TECNICAS DE ADMINISTRACION DE RECURSOS EN LOS SISTEMAS OPERATIVOS

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

TECNICAS DE ADMINISTRACION DE RECURSOS EN LOS SISTEMAS OPERATIVOS

LISSETTE DE CARMEN GALVEZ BAIZA

Modelo de trabajo profesional presentado para optar al grado académico de:

Licenciado en Ciencias de la Computación

GUATEMALA

1984

Vo. Bo.:

(f) _____
Licenciado Fabián Pira
Asesor

Tribunal:

(f) _____

(f) _____

(f) _____

Fecha de aprobación:

A mis queridos padres,

Manuel Gálvez Ruiz
Gilda Baiza de Gálvez

a mis tíos

Federico Castellanos Alarcón
Catalina Baiza de Castellanos

y a mis hermanos

Manuel, Aída, Edgar y Livia.

PREFACIO

La descomposición del sistema operativo en administradores de recursos, permite estudiar cada parte del sistema de manera relativamente aislada; dejando de lado los problemas de comunicación entre sus diferentes procesos. Esto quiere decir que tanto si estos últimos funcionan de forma estrictamente secuencial o si lo hacen realmente de forma concurrente entre ellos, las funciones a realizar son las mismas.

En este trabajo se ha puesto énfasis en la descripción de cada administrador y de sus elementos más caracterizados sin preocuparse demasiado de la jerarquía de organización y de comunicación entre ellos. También es preciso decir que en un estudio panorámico de este tipo, las descripciones son obligadamente breves y que cada elemento del sistema operativo se puede llegar a estudiar con toda la profundidad que se desee.

Han quedado fuera de este trabajo partes importantes de los sistemas operativos tales como los monitores de comunicación, sistemas de control de rendimiento, soporte o gestión de Base de Datos y otras; en definitiva, se ha descrito el

núcleo de los sistemas operativos y se ha considerado que estas partes citadas últimamente, forman una segunda capa de aplicaciones montadas de forma muy estrecha sobre el núcleo del sistema.

Tampoco se habla de sistemas operativos concretos ni de conceptos avanzados: máquinas virtuales, sistemas dirigidos al tratamiento de "objetos", sistemas operativos para sistemas multiprocesadores, etc.

Una vez descrito el núcleo del sistema operativo sería necesario continuar con:

- El estudio detallado de los elementos de éste núcleo, enfocando los conceptos mencionados y sus características.

- La organización jerárquica, comunicación y concurrencia entre ellos.

- Las aplicaciones de segundo nivel enlazadas muy estrechamente a ese núcleo.

- Los nuevos conceptos que requieren sistemas operativos con

nuevas funciones y/o estructuras.

El presente trabajo podrá utilizarse como referencia en los cursos universitarios de sistemas operativos, lo que podría solucionar el problema de un texto adecuado en idioma español.

En el apéndice A, se incluye la implementación de un sistema para control del tamaño promedio de particiones, programa que podría utilizarse como base para investigaciones futuras sobre tamaño óptimo de particiones de memoria.

En el apéndice B, se enfoca la configuración simulada de un sistema de 'hardware' pequeño, sobre el cual se emula un sistema operativo de tiempo compartido. Todos los programas desarrollados no fueron incluidos, con el objeto de dar solamente los lineamientos y metodología utilizada en el desarrollo del proyecto.

Será muy satisfactorio el que este trabajo sea de alguna ayuda para las personas que deseen ahondar en el estudio de los sistemas operativos.

Deseo agradecer enormemente, al Lic. Fabián Pira por sus certeras apreciaciones, su interés e incondicional ayuda; al Lic. Marcel Reichenbach, por su tiempo y enorme colaboración para la conclusión de este trabajo; muy especialmente, a Jaime Hazard, por su invalorable ayuda y apoyo; a Edwin Acevedo y a todas aquellas personas que, en una u otra, forma ayudaron a la realización de este estudio.

CONTENIDO

LISTA DE TABLAS Y FIGURAS

I. SISTEMAS OPERATIVOS

A. Introducción a sistemas operativos

La era de la computación electrónica se ha caracterizado por las diferencias en la tecnología de la máquina física ('hardware'), marcando con ello las llamadas 'generaciones'. Actualmente, los avances en conjunto de la máquina física ('hardware'), así como los de los algoritmos diseñados para hacer trabajar ésta ('software'), son los que marcan la actual vertiginosa evolución de las computadoras.

El desarrollo de sistemas complejos de 'software', no empezó sino hasta la tercera generación de computadoras. Esto ha sido motivado por las investigaciones teóricas sobre la asignación adecuada de los elementos del sistema que tienen entidad propia y pueden usarse para un trabajo específico.

La organización, protección y control de acceso y uso de estos elementos o recursos del sistema, es uno de los objetivos principales de un sistema operativo. Esta es una colección de algoritmos que trabajan sobre estructuras de datos, para dar al usuario un conjunto de facilidades que simplifiquen los trabajos de diseño, codificación, puesta a punto y mantenimiento de programas.

La función principal del sistema operativo es controlar los recursos con el fin de que los usuarios los puedan compartir. Compartir es una idea central en todos los sistemas operativos: compartir implica en muchos casos, 'competir' por el uso de un recurso determinado. Como consecuencia de esto, pueden haber interferencias no deseadas entre las demandas de los usuarios. El sistema operativo protege a usuarios y recursos contra este tipo de interacciones. Por otra parte, compartir también permite en muchos casos, cooperar para hacer un trabajo. El sistema operativo controla esta cooperación, evitando que se transforme en interferencia no deseada. De cualquier forma, se puede decir que el sistema operativo es el árbitro que soluciona los conflictos que se puedan producir como consecuencia de la competencia y/o la cooperación.

Existen diferentes tipos de sistema operativos orientados a aplicaciones específicas, lo que viene a aumentar el rendimiento del sistema operativo frente a las demandas de servicio:

A) sistemas operativos orientados a trabajos 'por lotes' ('batch').

B) sistemas operativos orientados a tiempo compartido.

C) sistemas operativos orientados a tiempo real.

Sin embargo, esta clasificación es, actualmente, más aparente que real, ya que hay sistemas operativos en los que se pueden encontrar mezcladas, las características de los tres mencionados anteriormente.

1. Sistemas operativos de procesamiento en lotes:

Es una modalidad en donde los trabajos del usuario son sometidos como lotes secuenciales en los dispositivos de entrada y no hay interacción entre un usuario y su trabajo durante el procesamiento. El usuario está completamente aislado de su trabajo y como resultado, el sistema responde con el tiempo delimitante del mismo trabajo.

Consecuentemente, el sistema operativo puede seguir una política de administración ('scheduling') relativamente flexible.

2. Sistemas operativos de tiempo compartido:

Son sistemas que proveen servicios computacionales para muchos usuarios en línea, concurrentemente; permitiendo a cada usuario interactuar con sus procesos. Este acceso simultáneo es iniciado por el tiempo de procesador y otros recursos

compartidos entre varios usuarios, en una forma que garantice alguna respuesta de cada comando del usuario en pocos segundos. El uso del computador, es asignado a cada proceso del usuario por un pequeño período de tiempo (tajada de tiempo), normalmente en un rango de milésimas de segundo; si el proceso no ha sido finalizado al final de este pequeño período de tiempo, éste es interrumpido y puesto en una cola de espera, permitiendo a otros procesos su turno en la máquina.

En el apéndice B, se encuentra la implementación del algoritmo principal para la optimización de administración de recursos dentro de un sistema operativo de tiempo compartido. Esta implementación podrá ayudar a la mejor comprensión de los fundamentos de trabajo dentro de un sistema operativo de este tipo en específico.

3. Sistemas operativos de tiempo real:

Estos sistemas sirven para manejar procesos externos en línea, que tienen un tiempo estricto de respuesta. Las señales de interrupción provocadas por las órdenes de los procescs externos, producen la atención del sistema; si no son atendidas con prontitud (en microsegundos, milisegundos, o segundos, dependiendo del proceso), el proceso externo es degradado seriamente.

Estos sistemas son utilizados para aplicaciones particulares, por ejemplo, control de procesos.

B. Panorama general y componentes

La mayoría de los sistemas modernos de cómputo, consisten de una interconección complicada de muchos dispositivos de 'hardware', como procesadores centrales, almacenamiento en disco y unidades de cinta, impresoras, lectoras de tarjeta, etc. Para cubrir la complejidad del manejo de todos estos dispositivos, el sistema proporciona programas, o 'software', que hacen del sistema de 'hardware' un uso más conveniente.

Es importante conocer las características básicas de los componentes de un sistema de cómputo, es decir las características del 'hardware' y del 'software'.

1. 'Hardware'

El 'hardware' de un sistema de cómputo está compuesto de una o más unidades de procesamiento central, generalmente llamadas unidades centrales de procesamiento (UCP) o procesadores, las cuales pueden ejecutar instrucciones en lenguaje de máquina; la memoria direccionable del procesador (llamada memoria principal); y los dispositivos periféricos (llamados comúnmente periféricos o dispositivos de

entrada/salida), las cuales proporcionan las facilidades de entrada y salida del sistema y espacio para almacenamiento secundario de información.

El procesador consiste de varios registros de longitud fija, para almacenamiento temporal de datos e instrucciones; una unidad de aritmética y de lógica, que ejecuta las instrucciones, una unidad de control y la memoria interna o principal.

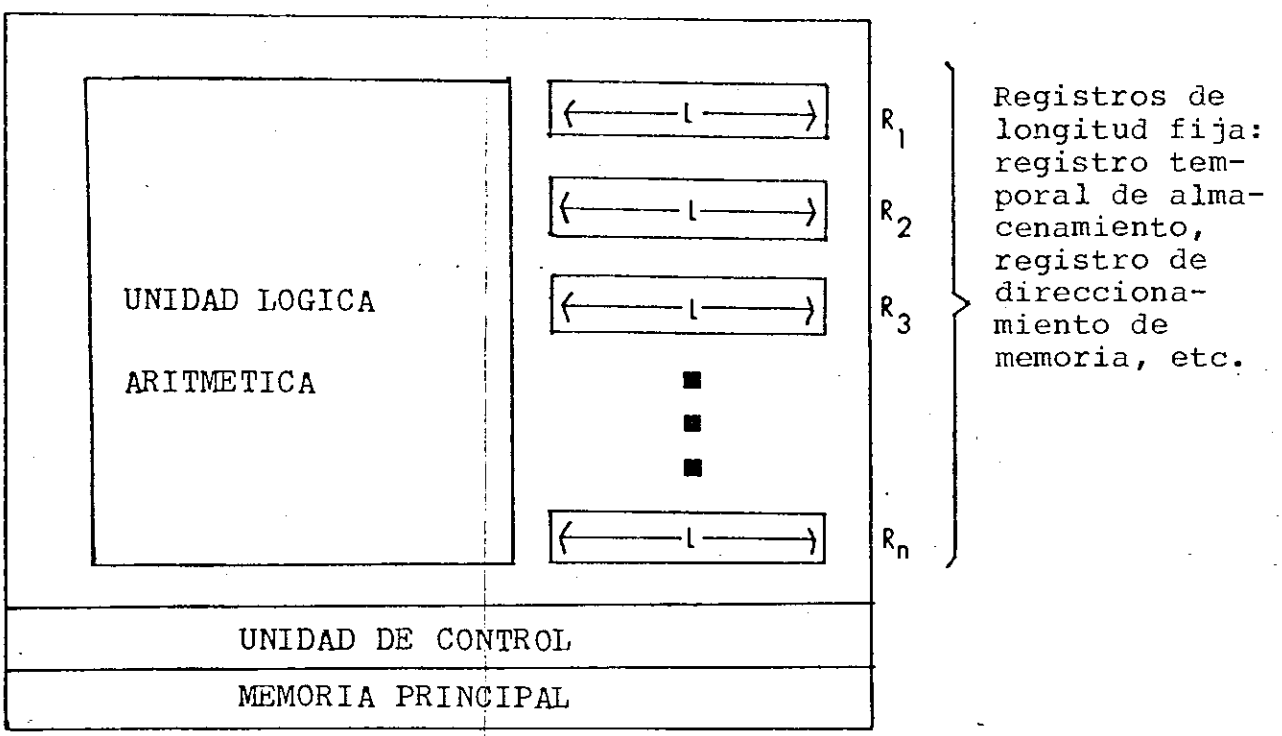


Figura 1.1 Procesador central

La memoria principal está dividida en bloques de bits

de longitud fija, llamados palabras, generalmente de la misma longitud de los registros. Para cada palabra en memoria, se asigna un nombre, que se conoce como su dirección o localización. Las palabras pueden contener tanto instrucciones como datos.

Uno de los registros del procesador, el registro de la dirección de instrucciones, contiene la dirección de la próxima instrucción que la unidad de control debe interpretar. Las instrucciones son ejecutadas generalmente en forma secuencial, atendiendo al incremento de las direcciones hasta que una instrucción provoca algún salto a una nueva localización, por medio de la alteración del contenido del registro de la próxima dirección.

Los dispositivos periféricos, mueven información entre la memoria principal y algún tipo de almacenamiento o medio de salida, tal como la lectora de tarjetas, discos magnéticos cintas, o impresoras. Los dispositivos difieren según la forma de almacenar información y el método y velocidad con que los datos van a ser utilizados. Algunos dispositivos, como unidades de cinta magnética y lectoras de tarjetas, son estrictamente secuenciales. Los datos pueden ser procesados solamente como un simple vector, en orden del 'primero que entra, es el primero que sale' (PEPS); así, si por ejemplo queremos usar el décimo 'ítem' de una lista de veinte,

deberemos pasar antes por los nueve primeros.

Otros dispositivos de almacenamiento proveen un uso más directo de datos, conocidos también como dispositivos de localización directa. Los tambores y discos magnéticos están organizados como un conjunto de pistas; de tal forma que, si se quisiera accesar una parte de los datos, como el n-ésimo elemento de la m-ésima pista, debido a que las cabezas de los discos pueden cambiarse relativamente rápido de una pista a otra, el promedio de tiempo requerido para localizar un elemento o ítem de datos es considerablemente menor que en el caso secuencial. En todos los dispositivos magnéticos, el promedio de tiempo en que una parte de los datos puede ser movido, una vez estos son localizados (llamado tiempo promedio de transferencia), es substancialmente más rápido que el tiempo de localización.

