMD BLOCKS
2 ANY          CMD BLOCKS
```

El número M especificaba el límite de salida. Si este era rebasado, el programa se detenía. Si no se especificaba este número, se tomaba como "default" M=1 block.

Descripción de cintas

Cuando un programador utilizaba cintas magnéticas, debía especificar directamente cada cinta en su programa. A continuación se presentan algunos ejemplos del manejo de cintas:

TAPE FREE
(N)

(El titulo del job se almacenaba en el bloque 0 de la cinta). N era el número de programador.
Se utilizaba cuando se requería una cinta nueva.

TAPE / (M)
(N)

(El titulo del job se almacenaba en el bloque 0 de la cinta). N era el número de programador.
Se utilizaba cuando un archivo estaba almacenado en varias cintas. M era el número de continuación.

TAPE / (M) END

Especificába la cinta final del archivo.

COPY TAPE FREE
(N)

(El titulo del job se almacenaba en el bloque 0 de la cinta). N era el número de programador.
Iniciaba el proceso de copiado cuando el programa lo requería.

COPY TAPE (B)
(N)

(El titulo del job se almacenaba en el bloque 0 de la cinta). N era el número de programador.
Cuando la cinta había sido previamente utilizada; B indicaba el número del bloque de la cinta.

OUTPUT TAPE
(MD TAPE FREE / (Tipo de equipo) (S) Blocks
(ND

(El titulo del job se almacenaba en el bloque 0 de la cinta). N era el número de programador.
Si se genera una salida extensa y se quiere grabarla en una cinta privada.

PRINT TAPE
(ND

(El titulo del job se alamcenaba en el bloque 0 de la cinta). N era el número de programador.
Imprime la cinta privada.

Miscelanea

En esta sección se suministraba al sistema información sobre:

- Almacenamiento utilizado
- Tiempo de proceso
- Número de "dumps" que requiere el programa

STORE S

Especificaba el tamaño máximo de almacenamiento principal y secundario en bloques de 512 palabras que estaba en uso durante la ejecución del programa

COMPUTING P Q. (Horas/Minutos/Segundos)

En donde P Q. era un número decimal que indicaba que estaba previsto que el programa corriera a lo más P Q. (H/M/S)

DRUMS D

Especificaba el número de tambores que requería el programa. Eran reservados para cubrir las transferencias programadas al tambor.

2.2 SISTEMA EXPERIMENTAL DE TIEMPO COMPARTIDO EN LINEA

A principios de la decada de los sesenta se desarrollaron distintas ideas que dieron origen a varios tipos de sistemas operativos. Estos sistemas poseian caracteristicas especificas y estaban orientados a prestar diversos servicios a sus usuarios.

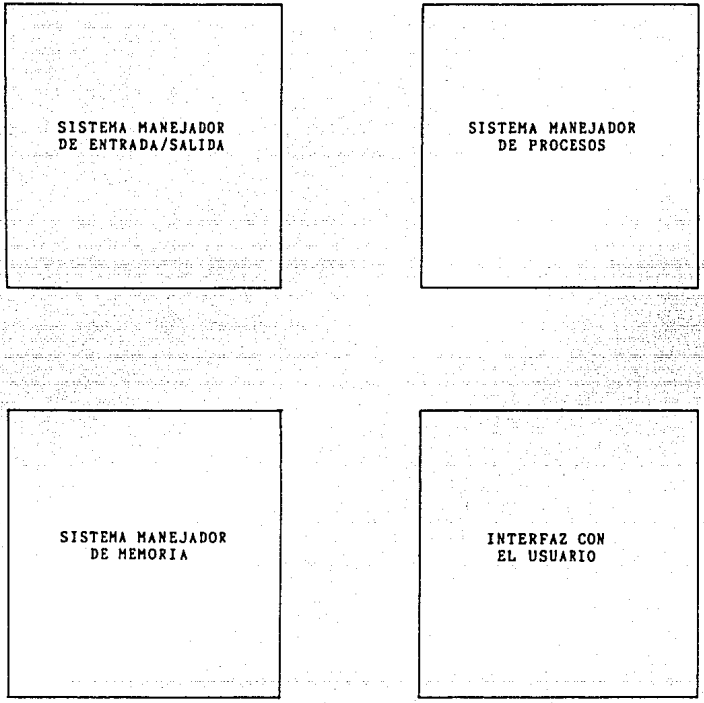
Entre los sistemas que surgieron se encontraban los de tiempo compartido "Timesharing". La caracteristica principal de los sistemas de tiempo compartido es permitir que varios programas usuarios compartan la computadora. Los sistemas de transacciones que existian, como "SABRE" utilizado por American Airlines, eran sistemas conversacionales que permitian trabajar a varios usuarios conjuntamente, sin embargo la gran diferencia con los sistemas de tiempo compartido era que éstos últimos surgieron como sistemas de cómputo de propósito general.

El grupo conocido como "Proyecto MAC" del MIT (Massachusetts Institute of Technology), estuvo investigando y desarrolló en un principio un sistema experimental de tiempo compartido que más adelante dió origen al sistema CTSS (Compatible Time Sharing System). Debido a sus caracteristicas y al hecho de ser uno de los primeros sistemas de tiempo compartido el CTSS se convirtió en el sistema más significativo de su época, tomándose como base su diseño para el desarrollo de varios sistemas posteriores.

El sistema experimental fue desarrollado para la máquina 7090 de IBM. Posteriormente esta máquina fue mejorada surgiendo así la máquina 7094 a la que fue trasladado el sistema CTSS.

El sistema experimental de tiempo compartido que dió origen al CTSS estaba compuesto básicamente de cuatro módulos: el sistema manejador de Entrada/Salida, el sistema manejador de Procesos, el sistema manejador de Memoria y el sistema que se encarga de interactuar con los usuarios. Ver Fig. 2.3

Fig. 2.3 Los cuatro componentes del sistema CTSS.



2.2.1 Arquitectura de la IBM 7090.

Antes de describir el funcionamiento de los módulos del CTSS, es importante presentar la arquitectura base de la 7090 para la que inicialmente fue desarrollado CTSS.

La IBM 7090 fue una máquina transistorizada que tuvo gran importancia debido a las innovaciones y ventajas que ofrecía. Esta máquina poseía una serie de registros índices almacenados en la unidad central de proceso, los cuales permitían realizar instrucciones más complicadas como direccionamientos indirectos.

La memoria con la que contaba esta máquina era de ferritas con 32,768 palabras de 36 "bits" cada una. La máquina contaba también con almacenamiento en tambores magnéticos.

Una de las características más importantes de la IBM 7090 era poder manejar por medio de "hardware" operaciones de punto flotante y de punto fijo. Esta facilidad, que muy pocas máquinas ofrecían, vino a resolver el problema de la representación limitada de un número.

En la IBM 7090 las operaciones de entrada/salida eran controladas por un conjunto de procesadores de propósito específico. Estos procesadores fueron el origen de lo que hoy se conoce como canales. Los procesadores podían comunicarse en forma directa con la memoria principal, sin embargo, la 7090 poseía una unidad de control que era la encargada de manejar la asignación de memoria ya fuera para la CPU o para los distintos procesadores de entrada/salida.

El sistema de cómputo utilizado en el MIT estaba configurado de la siguiente forma:

Contaba con cuatro canales: tres controlaban 19 unidades de cinta y el cuarto con la conexión de datos directos. A esta conexión se encontraban anexados tanto un "buffer" del equipo de tiempo real como un "rack" de control. Este "rack" tenía bajo su control una serie de dispositivos. En el caso particular de este sistema se contaba con tres teletipos.

Es importante señalar dos características especiales implementadas en la IBM 7090 del MIT. Éstas eran: un reloj con ciclos de 60 unidades y con un interruptor, así como un modo especial que permitía la protección de la memoria, reasignación dinámica y control sobre los intentos por parte de los usuarios a realizar operaciones de entrada/salida.

2.2.2 Sistema Manejador de Entrada/Salida.

El sistema de entrada/salida del CTSS era muy sencillo, solamente realizaba tres tipos de operaciones de entrada/salida: I/O por teletipos, I/O de discos y I/O de cintas.

I/O por teletipos: a cada "job" se le asignaba un "buffer" para las operaciones de I/O. Todas las entradas por teletipos eran vaciadas al "buffer" de entrada que le correspondía al "job". Así mismo las salidas por teletipo de un "job" se direccionaban al "buffer" de salida de ese "job". Si ocurría que este "buffer" se llenaba, el "job" era puesto en un estado de espera hasta que la información del "buffer" se imprimía. Es importante señalar que todas las operaciones de I/O que se realizaban por teletipos eran manejadas por el supervisor.

I/O de discos: estas operaciones de I/O se realizaban bajo dos circunstancias; cuando un usuario leía un archivo o cuando el sistema manejador de la memoria realizaba una transferencia de usuario entre memoria principal y memoria secundaria. Para el caso en que el usuario estuviera leyendo un archivo se consideraba que ese usuario estaba corriendo durante la operación de I/O y por lo tanto era transferido a memoria principal por el manejador de memoria.

I/O de cintas: las operaciones de I/O de cintas requerían al igual que las operaciones de I/O de discos que el usuario estuviera corriendo, es decir, que el usuario estuviera asignado al procesador.

Las operaciones de I/O de impresoras o lectoras de tarjetas sólo eran realizadas por los usuarios que corrían programas en lote. Estas operaciones eran enviadas a cinta, y posteriormente procesadas por una computadora fuera de línea.

2.2.3 Sistema Manejador de Procesos (Algoritmo de "Scheduling").

Dentro del Sistema Manejador de Procesos del CTSS, pueden identificarse dos divisiones: la primera consiste en manejar el algoritmo de "scheduling"; la segunda es propiamente el manejo de los procesos. Para este caso se ha dividido la información en dos incisos: en el presente inciso se presentará el algoritmo de "scheduling", mientras que el manejo propio de los procesos se verá en el inciso siguiente.

El algoritmo "scheduling" con el que contaba el sistema CTSS, tenía como objetivo principal proporcionar buen tiempo de respuesta a los usuarios interactivos.

El principio en el que estaba basado era el de niveles multiples de despachamiento, esto es, cada usuario que deseaba correr era asignado a una cola con cierta prioridad. Para esto, el sistema contaba con ocho colas de estados de "ready" o listos, numeradas del 0 al 7. Los niveles o prioridades de las colas estaban dados de mayor a menor por lo que la cola 0 era la de mayor nivel o prioridad, mientras que la cola 7 era la de menor nivel. Así, bajo estas condiciones los usuarios interactivos se encontraban en las colas de menor numeración, mientras que los usuarios con procesos de cálculo intensivos se encontraban en las colas de mayor numeración.

El sistema asignaba a todos los "jobs" un "quantum" de tiempo para correr. Este tiempo estaba dado por la ecuación:

Tiempo corrida = (500) ms. X 2n

en donde n es el numero de nivel de la cola. n = 0..7

Para la asignación de los procesos se consideraba el tamaño de los mismos. Por lo general, los procesos o "jobs" usuarios comenzaban en el nivel 2 o en el nivel 3. Si los "jobs" eran menores que 4 K, comenzaban en el nivel 2 y si los "jobs" eran mayores que 4 K comenzaban en el nivel 3.

De esta forma los "jobs" eran corridos en el inicio por un lapso de tiempo de 2 y 4 segundos respectivamente para cada nivel pero si durante este período de tiempo no alcanzaban a terminar, eran movidos a la siguiente cola la cual tenía menor prioridad.

Si un "job" que se encontraba en colas con prioridad menor no había corrido en el lapso de tiempo de un minuto, se le movía a la siguiente cola de nivel superior.

Por otra parte si un "job" se encontraba corriendo en una cola de nivel L y llegaba algun "job" a una cola de nivel superior J , J<L , el "job" de la cola de nivel L corría un tiempo igual al que tendría asignado si se encontrara en la cola de nivel J antes de ser movido de la memoria principal.

Por último los "jobs" que se procesaban en lote o "batch" eran asignados a colas de niveles inferiores a las que eran asignados los procesos interactivos, ya que estos "jobs" nunca esperarían o recibirían entradas desde un teletipo o una terminal.

2.2.2 Sistema Manejador de Procesos (Manejo de los procesos)

Como ya se mencionó, en el inciso anterior se presentó la parte correspondiente al manejo del algoritmo de "scheduling" que utilizaba CTSS. En el presente inciso se presentarán las políticas con las que CTSS manejaba sus procesos.

El sistema experimental de tiempo compartido fue implementado en un principio con cuatro usuarios, tres de estos usuarios trabajaban con programas en línea y el cuarto usuario realizaba programas en lote o "batch". Posteriormente se permitió hasta un número máximo de treinta usuarios para trabajar en línea.

Cada uno de los usuarios permanecía asignado al procesador hasta que ocurría alguno de los siguientes eventos:

- Sus periodos de tiempo terminaban
- Cometían algun error
- Terminaba su tarea
- Realizaban alguna entrada/salida desde la consola

Los usuarios permanecían en el procesador durante operaciones de entrada/salida que no fueran por consola, y sólo cuando concluían estas podían ser transferidos fuera del procesador.

Uno de los problemas a los que aún se enfrenta un sistema de tiempo compartido es la inevitable saturación del procesador a causa de la cantidad de programas de aplicaciones. Para programar la carga de trabajo, el sistema CTSS catalogaba a sus usuarios en grupos distintos. A cada grupo se le asignaba un número de líneas primarias que en realidad significaban el número de usuarios permitidos en primera instancia para cada grupo. Asimismo se asignaba un número de líneas secundarias que representaban el número de usuarios permitidos en segundo término para cada grupo. El sistema permitía la entrada de estos últimos sólo en caso de que el sistema no tuviera carga pesada. Sin embargo, estos usuarios eran los primeros en ser sacados del sistema cuando algún usuario de otro grupo que no hubiera agotado sus líneas primarias quisiera entrar.

2.2.5 Sistema Manejador de la Memoria (Manejo de Memoria)

Como se mencionó al principio, el sistema experimental fue desarrollado para una máquina IBM 7090, sin embargo al ser construida la IBM 7094 introdujo dos características importantes. La primera consistía en tener una palabra de 32 "bits" junto con un "bit" adicional, con lo cual se podía direccionar 64K. La segunda característica fue introducir dos series de siete registros de un "bit" para ser usados como registros de relocalización y acotamiento.

La memoria virtual de 64K se dividía en dos, la memoria A, que estaba destinada para almacenar los módulos de sistema operativo y la memoria B, la cual se utilizaba para los programas de usuarios.

Para evitar problemas de asignación de memoria, proteger a los usuarios y simplificar la organización del sistema, este sistema de tiempo compartido permitía tener un sólo usuario en memoria principal.

A cada usuario se le asignaba en un principio tanta cantidad de memoria como requería según el tamaño de su programa. Esta asignación comenzaba en la dirección cero de la memoria B . Si durante la ejecución se necesitaba más memoria, esta se asignaba siempre que no rebasara el límite máximo que tenía establecido. Para especificar estos límites se contaba con una serie de 7 registros de un "bit" en los cuales se guardaba la información.

Cuando un nuevo usuario requería correr sus programas, la información del usuario que en ese momento se encontraba en ejecución era transferida a memoria secundaria, desocupándose solamente el espacio suficiente para que el nuevo usuario entrara.

Aquí es en donde los registros límites desempeñaban un papel importante ya que sin ellos se corría el riesgo de que un usuario tratara de acceder a la información que se encontraba en memoria principal y que no le pertenecía.

Con este método de transferir de memoria principal a memoria secundaria, solo parte de la información que ahí residía, el sistema evitaba perdida de tiempo en las transferencias de datos.

2.2.6 Interfaz con el usuario (Descripción del Sistema)

El sistema en línea estaba organizado con base a comandos, mediante los cuales los usuarios se comunicaban con el sistema. Asimismo, el sistema contaba con archivos privados para cada usuario.

Estos archivos eran guardados en cintas magnéticas, una para cada usuario. Estas cintas eran imágenes y contenían datos grabados, como son el nombre y la clase. La gran ventaja que representaba tener los archivos de cada usuario en cintas, era que éstas podían ser grabadas fuera de línea.

Las cintas magnéticas que requería el sistema eran siete para el sistema "batch", una para el programa supervisor de tiempo compartido, una de archivo y una de vaciado para cada uno de los usuarios interactivos.

El supervisor de tiempo compartido manejaba todas las entradas/salidas del sistema en línea. Los comandos eran proporcionados al supervisor a través del teclado y tenían la ventaja de poder ser iniciados en cualquier momento sin importar el programa usuario que se encontrara en memoria.

Puede decirse que el uso del sistema en línea se iniciaba cuando un usuario tecleaba un comando completo. Este comando era colocado en una cola de espera y cuando se completaba un determinado periodo de tiempo, el supervisor iniciaba alguno de los comandos que se encontraban en espera.

Los comandos de sistema se encontraban almacenados en cintas magnéticas, y para que no se desperdiciara tiempo de procesador, el supervisor continuaba la ejecución del último programa usuario hasta que el programa del comando deseado estaba posicionado para lectura. En ese momento el programa usuario que estaba en el procesador era turnado a su cinta de vaciado, se leía el programa de comando desde cinta y se comenzaba un nuevo programa usuario.

Cuando la cola de comandos se encontraba vacía, el supervisor realizaba un simple "round robin", a los programas usuarios que estaban en ese momento en la cola de trabajo. Si ambas colas estaban vacías el programa usuario de lote o "batch", era ejecutado por un periodo de tiempo.

Los comandos estaban compuestos por segmentos separados cada uno de ellos por una línea vertical. El primer segmento correspondía al nombre del comando y los segmentos que seguían a éste eran parámetros del mismo. Cada uno de los segmentos estaba formado por seis caracteres y el caracter que indicaba el inicio del comando era el <return>.

Los comandos constituían la herramienta con la cual los usuarios del sistema en línea se comunicaban con el procesador. Con el fin de conocer un poco más de este sistema a continuación se exponen algunos comandos:

- LOGIN | A | B

  A = Número de problema usuario
  B = Número de programador

  Este comando debía ser tecleado al comienzo de cada sesión. Originaba que la cinta privada del usuario fuera rembobinada y limpiaba los registros de contabilidad de tiempo.

- LOGOUT

  Concluía la sesión del usuario. Rembobinada la cinta privada del usuario y registraba la contabilidad del tiempo consumido.

- INPUT

  Comando de edición de textos

- EDIT ! A ! B

  A = Titulo del archivo
  B = Clase del archivo

  Comando que era utilizado para la edición de textos.

- FILE ! A ! B

  A = Titulo del archivo
  B = Clase del archivo

Comando que era utilizado para la edición de textos.

2.3 RESUMEN.

El sistema operativo ATLAS y el sistema operativo CTSS constituyen dos representantes de la primera etapa de evolución de los sistemas operativos.

Son sistemas de distintos propósito, ya que ATLAS fue diseñado para trabajo "batch" mientras que CTSS se diseñó para uso interactivo. Sin embargo, a pesar de ésto existen características semejantes entre ellos que es importante señalar.

Una característica importante en las máquinas de la segunda generación es la implantación de registros de índice, que ambos sistemas utilizan para el direccionamiento de memoria.

Otra aportación importante en ambos sistemas, es la utilización de elementos especiales de "hardware" independientes de la CPU, para la realización de las operaciones de I/O. Así, el problema que representaba tener detenida la CPU, en espera de una operación de I/O se ve contrarestado.

En estos sistemas operativos la información correspondiente al usuario tenía asignado un espacio en memoria secundaria aún cuando no se estuviera ejecutando. El manejo de la memoria en esta forma, permitía tener varios procesos o usuarios dentro del sistema y no uno por uno como se utilizaba anteriormente.

Ahora bien, tener la posibilidad de alojar varios trabajos por ejecutar en el sistema, requería llevar un control muy estricto sobre ellos. Los sistemas operativos desarrollaron políticas para la elección de un proceso a ejecutar y asimismo se introdujeron normas para la correcta distribución de la CPU entre los procesos que la compartían.

Todas estas normas y políticas constituyen una de las mayores aportaciones de la primera evolución ya que gracias a ellas la CPU podía ser compartida entre varios procesos, incrementando con esto su productividad.

Con respecto al problema de protección de "jobs", estos sistemas operativos no introdujeron aportaciones puesto que se apoyaban en las facilidades que les proporcionaba el "hardware" para protección e interrupción de "jobs".

Algo que es importante y que marca una tendencia, en cuanto a manejo de procesos, es la clasificación de los procesos y usuarios según recursos utilizados y prioridades. A pesar de estar considerada dentro de las políticas para la implantación de un ambiente de tiempo compartido, es conveniente mencionarla por separado, porque, como se verá más adelante, constituye una base que todos los sistemas operativos posteriores han mantenido.

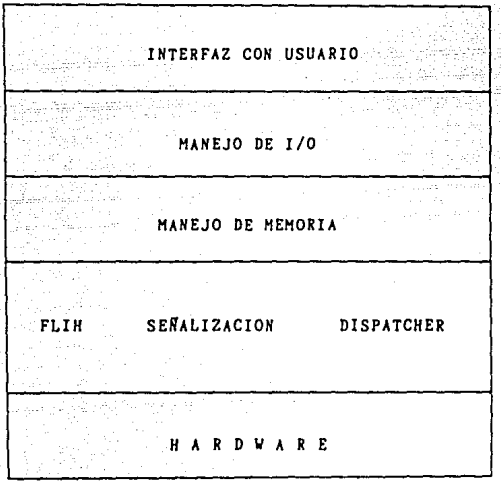
La utilización de sistemas auxiliares de cintas magnéticas permitían a los sistemas ampliar su capacidad de almacenamiento, pues podían disponer de mayor espacio y dejar para los sistemas de cinta todo aquello que no tenía por qué permanecer en memoria principal.

Ahora bien, considerando que el sistema operativo CTSS estableció las bases que dieron origen al sistema operativo UNIX, es interesante presentar los elementos sobresalientes que conformaban a CTSS.

Tomando la idea de Lister, sobre el "sistema operativo de papel", expresada en su libro "Fundamentals of Operating Systems", se desarrollará algo análogo con los sistemas operativos que componen la línea evolutiva de UNIX.

El objetivo de esquematizar en forma semejante al "sistema operativo de papel", es presentar los distintos niveles que van desarrollandose en cada uno de los sistemas operativos. Así para el caso del CTSS, su representación sería la siguiente:

Fig. 2.4 Representacion de los niveles de CTSS.



3. SISTEMA OPERATIVO MULTICS.

El sistema operativo MULTICS forma parte de una nueva generación de sistemas de tiempo compartido que surgieron a finales de la decada de los sesenta.

El MULTICS (Multiplexed Information and Computing System) fué desarrollado en forma conjunta por el MIT (Massachusetts Intitute of Technology), por los Laboratorios Telefónicos Bell y por el departamento de cómputo de la compañía General Electric.

Al hablar sobre el sistema MULTICS debe destacarse su principal objetivo: ser un sistema de servicio multiple, que mediante una facilidad central debía ser capaz de distribuir entre varios usuarios los recursos de la computadora.

El sistema MULTICS manejó dos conceptos importantes que daban solución a los problemas de capacidad y distribución existentes en aquel tiempo. La primera solución consistía en proporcionar a los usuarios un ambiente de memoria virtual. En MULTICS el procesador permitía direccionar toda la información almacenada, lo que permitía que un programa que se encontraba operando, direccionara cualquier información en memoria.

Por otra parte MULTICS realizó una generalización del concepto de tiempo compartido, distribuyendo en forma eficiente y efectiva los recursos del "hardware" entre los distintos usuarios del sistema. Las facilidades de MULTICS se pueden resumir en :

- Extender la capacidad de la memoria con el ambiente de memoria virtual.
- Compartir tanto los recursos del sistema como la información entre los distintos usuarios.

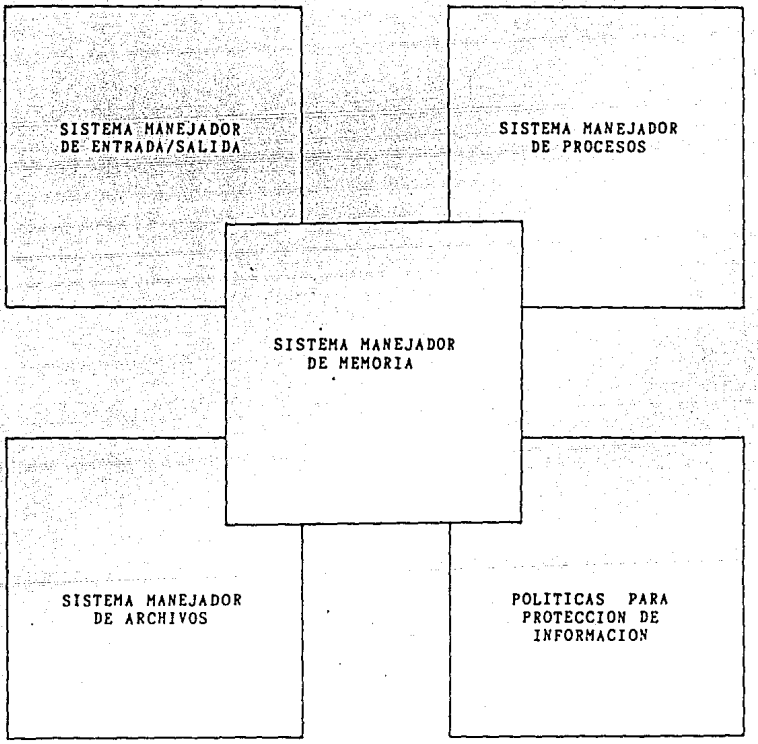
Cabe aquí mencionar dos circunstancias de interés que se presentaron, la primera de ellas fue la utilización de un lenguaje de alto nivel, como lo es PL/1, para escribir gran parte del sistema y la segunda fue la implantación de MULTICS con la ayuda del sistema de tiempo compartido CTSS.

MULTICS no solamente fue desarrollado con la ayuda del CTSS, sino que mucho de su diseño está basado en su predecesor. Son muchas las características de MULTICS heredaras de CTSS, y a lo largo de este capítulo se irán señalando.

Si se analiza en primera instancia el sistema operativo MULTICS, como se hizo con los sistemas operativos ATLAS y CTSS, se encontrará que MULTICS está formado por cinco módulos, éstos permitían al sistema administrar sus recursos en forma más eficiente que los sistemas operativos analizados anteriormente. Los cinco módulos principales de MULTICS se esquematizan en la Fig. 3.1.

Antes de pasar a la descripción de los módulos se presentará la arquitectura base de la máquina GE-645, para la que fue desarrollado MULTICS.

Fig. 3.1 Los módulos del sistema MULTICS.



3.1 Arquitectura de la GE 645.

La máquina GE 645 estaba considerada por sus características como una computadora de la tercera generación y fue desarrollada por la compañía General Electric. Entre las características más importantes de esta generación de computadoras pueden mencionarse:

- Introducción del uso de los circuitos integrados que reducían en forma notable el tamaño físico de las máquinas, así como el costo de las mismas.
- El diseño de los procesadores se ve simplificado por el uso de la microprogramación.
- Técnicas como multiprogramación, el proceso en cadena y el multiprocesamiento fueron implementadas logrando aumentar así la velocidad efectiva a la cuál se ejecutaban los programas.
- Desarrollo de facilidades para compartir en forma automática los recursos del sistema.

Con las mejoras realizadas en el "hardware",se hizo necesario desarrollar "software" capaz de ejercer control sobre las nuevas facilidades proporcionadas.

El sistema operativo MULTICS se encontraba estrechamente relacionado con el "hardware" de la GE 645.

MULTICS introduce el concepto de segmento, el cuál constituía un elemento importante para que el sistema llevara un estricto control de la memoria. Los segmentos eran bloques de memoria que contenían información, así como un conjunto de atributos para el acceso de la misma ya fuera para consulta o para alterar algún dato.

El segmento constituía la unidad básica que utilizaba MULTICS para permitir a los usuarios del sistema compartir la información almacenada. Una de las características más importantes de los segmentos era la de ser directamente direccionables desde un programa.