Esta propiedad de los dispositivos, los hace ventajosos para mover información en grandes bloques. Así, si almacenamos diez palabras en un disco, es mucho más rápido hacerlo en un solo bloque en un solo paso, que el hacerlo a la vez. Para bloques pequeños de datos, el tiempo de transferencia es comparable con el tiempo de localización; el mover las palabras una a la vez es aproximadamente cien veces más lento que moverlas todas a la vez.

El rango de velocidad entre los diferentes tipos de dispositivos y su conexión al sistema de cómputo es relativamente uniforme. Las unidades periféricas se conectan a la unidad central de procesamiento, mediante unos dispositivos especiales llamados canales o procesadores de entrada/salida que tienen su propio lenguaje.

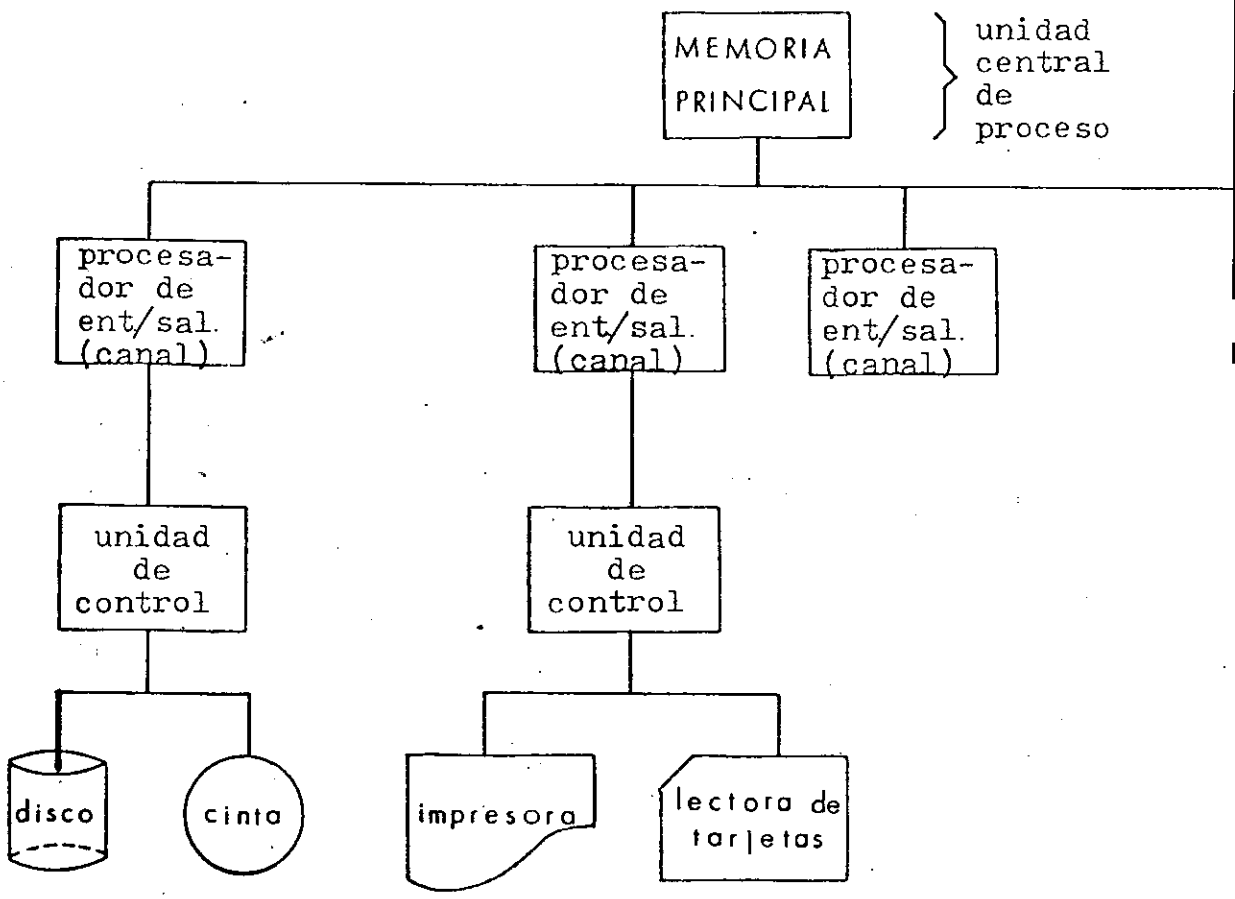


Figura 1.2 'Hardware' básico

La conexión entre el periférico y el canal, se

efectúa mediante una unidad de control de dispositivos. Esta unidad es un dispositivo electrónico que acepta órdenes de un canal y provoca que el dispositivo opere en alguna forma, recibiendo de o emitiendo datos al canal; esto es leer o escribir. Cuando un programa inicia una operación de entrada/salida, el sistema recibe el requerimiento, genera el programa de canal necesario y lo envía. Usando la información contenida en el programa de entrada/salida, el canal selecciona el dispositivo y provoca la acción adecuada; por ejemplo, leer de o escribir en un medio de almacenamiento. Después de que la operación de entrada/salida se completa, el canal debe notificar al sistema sobre la finalización del trabajo. Esta notificación se realiza a través de una interrupción de entrada/salida del 'hardware'.

Una interrupción, es una transferencia forzada del control para atender alguna condición externa al programa que se está ejecutando. La posibilidad de interpretar interrupciones es proporcionada por bits de interrupción especiales localizados en el procesador.

Cuando se acepta una interrupción, el procesador central interrumpe lo que está haciendo y salta a la localización de memoria principal que atiende la condición que originó la interrupción. Esta rutina examina, los registros de interrupción para averiguar la razón de la misma.

Por ejemplo, una interrupción puede provocarse porque un dispositivo ha terminado su ejecución o porque trató de ejecutar una instrucción ilegal. Entonces se desarrolla la acción adecuada, tal como enviar otra orden al canal o detener el programa. Los canales y el procesador central pueden ejecutar procesos o instrucciones, concurrentemente. En la mayoría de las máquinas, la unidad central de procesamiento es la encargada de decir al canal lo que tiene que hacer. Y las interrupciones son usadas para sincronizar la operación de estos dispositivos independientes.

2. 'Software'

El 'hardware' no provee exactamente todas aquellos servicios que son necesarios para los usuarios de sistemas de cómputo.

Así, es prácticamente imposible para los usuarios, utilizar eficientemente el sistema sin la ayuda del 'software'.

Se puede definir como 'software' el conjunto de programas y datos que almacena un computador y que ponen en uso los servicios del 'hardware'. En él se incluyen programas que no están diseñados para resolver problemas específicos de procesamiento de datos. Son programas que son utilizados para

incrementar la producción y mantenimiento de otros programas.

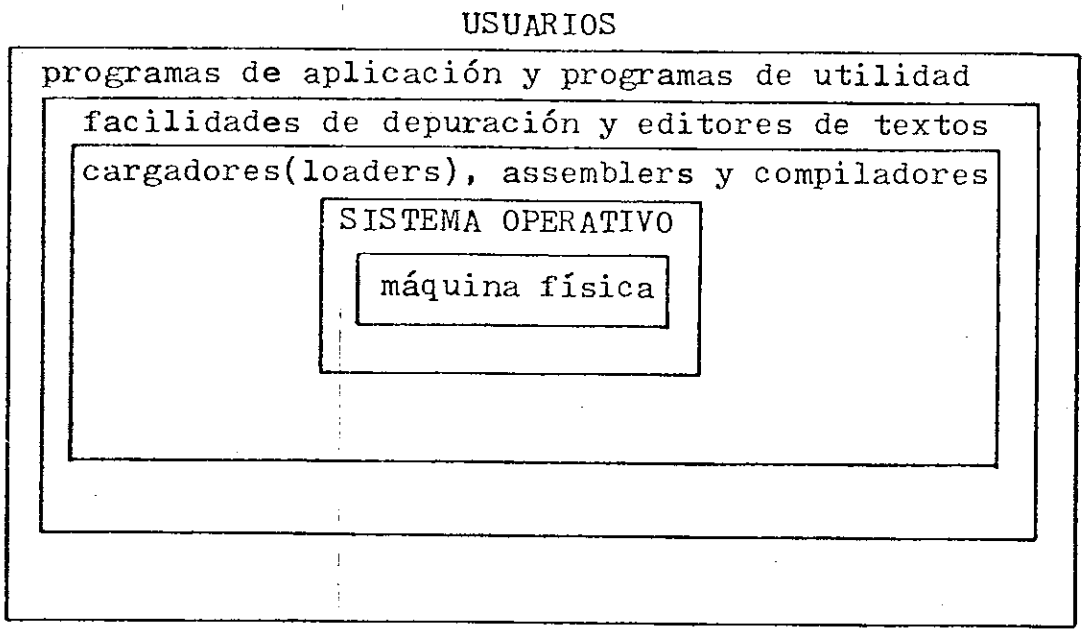


Figura 1.3 Niveles del 'software'

2.1 Compartimiento de recursos

Un recurso es cualquier objeto que pueda ser localizado dentro del sistema. Los recursos de 'hardware' son raramente compartidos, en el sentido de que los programas puedan utilizar el mismo recurso simultáneamente.

Por el contrario, se dice que está compartido, si varios programas están ejerciendo control sobre el recurso, en un subintervalo de un período de tiempo para cada programa.

Por ejemplo, cinco programas pueden compartir igualmente un procesador si a cada programa se le permite ejecutar 1 de cada 5 segundos. Todos los dispositivos de 'hardware' son compartidos en esta forma; sin embargo, la escala de tiempos bajo la cual los programas pueden ejercer control sobre un dispositivo determinado, varía considerablemente. Por ejemplo, un programa puede utilizar parte de la memoria principal por varios minutos, pero nunca puede ejercer control sobre el procesador por más de un segundo a la vez.

Básicamente, existen dos formas de implementar los sistemas compartidos. El primero, es el método descentralizado, en el cual si varios programas quieren compartir un recurso, y si saben de la existencia de cada uno de los demás, entonces pueden cooperar pasándose el recurso entre ellos. En la mayoría de los sistemas, los usuarios no se preocupan de comunicarse con los demás. El segundo, es el método centralizado en el que se carga un programa separado para controlar la utilización de los recursos equitativa y eficientemente; así, esos programas no necesitarán cooperar más entre sí, para obtener el recurso que

demandan. Cuando esto sucede, el programa requiere únicamente el servicio del asignador de recursos.

Las nociones de la asignación de recursos y de los recursos compartidos, son el concepto central de un sistema operativo.

2.2 Otras facilidades de 'software'

Los dispositivos periféricos mueven datos a velocidades que son substancialmente más lentas que la velocidad a la que el UCP puede procesarlos. Para evitar que la unidad central tenga que esperar, el dispositivo podría mover los datos a la memoria principal antes que el UCP los necesite explícitamente. Cuando la UCP inicia el procesamiento de datos de la memoria principal, el dispositivo va a estar listo para empezar y mantener al procesador ocupado hasta que tenga que parar o esperar. En inglés, esta actividad es llamada 'buffering', y el área donde el dispositivo almacena la información avanzada, es llamado 'buffer' o memoria compensadora.

Una técnica muy común de 'buffering' en sistemas de cómputo, es llamada 'spooling'. Generalmente, el procesador obtiene bastante información de la entrada de datos desde una lectora de tarjetas, pero la lectora es muchísimo más

lenta que el procesador. Para compensar esto, las tarjetas son movidas a un pequeño buffer de memoria en disco. De tal forma, el disco actúa como una memoria intermedia o temporal entre el UCP y la lectora de tarjetas, haciendo que el UCP lea las tarjetas del disco. Una técnica similar, puede ser usada para la impresora, utilizando un buffer de salida. El programa que implementa el spooling de datos es una de las partes más complejas que integran un sistema de cómputo.

Una de las facilidades más importantes, ofrecidas por un sistema de cómputo, es la traducción de lenguajes por medio de compiladores o traductores, ensambladores e intérpretes. Como otras facilidades de 'software', los traductores de lenguajes pueden interactuar con el sistema cuando están ejecutando. El medio por el cual interactúan el sistema y el traductor, se llama 'interfaz'. Este requiere un diseño muy cuidadoso.

Por ejemplo, el traductor necesita obtener determinados recursos del sistema, tal como un archivo de almacenamiento o una terminal de pantalla. Esto implica que el traductor debe comunicarse con la parte del sistema que gobierna la asignación de recursos.

Similarmente, si el traductor genera el código de máquina, ese código podría requerir las convenciones del

sistema, tal como las reglas para encadenar programas.

En conclusión, el interfaz puede ser tan simple y flexible que permita la conexión de una gran variedad de traductores.

El área de traducción ha recibido una atención especial de tal forma que este campo ha progresado al punto en que se puede comprar un traductor en la misma forma que los dispositivos de 'hardware'.

Una segunda facilidad muy importante provista por los sistemas de cómputo, es la habilidad para manejar una gran cantidad de datos. Los datos son organizados en unidades lógicas llamadas archivos y después almacenados en una forma que esté relacionada con su uso. Los elementos que se encuentran dentro de un archivo son usualmente llamados registros, los que pueden ser localizados con distintas técnicas dependiendo de cómo están estructurados los archivos. Por ejemplo, en un archivo estructurado secuencialmente, los registros son localizados en el mismo orden en que fueron almacenados inicialmente. Un ejemplo de esto podría ser un archivo en cinta magnética. La capacidad de almacenar grandes bloques de datos estructurados, dentro del sistema y por largos períodos de tiempo, es de gran utilidad. Los archivos son colocados en almacenamiento secundario y no en memoria principal.

Los sistemas de archivos hacen más fácil el compartir los datos y, si ellos están bien estructurados, pueden brindar una gran ventaja en la rapidez del tiempo de localización de un elemento arbitrario.

El sistema de archivos requiere del uso de los recursos del sistema de cómputo, por lo que éste debe interactuar con el sistema a fin de obtener dispositivos periféricos, almacenamiento en memoria principal y tiempo de procesamiento. Así, el interfaz entre el sistema de archivos y el sistema de cómputo requiere el mismo cuidado en el diseño, como el discutido en los traductores.