La memoria de la GE 645 estaba organizada en palabras de 36 "bits", cada dirección para el acceso de la memoria era un par ordenado de (nombre,i) en donde nombre referenciaba al segmento e i era el número de palabra en ese segmento.

Por "hardware" la máquina soportaba hasta 2 a la 18 segmentos por proceso con una longuitud de 2 a la 18 palabras, sin embargo el sistema operativo restringía el tamaño del segmento a 2 a la 16 palabras por segmento.

Cada uno de los segmentos se dividía en páginas las cuales eran en realidad las unidades de memoria que se trasladaban a la memoria principal cuando la información que contenían era requerida. Por lo general en MULTICS el tamaño de las páginas era de 1024 palabras, sin embargo esta cantidad podía variar.

La GE 645 tenía dos formatos para las instrucciones; las de tipo cero servían para referenciar los datos o instrucciones dentro del mismo segmento, mientras que las instrucciones de tipo uno eran usadas para referenciar los datos de instrucciones de cualquier segmento. Esta máquina contaba además con ocho registros básicos, semejantes a los que poseía la máquina bajo la cual corría CTSS, y que estaban formados por 24 "bits" cada uno. Los registros básicos tenían la función de almacenar apuntadores efectivos a la palabra descriptora. Estos apuntadores en conjunción con otros parámetros formaban la dirección de las palabras a las que querían acceder.

3.2 Sistema Manejador de Entrada/Salida.

El sistema de manejo de entrada/salida del sistema MULTICS presentaba ventajas sobre los manejadores de entrada/salida existentes. En los primeros sistemas operativos era necesario realizar en forma explícita la llamada a las rutinas encargadas de la entrada/salida, sin embargo en el sistema MULTICS no ocurría esto.

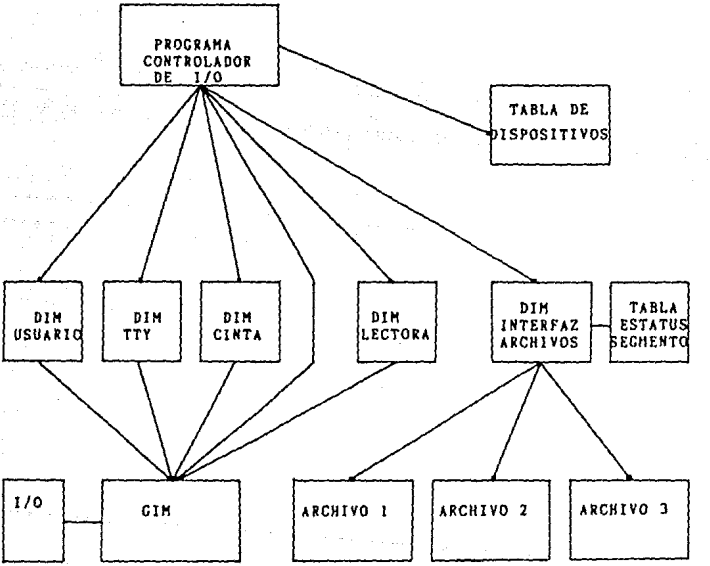
El sistema de archivos de MULTICS se encargaba tanto de dar a conocer, como de ligar, los archivos que se encontraban en lo que se denominaba el archivo jerárquico, por lo que la transferencia de los archivos entre la memoria principal y secundaria resultaba ser transparente para el usuario.

Adicionalmente MULTICS proporcionaba una interfaz independiente de los dispositivos, esto permitía que el usuario, al codificar el requerimiento de una operación de entrada/salida, no necesitaba especificar el tipo de modelo de dipositivo en particular, sino que describía el dispositivo en términos de las funciones que debía realizar.

En el sistema operativo MULTICS tanto los dispositivos de entrada como los de salida, eran agrupados bajo un nombre genérico. A esta agrupación se le conocía como "stream" de entrada/salida. Al "stream" se le podían asignar varios dispositivos particulares, de esta forma cuando se quería realizar una operación de escritura o de lectura, el sistema debía consultar la tabla de asignación para determinar los dispositivo asociados a un "stream" en particular, y después de interpretar el requerimiento, convertirlo en una acción de entrada/salida.

El módulo de control de entrada/salida transmitía el requerimiento del dispositivo específico al módulo de "interfaz" de dispositivos (DIM), el cuál se encargaba de que el requerimiento de entrada/salida fuese completado. Existía un DIM por cada tipo de dispositivo. Fig 3.2.

Fig. 3.2 Esquema del sistema de Entrada/Salida de MULTICS.



La comunicación con los canales, es decir, la iniciación de su actividad, el proporcionar los comandos que necesitaban y el recibir e interpretar el estado en el que se econtraba la información que transmitían los canales , se realizaba por medio del controlador generalizado de entrada/salida (GIOC), el cual se encargaba de manejar y multicanalizar la comunicación entre la memoria y los canales de entrada/salida.

El DIM compilaba un programa que proporcionaba al GIOC. Este programa reflejaba las características del dispositivo en particular, el cual estaba asignado al "stream", y además contenía líneas de control para el caso de cintas magnéticas.

Después de compilado el programa, el DIM llamaba al GIM que servía de "interfaz" con el GIOC. Éste último debía interactuar junto con el DIM hasta que el requerimiento de entrada/salida fuese completado. El GIM era el responsable del manejo y monitoreo total del GIOC, se encargaba de contestar las interrupciones, reconocía la terminación de una tarea y transmitía al DIM el estado de la información.

Una de las ventajas del sistema operativo MULTICS era que realizaba una llamada explícita de entrada/salida cuando requería transferir datos entre memoria principal y secundaria. De esta forma MULTICS ofrecía una "interfaz" DIM para el sistema de archivos y funcionaba en forma semejante a los otros DIM's. La diferencia estaba en que este DIM no realizaba una llamada al GIM, sino que era usado para que un segmento fuese visto como un dispositivo de entrada/salida. El DIM del sistema de archivos tenía una memoria en donde almacenaba información de cada segmento que estaba siendo referenciado, como si fuera un dispositivo. Cuando se realizaba una llamada al programa de control de entrada/salida para asignar un "stream" a un segmento, el segmento requerido era dado a conocer al proceso o programa que lo solicitó, de esta manera MULTICS junto con la ayuda del "hardware" introducía nuevos conceptos que lo hacían un sistema operativo muy versátil.

3.3 Sistema Manejador de procesos.

En un ambiente de multiprogramación en el que es posible que coexistan dos o más procesos, es necesario establecer un control eficiente sobre la ejecución de los mismos. Tener procesos coexistentes implica que estos procesos competirán por la obtención de los recursos del "hardware", por el "software" y por las bases de datos.

El sistema operativo MULTICS, para resolver estos conflictos, contaba con lo que podemos denominar manejador de procesos. Este tenía la responsabilidad de elegir el "job" o proceso a ser ejecutado, administrar los recursos entre los procesos que estaban en ejecución, así como mantener información del estado en que se encontraban los procesos.

En el MULTICS se podían distinguir a grandes rasgos tres tipos de procesos que compartían los recursos del sistema de cómputo.

Los procesos que aparentemente no se encontraban relacionados.- En un ambiente como el de MULTICS varios "jobs" compartían el procesador y no por ello necesitaba existir relación entre ellos, sin embargo, al compartir tanto los programas supervisores del sistema operativo como también algunas bases de datos, se establecía cierta relación entre ellos. Cuando se encontraban dos o más procesos compitiendo por el procesador, el programa o proceso que en ese momento estaba en el procesador corría en modo de ejecución supervisor y por lo tanto tenía la obligación de poner en alerta a los programas o procesos que esperaban que el procesador fuera desocupado.

Los procesos de usuario y los procesos de sistema.- En el sistema existían un conjunto de programas o procesos los cuales eran los encargados de proporcionar una serie de servicios. Estos procesos eran conocidos como procesos del sistema y mantenían comunicación con los procesos usuarios a través de la ejecución de subrutinas, las cuales ejecutaban los pasos que se requerían para comunicar una solicitud de trabajo a los procesos de sistema. Es importante mencionar que los procesos de sistema eran completamente independientes y que atendían los requerimientos que les eran solicitados de acuerdo a su conveniencia, es por esto que su ejecución se realizaba en forma paralela a la de los procesos usuarios. Mientras tanto, los procesos usuarios no permanecían esperando la terminación de su requerimiento sino que continuaban su ejecución, preguntando en forma periódica si su requerimiento había sido atendido o aún no.

Los procesos cooperativos.- Este último tipo de procesos coexistentes tenían como característica la de ser procesos que se ejecutaban en forma paralela, lo que implicaba que era posible ejecutar ciertas instrucciones del proceso en forma simultánea, siempre que la ejecución de algunas de las instrucciones no fuera dependiente de la ejecución previa de otras instrucciones. Esta forma de operación ofrecía ciertas restricciones como que para poder realizar la ejecución en paralelo de dos o más procesos, debía contarse con dos o más procesadores que ejecutaran dos o más procesos en forma paralela. El sistema MULTICS ofrecía las herramientas necesarias para que se pudiera llevar a cabo un procesamiento multiple.

Para controlar el órden de ejecución de varios procesos en forma simultánea, el sistema operativo MULTICS se ayudaba de varios módulos que manejaban la transición de estados de los distintos procesos que se encontraban dentro del sistema. La expresión "transición de estados de un proceso" significa que si un proceso estaba corriendo podía pasar a un estado de espera o si se encontraba en un estado de espera podía pasar a un estado de alerta o listo para ejecutarse.

Un proceso pasaba a un estado de espera cuando requería una página que no se encontraba en memoria principal y era necesario traerla o cuando requería alguna intervención de entrada/salida. Cuando un proceso pasaba a un estado de alerta era porque el requerimiento solicitado había sido atendido y estaba esperando que el procesador estuviera disponible para entrar a ejecución.

El módulo conocido como controlador de tráfico era el encargado de llevar el control del estatus del proceso. Entre las funciones que tenía encomendadas estaban asignar al procesador un programa o proceso teniendo que actualizar los registros del procesador, al estatus correcto del programa o proceso para que este último pudiera ejecutarse. Debía ,asimismo, retirar del procesador el programa o proceso que hubiera consumido su tiempo permitido de proceso o que requiriera esperar alguna operación de entrada/salida. Además, el controlador de tráfico debía guardar la información correspondiente a los registros del procesador para que en su próxima intervención, el programa pudiera continuar su ejecución.

Las tareas que realizaba el controlador de tráfico para distribuir el procesador entre las distintas clases de procesos eran sencillas. Su actividad se centraba en el mantenimiento y la actualización de una lista de procesos coexistentes denominada tabla de procesos activos (APT), la cual poseía una entrada para cada proceso. Esta tabla contenía información sobre el estado de ejecución del proceso, es decir, si estaba ejecutandose, en espera de algún evento o listo para ejecución. Además, almacenaba la prioridad asignada al proceso y el tiempo asignado para ejecución, entre otras cosas.

El "scheduler" o despachador era un módulo del controlador de tráfico que se encagaba de asignar el tiempo Q que era permitido a un proceso permanecer en la CPU. Adicionalmente el sistema tenía registrado un tiempo R , que era el tiempo real utilizado por el proceso.

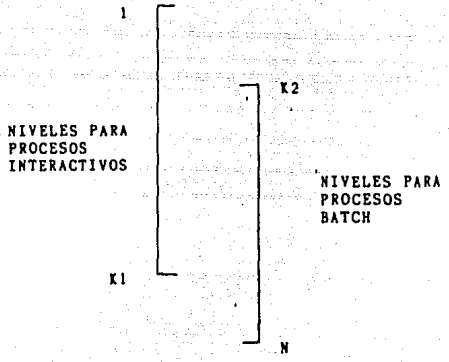
Cuando el tiempo Q había sido agotado, el proceso se convertía en inelegible para utilizar el procesador, asimismo se le asignaba un nuevo tiempo Q', se actualizaban sus valores en la tabla de procesos activos y se colocaba en estado de alerta. Pero en esta ocasión el proceso era colocado en una cola que tenía asignado menor tiempo de ejecución que aquella a la que fue asignado en un principio, con esto la prioridad del proceso cambiaba y ahora era menor que la que tuvo anteriormente. Sin embargo, cuando un proceso entraba en un estado de espera por algún evento del sistema, el proceso no perdía eligibilidad ni prioridad. Así, cuando el evento esperado ocurría, el proceso seguía ejecutándose con la misma prioridad que tenía antes de interrumpir su ejecución.

Esta manera de manejar el tiempo de estancia en la CPU para cada proceso, no era una idea nueva sino que se tomó de la ya existente en el sistema CTSS.

El "scheduler" era también el encargado de seleccionar el proceso que sería elegible para el procesador y colocarlo en un estado de alerta. En MULTICS existían una serie de colas, las cuales tenían cierto nivel de prioridad. A cada proceso usuario se le asignaba un rango de prioridades (L1,L2), el cual estaba asociado con el tipo de servicio de tiempo compartido que se le daría al proceso.

El sistema asignaba a los procesos interactivos un rango de valores de (1,K1) mientras que los procesos "batch" tenían un rango de (K2,N). Fig 3.3. En el sistema MULTICS los niveles de prioridad mayor eran para procesos que estaban esperando utilizar el procesador por un espacio de tiempo breve mientras que los niveles de prioridad menor estaban reservados para procesos interactivos de mayor tiempo de ejecución y para procesos "batch".

Fig. 3.3 Relación de prioridades de ejecución en MULTICS.



Cuando un proceso se hacía elegible, éste debía ser cargado, lo que implicaba construir ciertos segmentos descriptores y preparar todas las páginas designadas al momento de descargar el proceso.

Para llevar un control sobre los procesos que podían cargarse al sistema, MULTICS agrupaba a sus usuarios dentro de conjuntos de control de carga. A cada uno de estos conjuntos se le asignaban ciertas unidades que en realidad representaban los recursos utilizados por el proceso. Por ejemplo, un proceso "batch" no tenía pausas en consola, como ocurría con un usuario interactivo, por lo que el primero ocupaba los recursos en forma continua y se asignaba a un conjunto que tuviera más unidades que el conjunto al que se asignaba un usuario interactivo.