Hay una gran cantidad de facilidades de 'software' que son ofrecidas en los sistemas de cómputo. Los paquetes de subrutinas matemáticas, las librerías de subrutinas estadísticas y librerías de programas utilitarios como los clasificadores, son agregados para facilitar la conveniente explotación del sistema por muchos usuarios.

C. Administrador de Recursos

1.- Funciones

Tal como ya se ha dicho en la introducción, un administrador de recursos,

a) ha de saber en todo momento si dispone del recurso y en

qué cantidad.

b) ha de decidir quién toma un recurso, cuándo conviene que lo haga y qué cantidad toma de él.

c) ha de llevar a cabo la asignación del recurso a un proceso que lo pida.

d) ha de recuperar el recurso y el proceso lo deja libre.

El punto a) lleva a hablar de las estructuras de datos del sistema; tablas, vectores, listas, punteros, colas, etc. El punto b) diseña la estrategia de asignación y utilización del recurso. Los puntos c) y d) llevan a cabo las asignaciones decididas en b).

Los elementos de b), c) y d) son pues, módulos del sistema, implementados con algoritmos que trabajan sobre los datos de las estructuras de a), creándolas, borrándolas o destruyéndolas, actualizándolas, enlazándolas, y recuperándolas. Estos procesos se activan en diferentes momentos como respuesta a las demandas de servicio de otros procesos del sistema o como respuesta a señales externas (interrupciones). Así los procesos de un administrador pueden trabajar concurrentemente con cualquier otro proceso del sistema o del usuario.

En lo sucesivo, se tratarán las partes o módulos fundamentales de un sistema operativo, en base a esta descomposición. Se hablará pues de los administradores de procesos, de memoria y de entrada/salida. Es necesario hacer notar que el administrador de procesos tratará de los problemas de los procesos que se asignan a la UCP (procesos de cálculo) dejando que el administrador de entrada/salida maneje los problemas de los procesos especificados. Para una mejor comprensión del trabajo de un administrador de recursos, refiérase al apéndice B.

2. Asignación y recuperación de recursos

Mientras se está ejecutando un proceso, hace uso de los recursos del sistema. Conviene identificar tres momentos o situaciones diferentes en los que puede encontrarse un proceso durante su ejecución:

- la asignación inicial de un recurso por parte del administrador.

- la recuperción final del recurso al acabar el proceso o programa.

- la demanda y asignación dinámica de recursos que piden los

procesos mientras ejecutan.

Los recursos, por razón de su naturaleza, se dividen en dos grupos:

1. aquellos, de uso concurrente que una vez asignados, pueden ser recuperados en cualquier momento por el administrador, que los puede asignar a otro proceso que los haya pedido, recuperando después la asignación inicial. Por ejemplo, un disco, una memoria o una unidad central de proceso.

2. aquellos de uso secuencial que una vez asignados a un proceso, el administrador no puede recuperarlos sino hasta que el proceso que los tiene asignados los libera. Por ejemplo una cinta magnética o una impresora. En este último caso, la asignación inicial permanece hasta que se hace la recuperación final.

La demanda y la asignación dinámica, permiten una mejor utilización del recurso, pero introducen un grado más de complejidad en el diseño de los algoritmos del administrador, por razón de la ya mencionada competencia entre procesos.

II. TECNICAS DE ADMINISTRACION DE PERIFERICOS

A. Monitor o Supervisor

Un sistema operativo típico, tiene un conjunto de programas colectivamente conocidos como 'Supervisor ', cuya función es la de proveer los servicios necesarios para supervisar la ejecución de un número de programas del usuario. El control se transfiere al supervisor cada vez que el flujo normal de procesamiento es interrumpido por cualquier cambio.

El propósito de una llamada al supervisor es el de proveer un mecanismo que permita al programa, interrumpir el flujo normal de procesamiento y requerir al supervisor el adoptar las medidas necesarias para tratar la condición que dió origen a la interrupción.

Las interrupciones al supersivor más comunes, son las de entrada y salida. En un sistema de multiprogramación es esencial tener un sistema de control de dispositivos de entrada/salida, especialmente de aquellos dispositivos compartidos por un determinado número de

programas.

B. Programación de entrada/salida y Administración de Periféricos

Las funciones básicas del administrador de dispositivos y de entrada/salida son:

a.- Conocer el estado de todos los dispositivos del subsistema de entrada/salida; conocer cómo están relacionados y si hay uno o más caminos libres que unan a los elementos entre sí, función realizada por el controlador del tráfico de entrada/salida.

b.- Decidir qué proceso puede utilizar al dispositivo en función de su clase: dedicado, compartible o virtual; hacer el tratamiento de un tipo concreto de dispositivos, función que realiza el manejador del dispositivo. Y de las demandas de operación de los programas, las que controla el planificador de la entrada/salida.

c.- Hacer la asignación de un dispositivo a un proceso o trabajo.

d.- Hacer la recuperación del dispositivo.

Es preciso explicar qué se entiende por clase de dispositivo:

a.- Un dispositivo dedicado es un dispositivo que se asigna a un trabajo mientras dura; por ejemplo una cinta, una impresora, etc.

b.- Un dispositivo compartido es un dispositivo que puede ser utilizado concurrentemente por más de un proceso; por ejemplo un disco.

c.- Un dispositivo virtual es un dispositivo que se implementa sobre otro a nivel de software. Por ejemplo un dispositivo determinado puede convertirse en virtual, implementándolo sobre disco, como sucedería con impresoras y lectoras de tarjetas virtuales.

1. Controlador del tráfico de entrada/salida

Si desde un proceso se pide realizar una operación de entrada/salida sobre un dispositivo, tiene que existir necesariamente un camino a

través del subsistema de entrada/salida para llegar a él, en el sentido de que el dispositivo tiene que estar conectado a una unidad de control. Esta a su vez, deberá estar conectada directamente a memoria a través de una línea para transmitir datos o bus de datos; o bien a través de un procesador especializado de entrada/salida, canal. El controlador de tráfico por tanto tiene que saber:

- Si hay un camino libre para servir al requerimiento de entrada/salida.

- Si no existe el camino, saber si hay alguno alternativo.

El controlador del tráfico mantiene actualizadas las tablas, los punteros y los bloques de control necesarios de la unidades que gobierna. En cualquier momento los procesos del planificador de entrada/salida y los manejadores de dispositivos, tienen que poder consultar a estas tablas.

En la Figura 2.1, se indica cómo están formados los bloques de control por punteros a otros bloques y por punteros a colas de procesos de en-

trada/salida que están esperando la terminación de la operación que tiene ocupado a algún elemento del subsistema de entrada/salida. El planificador de entrada/salida trabaja sobre estas colas.

"NOMBRE" Tabla de nombres simbólicos	Identificador del dispositivo Estado del dispositivo Lista de unidades de control conectadas Lista de procesos de E/S esperando el dispositivo Tabla de bloques de control	Identificador de la unidad de control Estado de la unidad de control Lista de dispositivos conectados Lista de procesadores conectados Lista de procesos esperando la unidad de control Conjunto de bloques de control de las unidades de control	Identificador del procesador Estado del procesador Lista de las unidades de control conectadas Lista de procesos de E/S esperando a que el procesador esté libre Conjunto de bloques de control de los procesadores especializados de E/S

Figura 2.1 Tabla de nombres simbólicos

2. Asignación de dispositivos

El hecho de asignar una unidad lógica o simbólica definida por el programador, a una

unidad física, no es nada más que hacer corresponder el nombre simbólico con el bloque de control del dispositivo concreto, tal como se indica en la figura anterior.

La asignación de unidades simbólicas a unidades físicas se hace normalmente al empezar un trabajo, y es cuando las rutinas de asignación:

- Comprueban si existe un camino hasta el dispositivo.

- Intentan hacerlo con un dispositivo del mismo tipo que esté libre, en caso se quisiera hacer la asignación de un dispositivo dedicado, que ya está ocupado, por ejemplo, otra cinta, o bien, si no se consigue, abortan o suspenden la ejecución del programa.

Una vez hecha la asignación, una orden de entrada/salida dirigida a un dispositivo concreto sigue el camino definido por los punteros de los bloques de control.

3. Planificador de entrada / salida

Si imaginamos que todos los requerimientos de entrada/salida se sitúan en una cola, el primer trabajo del planificador es ordenarla y hacer la gestión de las nuevas órdenes que llegan y de las que se han de borrar una vez atendidas.

En los sistemas con múltiples caminos, el planificador trabaja con la información del controlador de tráfico para decidir el camino del requerimiento de entrada/salida. Una vez decidido, el planificador puede pasar el requerimiento a las colas de los elementos que lo forman. Desde un punto de vista conceptual, el planificador ha creado el entorno de datos del proceso de entrada/salida.

4. Activación contínua de los procesos de entrada/salida

Ya se ha mencionado el hecho de que los procesos de entrada/salida son concurrentes con los de cálculo. En definitiva podemos tener un proceso de cálculo asignado a la UCP , y diferentes procesos de entrada/salida trabajando para diferentes procesos de cálculo que están es-

perando ser ejecutados. Para mantener el máximo grado de paralelismo, se han de ejecutar tantos procesos de entrada/salida como sea posible.

Desde un punto de vista global, se tiene que intentar que el sistema operativo mantenga la máxima actividad de los procesos de entrada/salida, mientras haya requerimientos en la cola. El administrador de entrada/salida y de dispositivos ha de intentar servir los requerimientos, despachando los procesos de entrada/salida correspondientes, tan pronto y tan frecuentemente como les sea posible.

El funcionamiento es el siguiente: después de situar el requerimiento en la cola, se intenta despachar el proceso de entrada/salida asociado. Tanto si se puede hacer como si no, porque el camino está ocupado, se devuelve el control.

Esta estrategia justifica el ejemplo en el que una vez solicitada la entrada/salida el proceso de cálculo que la ha pedido, puede continuar trabajando:

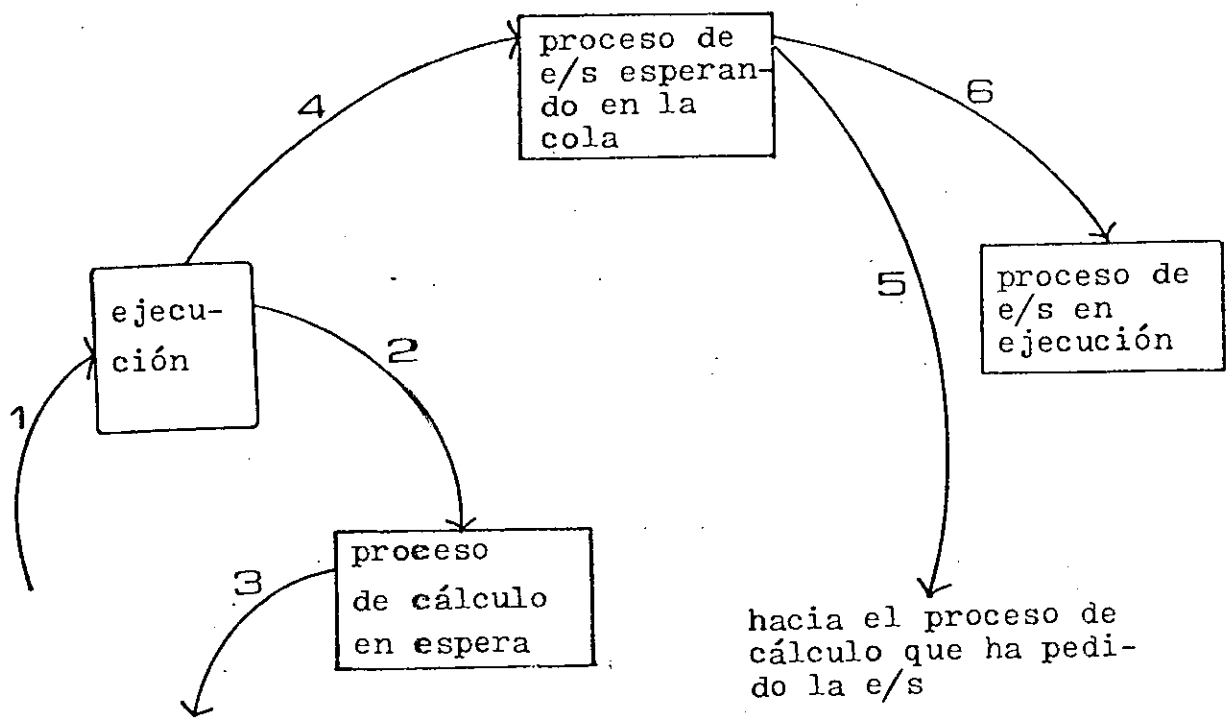


Figura 2.2 Despachador de procesos

La bifurcación de la transición 6 indica el hecho de que se intenta despachar el proceso de entrada / salida y, tanto si se puede como si no, se retorna el control directamente al proceso de cálculo del usuario. La terminación de la operación provoca una señal que hace que se pase el control al administrador de entrada y salida. Este borra el requerimiento ya

satisfecho de la cola (destruye el proceso), provoca la transición 3, da servicio al próximo requerimiento de la cola y pasa el control al planificador de procesos.