Este concepto de agrupar a sus usuarios según ciertas características y recursos utilizados, fue una idea tomada del sistema CTSS que MULTICS generalizó.

El operador del sistema establecía el número máximo de unidades que permitiría por grupo. Con ésto, cuando algún conjunto de control de carga excedía sus unidades, sus usuarios eran sacados del sistema después de un período de gracia.

3.4 Sistema Manejador de Memoria (Memoria Virtual).

El sistema operativo MULTICS proporcionaba a sus usuarios un ambiente de memoria virtual, esto implicaba la utilización de ciertas estructuras que le permitieran realizar las funciones de paginación y de intercomunicación de los segmentos. Entre estas estructuras estaban:

- Tabla de páginas.- Esta estructura contenia apuntadores en forma individual a las páginas del segmento que se encontraban en memoria principal.

- Descriptor de segmentos.- Era una tabla en la cual se tenían algunas características de los segmentos de un proceso. Se tenía una entrada por segmento.

- Palabra descriptora .- Era una estructura de 36 "bits" del descriptor de segmentos. Tenía un apuntador a la tabla de páginas en caso de que el segmento se encontrara en memoria principal, de otra forma se indicaba que el segmento no estaba en memoria principal.

- Campo descriptor.- Era conocido también como el campo de control de acceso . Los "bits" en éste campo eran alterados por el supervisor e interpretados por el "hardware".

Los segmentos podían dividirse en dos clases: los segmentos que eran propiamente datos y los segmentos que eran programas. Estos últimos se dividían a su vez de acuerdo a su modo de ejecución en ordinarios o esclavos y en segmentos de ejecución o amos. La diferencia entre ambos era que los primeros eran codificados y alterados por los usuarios mientras que los segmentos de ejecución no podían ser alterados ya que tenían la capacidad de ejecutar instrucciones privilegiadas como las del programa supervisor.

Cada proceso en MULTICS contaba con una colección de segmentos con identificación y tamaño únicos. A esta colección se le conocía como el espacio de direccionamiento por proceso. Este espacio debe entenderse como la colección de segmentos (cantidad de direcciones) a las que un programa o proceso podía hacer referencia. En sistemas que poseen un ambiente de memoria virtual como era el caso de MULTICS, el espacio de direccionamiento del proceso es mayor que el tamaño físico de la memoria principal.

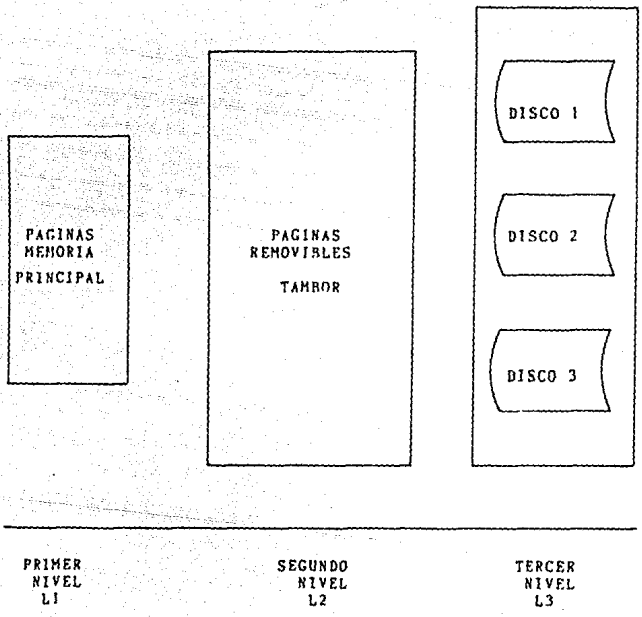
En muchos sistemas el manejo de paginación de la memoria y el manejo de la información se realizan en forma separada, sin embargo en MULTICS no ocurría así. El sistema trataba todos los archivos como segmentos y realizaba todas las entradas/salidas de archivos por medio de mecanismos de demanda de páginas.

Para realizar el mapeo de los segmentos, ya sea de la memoria principal al espacio de direccionamiento del proceso o viceversa, el sistema de archivos era invocado y basándose en la información que guardaba en las tablas de mapa de páginas y mapa de segmentos, el sistema se encargaba de trasladar el segmento o la página de éste que se requería.

El sistema MULTICS estableció una jerarquía de almacenamiento que empleaba tanto dispositivos de alto-rendimiento-alto-costo como dispositivos de bajo-rendimiento-bajo-costo. Los programas que corrían en MULTICS podían encontrarse en tres diferentes dispositivos de almacenamiento. Fig. 3.4.

El que el sistema manejara varios posibles lugares de almacenamiento, originaba el problema de decidir en donde le correspondía estar a cada página. Para esto MULTICS se basaba en las siguientes reglas:

Fig. 3.4 Relación de niveles de almacenamiento.



- Si se requería una página en memoria principal, ésta era extraída de la memoria de nivel dos (N2), si es que estaba ahí, si no se extraía de la memoria de nivel tres (N3).

- Si la memoria de nivel uno (N1) estaba llena y se requería mover una página, ésta era colocada en la memoria de nivel dos (N2). Aquí se verificaba si existía o no una copia de la página en el nivel dos (N2), si no existía se creaba el espacio para ella y si existía se reemplazaba.

- En caso de que la memoria de segundo nivel (N2) se encontrara llena cuando se quería introducir una página en ella, se debía mover alguna de las páginas que contenía hacia la memoria de nivel tres (N3).

El criterio que utilizaba MULTICS para seleccionar las páginas a ser movidas era el buscar aquella página que hubiera permanecido más tiempo sin ser referenciada y esa era candidata a salir ya sea de la memoria de nivel uno (N1) o de la de nivel dos (N2).

3.5 Sistema Manejador de Archivos.

El sistema de archivos era un conjunto de módulos del sistema que tenían como objetivo el facilitar al usuario y a los procesos el manejo de la información que se encontraba en el sistema. Entre las funciones que ejecutaba estaban:

- Mantener la pista de toda la información en el sistema a través de varias tablas que contenían el nombre, localización, derechos de acceso de la información.

- Determinar en dónde sería almacenada la información y quien tendría acceso a ella.

- Asignar la información al proceso o programa que la requiriera. Esto implicaba encontrar la información requerida, hacerla accesible al proceso así como establecer control de acceso.

- Una vez que había sido actualizada la información en memoria principal, debía actualizar la información en la copia que se encontraba en memoria secundaria.

En el manejo de la información, MULTICS tenía dos conceptos que es importante resaltar. El primero es que trataba a un archivo como si fuera un segmento y el segundo es que tenía una estructura de archivos jerárquica. El tratar a un archivo como segmento permitía que varias de las funciones que realizaba el sistema de archivos fuesen en realidad ejecutadas por el "hardware". Por otra parte, una estructura jerárquica daba flexibilidad a los usuarios al establecer derechos de protección para compartir la información, así como acceder a los archivos.

El sistema de archivos básico, poseía dos módulos de gran interés, ya que constituyen una aportacion de MULTICS; el módulo de control de segmentos (SCM) y el módulo de control de directorio (DCM).

El módulo de control de segmentos era responsable de interpretar la referencia hecha por un usuario a cierto segmento, es decir establecía el mapeo de un nombre simbólico a un número de segmento.

Este módulo era el que determinaba a cuál de los segmentos que eran conocidos para el programa o proceso se estaba haciendo referencia. En caso de que no fuera ninguno de los segmentos conocidos, el SCM debía investigar si se trataba de la creación de un nuevo segmento y de ser así debía decidir en qué lugar le correspondía estar dentro de la jerarquía de archivos.

El SCM obtenía la información que necesitaba para tomar sus decisiones, por una parte de la tabla de segmentos conocidos (KST), la cuál tenía registrados los segmentos que eran parte de un proceso y por otra parte de la tabla de segmentos activos (AST), la cuál contenía todos los segmentos usados por los procesos activos.

El módulo de control de directorio (DCM) estaba involucrado en hacer modificaciones a la estructura del directorio. Todos los requerimientos de usuario que tenían que ver con la creación, destrucción o alteración de archivos y sus descripciones, así como todas las preguntas por el estado o la localización de segmentos debían invocar al DCM, ya que solamente este módulo tenía la suficiente autoridad para leer y alterar el contenido del directorio de segmentos.

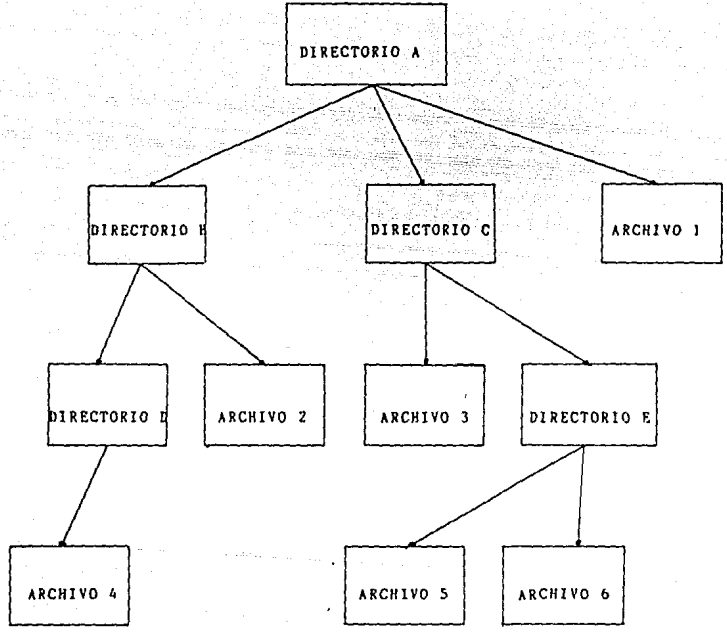
Para visualizar la estructura del sistema de archivos hay que hacer mención a dos campos que existían en la estructura del directorio de MULTICS. Estos dos campos son "branch" y "links". El "branch" era una descripción detallada del segmento que contenía su localización en memoria secundaria.

Cuando se creaba un "branch" se le asociaba un único identificador además de una serie de "alias" o sobrenombres con los cuales los usuarios harían referencia al segmento en el código fuente. La utilización de "alias" en MULTICS era muy necesaria ya que permitía enlazar o vincular dos o más segmentos nombrados individualmente, por ejemplo; si tenemos los segmentos [A], [B] y [C] estos tres podían ser enlazados en un nuevo segmento y este segmento sería conocido por los tres nombres. Otro propósito en el uso de "alias" era el permitir que un mismo segmento fuera referenciado con distintos nombres; ejemplo "DELETE" y "REMOVE" en este caso son una misma función.

Un "link" era una entrada especial cuyo objetivo era de apuntar a otra entrada, normalmente en algún otro directorio. Las ligas permitían tener referencias cruzadas ya que la correspondencia entre segmentos y "branch" debía ser uno a uno. En MULTICS no era posible que dos directorios compartieran el mismo segmento, sin embargo, utilizando "links" estas podían apuntar a cierto "branch" aparentando de esta forma que el segmento correspondiente pertenecía a dos directorios.

La Fig. 3.5 muestra la estructura de archivos de MULTICS.

Fig 3.5 Modelo Conceptual de la Estructura de Arbol del Sistema de Archivos de MULTICS.



3.6 Protección.

En un ambiente operativo en donde coexisten varios procesos, como el que proporcionaba MULTICS, la efectividad en la protección de la información es la que garantiza la integridad de los procedimientos y datos del usuario.

La forma de protección utilizada por las primeras computadoras consistía en establecer dos tipos de programas y su modo de ejecución: los programas supervisores y los programas de aplitación. Los programas del supervisor podían ejecutar todo tipo de instrucciones mientras que los programas usuarios estaban limitados en su alcance.

Esta protección no era lo suficientemente confiable ya que, si bien es cierto que los programas usuarios no podían dañar a los programas supervisores, si podía ocurrir que un programa de aplicación alterara o destruyera otro programa de aplicación.

El sistema operativo MULTICS introdujo el concepto de anillos de protección, los cuales tenían como base dos premisas:

- La necesidad de conocer, es decir, un procedimiento solo debía tener acceso a aquellos datos o procedimientos que fueran necesarios para su ejecución.

- Controlar el grado de daño causado por errores, es decir, los procesos debían de asignarse a los distintos anillos dependiendo de su confiabilidad.

Este concepto generalizaba la jerarquización de los estados en que podía operar el sistema. Ya no se tenía un programa usuario y un supervisor, sino que estaban N programas usuarios y un solo supervisor el cual debía mantener una base de datos de los procesos usuarios cada vez que estaba en ejecución.

MULTICS ofrecía, además, la posibilidad de que dos o más procesos compartieran los mismos segmentos por lo que debía establecer un control en el acceso que tendrían los otros usuarios hacia un segmento.

Los anillos concéntricos eran un mecanismo que ofrecían protección a segmentos interprocesos. Estos segmentos eran compartidos por dos o más procesos, por lo que aumentaba la probabilidad de que fueran dañados por alguno de los procesos. Sin embargo, con el concepto de anillos se protegía a los segmentos de cualquier posible destrucción.

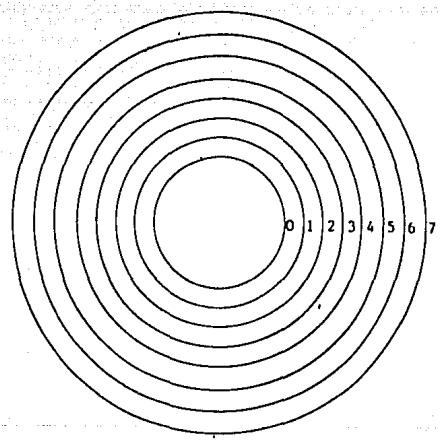
Cada segmento era asociado a uno o más anillos, de los cuales los anillos centrales (0-3) correspondían al sistema operativo y los anillos (4-7) estaban disponibles al usuario. Cuando un procedimiento era asignado a un anillo de categoría R , durante su ejecución este procedimiento podía llamar a cualquier procedimiento que se encontrara en un anillo de categoría menor a la de R , pero no podía referenciar aquellos que se encontraban en anillos con categorías mayores a la suya. Fig. 3.6.