Para clarificar lo anterior podemos observar el siguiente gráfico:

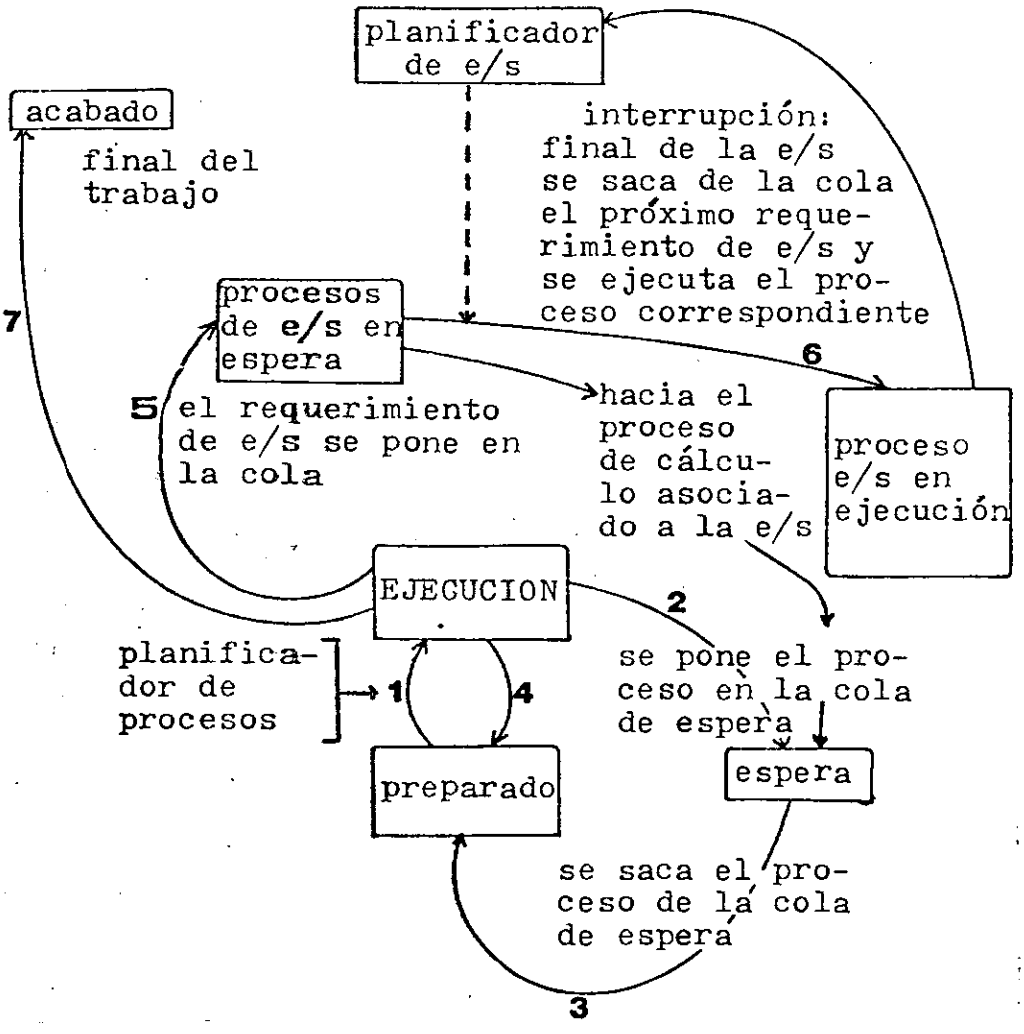


Figura 2.3 Transiciones de los procesos de entrada/salida

5. Despachador de la entrada/salida

El trabajo del despachador de entrada/salida es iniciar el subsistema de entrada/salida. Esto lo hace cargando los registros 'hardware' de los procesadores de entrada/salida con datos del proceso de entrada/salida que quiere iniciar o bien cuando se trata de un sistema que utiliza canales, cargando la dirección de inicio, al programa de canal como se verá más adelante, al tratar sobre manejadores y programas de canal.

En cualquier caso se pasa la información siguiente:

- Dirección del área de entrada/salida
- Número de bytes a leer o escribir
- Tipo de operación (lectura, escritura o control).
- Información de control.

Si la operación se inicia correctamente de-

vuelve el control de entrada/salida o de lo contrario realiza el análisis y el tratamiento de error.

C. Estructura del Ensamblador e instrucciones privilegiadas

1. Papel de los manejadores y programas de canal

En secciones anteriores se ha hablado de órdenes de entrada/salida que llegan al sistema operativo desde procesos de cálculo, diciendo sencillamente que se pedía servicio al sistema operativo para ejecutarlas. Desde el punto de vista del programador que está programando entrada/salida en un lenguaje de bajo nivel, ésto se traduce en reservar espacio para bloques de control, de preparar secuencias de órdenes de operación, para después pedir servicio al sistema operativo con el objeto de realizar las secuencias de operaciones deseadas.

Las secuencias de operaciones descritas anteriormente, constituyen un 'programa' en el sentido de que los procesadores especializados de entrada/salida aceptan e interpretan adecuadamen-

te esta secuencia de órdenes. En el caso de sistemas con canales, la secuencia de órdenes se conoce con el nombre de 'Programa de Canal'.

Cualquier orden de operación lleva asociada información de:

- El tipo de operación a realizar (leer, escribir, de control).

- La unidad lógica en la que se realizará la operación.

- La variable de control asociada a la terminacion de la operación.

- La dirección del área de datos.

- La longitud de bytes a transferir.

- La dirección de un bloque de control donde se recoge la información obtenida por el proceso de entrada/salida.

Una vez terminada la operación, el programador tiene que preparar rutinas para

hacer el análisis y el tratamiento de las condiciones producidas.

En resumen, la obtención de un registro de datos se tiene que programar con una secuencia de órdenes de entrada/salida y de instrucciones que constituyan el método de acceso correspondiente a la estructura de datos del archivo. Para una misma aplicación, donde los parámetros de los datos (longitud, formato, etc.) son idénticos,éstas secuencias se pueden agrupar, lo cual constituye los fundamentos de las rutinas de servicio de entrada /salida.

2. Rutinas de servicio en la entrada/salida e instrucciones privilegiadas

Si trabajan con un tipo determinado de estructuras de datos, es evidente que los algoritmos de ingreso al archivo que los contiene serán siempre los mismos; es decir, dada la necesidad de leer, escribir, localizar y extender un archivo, los algoritmos de acceso correspondientes a una misma organización, serán los mismos. La única diferencia será debida a parámetros tales como la longitud del registro, el

factor de bloqueo, el formato, etc. Las rutinas de servicio a la entrada/salida forman parte del 'software' proveído por el fabricante de la máquina; en su núcleo central preparan secuencias de órdenes de entrada/salida y hacen el análisis y el tratamiento de algunas de las condiciones de terminación de una operación. Estas secuencias corresponden a algoritmos concretos de acceso a datos estructurados según una determinada organización: secuencial, al azar, por niveles deíndice, etc. Sin querer ser exhaustiva, dan servicio para:

- Agrupar y desagrupar registros en bloques.

- Facilitar el acceso a registros lógicos independientemente del tamaño del registro físico.

- Implementar los algoritmos de acceso en datos organizados dentro de índices.

- Añadir y mantener información de control necesaria para implementar diferente formato de datos(fijo, variable, etc.).

- Implementar la entrada/salida sobre áreas

alternas o doble área de entrada/salida.

- Preparar y modificar secuencias de órdenes de entrada/salida y pedir servicio al supervisor para hacerlas.

- Implementar acciones concretas en caso de errores.

- Llevar control del número de registro, leídos o grabados.

- Añadir, sacar o interpretar caracteres de control.

La información que utilizan las rutinas de servicio en la entrada/salida está relacionada con la definición de la estructura del archivo y de las características del dispositivo que los contiene. Así, las rutinas de servicio citadas trabajan con la información de un bloque de control o bloque de definición del archivo que tiene información de:

- El tipo de registro (fijo, variable, etc.).

- La longitud del registro (número de bytes).

- El factor de agrupamiento.

- El tipo de organización del archivo.

- Los parámetros y las estructuras necesarias para localizar a los registros de datos: áreas de índices, claves, etc.

- Las direcciones de rutinas de tratamiento de casos especiales.

- La unidad lógica y el nombre simbólico del archivo.

- El número de áreas de entrada/salida que se utilizan.

- El tipo de ingreso al archivo (secuencial, al azar, etc..).

- Los derechos de ingreso al archivo: claves de protección y forma de llegar a ellas.

- La posición de principio del archivo dentro del volúmen.

- La posición del bloque físico dentro del archivo.

- Los códigos especiales dependientes del dispositivo.

- Las características físicas del dispositivo.

La mayoría de estos datos se fijan al momento de programación; esto quiere decir que un programa quede preparado para trabajar con archivos con características bien definidas. En ciertos sistemas operativos, se permite que el usuario defina o modifique algunos parámetros del archivo en el momento de la asignación inicial o que pueda definir y utilizar archivos independientes del dispositivo que los contiene.

D. Interrupciones de entrada/salida

Un procesador de entrada/salida comunica a la UCP que ha terminado su trabajo, provocando una interrupción. Las diferentes unidades del subsistema de entrada/salida no terminan la operación simultáneamente; en una arquitectura con canales,

esto permite liberar el canal y/o la unidad de control mientras todavía trabaja el dispositivo de entrada/salida, permitiéndo una mejor utilización de los elementos físicos del subsistema. Los algoritmos de tratamiento de la entrada/salida son, en estos casos, muy dependientes de la arquitectura de entrada/salida.

En sistemas con canales, cualquier interrupción, retorna al administrador del canal, que es el elemento de software que se encarga de las particularidades del funcionamiento de cada tipo de canal concreto (multiplexor, selector, canales especializados para cinta, etc.), y que aplica la mejor estrategia de servicio del canal a los dispositivos que están conectados a él.

En cualquier caso, tanto en una arquitectura como en la otra, los manejadores de canal o de dispositivo se tienen que preocupar de cumplir con la premisa general que se ha dicho antes, e iniciar otra operación de entrada/salida tan pronto como haya libre un camino. Por eso, una vez atendida la interrupción, piden al planificador de entrada/salida la siguiente operación y despachan el proceso de entrada/salida correspon-

diente.

E. Manejadores y Programas de Canal

El organizador de dispositivos ('driver' o 'device handler') generalmente se refiere al módulo del sistema operativo que se encarga de controlar un dispositivo específico de entrada/salida. El administrador recibe las direcciones y los punteros correspondientes a las estructuras de datos que pertenecen al proceso de entrada/salida a ejecutar. En sistemas con canales, el manejador de dispositivos trabaja a un nivel más alto que el de canal.

En este punto, conviene definir un término que se relaciona estrechamente con los manejadores de dispositivos, los dispositivos virtuales. Este concepto se utiliza para referirse a aquellos recursos que aparentan existir pero que realmente no existen. Tal es la aplicación para los dispositivos de los que se simula su existencia.

Las funciones de un manejador de dispositivos se podrían resumir en:

- Traducción e interpretación de las órdenes y los parámetros de la entrada/salida (preparando un programa de canal en el caso de arquitecturas que los posean).

- Traducción e interpretación de los caracteres de controles enviados (espaciado, tabulación, saltos de páginas, etc.)

- Optimización del acceso al dispositivo. Por ejemplo el organizador de disco puede reordenar la cola de mandos de entrada/salida en función de diferentes estrategias: minimización de la distancia que ha de desplazarse el brazo, movimiento lineal y constante del brazo hacia adelante, minimización de rotaciones desaprovechadas con la ayuda de dispositivos físicos de identificación de la posición rotacional, duplicación de registros de datos en diferentes lugares del disco, etc.

- Encadenamiento de órdenes sucesivas para obtener, de un dispositivo, un número de bytes mayor que su unidad básica de grabación (por ejemplo sectores de 256 o 512 bytes); para leer una cierta cantidad de bytes, el sistema tendrá

que leer uno o más sectores. Este tipo de organización simplifica el diseño de las unidades de control, pero introduce una complicación en el 'software'. Otras arquitecturas permiten una lectura de todos los sectores como sea preciso sin intervención del 'software' entre cada lectura de sectores.

- La implementación de dispositivos virtuales cambiando el proceso de entrada/salida sobre el dispositivo a virtualizar por un proceso equivalente sobre un dispositivo compartible. El sistema operativo prepara el retorno de datos de entrada/salida de manera que el manejador del dispositivo compartible pueda arrancar el proceso de entrada/salida correspondiente (por ejemplo, la dirección del disco donde escribir una línea o leer una tarjeta). Este concepto se utiliza para hacer la operacíon simultánea de dispositivos donde hay cuatro procesos: uno de lectura de tarjetas y lectura y escritura en disco; uno de lectura de disco y escritura de líneas sobre impresora; uno de conversión de una demanda de lectura (sobre lectora) en una demanda de lectura sobre la correspondiente lectora virtual en disco

y uno de conversión de una demanda de escritura de una línea de impresora sobre la impresora física a una escritura sobre una impresora virtual, simulada sobre disco. Los dos primeros procesos son los encargados de preparar los datos para las lectoras virtuales y para sacar los resultados de las impresoras virtuales. La simulación es completamente inadvertida para el programa y para el programador.

- El tratamiento y la recuperación de errores del dispositivo, pasando la información correspondiente a los bloques de estado del dispositivo.

- Pasar información al usuario para indicar la terminación de la operación de entrada/salida o condiciones del dispositivo: final de tarjetas, registro no encontrado, error no recuperable, etc.

- Dar servicio en caso de fallo de potencia a fin de preservar lo más posible la integridad y coherencia de los archivos. Esto puede implicar el escribir directorios o no emprender ninguna acción, dependiendo del tipo de dispositivo.

- Tratar la condición de "operación no atendida" al cabo de un cierto período de tiempo. La acción del organizador puede ser avisar continuamente al operador del sistema o abortar el trabajo.

En el caso de arquitecturas de entrada/salida sin canales, los organizadores, ademas, hacen:

- El 'software' de multiplexor para dispositivos lentos a base de transferir cada vez un byte de datos desde los registros 'hardware' de la interfase a las areas de entrada/salida del usuario.

- El tratamiento de interrupciones una vez enmascaradas éstas, (a base de aumentar la prioridad de la UCP frente a la de interrupciones de dispositivos), y hecho el tratamiento correspondiente, el manejador baja su prioridad, se constituye en un proceso de sistema y se pone a la cola junto con otros procesos del mismo tipo. Todos ellos se ejecutarán de forma estrictamente serial, pero serán prioritarios frente a los procesos del usuario.

- Despachar la operación de entrada/salida a

base de preparar los registros correspondientes de la interfaz con el 'hardware'.

Para poder realizar todas estas funciones, los manejadores usan rutinas de servicio del ejecutivo para hacer cosas tales como:

- Comprobar qué áreas de entrada/salida pertenecen al espacio de direcciones del programa que ha pedido la entrada/salida. Hacer la traducción de direcciones virtuales a direcciones físicas,

- Hacer un primer tratamiento de una interrupción de fin de la entrada/salida,

- Asociar una interrupción a un proceso,

- Sacar una demanda de entrada/salida de la cola (pidiendo servicio al planificador de entrada/salida),

- Mover un byte desde la interfaz 'hardware' al área del usuario,

- Transferir información desde el S.O. al espacio de direcciones del programa.