La categoría de los anillos iba de mayor a menor, es decir el anillo número 0 era el de mayor categoría mientras que el anillo número 7 era el de menor. El sistema MULTICS contaba con ocho anillos en los cuales distribuía sus procesos. De esta manera aislaba efectivamente el problema de protección y limitaba un posible daño al sistema a causa de errores en los programas de aplicaciones.

Las reglas que se aplicaban a los procedimientos eran:

- Un procedimiento que residía en una anillo número E tendría la libertad de llamar a cualquier procedimiento que residiera en el anillo E o en un anillo con número mayor al de E. Asimismo podía hacer referencia a datos, tanto como lo permitía cada control de acceso del segmento.

Fig. 3.6 Estructura de anillos de protección.



- A un procedimiento que se encontraba en el anillo número E , le debía ser negado el privilegio de llamar un procedimiento en un anillo menor al suyo. El acceso podía ser permitido pero debería ser controlado por el sistema operativo.

- A un procedimiento que residía en el anillo número E, nunca se le debería permitir el acceso a segmentos de datos que se encontraran en un anillo de número menor.

Este mecanismo de protección fué implementado como tal por primera vez en MULTICS y fué una de las aportaciones más importantes del sistema, ya que sirvió de base para desarrollos posteriores en otros sistemas.

3.4 Resumen.

El sistema operativo MULTICS fue sin duda un sistema inovador, que reune los elementos de un sistema de tiempo real y de tiempo compartido, así como los elementos para implementar un sistema de memoria virtual.

La introducción de los conceptos de paginación y de segmentación constituyen una de las aportaciones de MULTICS. Hasta ese momento la utilización de la memoria se había visto limitada, pero la solución propuesta en MULTICS resolvió el problema de espacio, al tiempo que marcó una tendencia para los sistemas posteriores. Aquí debe mencionarse que MULTICS se apoyó en la estructura conocida como segmento, y no fue sólo un capricho, sino que representó un obstáculo por vencer en los sistemas operativos posteriores, ya que, como se expuso a lo largo de este capítulo, el segmento era manejado por "hardware"; esto lleva a considerar que muchas de las funciones que intervenían para el manejo de los segmentos eran realizadas también por "hardware".

Los mecanismos de protección implementados por MULTICS solucionaron el problema que planteaba el tener un programa supervisor controlando a N programas usuarios. Los sistemas operativos anteriores no implantaron sistemas de protección como tales. Sólo establecían dos modos de ejecución y este control lo ejercía basicamente el "hardware", por lo que los mecanismos de protección implementados por MULTICS constituyen uno de los primeros pasos dentro de este campo.

Una aportación muy importante del sistema operativo MULTICS lo constituye su estructura jerárquica de archivos. Al introducir esta estructura MULTICS da flexibilidad en el manejo de los archivos y establece antecedentes para desarrollos posteriores más sofisticados.

Por otra parte, es cierto que MULTICS heredó ciertas características de su diseño, del sistema operativo CTSS. Esto no es raro, en primer lugar porque muchos de los sistemas toman diseños anteriores y los optimizan; y en segundo lugar porque participaron en el diseño de MULTICS, personas que también participaron en el diseño del CTSS, por lo que era de esperarse que lo mejor del CTSS fuera implementado en MULTICS.

El diseño para el control de procesos implementado en MULTICS estuvo basado en el que fue implantado en CTSS; tanto el control del tiempo como la agrupación de los procesos según sus recursos son basicamente manejadas igual.

Una idea tomada del CTSS fue el manejo de los dispositivos de I/O mediante procesadores especiales. Sin embargo MULTICS desarrolla aún más esta idea, ya que introduce una interfaz para el manejo de I/O dependiendo de cada dispositivo. El sistema agrupa a los dispositivos según sus caracteristicas, de esta forma el programa usuario no referencia a uno en particular, sino que hace el llamado al dispositivo de acuerdo a sus caracteristicas.

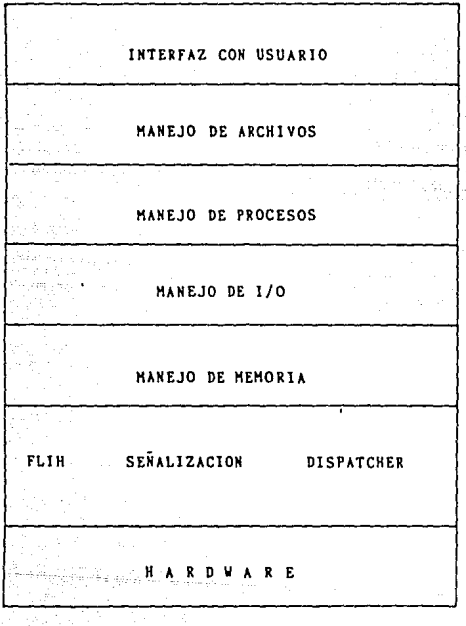
Otra idea importante tomada por MULTICS de CTSS es la del manejo del espacio en memoria principal. Cuando se encontraba un usuario en memoria principal, CTSS procuraba sólo desocupar la cantidad de memoria requerida para que el siguiente usuario pudiera trabajar, de esta forma cuando el usuario regresaba ya no era necesario trasladar toda su información a memoria principal, debido a que parte de ella se encontraba ahí.

Pues bien, MULTICS generalizando la idea de CTSS, utilizaba tres niveles de memoria y apoyandose en rutinas de paginación, no siempre requería sacar toda la información que se encontraba en la memoria principal, por lo que las probabilidades de que la información requerida se encontrara en memoria , eran muy grandes.

MULTICS fue sin duda un logro muy importante dentro de la línea evolutiva del sistema operativo UNIX.

En forma análoga a como se realizó con CTSS, se pueden esquematizar los distintos niveles de "software" que conforman al sistema operativo MULTICS en la Fig. 3.7.

Fig. 3.7 Representación de los niveles de MULTICS.



4. SISTEMA OPERATIVO UNIX.

La primera versión del sistema operativo UNIX surge en el año de 1969 y fue desarrollada por Ken Thompson, quien trabajaba para los laboratorios Bell.

Uno de los principales objetivos que se pretendía alcanzar con UNIX, era proporcionar a los programadores un ambiente de trabajo agradable, en el que se concentraran en su trabajo y no tuvieran demasiados problemas en el manejo del sistema operativo.

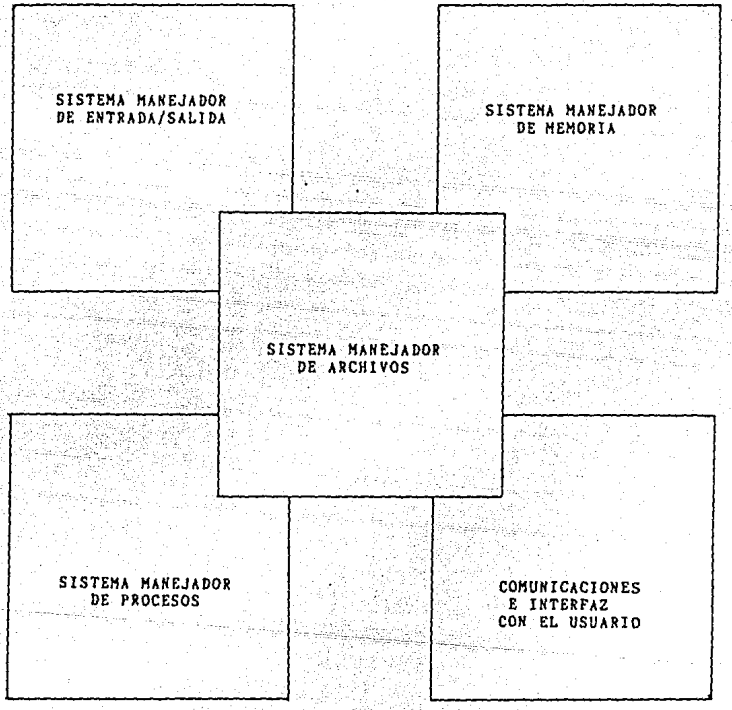
UNIX es un sistema operativo de propósito general, el cual recopila varias ideas de sistemas operativos ya existentes y de esta conjunción surgió un sistema muy versátil, elegante y fácil de manejar. Se puede citar a MULTICS y CTSS como sistemas operativos que tuvieron influencia en UNIX.

La primera versión de UNIX corrió en una computadora de Digital Equipment Corporation PDP 7, posteriormente fue reescrito para pasarlo a una PDP 11. Por el año de 1973, UNIX se convierte en el primer sistema operativo escrito, casi en su totalidad, en un lenguaje de alto nivel. Dennis Ritchie había desarrollado un nuevo lenguaje de programación llamado C, y entre él y Thompson reescribieron UNIX en este lenguaje.

Al reescribir UNIX en lenguaje C, el sistema resultó más fácil de entender y fue posible introducir mejoras de multiprogramación y la facilidad de compartir código reentrante entre varios programas usuarios. Adicionalmente UNIX se convierte en un sistema operativo fácil de instalar en computadoras con arquitectura distinta a la PDP 11. Es por esto que al aparecer en 1980 las microcomputadoras con microprocesadores de 16 "bits", UNIX se convierte en la solución al problema de reescribir todo el "software" ya existente. La Fig. 4.1 muestra los principales módulos que forman UNIX.

A continuación se presentará someramente la arquitectura de la PDP-11 para la que UNIX fue desarrollado.

Fig. 4.1 Los módulos del sistema UNIX.



4.1 Arquitectura de la PDP 11.

La arquitectura de la PDP 11 presenta una característica sobresaliente; el UNIBUS, que es un canal de datos único. Se tiene conocimiento que fue el primer canal de datos para minicomputadoras. La función de este canal es permitir a los dispositivos enviar, recibir o intercambiar datos sin intervención del procesador ni de "buffers" en memoria.

El UNIBUS es una ruta común que conecta al procesador, la memoria y todos los dispositivos periféricos de entrada/salida. Cada dispositivo en el UNIBUS tiene asignada una dirección que le permite ser accesado como si fuera una memoria; facilitando de esta manera la manipulación de los dispositivos periféricos.

En la PDP 11 los elementos de la memoria, tales como memoria principal o cualquier memoria de "read-only", tienen direcciones ascendentes que empiezan en cero, a diferencia de los registros que almacenan datos de entrada/salida o del estado de los dispositivos periféricos, los cuales tienen sus direcciones en los 8 "kbytes" más altos del espacio direccionable.

El UNIBUS consiste de 56 líneas a las cuales todos los dispositivos incluyendo el procesador estan conectados en paralelo. De estas líneas 51 son bidireccionables y 5 son unidireccionables.

La comunicación entre dos dispositivos mediante el UNIBUS sostiene una relación de maestro-esclavo; durante cualquier operación del canal, un dispositivo controla el canal mientras se comunica con otro dispositivo. Cuando dos o más dispositivos tratan de ganar control del canal al mismo tiempo, existe una prioridad que decide a quien se le cederá dicho control.

A la comunicación que existe entre dos dispositivos se le llama ciclo de canal y en cada ciclo sólo se puede transferir una palabra o "byte". Un ciclo de instrucción incluye uno o más ciclos del canal.

La PDP 11 es una máquina de palabra de longuitud variable y puede trabajar con números de 16 o 8 "bits". La palabra de 16 "bits" se divide en dos "bytes" de 8 "bits" cada uno. Las posiciones de 0-7 constituyen el "byte" bajo y las posiciones del 8-15 constiutyen el "byte" alto.

El procesador central de la PDP 11 , está compuesto por tres bloques: la unidad de control, la unidad aritmética y los registros de propósito general.

La PSW (procesor status word), es la palabra de 16 "bits" en la cual se almacena información sobre el estado del procesador. La información contenida incluye los modos de operación actual y anterior del procesador, la prioridad, un indicador y códigos de condición que describen el resultado de la última instrucción ejecutada.

Los registros generales,como su nombre lo indica, pueden ser usados como:

- Acumulador.- Cuando una suma es acumulada en el registro de propósito general.

- Apuntador.-Cuando el registro de propósito general apunta al operando.

- Registro de autoincremento.-Cuando el registro de propósito general apunta al operando, la dirección es usada y después incrementada automáticamente.

- Registro de autodecremento.-Cuando el registro de propósito general apunta al operando, la dirección es automáticamente decrementada y después usada.

- Registro de índice.-Cuando el registro de propósito contiene el valor de un índice que es sumado a una dirección base, esta suma la dirección del operando.

Todos los modos de direccionamiento usados en la PDP 11 están asociados con los registros de propósito general y puede accederse cualquier dato con la ayuda de alguno de estos registros.

Los registros 6 y 7 tienen un uso particular. El registro 6 es el apuntador al "stack" y contiene la dirección de retorno en caso de una interrupción, mientras que el registro 7 es el apuntador al programa y contiene la dirección de la siguiente localidad a ser referenciada.

4.2 Sistema Manejador de Entrada/Salida.

El sistema de entrada/salida de UNIX posee características semejantes al sistema de entrada/salida de MULTICS. Al igual que este último, UNIX esconde a sus usuarios detalles de "hardware" de los dispositivos a manejar, implementando facilidades en su sistema de archivos, así como en las interfaces diseñadas para el manejo de protocolos.

El sistema de entrada/salida de UNIX es, sin embargo, más complejo y contempla básicamente tres tipos de dispositivos de I/O. La interfaz de "socket", los dispositivos de bloque o estructurados y los dispositivos de carácter o no estructurados.

Como la interfaz de "socket" está muy relacionada con redes y protocolos se presentará posteriormente.

Antes de continuar, es conveniente mencionar la existencia de dos elementos importantes: uno es el código para manejadores de dispositivos y el otro son los manejadores de dispositivos. Tanto el código como los manejadores de dispositivos tienen la función de direccionar dispositivos específicos así como aislar los detalles de pistas y cilindros. Existe un manejador por cada dispositivo específico.

Como se recordará este manejo de los dispositivos fue implementado en MULTICS, UNIX toma la idea y la perfecciona.

Los dispositivos de bloque se caracterizan por ser dispositivos que se direccionan en bloques de 512 "bytes", utilizan un "buffer caché" para transmitir sus bloques; puede accederse a ellos por medio de archivos especiales o en forma indirecta por medio del sistema de archivos.

Tanto discos como cintas magnéticas son considerados dispositivos de bloque.

Los dispositivos de caracter se distinguen por no utilizar el "buffer caché" para sus transferencias. Esta definición es amplia y abarca a distintos tipos de dispositivos como son terminales, impresoras, interfaces distintas (a excepción de aquélla que maneja las redes).

Para los dispositivos estructurados y no-estructurados se maneja un arreglo de varias entradas para los distintos manejadores de dispositivos. Cada dispositivo se encuentra representado en el sistema de archivos tanto por una clase como por un número. El número se divide en dos partes, de las cuales la primera tiene como objetivo apuntar, en el arreglo correspondiente a su clase (estructurado o no-estructurado), el manejador de dispositivos apropiado; la segunda parte es utilizada por el manejador de dispositivos e interpretada como una partición lógica o una terminal.

El uso del arreglo de entradas para cada clase es conocido como tabla de configuración y constituye la única conexión entre el código del sistema y el manejador del dispositivo. Esta característica proporciona la facilidad de reconfigurar el sistema sin mucho problema, además de ser una de las características que permite a UNIX ser un sistema portable.

UNIX desarrolla un concepto utilizado anteriormente en MULTICS y que viene a incrementar la eficiencia del S.O.; una pequeña memoria conocida como memoria de "buffer" bloqueado, que tiene como objetivo reducir el número de transferencias de I/O que requiere el sistema de archivos.

Al emplear este "buffer" la información que más se utiliza sólo es leída una vez, ya que en las ocasiones sucesivas en las que sea requerida esa información será tomada desde la memoria del "buffer". En forma similar cuando se requiere escribir un bloque de éstos, la escritura se realiza en el "buffer" y no en los dispositivos magnéticos. Sin embargo, cuando se trata de escribir la información, es necesario tomar medidas de precaución para el caso de que el sistema falle.

Por esta razón la transferencia del "buffer" al dispositivo se realiza cada determinado tiempo y no hasta que el proceso ha sido completado.

La memoria de "buffer" bloqueada está constituída de varias etiquetas. Estas etiquetas contienen un número de dispositivo, el número de un bloque en el dispositivo, un apuntador a la memoria física, el tamaño de la memoria física apuntada y la cantidad de memoria que contiene datos.

El manejo de esta memoria es dinámico por lo que las etiquetas se encuentran asociadas por medio de listas. Estas listas tienen como objetivo guardar información sobre las etiquetas y de esta forma administrar la memoria de "buffer" bloqueada. Para poder dar un seguimiento al manejo dinámico de la memoria, es necesario dar algunas definiciones:

- "Buffers" Reservados.- Contienen información que se encuentra en memoria principal.

- "Caché".- Contiene bloques que no se encuentran en uso pero que pueden ser solicitados nuevamente.

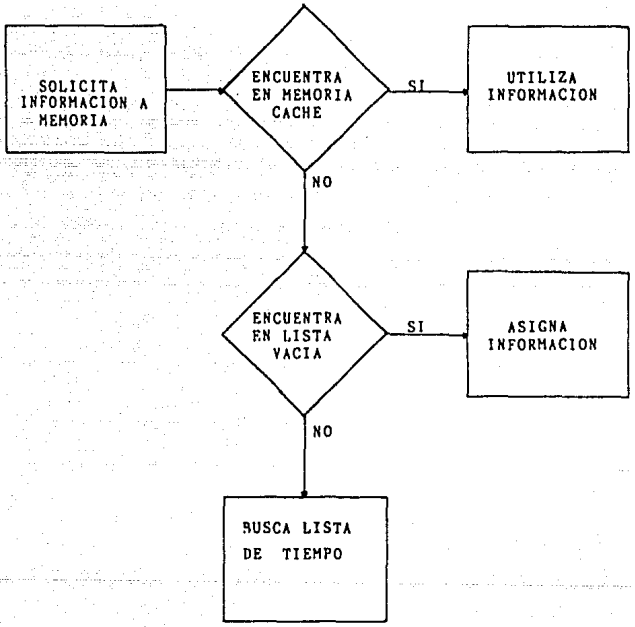
- Lista de Tiempo.- Contiene bloques con poca probabilidad de ser utilizados.

- Lista Vacia.- Contiene las etiquetas las cuales no tienen asociados bloques de memoria o de discos.

- Lista de manejadores de dispositivos activos.- Cada dispositivo de bloque tiene una lista de "buffers" en donde las entradas/salidas están activas o suspendidas.

En la Fig. 4.2 se muestra en forma gráfica la secuencia que sigue el sistema para la administración de la memoria.

Fig. 4.2 Secuencia de la administración de la memoria.



Es conveniente mencionar que los dispositivos de bloque tienen también una interfaz de caracter. Esta interfaz puede accederse a través de archivos especiales y lo que en realidad hace es transferir los datos directamente entre el espacio de direcciones virtual de un usuario y el dispositivo específico.

Después de haber analizado en forma general el manejo de entrada/salida de los dispositivos de bloque, se verá el manejo de los dispositivos de caracter.

En realidad el sistema de caracter maneja también bloques de información, pero a diferencia del sistema de bloques, son por lo general más pequeños. Estos bloques se encuentran en distintas listas concatenadas entre sí y que clasifican los bloques según su estatus.

Al momento que se desea escribir, por medio de una rutina se colocan los bloques de caracteres en una cola de salida para el dispositivo específico al cual se quiere transmitir la información.

Para la escritura de caracteres sucede algo semejante, con la particularidad de que los manejadores de terminales soportan el uso de dos tipos de colas de entrada; la cola "RAW" y la cola "CANONICA". Las colas "RAW" reciben los caracteres conforme llegan de las terminales, en este caso no es necesario que el usuario realice una llamada a la subrutina de lectura. Cuando sucede que el usuario realiza una llamada a la rutina de lectura, los datos son extraídos de la cola "CANONICA". En caso de no existir datos en esta cola, el proceso que solicitó la lectura será suspendido hasta que existan suficientes datos en la cola "CANONICA" para realizar la transferencia de datos.

4.3 Sistema Manejador de Procesos

La forma de manejar los procesos por parte de UNIX, es una de sus características más distintivas. Las decisiones que se deben tomar para la asignación de los recursos, la identificación de los procesos y la administración del CPU, son simplificados haciendo con esto a UNIX uno de los sistemas más eficientes en el manejo de usuarios interactivos.

Dentro de UNIX los usuarios efectuan sus programas en un ambiente conocido como proceso usuario. El proceso usuario ejecuta sus programas, sin embargo cuando se requiere una función especial del sistema, el proceso usuario llama al sistema como si fuera una subrutina. De esta forma el proceso usuario aloja entonces rutinas del sistema y pasa a ser llamado proceso de sistema. Por lo anterior se puede concluir que el proceso usuario y el proceso de sistema son el mismo programa en ejecución pero en tiempos distintos.

En realidad no existen bloques de control dentro del espacio de direccionamiento de un proceso usuario, sino que toda la información necesaria se encuentra alojada en el núcleo del proceso usuario.

La estructura conocida como estructura de proceso tiene como objetivo guardar información relativa al proceso; ésta es identificador de usuario, prioridades, apuntadores a otras estructuras distintas. Por otra parte el espacio de direccionamiento de un proceso usuario está dividido en tres segmentos: el segmento de texto, el segmento de datos y el segmento de datos del sistema.

Cualquier proceso usuario se ejecuta desde el segmento de texto, el cual sólo se permite utilizarlo de lectura. Es por esta característica que este segmento puede ser compartido por varios procesos usuarios.

Para almacenar la información relativa a los segmentos de texto, existe una tabla que contiene, entre otras cosas, la dirección de memoria secundaria y, en el caso de que el segmento esté cargado, contiene la dirección de memoria principal.

Anteriormente se mencionó que la estructura del proceso tiene apuntadores a ciertas estructuras, pues bien la tabla de textos es direccionada desde la estructura del proceso.

Como su nombre lo dice, el segmento de datos aloja los datos relativos al proceso. Este segmento no es compartido entre varios procesos usuarios y su longitud depende del sistema.

Por último, el segmento de datos del sistema contiene toda la información relativa al proceso cuando se encuentra activo. Como medida de protección este segmento no puede ser accesado desde el proceso usuario.

Se mencionó que un proceso es propiamente un programa en ejecución y para que la ejecucion de este se realice UNIX tiene una filosofía muy particular. El primer programa que se ejecuta es el INIT, al cual se le asigna el identificador de proceso # 1; este programa tiene como responsabilidad crear un proceso para cada terminal, además de abrir las terminales adecuadas para entrada/salida en los archivos cero, uno, y dos.

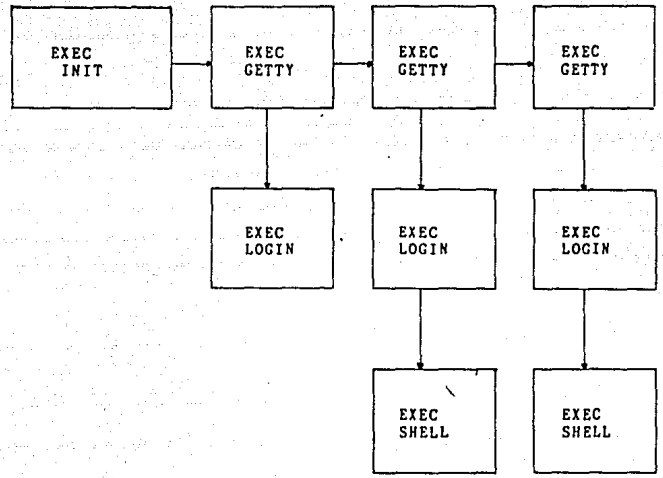
En UNIX el archivo cero denota al archivo estándar de lectura, el archivo uno es el archivo estándar de salida y el archivo dos es el que se mantiene asociado a la terminal para los casos en que el direccionamiento de salida se realizó hacia otro archivo distinto al estándar.

Una vez que cada terminal está disponible para uso interactivo, INIT ejecuta un nuevo proceso conocido como GETTY. Este proceso se encarga de recibir el nombre del usuario y pasarlo como parámetro al programa LOGIN, éste recibe el "password" del usuario y determina si el acceso al usuario está permitido, asignando un identificador al proceso que se crea con la entrada del usuario del sistema. En este punto se genera una ejecución al programa SHELL. Este programa es un intérprete de comandos que lee la información tecleada por el usuario y la interpreta como una solicitud para ejecutar otro programa. Esta secuencia es explicada graficamente en la Fig. 4.3.

En UNIX cada proceso se crea sólo con la llamada de FORK. Este programa al momento de su ejecución crea dos procesos idénticos pero independientes con una copia del espacio de direccionamiento original y compartiendo todos los archivos que se encontraban abiertos. Es el FORK el que asigna el identificador al proceso haciendo una distinción; al proceso hijo le asigna un valor de 0, mientras que el valor del padre nunca es 0,de esta forma el sistema identifica cuál proceso se está ejecutando.

Aquí es importante mencionar la función que realiza el identificador de usuario. Este valor es utilizado por el núcleo del sistema operativo para poder permitir accesos a ciertas llamadas del sistema, así como accesos a archivos. Estos usuarios pueden ser agrupados de acuerdo a sus características y privilegios. Al igual que MULTICS, UNIX se apoya en esta identificación de los usuarios para administrar sus recursos, sin embargo, existe una característica sobresaliente que hace de UNIX un sistema más evolucionado.

Fig. 4.3 Secuencia de los procesos.



Los identificadores de los ususarios se pueden dividir en dos clases: el identificador de usuario efectivo y el identificador de usuario real. El primero es utilizado para el acceso a los archivos, es decir, si al estar ejecutándose un proceso realiza una llamada a un archivo y éste posee una indicación de SETUID, el sistema asignará al proceso que está ejecutándose el identificador de usuario del propietario del archivo. Con esto el proceso usuario que realizó la llamada podrá tener privilegios adicionales. El identificador de usuario real permite , ante la explicación anterior, saber quién era realmente el usuario antes de haber adoptado otro identificador de usuario.

El sistema operativo UNIX está diseñado para dar prioridad a los procesos que corren en forma interactiva. Al igual que sus antecesores, UNIX presenta un esquema de prioridades, en el que se basa la distribución de los recursos de la CPU.

A los procesos que ejecutan entrada/salida o alguna tarea importante, se les asignan prioridades negativas, en tanto los procesos comunes de usuario son marcados con prioridades positivas y tienen menos posibilidades de ser elegidos en comparación con los procesos de sistema.

En forma semejante a como se maneja en MULTICS, UNIX previene el problema de que un "job", sólo por su baja prioridad, no tenga acceso a la CPU. Mediante la utilización de un sistema dinámico de prioridades, UNIX analiza sus procesos, entre mayor tiempo de CPU acumule un proceso, la prioridad del misma se verá disminuida. De igual manera, si un proceso no ha tenido acceso a la CPU el tiempo suficiente, el sistema aumentará su prioridad para que pueda concluir su ejecución.

La sincronización de procesos consiste en decidir a qué proceso se le dará entrada a la CPU. Para esto el sistema debe tomar en cuenta varios factores. El primero de ellos se basa en ver si un evento cualquiera, ya sea una operación de entrada/salida o la espera de cierto dispositivo ha ocurrido. Y segundo, en caso de que varios procesos esperen por el mismo evento cuál de ellos será el elegido para continuar su ejecución.

UNIX representa a los eventos con un número arbitrario y se escogen que sean direcciones de tablas asociadas con esos eventos. Cuando uno de esos eventos termina o es cubierto en la tabla, entonces el sistema lo señala como atendido. En este momento, si existían procesos que esperaban por este evento, son puestos en alerta y solo uno de ellos será escogido dependiendo de sus prioridad.

En UNIX los eventos no tienen una cantidad de memoria asociada con ellos, sino que los eventos existen simplemente al ser usados. Debido a esto UNIX debe implementar ciertas características que le ayuden a resolver ciertos problemas como es:

- El que sean puestos en alerta procesos que posiblemente no les sirve el evento por completo que acaba de ser señalado.