III. TECNICAS DE ADMINISTRACION DE MEMORIA.

A. Asignación contigua simple

1. Conceptos:

La asignación contigua simple es una técnica usada en minicomputadores con un sistema operativo simple, donde no existe multiprogramación, y existe una correspondencia uno a uno entre un usuario, un trabajo, un paso de un trabajo y un proceso.

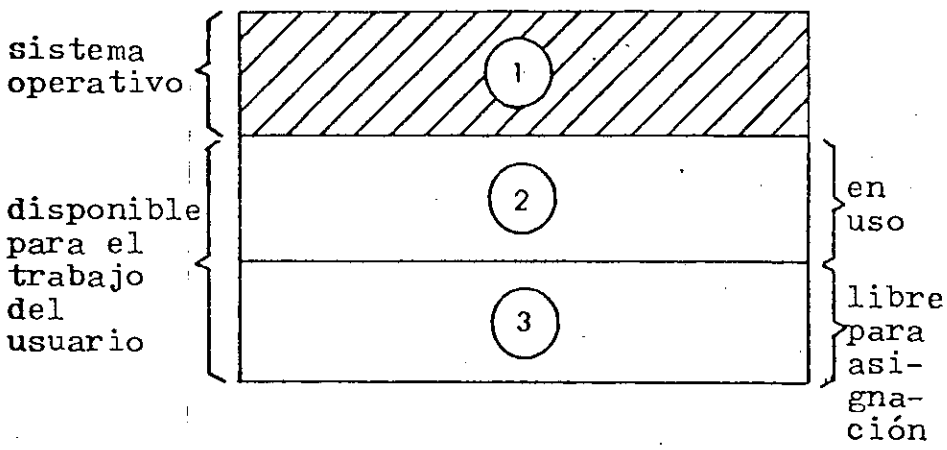


Figura 3.1. Asignación de memoria, contigua simple

2. Soporte de 'hardware'

Esta técnica de manejo de memoria no requiere un 'hardware' especial. Podría requerir únicamente un sistema de protecciones para evitar el acceso al sistema operativo.

3. Soporte de 'software'

Es necesario un 'software' muy sencillo con el objeto de asignar la memoria. Este proceso es llevado a cabo por un algoritmo llamado por el 'planificador' de trabajos del administrador de procesos, que tiene la estructura siguiente:

ALGORITMO: 1. [verificación de espacio]
es el tamaño del trabajo ±= que el espacio disponible en memoria principal ?
si ----------° 2
no ----------° 4

2. [asignación de espacio]
Asigne la memoria al trabajo .

Cargue y ejecute el trabajo.

3. [fin de proceso]

   al finalizar el trabajo desasigne la memoria y encuentre otro trabajo a correr.

   Regrese a paso 1.

4. [mensaje de error]

   imprima ' el trabajo no puede correr '

5. [fin]

   STOP

4. Ventajas

- No hay necesidad de mantener un direccionamiento de la memoria ya que toda ella es asignada a un solo trabajo.
- Simplicidad en el sistema operativo, lo que implica poco uso de la memoria principal.
- Fácil entendimiento del uso del sistema.
- No hay problema de planificación de trabajos.

5. Desventajas

- La memoria no es totalmente utilizada.
- Un gran porcentaje de tiempo del UCP espera por algún requerimiento de entrada/salida, lo que hace que se degraden, tanto el procesador, como la memoria.
- Poca flexibilidad en cuanto a la relación de los programas con el tamaño de la memoria principal disponible.

B. Introducción a la multiprogramación

El operar más de un trabajo a la vez y distribuir los recursos entre estos trabajos, es en lo que consiste la multiprogramación. Esta en sí es un técnica que permite la ejecución de dos o más procesos concurrentes.

El tiempo de espera de entrada/salida se reduce, incrementando el grado de multiprogramación.

C. Asignación particionada

1. Conceptos

En la asignación particionada la memoria principal es dividida en regiones separadas de memoria o particiones de memoria. A cada

partición se la puede asignar un trabajo diferente.

Con esta técnica se debe mantener el estado de cada partición, cuáles están libres y cuáles ocupadas, quién tiene cada partición (esto es manejado por el planificador de trabajos).

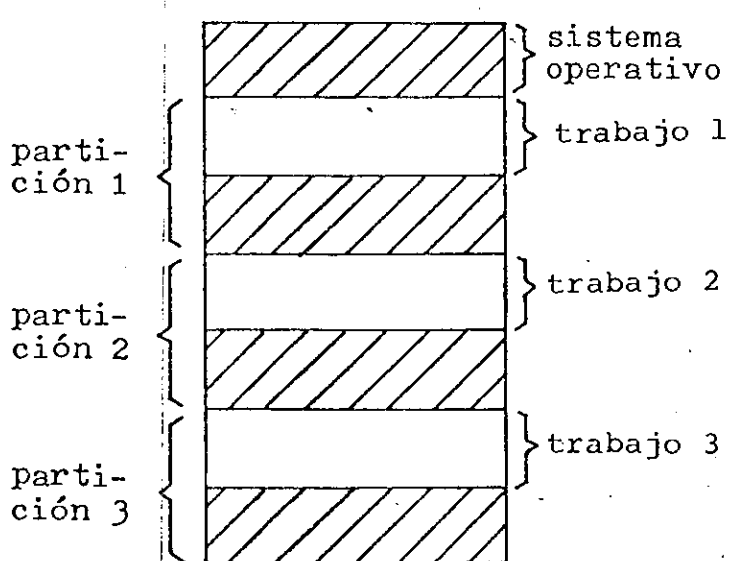


Figura 3.2 Asignación particionada de memoria

Un problema específico de la asignación particionada de memoria, es tratado en el apéndice A de este trabajo. Allí se enfoca el caso particular del tamaño óptimo de las particiones de memoria.

2. Soporte de 'hardware'

Se requiere un mecanismo de protecciones para prevenir que un trabajo caiga sobre el sistema operativo o sobre otros trabajos.

3.3.3 Soporte de 'software'

Partición estática:

En esta forma la memoria está dividida en particiones, antes de cualquier procesamiento de algún trabajo. Así, cada trabajo debe especificar la mayor cantidad de memoria que necesitará. La partición que pueda contener el trabajo, le es asignada.

Esta técnica es apropiada cuando el

tamaño y la frecuencia de los trabajos es conocida. De esta manera, el tamaño de las particiones corresponden a los tamaños más comunes de los trabajos.

Sin embargo, esta técnica no es aplicable cuando el tamaño y la frecuencia de los trabajos se desconoce o es muy diversa.

Partición dinámica:

En esta modalidad, las particiones son creadas durante el procesamiento de un trabajo, haciendo corresponder el tamaño de la partición al tamaño, del trabajo.

Se crea una tabla para mantener el estado de las particiones libres o no asignadas y otra para mantener el estado de las particiones ocupadas con sus respectivas localizaciones y longitudes.

En la siguiente gráfica podemos observar el comportamiento de la memoria trabajando con partición dinámica:

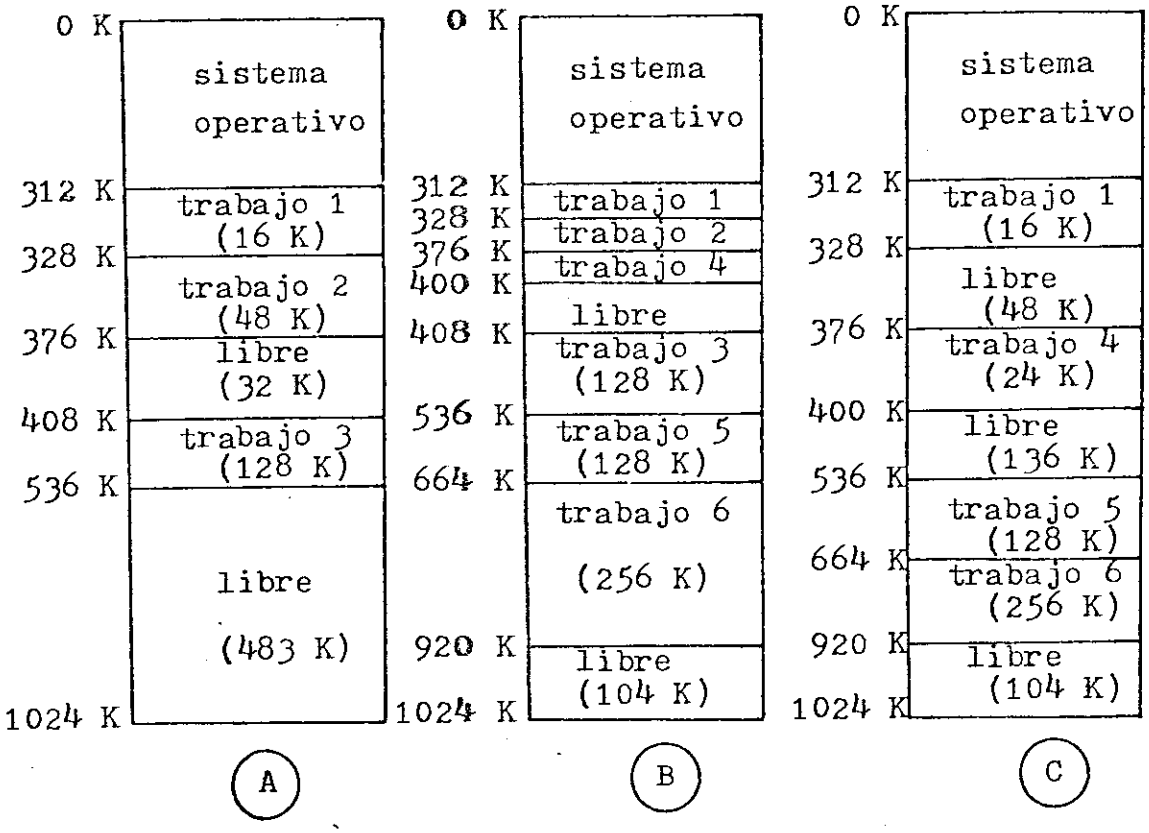


Figura 3.3. Comportamiento de las memorias con particiones dinámicas.

Se observa en a) que existen 2 espacios disponibles, los cuales son asignados en b), a un trabajo 4, sobrando 8k de memoria

disponible y a los trabajos 5 y 6, sobrando 104K de memoria disponible. Cuando porciones de memoria son desasignadas, sucede lo que se observa en c): las particiones libres contiguas se unen.

Para la asignación de particiones en la forma dinámica, existen 2 algoritmos principales:

a) La primera que encaja ('first fit')
b) La que mejor encaja ('best fit')

a) La primera que encaja ('first fit')

En este algoritmo la tabla de particiones libres es ordenada según la localización en memoria, de las particiones libres.

Así, al querer asignar una partición, se observan las áreas libres que se encuentran en las localizaciones más bajas de memoria y se va observando en las siguientes, hasta encontrar la primer área libre, lo suficientemene grande para encajar con la partición requerida.

Ventajas:

1. Generalmente es suficiente buscar en la mitad de la tabla la partición adecuada.

2. Utiliza las áreas libres localizadas en las direcciones más bajas de memoria, siempre que sea posible.

b) La mejor que encaja ('best fit')

Esta técnica ordena la tabla de áreas libres por el tamaño de las particiones. De esta forma la primer área encontrada, cuyo largo es lo suficientemente grande para la partición deseada, será la que mejor encaja.

Ventajas:

1. En promedio, el área libre que mejor encaja puede ser encontrada buscando en la mitad de la tabla solamente.

2. Si existe un área libre de exactamente el tamaño requerido, ésta será seleccionada, lo cual no sucede necesariamente en la técnica de la primera

que encaja.

3. Si no existe un área libre de exactamente el tamaño deseado, la partición es colocada en el área libre mas pequeña y no destruye algún área libre muy grande que pudiera ser requerida posteriormente.

Desventajas:

1. El mayor problema con el algoritmo de mejor encaje es que, generalmente el tamaño del área libre asignada, no es exactamente del tamaño requerido y es necesario dividirla en dos partes. Debido al criterio de mejor encaje, el área libre resultante es a menudo tan pequeña que generalmente ya no puede ser utilizada. En consecuencia, al utilizar esta política, se obtiene un gran número de áreas libres muy pequeñas en lugar de un área libre grande.

4. El problema de la fragmentación

La fragmentación es el desarrollo de un gran número de pequeñas áreas libres separadas.

Un método que podría ayudar a minimizar el problema de la fragmentación, en la asignación particionada de memoria, sería la cuidadadosa selección del algoritmo a utilizar. Para cada uno de los algoritmos discutidos (partición estática, dinámica 'first fit' y dinámica 'best fit') existen situaciones en donde su uso va a ser más apropiado y otras en donde lo será menos.

De tal manera, una situación particular será considerada como mala para un algoritmo dado, si no puede satisfacer un requerimiento inmediatamente.

Otro método que puede decrementar el problema de la fragmentación, es la asignación múltiple de particiones. En este método, un trabajo consiste de varias porciones de memoria separadas, que pueden ser subrutinas y arreglos de datos. Cada porción individual va a estar lógicamente contígua a la anterior, pero su localización física en memoria, no será necesariamente contigua.

5. Ventajas

1. Facilita la multiprogramación, pues hace más eficiente la utilización del procesador y de los dispositivos de entrada/salida.

2. No requiere un 'hardware' especial costoso.

3. Los algoritmos usados son simples y fáciles de implementar.

6. Desventajas

1. La existencia de la fragmentación.

2. Si la memoria no se fragmenta, las áreas libres podrían no ser lo suficientemente largas para una partición.

3. Requiere mucho más memoria que el sistema de asignación contigua simple.

4. La partición de un trabajo, está limitada al tamaño físico de la memoria.

D. Memoria particionada relocalizable

1. Conceptos:

Una solución obvia para el problema de la fragmentación es combinar periódicamente todas las áreas libres en áreas contíguas. Esto podría ser hecho, moviendo los contenidos de todas las particiones asignadas, para que permanezcan contiguas. Este proceso es conocido como compactación o recompactación, si en un proceso es hecho varias veces.