- Preveer que un proceso quede esperando por un evento que ya fue previamente señalado.

Una de las medidas preventivas que toma UNIX, es aumentar la prioridad del procesador durante una sección crítica de tal forma que no puedan ocurrir interrupciones. Con esto UNIX asegura que ningún proceso pueda quedar esperando un evento que no ocurre nunca. Ahora bien, para el caso de que un proceso sea puesto en alerta antes de que el evento que requiere se encuentre listo, el "scheduler" de UNIX realiza una selección de los procesos de acuerdo a sus prioridades, por lo que es posible que el proceso ya se encuentre esperando para ser ejecutado, sin embargo el "scheduler" tiene otros procesos a quienes atender antes que a él.

4.4 Sistema Manejador de Memoria.

En un principio el desarrollo del sistema operativo UNIX, en cuanto al manejo de memoria, se vió limitado por el "hardware", pero UNIX es un sistema operativo vigente, es decir, continuamente se le incorporan nuevas facilidades y se atacan los problemas que van surgiendo en la práctica cotidiana.

Una de estas facilidades desarrolladas para UNIX, la constituyen sus algoritmos de paginación. En versiones anteriores de UNIX, se recurría al traslado de información desde memoria principal a memoria secundaria. Este intercambio conocido como "swapped" se ve minimizado en las nuevas versiones, al apoyarse en la paginación para el manejo de la memoria.

Un sistema de memoria virtual paginada tiene como objetivo minimizar la fragmentación de la memoria permitiendo que más procesos permanezcan en memoria principal. Es importante mencionar que de acuerdo a su ubicación, los distintos bloques en los que se ve dividida la memoria reciben distintos nombres. Los "frames" son los bloques de información que se encuentran en la memoria principal, mientras que las páginas son los bloques de información pero dentro del medio magnético.

Se mencionó anteriormente que cada proceso en UNIX maneja tres segmentos de información: un segmento de texto, un segmento de datos de usuario y un segmento de datos del sistema. Esta información debe residir en memoria, sin embargo no siempre es posible.

La implantación de algoritmos de paginación se lleva a cabo en función de las características del "hardware". Dentro de los algoritmos manejados por UNIX para paginación, se encuentra una modificación al LRU (Least Recently Used). En forma general el algoritmo se comporta de la siguiente manera:

Un programa especial "Clock Hand", recorre todos los "frames" de memoria disponibles para paginación. Al pasar marca las páginas como inválidas. Si la página es solicitada por algún proceso o usuario, antes de que el programa especial la recorra nuevamente, esto provoca que ocurra una falta de página y es necesario marcar la página nuevamente como valida. Pero si al momento en que el programa especial recorre de nuevo todos los "frames" de memoria, la página marcada previamente como invalida no ha sido referenciada, esta página será asignada para otros usos.

La información relativa a los "frames" de memoria principal se encuentra almacenada en un mapa de memoria. Aquí se indica cuáles de los "frames" están libres y cuáles estan ocupados, indicando en estos ultimos que procesos están alojando.

Cuando un proceso en ejecución requiere de una página y ésta no se encuentra dentro de los "frames" de memoria principal, ocurre lo que se conoce con el nombre de falta de página. En este momento el núcleo del sistema implementará las acciones necesarias para que la información requerida se encuentre en memoria. Aquí pueden ocurrir tres cosas: que la página solicitada no se encuentre en memoria principal; que la página solicitada se encuentre en memoria principal pero que esté marcada como inválida; y que la página se encuentre en la lista de "frames" libres.

En el primer caso el núcleo buscará un "frame" libre y trasladará sobre la información desde disco. En el segundo caso la página será marcada como válida y el proceso tendrá en memoria principal la información que requiere. Para el tercer caso debe recordarse que el segmento de texto de un proceso en UNIX puede ser compartido por más de un usuario. Pués bien, cuando esto ocurre y el último proceso que compartía esta información termina, el núcleo respeta la información en los "frames" correspondientes al segmento de texto y sólo los coloca como "frames" libres. De tal suerte que si llega un proceso nuevo que utilice esta información puede recuperarla de los "frames" libres sin necesidad de cargarlas desde disco.

Para el caso en el cual se requiere sacar de memoria una página existen varios factores a considerar. El encargado de decidir cuáles páginas deben salir es el programa especial "Clock Hand". Este programa recorre el mapa de memoria, checando el estatus de los distintos "frames". Si el "frame" está vacio o está siendo usado, no es marcado en ninguna forma: pero si el "frame" no está siendo usado pero tampoco se encuentra marcado como inválido, entonces es marcado como inválido con opción a ser recuperado.

Sin embrago, si el "frame" se encuentra marcado como invalido, se trasladará la información a disco (en caso de haberse modificado) y ese "frame" pasará a formar parte de la lista de "frames" libres.

El programa especial "Clock Hand", forma parte del proceso "Pagedaemon", el cual tiene como objetivo mantener el nivel de la lista de "frames" de memoria libres, lo suficientemente alto para no ocasionar problemas al sistema por la falta de "frames" libres. Este proceso es disparado cuando ciertos límites de "frames" libres, que son especificados al momento de arranque del sistema, son superados.

Ahora bien, a pesar que el método de paginación antes descrito optimiza muchísimo el manejo de la memoria, es importante estar concientes que este método puede llegar a ocasionar contención y convertirse en un problema serio.

Muchos sistemas previos a UNIX y algunos actuales utilizan un esquema conocido como "Swapping" o transferencia de información entre memoria principal y secundaria. Este esquema funciona de la siguiente manera:

Los segmentos de datos de usuario y de datos del sistema son almacenados en forma continua en memoria principal. La asignación de memoria y del espacio para la transferencia es manejado la primera vez. Por lo que cuando por razones de crecimiento el espacio del proceso en la memoria principal debe ser incrementado, el sistema se ve obligado a localizar en memoria principal un espacio lo suficientemente grande para que el proceso sea copiado. Cuando no se puede encontrar este espacio, el proceso es copiado a memoria secundaria ya con su nuevo tamaño, y ahí esperará a entrar nuevamente a memoria principal.

El segmento de texto, por ser compartible entre varios usuarios, no necesita ser trasladado a memoria secundaria. Esto reditúa en dos beneficios. El tráfico de transferencias se ve reducido y la cantidad de memoria disminuye al compartir los procesos el segmento de texto.

Los criterios en los cuales se basa el "scheduler" para decidir qué procesos deben permanecer en memoria principal y cuales deben salir de la misma son muy sencillos.

Aquellos procesos que han permanecido ociosos por algún tiempo, aquellos que han ocupado la memoria principal por mucho tiempo o aquellos procesos que están esperando por la ocurrencia de un evento lento , son los candidatos más próximos a salir de memoria principal.

Por otra parte los eventos más aptos para ocupar la memoria son aquellos que han permanecido fuera por algún tiempo considerable y aquellos procesos que son pequeños y no consumen mucha memoria.

4.5 Sistema Manejador de Archivos

El sistema de archivos de UNIX está integrado por tres tipos de estructuras: los directorios, los archivos ordinarios y los archivos especiales. La filosofía en que se basa el sistema de archivos de UNIX no es nueva; es una implementación del sistema de archivos de MULTICS, mejorando en mucho el manejo de los archivos dentro de los directorios pero conservando las bases del sistema jerárquico utilizado en MULTICS.

Un archivo ordinario lo constituyen una secuencia de "bytes" que contienen la información que el usuario asigne. Al igual que en MULTICS, UNIX no impone ninguna estructura a sus archivos, dejando con esto que el usuario controle esta estructura desde los programas usuarios. Todos los archivos se encuentran organizados dentro de una estructura jerárquica. Los directorios son realmente archivos y estos contienen a su vez directorios y/o archivos.

Los directorios contienen toda la información necesaria para el acceso de los archivos, por lo que el acceso a los directorios esta restringido solo al sistema, ocasionando que ningun programa no privilegiado escriba sobre ellos.

Para poder identificar los archivos, el sistema de archivos de UNIX utiliza el concepto de trayectorias, que no es otra cosa sino una secuencia de nombres de directorios, los cuales debe accesar el sistema antes de poder llegar al archivo en cuestión. Para aclarar esto se observará un ejemplo; se tiene el archivo ejem1 ubicado en el directorio dir3; a su vez este directorio se encuentra ubicado en el directorio dir1 que a su vez se encuentra ubicado en el directorio "Root" del sistema. La forma en que el sistema de archivos identifica al archivo ejem1 es:

/ dir1 / dir3 / ejem1

Es importante observar que el directorio "Root" por ser la raiz de todos los directorios, no se especifica.

Hay dos tipos de trayectorias, aquella que contempla al archivo desde el directorio "Root", como en el ejemplo anterior; a este tipo de trayectoria se le conoce como trayectoria absoluta. El otro tipo de trayectoria es aquella que sólo referencia al archivo, y es conocida como trayectoria relativa.

Existe un concepto conocido como "Links" el cual conserva UNIX tal cual fue utilizado en el sistema operativo MULTICS. El sistema permite que un archivo sea conocido bajo distintos nombres en varios directorios; para que esto se realice el sistema utiliza los "Links".

El archivo no existe físicamente en ningún directorio en particular, sino que en el directorio se encuentra el "Link" de archivo y un apuntador a la información que describe. Con esto se logra que el archivo sea independiente del directorio.

Existe un tercer tipo de archivo ya antes mencionado. Es aquel conocido como archivo especial. Estos archivos son en realidad una interface entre el núcleo y los dispositivos de entrada/salida. Estos archivos pueden ser accesados para lectura y escritura como cualquier archivo ordinario, la diferencia consiste en que cualquier solicitud de lectura/escritura se traduce en la activación del dispositivo asociado.

El sistema de archivos cuenta con varias rutinas para el manejo de los archivos. Cualquier operación que se quiera realizar con un archivo, abrirlo, cerrarlo, crear un nuevo archivo, establecer "Links" y desligarlos, todas las operaciones tienen en el sistema una subrutina correspondiente que realiza la operación.

Existe un detalle en la subrutina que abre los archivos así como en aquella que los crea; ambas generan un valor entero, conocido como el descriptor de archivos, el cual se utiliza como parámetro cuando se requiere realizar una operación sobre el archivo.

UNIX maneja el concepto de sistema de archivos virtual, que no es otra cosa que varios sistemas de archivos físicos, cada uno en un dispositivo distinto. En realidad existe un sistema de archivos básico siempre disponible y en la medida en que son requeridos los otros sistemas es que se van integrando en forma jerárquica.

Si se recuerda un poco, dentro del sistema de entrada/salida existe una facilidad que realiza la transferencia de la información conocida como el sistema de entrada/salida bloqueado. Pues bien todos los accesos al sistema de archivos virtual se efectúan por medio del sistema de entrada/salida bloqueado.

Físicamente cada disco se encuentra dividido en varios discos lógicos que contienen los sistemas de archivos. El disco lógico se encuentra dividido en varios sectores. Cada uno aloja programas y/o parámetros del sistema de archivos. Dentro de los sectores en que se encuentra dividido el disco lógico, existe un arreglo de elementos conocidos como "Nodos-I", los cuales contienen la mayor información sobre un archivo en específico.

Existe un detalle especial en el direccionamiento de los datos en un archivo. Cada "Nodo-I" direcciona 13 dispositivos, las primeras 10 direcciones apuntan a los diez primeros bloques del archivo; si el archivo en cuestión es mayor de diez bloques, las onceava, doceava y treceava direcciones son apuntadores a bloques indirectos que contienen 128 direcciones adicionales.

El directorio es el encargado de mantener una secuencia de pares formados por el nombre del archivo y su "Nodo-I". Por otra parte el núcleo del sistema aloja información referente a los archivos bajo otra filosofía. El núcleo almacena el número de dispositivo y el "Nodo-I" para identificar el archivo.

4.6 Interfaz con el usuario.

UNIX es un sistema operativo que proporciona a sus usuarios una interfaz muy diferente a la proporcionada por los sistemas analizados anteriormente.

El interprete de comandos en UNIX, el "shell", es un proceso de usuario como cualquier otro; este proceso es un programa en si, y como tal puede ser sustituido. A diferencia de otros sistemas operativos UNIX no tiene incorporado a su núcleo el intérprete de comandos, por lo que existen varios "shells" que comparten la sintaxis básica y que aplican para UNIX.

Debe mencionarse que existen "shells" orientados para el manejo de pantallas y menús, pero en realidad los más comunes son los orientados hacia comandos de línea.

Para indicar que el "shell" esta listo para recibir comandos, presenta un "promt" y el usuario digita un comando en una sola linea. La mayoria de los comandos pueden tener argumentos que deben digitarse después del nombre del comando y separados por blancos.

Cualquier comando se encuentra representado por código ejecutable. Este código puede encontrarse en cualquier directorio de una lista que contiene el "shell" conocida como trayectoria de busqueda.

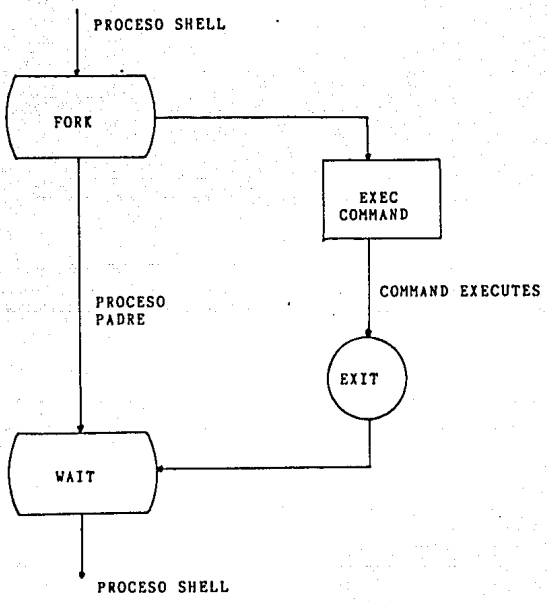
La ejecución de un comando se realiza por una llamada al "FORK" seguida de un "EXEC" del código objeto. En este momento el "shell" entra en un estado de "WAIT", hasta que la ejecución del comando se completa.