Es muy difícil mantener un programa corriendo correctamente después de la compactación, pues existen varios parámetros de dirección que deben ser alterados, como los registros base, las instrucciones de referencia a memoria, listas de parámetros y las estructuras de datos que utilizan apuntadores de dirección. Es por ello que se utilizan los mapas de memoria que mantienen el espacio de dirección de un trabajo, que no será el mismo de las direcciones físicas de memoria utilizadas.

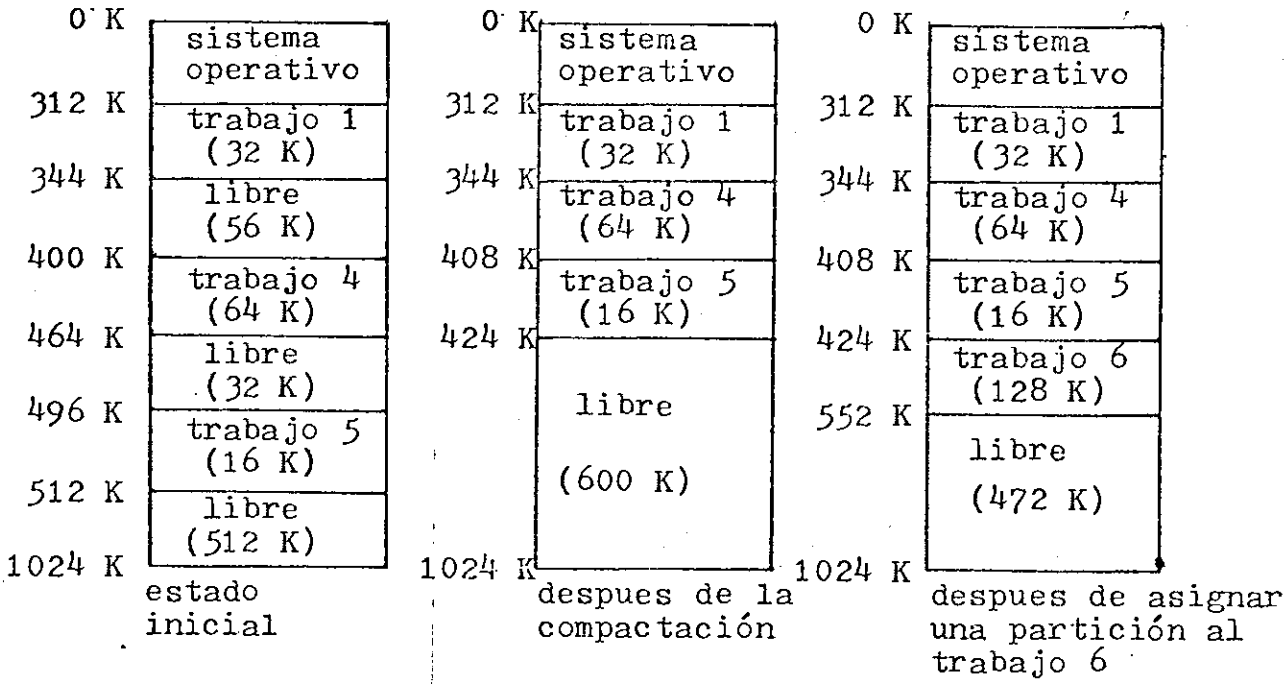


Figura 3.4. Memoria particionada relocalizable.

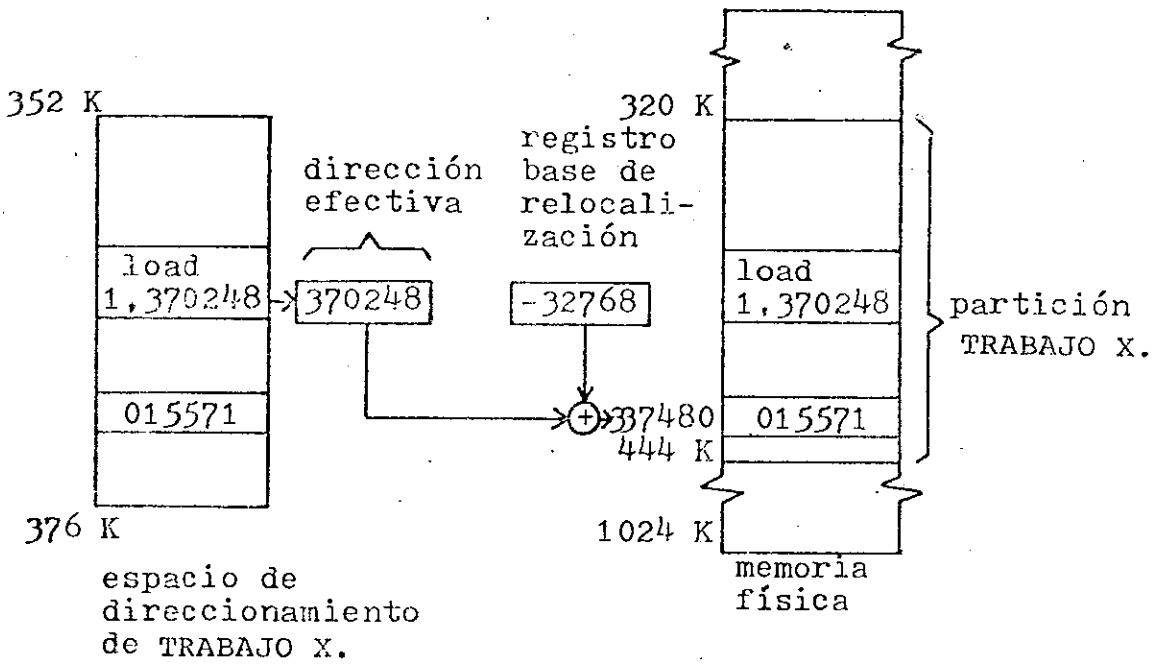
2. Soporte de 'hardware'

Se requiere un soporte especial de 'hardware'. Para efectuar una compactación adecuada, se podrián requerir 2 bits más por cada palabra de memoria, con el objeto de especificar el tipo de dato que se almacena.

Sin embargo, esto aumentaría el tiempo de compactación pues se tendrían que examinar los 2 bits de cada palabra.

Otra forma de implementación del 'hardware', podría

ser el utilizar la localización dinámica, que consiste en agregarle a la dirección relativa, el contenido de un registro base de relocalización. Este último registro contendría el desplazamiento que han sufrido los datos, después de una compactación, en relación a su posición inicial.



3. Soporte de 'software'

El soporte de 'software' radica en algoritmos tan sencillos como el que se ejemplifica a continuación.

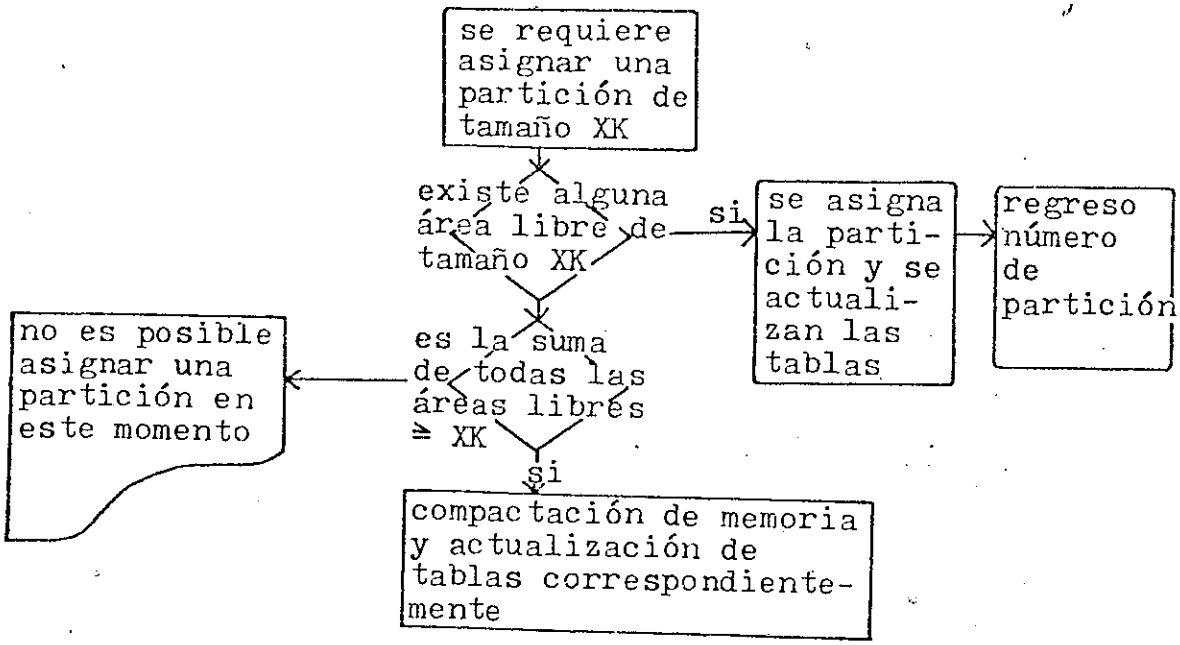


Figura 3.6. 'Software' de localización dinámica

4. Ventajas

- Elimina la fragmentación y hace posible asignar más particiones.
- Permite un alto grado de multiprogramación que provoca un incremento en la utilización del procesador y de la memoria.

5. Desventajas

- La implementación de 'hardware' para la relocalización, puede disminuir la velocidad y aumentar el costo del computador.
- El tiempo de compactación puede ser substancial.
- Parte de la memoria puede seguir desperdiciándose si la cantidad de área libre es menor que el tamaño de la partición requerida.
- La memoria puede contener información que nunca va a ser utilizada.

E. Paginación

Una idea para ayudar a eliminar la fragmentación, fue la de separar los conceptos de espacio de dirección y las localizaciones de memoria.

Si consideramos como ejemplo, una computadora con un campo de dirección de 16 bits en sus instrucciones y 4096 palabras de memoria, un programa en esta computadora podría dirigirse a 65,536 palabras de memoria; la razón es que 65,536 direcciones existen. El número de direcciones de las palabras de un computador, depende solamente del número de bits de que dispone el registro de dirección y puede no estar relacionado con el número de palabras disponibles en memoria principal.

La idea de separar el espacio de direcciones y las direcciones de la memoria propiamente dichas, se podría plantear como sigue:

En cualquier instante del tiempo, 4,096 palabras de memoria pueden ser directamente localizadas, pero no necesariamente deben responder a las direcciones 0 a 4,095. Se podría, por ejemplo decir a la computadora que a partir de la dirección 4,096 en adelante, utilizará de la dirección 0 en forma ascendente, de tal forma que cuando la dirección 8,191 fuera referida, se utilizaría la palabra de memoria 4,095; en otras palabras hemos equiparado el espacio de direcciones a las direcciones de memoria reales.

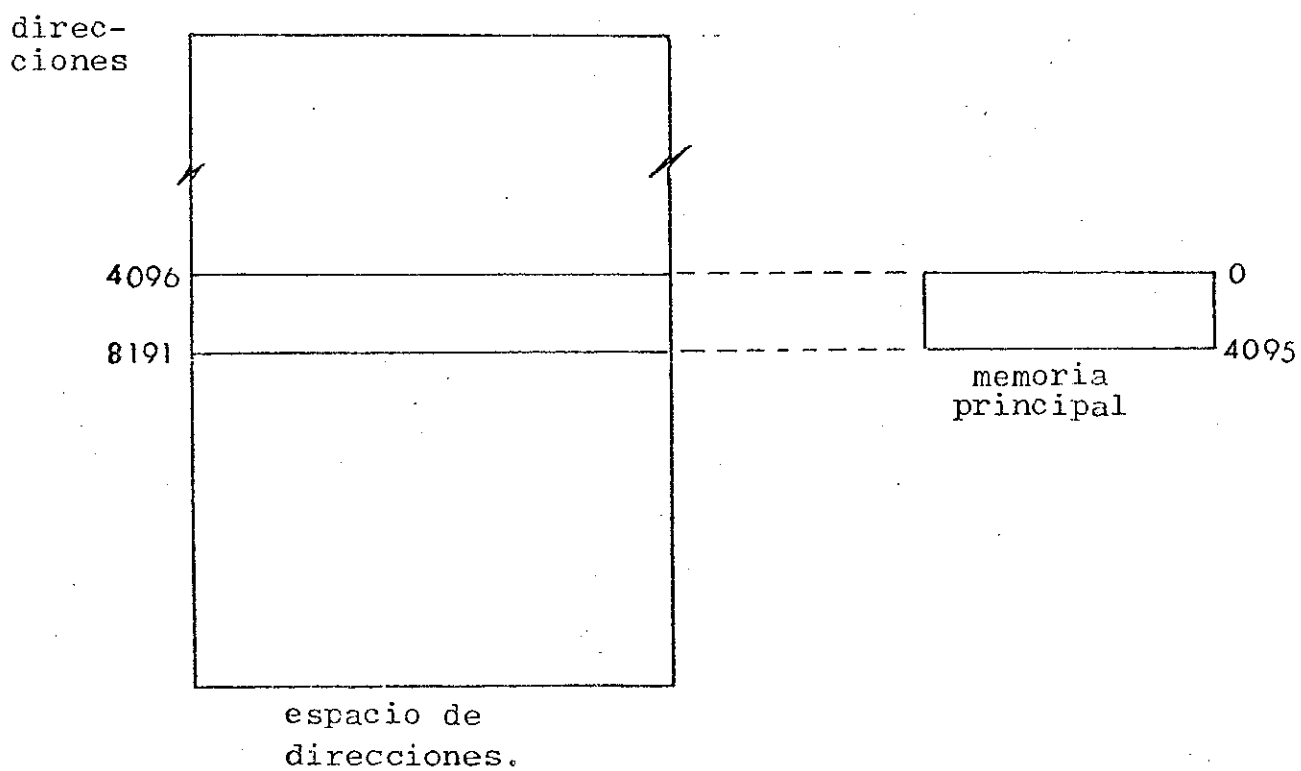


Figura 3.7. Direccionamiento de memoria

En términos de este dibujo de direcciones trazadas, del espacio de direcciones a las localizaciones de memoria reales, una máquina de 4 k sin memoria virtual, simplemente tiene un diagrama arreglado entre las direcciones 0 a 4,095 y las 4,096 palabras de memoria. Una pregunta interesante es: qué pasa si un programa salta a una dirección entre 8,192 y 12,287 ?. En una máquina sin memoria virtual, el programa provocaría un error que imprimiría un mensaje de

'MEMORIA REFERIDA INEXISTENTE'

y terminaría el programa.

En una máquina con memoria virtual, la siguiente secuencia de pasos ocurriría:

1. Los contenidos de la memoria principal serían salvados en la memoria secundaria.
2. Las palabras 8,192 a 12,287 serían localizadas en memoria secundaria.
3. Las palabras 8,192 a 12,287 serían cargadas a la memoria principal.
4. El mapa de direcciones sería cambiado a un mapa de direcciones 8,192 a 12,287 sobre las localizaciónes de memoria 0 a 4,095.
5. La ejecución continuaría como si nada raro hubiese sucedido.

Esta técnica es llamada PAGINACIÓN y los trozos de programas leídos en la memoria secundaria son

denominados páginas.

También es posible encontrar direcciones reales del espacio de direcciones, de una forma más compleja. Para enfatizar, llamaremos a las direcciones que el programa pueda referir, como el espacio virtual de direcciones, y al conjunto de direcciones reales de memoria, el espacio físico de direccionamiento.

Un mapa de memoria podría relacionar las direcciones virtuales con las direcciones físicas, asumiendo que hay suficiente espacio en la memoria secundaria (en un tambor o disco), para almacenar el programa completo y su información.

Los programas se escriben como si hubiera suficiente memoria principal, aunque no sea este el caso. Estos se pueden cargar desde o almacenar, en cualquier palabra del espacio virtual de direcciones, o saltar a cualquier instrucción localizada en el mismo, sin considerar que actualmente no hay suficiente memoria física. En realidad, el programador puede escribir sus programas sin estar consciente que la memoria virtual existe; él simplemente piensa que la máquina tiene una memoria muy grande.

Este punto es crucial y será contrastado mas tarde con segmentación, donde el programador debe estar consciente de la existencia de segmentos. La paginación le da al programador, la ilusión de una larga, continua y lineal memoria principal. El mismo tamaño del espacio de direcciones existe cuando en efecto, la memoria principal disponible puede ser más pequeña que el espacio de direcciones. La simulación de esta gran memoria principal por paginación, no puede ser detectada por el programa.

De tal forma, el programador puede trabajar como si la paginación no existiera y se dice que este fenómeno es transparente para el usuario.

1. Implementación de paginación

Un requisito esencial para una memoria virtual es una memoria secundaria en la cual se guarda el programa completo. Es conceptualmente más simple, si uno piensa en la copia del programa, en la memoria secundaria como el original y las piezas traídas dentro de la memoria principal como copias.

Naturalmente es importante guardar el original actualizado, pues cuando se han hecho cambios a la copia en la memoria principal, deberán ser reflejados en el original

también.

El espacio virtual de direccionamiento es dividido dentro de un número de páginas igualmente clasificadas, según el tamaño, sabiendo que los tamaños de las mismas son siempre un múltiplo de 2. El espacio físico de direccionamiento es dividido en pedazos en una forma similar; cada pieza tiene el mismo tamaño, así que cada una es capaz de mantener exactamente una página. Estos pedazos de memoria principal dentro de los cuales van las páginas, son llamados estructura de páginas o bloques.

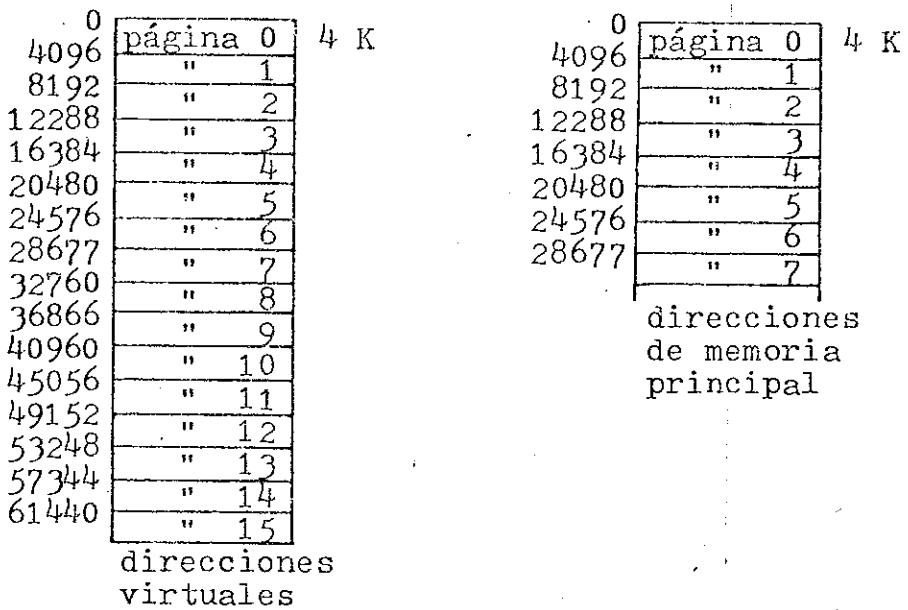


Fig. 3.8. División de un espacio de direccionamiento de 64 K

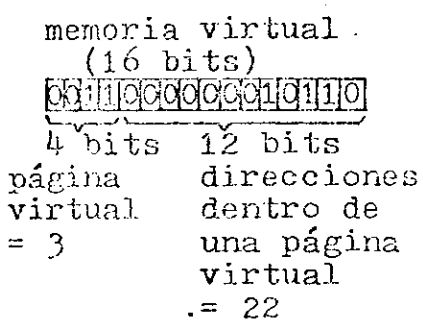
La memoria virtual de la figura, sería implementada como significado de una tabla que contiene 16 palabras. Cuando el programa trata de referenciar su memoria ya sea

para tener información almacenada, traer instrucciones o saltar, generaría primero una dirección de 16 bits correspondientes a una dirección virtual entre 0 y 65,535.

En este ejemplo, la dirección (16 bits) se toma como un número virtual de página (4 bits) y una dirección (12 bits) dentro de la página seleccionada.

2. Acceso a páginas

En la siguiente figura la dirección de 16 bits es 12,310 lo cual es considerado como dirección 22 de la página 3. La relación entre páginas y memoria virtual se muestra tabularmente.



página	dirección virtual
0	0-4095
1	4096-8191
2	8192-12287
3	12288-16383
4	16384-20479
5	.
6	.
7	.
8	.
9	.
10	.
11	.
12	.
13	.
14	.
15	61440-65535

Fig. 3.9. Memoria Virtual de 16 Bits.

Cuando se determmina que la página virtual 3 es necesaria, el sistema operativo debe encontrar dónde está localizada. Existen 9 posibilidades: 8 estructuras de página en la memoria principal o algún lugar de la memoria secundaria. Para descubrir cuál de estas posibilidades es, el sistema operativo ve en las tablas de páginas, cuál tiene una entrada para cada una de las 16 páginas virtuales.

La decisión de restringir todos los segmentos de información de tamaños uniformes, provoca un impacto significante en el manejo de la memoria.

Si una página no es accesible, el control es transferido a un procedimiento de errores el cual hace la página accesible. Este método es llamado 'demanda de paginacion', en donde debe tomarse en cuenta, el estado de la página y la selección del lugar donde se va a cargar o guardar una página. Inicialmente y después que una página ha sido borrada, su descriptor esta en estado vacío, es así como el referenciarla sería un error. Un comando de carga puede ser puesto a la cola mientras es servido. Durante este tiempo la situación en la memoria principal puede cambiar, y aparecer espacios libres, es por esto que hay que esperar a que el comando de carga sea colocado en el 'buffer'.

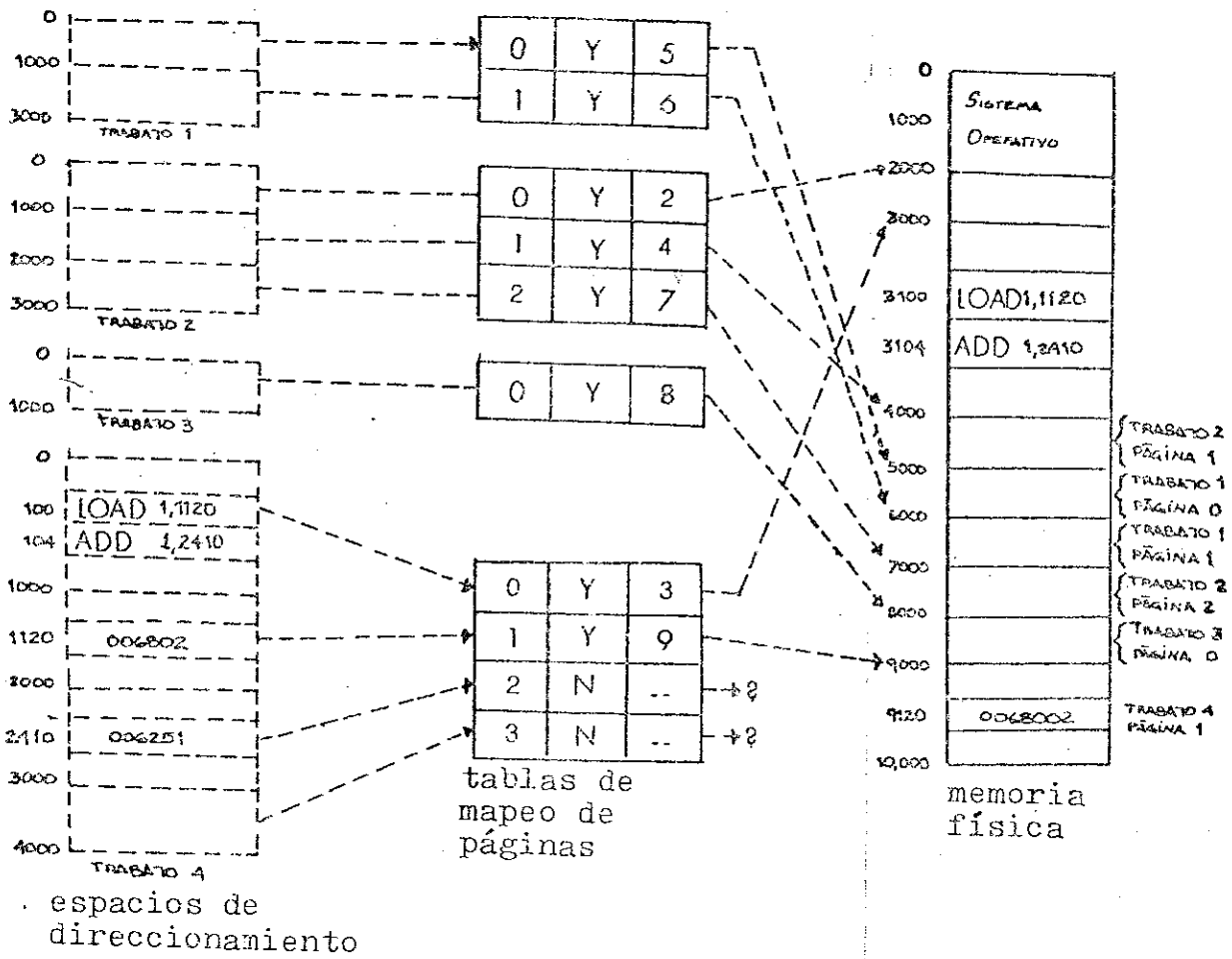


Fig. 3.10. Mapeo de la memoria para demanda de páginas.

3. Tamaño de página y fragmentación.

Si el usuario del programa y la información accidentalmente llenan un número de páginas exactamente,

no habrá espacio perdido cuando estén en la memoria. Si por el contrario, no se llena un número exacto de páginas, habrá un espacio sin uso en la última página.

Por ejemplo, si el programa y la información necesitan 26,000 palabras por página, las primeras 6 páginas estarán llenas y la última contendrá 1,424 palabras, desperdiciando 2,672. Cuando la séptima página está presente en la memoria, las palabras sobrantes se cargarán pero tendrá una función inútil. El problema de estas palabras desperdiciadas se llama fragmentación de paginación.

Si el tamaño de página es de 'n' palabras, la cantidad de espacio promedio desperdiciado en la última página de un programma será de n/2 palabras. Esta situación sugiere un tamaño pequeño de página para minimizar el desperdicio, aunque esta solución acarrearía un mayor número de páginas y una tabla grande; si la tabla está mantenida en el hardware, significa que son más registros para almacenar, hecho que aumenta el costo del computador. Además será necesario más tiempo para cargar y resguardar estos registros cuando un proceso ha iniciado o finalizado.

Si se consideran páginas pequeñas harán ineficiente el uso de la memoria secundaria. Otro método que puede

decrementar el problema de fragmentación, es la asignación múltiple de particiones. En este método un trabajo consiste de varias porciones de memoria separadas, cada porción individual va a estar lógicamente contigua al anterior, pero su localización física, no. Por una parte facilita la multiprogramación, pues hace más eficiente la utilización del procesador y de los dispositivos de entrada/salida; no requiere un hardware costoso, los algoritmos usados son simples y fáciles de implementar. Por otro lado, aunque la memoria no se fragmenta, las áreas libres podrían no ser lo suficientemente largas para una partición y ésta estaría limitada al tamaño físico de la memoria. Para una mejor comprensión del tamaño óptimo de páginas, refiérase al Apéndice A.

Existe una partición estática, en la cual se divide la memoria antes de cualquier procesamiento de algún trabajo; esta técnica es apropiada cuando el tamaño y la frecuencia de los trabajos es conocida. La partición dinámica, es aquella en que las particiones creadas durante el procesamiento de un trabajo, hacen corresponder el tamaño de la partición al tamaño del trabajo.

F. Segmentación

La idea básica de segmentación es que no es necesario

cargar un programa y sus respectivos datos en un área contigua de la memoria; se puede cargar la sección del programa y la sección de datos en dos áreas separadas, teniendo la obvia ventaja de una más flexible localización de memoria. Cuando un espacio es continuamente localizado es más difícil encontrar un área contigua para una larga pieza de información que dos pequeñas áreas individuales.