Es posible que la ejecución del "shell", no sea suspendida durante la ejecución de un comando. A esta forma de ejecución de un comando se le conoce como "background".

El "shell" además de ser un interprete de comandos, es un lenguaje de programación que permite la creación de procedimientos para ser interpretados por el "shell".

El "shell" de UNIX, como lenguaje para programar procedimientos, ofrece la facilidad de contar con sentencias para controlar secuencias repetidas, probar condiciones y conservar valores temporales. Esto permite realizar procedimientos complejos sin perder el control de los mismos. Ver Fig. 4.4.

Fig. 4.4 El "SHELL" de UNIX.



4.7 Sistema Manejador de Comunicaciones

Dentro de las aportaciones de los sistemas de cómputo recientes se encuentra la integración de programas para el manejo de las comunicaciones.

Esta aportación, si bien, no es posible verla como un desarrollo de los sistemas operativos, sí representa un cambio que afecta tanto a los sistemas operativos como a la forma de trabajo.

La comunicación en un sistema de cómputo actual, representa un incremento en la productividad, ya que es posible no solo distribuir el procesamiento entre varias máquinas, sino también compartir datos y recursos.

Es conveniente señalar dos tipos de comunicación en los sistemas de computó recientes: el primero ocurre cuando un programa que se encuentra ejecutándose en una máquina, interactúa con otro programa que corre en otra máquina. El segundo se presenta cuando dos procesos se encuentran corriendo en la misma máquina y se comunican entre sí. Al primer tipo se le conoce como "networking" mientras que al segundo se le conoce como "interprocess comunication".

Las comunicaciones entre-procesos que proporciona UNIX, son limitadas debido a que no permite que la memoria sea compartida. He aquí una situación en la cual si se contára con la facilidad de compartir memoria, los accesos serían mucho más rápidos.

UNIX implementa una estructura conocida como "<PIPE>", mediante la cual establece la comunicación entre procesos. Esta estructura permite que dos procesos se comuniquen entre sí por medio de un flujo de "byte" unidireccional, el cual se implementa como si fuera un archivo común, pero considerando ciertas excepciones.

Un "<PIPE>" no está contemplado en el sistema de archivos, por lo que su correspondiente descriptor de archivos es manejado directamente por las subrutinas que hacen uso de él, por ejemplo "READ", "WRITE", "CLOSE", etc.

El tamaño de un "<PIPE>" es fijo y no es muy grande, aproximadamente 4096 "byte"; debido a esta característica es posible mantener la información contenida en los "<PIPE>" dentro del sistema cache de "buffer" bloqueado en la memoria principal.

Dentro de todos los desarrollos realizados sobre UNIX, existen algunos que implementan al "<PIPE>" como un caso especial de los "sockets", de tal forma que pueda proporcionar tanto facilidades locales como facilidades a nivel "networking".

Para aclarar que es en realidad un "socket", es conveniente analizar sus características y funciones. Un "socket" está definido como un punto final de comunicación, el cual se encuentra limitado por cierto direccionamiento. Este punto final tiene un dominio o espacio de influencia dentro del cual los procesos que intercambian información utilizan el mismo formato de direccionamiento.

Existen varios tipos de socket, que representan las clases de servicio que pueden proporcionarse.

- "Sock Stream".- Proporciona un flujo de datos secuenciales, no duplica ni pierde información. Durante el envio no existen límites de registro.
- "Sock Seqpaket".- A diferencia del anterior, este tipo establece ciertos límites para el registro.

- "Sock Dgram".- Envia mensajes de tamaño variable en cualquier dirección, sin embargo no proporciona seguridad en que el mensaje llegará, no sera duplicado, conservará el orden de envio etc.

- "Sock Rdm".- A diferencia del anterior, este tipo proporciona el servicio de envio de mensajes con la garantía de que llegarán a su destino.

- "Sock Raw".- Proporciona la facilidad de que un proceso accese en forma directa otro protocolo.

A pesar de que estos tipos de "sockets" existen, no todos son soportados en el sistema.

El manejo que realiza el sistema con respecto a los "sockets" es muy parecido al manejo que realiza de los archivos. Un "socket"es creado mediante una llamada del sistema conocida como "socket" y de acuerdo a ciertas especificaciones se genera un entero conocido como descriptor de "socket";sin embargo la estructura del archivo no apunta a una estructura de "nodo-I" sino a una estructura de "socket".

El sistema contempla otras dos subrutinas para manejo de los "sockets". La subrutina "BLIND", que se encarga de apuntar el nombre de "socket" y la subrutina "CONNECT", que inicia la conexión de un proceso cuando desea direccionar un "socket".

La subrutina "LISTEN" es utilizada para dar a conocer al "kernel" sobre la posibilidad de aceptar conexiones, mientras que la subrutina "ACCEPT" permite aceptar dicha conexión.

El modelo que se sigue para que dos procesos puedan comunicarse entre si es muy sencillo. El proceso cliente es aquél que solicita un servicio al proceso servicio. Si es posible proporcionar este servicio el proceso cliente llama a la subrutina "CONNECT" y trata de enlazarlo. Mientras el proceso servicio crea un "socket" y utiliza la subrutina "BIND" para apuntarlo. Posteriormente utiliza la subrutina "LISTEN" y finalmente la de "ACCEPT". De esta forma la conexión se establece y queda abierta la posibilidad para futuros intercambios.

Para poder eliminar la línea de conexión entre los dos procesos es necesario hacer una llamada explícita a la subrutina de "CLOSE".

4.8 Resumen

UNIX se ha conservado como un sistema operativo popular, cuyo principal objetivo es servir como un Sistema Operativo de próposito general. Esta popularidad de UNIX se ha mantenido debido a sus características más relevantes.

La utilización por parte de UNIX de un sistema de archivos jerárquico, le permite organizar la información dentro del sistema de manera que se puede conocer donde y cómo encontrar la información deseada. Asimismo la estructura jerárquica permite a los usuarios interactuar con los archivos, ya que cualquier usuario al iniciar su sesión tiene asignado un directorio y a partir de ahí, el puede consultar, crear y borrar directorios, siempre y cuando su autoridad se lo permita.

Una segunda característica importante de UNIX se encuentra relacionada con la anterior; UNIX maneja una simplificación de archivos poco común en los sistemas operativos. Para él, un archivo es una secuencia de "bytes", no importa la forma en que este archivo sea accesado. Los métodos de acceso así como los bloques de control son eliminados en UNIX, responsabilizando al usuario del manejo del archivo.

Es importante resaltar que UNIX maneja en forma compatible archivos, dispositivos y I/O entre procesos. Tanto los archivos como los dispositivos son tratados tan similar como es posible. Se recordará que cada dispositivo tiene asociado un archivo y que una llamada a este archivo ocasiona en realidad la activación del dispositivo. Se mencionó también que la comunicación entre procesos se realiza por medio de las llamadas de lectura y escritura, y que el canal de comunicación que se establece no es otra cosa que un archivo que será accesado por ambos procesos usuarios.

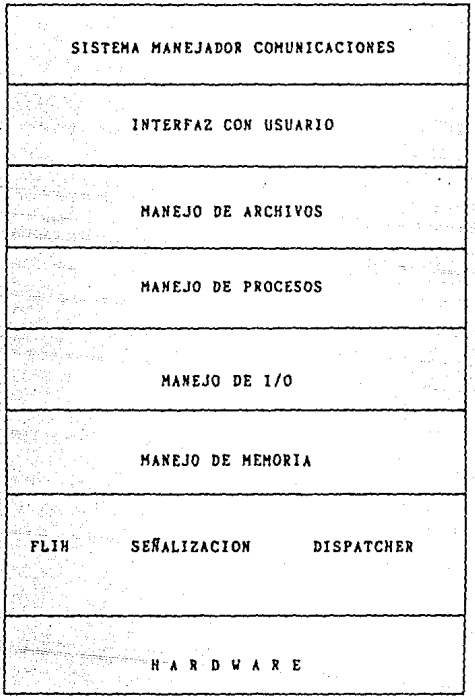
Al implementar UNIX esta semejanza entre archivos y dispositivos y I/O entre procesos, tiene la posibilidad de conservar la misma sintaxis, con lo que un programa prodrá recibir como parámetro ya sea un dispositivo, un I/O entre procesos o un archivo; otra ventaja radica en que los mecanismos de protección son iguales para los tres.

El manejo de la memoria es mucho mas evolucionado con relacion a los sistemas que le anteceden. La paginación previamente utilizada en MULTICS, es utilizada bajo normas similares, pero evitando el tener varios niveles de memoria.

El manejo de las comunicaciones en UNIX es una faceta nueva que sus antecesores no contemplaron. Aquí en este momento se habla ya de dos tipos de comunicaciones, Networkink y Interprocess Comunication. La primera de ellas se enfoca a la comunicación entre distintos host, la segunda de ellas es la comunicación que UNIX permite que exista entre sus procesos.

En forma análoga a los sistemas anteriores, la Fig. 4.5 describe los niveles de UNIX.

Fig. 4.5 Niveles del sistema UNIX.



C O N C L U S I O N E S

Al inicio de este trabajo se planteó la problemática que representaba, en los primeros años del procesamiento electrónico de datos, no contar con sistemas operativos.

El surgimiento de los primeros sistemas operativos se dió como respuesta a varios problemas. El primer problema a solucionar era lograr que la máquina, en especial la CPU, fuera mejor aprovechada; esto es, reducir el tiempo que la CPU permanecía inactiva entre la ejecución de cada proceso.

El segundo problema estába muy relacionado con el primero. Los métodos operacionales de la máquina eran totalmente manuales, el operador decidía que trabajo ejecutar, preparaba las direcciones y dispositivos que requería dicho trabajo, digitaba las instrucciones para que el programa comenzara a ejecutarse y por último al terminar la ejecución, debía recolectar los resultados generados. Este tipo de operación era lento y poco productivo ya que un solo trabajo acaparaba toda la máquina.

Un tercer problema consistía en que los programadores no contaban con una interfaz suficientemente amigable con la máquina, que les permitiera establecer un diálogo agil con ella. Si bien es cierto que la evolución de los lenguajes de programación motivó el desarrollo de programas de aplicación interesantes, tambien es cierto que al tener un sistema operativo, la relación hombre-máquina no era tan ágil como los programadores de estas aplicaciones hubiesen querido: esto lo unico que ocasionaba era frenar el desarrollo de aplicaciones más complejas.

El propósito de realizar un análisis de la línea evolutiva de UNIX, fue observar en los distintos sistemas operativos que forman esta línea, los elementos principales que surgieron como solución a algún problema y ver de que forma evolucionaron hasta constituir un sistema operativo tan completo y versátil como lo es UNIX.

Los sistemas operativos ATLAS y CTSS, corresponden a un mismo período aunque fueron diseñados con fines muy distintos.

ATLAS introdujo el concepto de memoria a un solo nivel, mientras que CTSS fue uno de los primeros sistemas que conjuntó las características de un sistema de tiempo real y un sistema de tiempo compartido.

CTSS introdujó características de diseño especiales, que posteriormente fueron adoptadas y desarrolladas por MULTICS. Entre las características importantes estan:

- Manejar los procesos usuarios con prioridades, de acuerdo a los recursos que utilizaban.
- La política de administración de la memoria principal permitía ahorrar tiempo en la transferencia del control para ejecución de un programa usuario a otro.

El sistema MULTICS, fue el directo decendiente de CTSS. El grupo de trabajo que desarrolló MULTICS, pensó hacerlo un sistema más general que el CTSS. MULTICS no solo tuvo características de un sistema de tiempo real y tiempo compartido, sino además tuvo características que lo clasificaron como un sistema de memoria virtual.

MULTICS tuvo aportaciones muy importantes. La utilización de la paginación y segmentación, fue una solución al problema de no tener suficiente memoria principal.

El manejo de los procesos así como la administración de la memoria, fueron características que MULTICS adoptó de CTSS.

La conceptualización de un sistema de archivos jerárquico junto con el manejo de los dispositivos de I/O, constituyen aportaciones que más adelante UNIX adoptará.

Finalmente se completa la línea evolutiva con UNIX. La primera característica importante que presenta es ser un sistema operativo escrito casi en su totalidad en un lenguaje de alto nivel, lo que lo hace un sistema operativo portable.

Otra características importante de UNIX es su manejo de archivos. Para él, todo puede ser visto como un archivo. Manejar un archivo, un dispositivo o una llamada de I/O entre procesos no implicaba gran diferencia.

Junto con la característica anterior, la utilización de un sistema de archivos jerárquico es considerada también una de las aportaciones más importantes de UNIX.

UNIX es sin duda un sistema operativo que ha recibido mucho impulso en los últimos años. Han sido su portabilidad, sus características de funcionamiento y sus posibilidades de acceso lo que lo han llevado a convertirse en un estandar de la industria de la computación.

B I B L I O G R A F I A

Fundamentals of Operating Systems.
A.M. Lister.
The Macmillan Press LTD.

Operating Systems.
Madnick - Donovan.
Mc Graw Hill.

The Bell System Technical Journal.
"UNIX Time Sharing System".
Vol. 57, número 6.

Computer Arquitecture and Organization.
J. P. Hayes.
Mc Graw Hill.

Programming Systems & Lenguages.
S. Rosen.
Mc Graw Hill.

Perspectivas de la Revolución de los Computadores.
H. Aiken, Ch. Babbage, J. von Neumann
C.E. Shannon, A.M. Turing, W.G. Walter y otros.
Alianza Editorial

Programming the IBM 7090.
J. A. Saxon.
Prentice Hall.

Computer Programming Systems.
P. Fisher - G. Swindle.
Holt, Rinehart and Winston.

The Manchester University ATLAS Operating System.
"Internal Organization".
T. Kilburn, D.J. Howarth, R.B. Payne and F.H. Summer.

The Manchester University ATLAS Operating System.
"User's Description".
D.J. Howarth, R.B. Payne and F.H. Summer.

The ATLAS scheduling System.
D.J. Howarth, P.D. Jones and M.T. Wyld.

From Hardware to Software, and Introduction of Computers.
Graham Lee.
The Macmillan Press LTD.