Particionando la información entre muchos segmentos de programas y muchos segmentos de datos, no solo facilita el poner la información en memoria primaria sino también perfeccionar su uso en otras formas. La segmentación hace que sea innecesario para toda la información de un proceso a ser cargado en memoria antes que el proceso pueda correr. Es suficiente cargar el segmento de programa entre cada contador de programas de un proceso. Esto permite que el proceso esté listo para correr estando parcialmente cargado; la segmentación permite que el tamaño requerido de un proceso sea mucho menor que el tamaño total de la información.

Obviamente, hay algo envuelto por encima yendo de un segmento a otro. Así, el propósito es minimizar el número de movimiento de segmentos. Este propósito puede en parte, ser conseguido si instrucciones o items de datos

con el mismo lugar de control son puestos en un segmento. El lugar natural de control para instrucciones es un bloque de programas, un procesamiento, o un subrutina; si una instrucción de alguna subrutina es ejecutada, es más probable que otra instrucción sea mejor ejecutada que una instrucción fuera de la rutina.

Observando los datos, la unidad natural es una estructura de datos como arreglos o listas. El programa y los datos de un proceso son particionados entre un conjunto de segmentos, los cuales son puestos aleatoriamente en memoria.

G. 'Swapping' y relocalización

En una máquina con multiprogramación puede ser que algunos programas estén listos para correr, mientras otros están esperando por la completación de la transmisión de datos. El tiempo de espera es generalmente largo. La transmisión de datos a través de dispositivos lentos como una lectora de tarjetas o un teletipo toma de 0.1 a 1 segundo; idealmente, la memoria principal solo contendría los procesos que estén listos para correr.

Este objetivo puede ser llevado a cabo de varias

formas. Una de ellas es el método 'swaping'. El principio de este método es que los programas y los datos de un proceso en espera son cambiados de la memoria principal y son guardados en un área de almacenamiento de copias. Cuando el período de espera finaliza y el proceso está otra vez listo para correr, es colocado en la lista de traslados. Cuando ha llegado a la cabeza de la lista de traslados, el proceso es trasladado a la memoria principal tan pronto como haya suficiente espacio.

Primero diseñamos los procedimientos del sistema para cambiar programas y datos de la memoria principal. Luego discutimos los procesos del sistema que controlan la memoria de copias y repartimos el espacio de memoria principal a los programas que pueden correr. Finalmente, veremos un caso especial de sistema de multiprogramación de tiempo compartido. Este es una extensión del sistema de tiempo compartido que no tiene multiprogramación.

Primero, un asunto de diseño de la estrategia, no se seguirá la inconveniente práctica de repartición de ventanillas, la cual es común en bancos y oficinas de correos. Alli un cliente debe decidirse y escoger una ventanilla cuando entra, y debe permanecer en esa ventanilla no importando cuánto tenga que esperar. La gente que

entra después en ocasiones es atendida mucho antes que él. Lo correcto sería asignar al cliente una ventanilla cuándo llegue su turno de recibir el servicio. Análogamente, un proceso en un sistema de multiprogramación no debe ser atado a un área particular en la memoria principal; un proceso completo en turno para correr sería cargado dentro de la primera área de memoria lo suficientemente grande disponible. Esto sin embargo implica que un proceso que fue trasladado, pueda ser cargado a la memoria principal más que un proceso que fue previamente corrido. Así las referencias a los items de datos o lugares en un programa no serían expresados en términos de localizaciones físicas de memoria principal, pero sí en términos de posiciones relativas al principio del área en la memoria principal repartida para el proceso. Tales referencias en un programa son expresadas como direcciones virtuales, el programa es llamado un programa relocalizable.

La traducción de direcciones virtuales a direcciones físicas puede ser fácilmente ejecutada. La exacta localización es derivada sumando los contenidos de un registro base o registro de relocalización a una dirección virtual. El valor de este registro es la localización inicial del área en memoria principal en la que el programa relocalizable y sus datos son cargados. Así, la regla de traducción de direcciones aplicada es:

LOCALIZACION = BASE + DIRECCION VIRTUAL

Un programa corre en un bloque contiguo de espacio de memoria principal cuyo límite inferior es determinado por el registro de relocalización. De esta forma hay muchos programas en la memoria principal al mismo tiempo, se necesita estar seguro de que un programa no accese las localidades del límite inferior del espacio repartido. El hardware de muchas máquinas es diseñado para ejecutar un simple chequeo de direcciones. Además la localización inicial, ya sea el tamaño o el máximo del área repartida en memoria principal es guardado en un registro. Cuando una dirección virtual es traducida en una localidad, el hardware chequea si el resultado está dentro del área repartida. Si el chequeo falla, las operaciones probadas resultan con un error de dirección. Este simple chequeo asegura que un programa no pueda accesar información fuera de su área, y protege un programa y sus datos contra errores o acciones maliciosas de otros programas. Por supuesto, no será posible para un programa violar la protección asignando un valor arbitrario al registro conteniendo la base, el tamaño o el máximo.

Asumiendo que las áreas en memoria principal, en las que un programa y sus datos son colocados, son todas del

mismo tamaño. Un bloque contiguo de espacio de memoria principal en uso por un procesador Pi es movido de la memoria principal y guardado en el almacenamiento de copias por un procedimiento especial llamado SWAPOUT(i). Un bloque contiguo es movido de la memoria de copias por un procedimiento especial llamado SWAPIN(i), donde i es el proceso 'indizado'.

La función SWAPOUT(i) guarda la información en dispositivos de memoria secundaria (por ejemplo, un disco). Esto es activado colocando un mensaje en el dispositivo de cola. Si la cola está vacía, el mandato es también colocado en un buffer; de otro modo simplemente se reparte la cola. El mandato consiste de la base y el máximo del área de memoria principal, la base del área en el almacenamiento de copias donde el contenido sería guardado, y una indicación de que ésta es una orden de guardar la información.

Se asume que el almacenamiento de copias es lo suficientemente grande y que un área libre puede ser encontrada siempre.

El sistema 'swaping' mantiene una cola 'swap-in' la cual contiene el proceso listo para regresar al almacenamiento de copias. Si un proceso es colocado en una cola de 'swap-in' vacía y hay lugar en la memoria principal, el proceso es

inmediatamente trasladado; de otro modo es atado a la cola de traslados..

Similar a 'swapout', la función 'swapin' prepara y envía un mensaje a la cola del dispositivo de memoria secundaria, pero envía un orden de carga en lugar de una orden de grabación.

'Swapin' y 'swapout' insertan un mensaje dentro de la cola de copias sin esperar hasta que la transferencia sea completada. Esto tiene muchas consecuencias, en primer lugar, el área en la memoria principal, ocupada por un proceso el cual es cambiado, no puede ser usada como espacio libre en 'swapout' hasta que la información sea guardada. En segundo lugar, esto es posible para un proceso que todavía no esta completamente trasladado cuando una respuesta llega y swapin decide como cambiar este proceso de regreso. Esto no causa un problema real, porque la orden de carga es colocada en la cola más que la orden de grabación. Así, por el tiempo en que la orden de carga es ejecutada, la orden de grabación ha sido procesada. Finalmente la completación de una transferencia en ambas direcciones debe ser notificada por el proceso de control del dispositivo de almacenamiento de copias.

Se llama BSC ('Back Storage Control') al proceso que

controla el almacenamiento de copias. El proceso BSC es activado cuando el dispositivo ha completado la transferencia.

El trabajo primario es borrar la orden común y enviar el siguiente mensaje al dispositivo. Si una orden de grabación ha sido completada, el estado de almacenamiento primario es actualizado, porque el área en el almacenamiento primario, ocupada por el proceso de grabación, está realmente libre. El espacio libre es dado al primer proceso en la cola de cambios, o si la cola está vacía esto es grabado como libre. El espacio libre es enteramente controlado por el proceso BSC.

1. TRASLAPES

Es una forma más refinada de 'swapping', la cual traslada únicamente porciones del espacio de dirección de trabajos y es llamada manejo de traslapes.

A continuación se ilustra la relación entre los procedimientos en un espacio de dirección de trabajos.

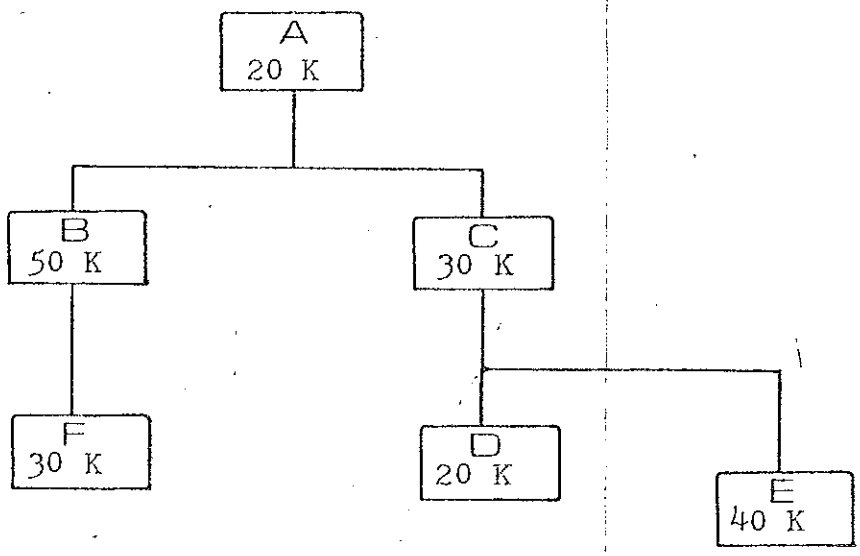


Fig. 3.11. Manejo de memoria por traslapes.

Asignando direcciones de procedimientos como se muestra en la figura siguiente, en lugar de usar 190 K bytes para mantener el espacio de direccionamiento de trabajos en memoria, un máximo de 100 k bytes son necesarios.

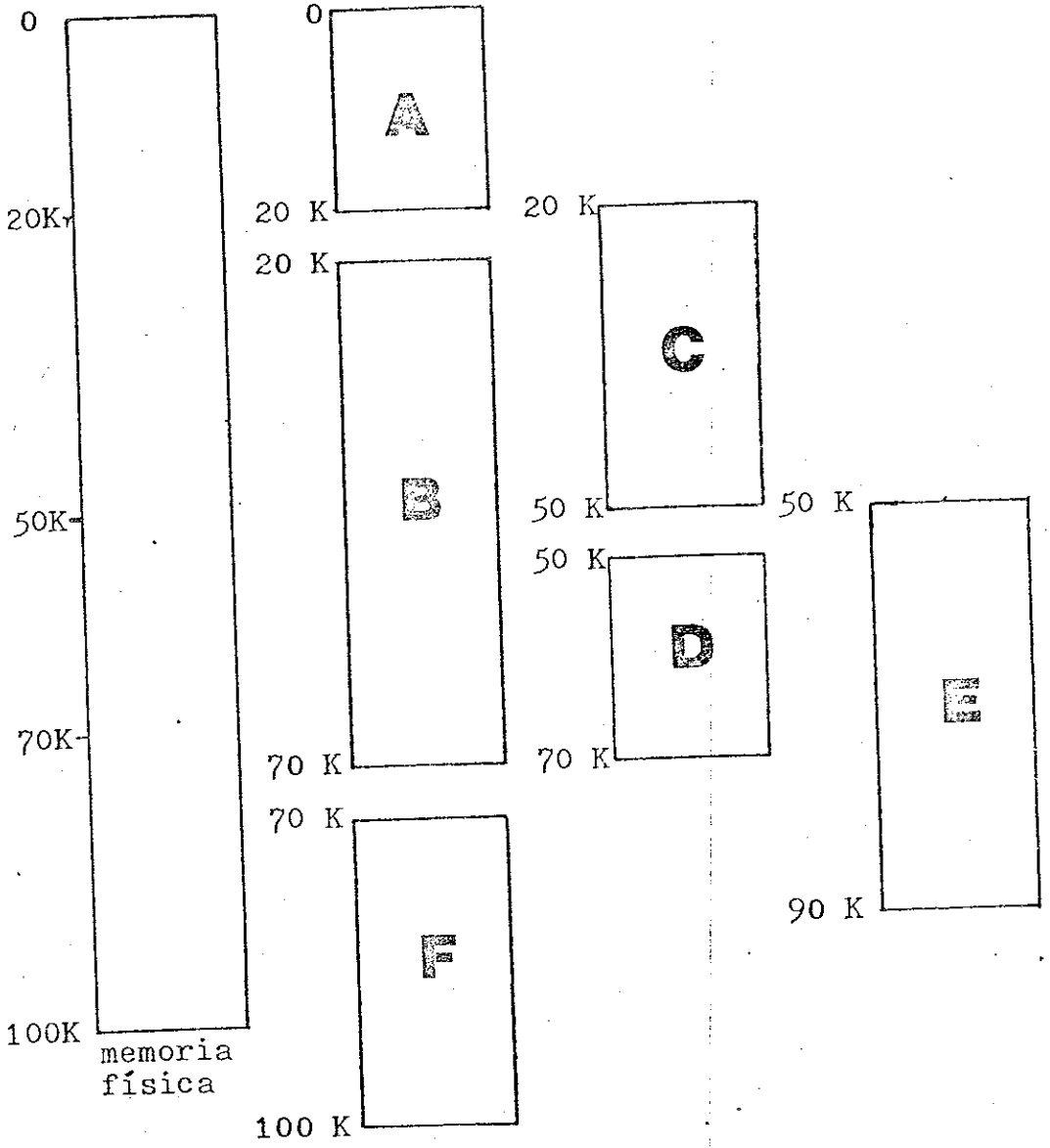


Fig. 3.12. Asignación de segmentos de traslape.

IV. ADMINISTRADOR DE PROCESOS

A. Definición

Habitualmente se utiliza el término proceso, sin precisarlo debidamente y se deja a la intuición de cada quién la comprensión de su significado. En general, se acepta como definición de proceso o tarea secuencial, la actividad resultante de ejecutar un programa con sus datos en un procesador secuencial. No puede asociarse esta idea a la de procesador o a la de programa aisladamente, sino debe hablarse de un par (procesador, programa) en ejecución y entender por programa, una colección de instrucciones que describen un proceso. Si no es esta la concepción del último término, podrá darse el caso de que una única tarea, corresponda a la ejecución de varios programas. También podrá ocurrir que más de un proceso o tarea, se creen sobre un mismo conjunto de instrucciones en un momento determinado. Este caso, sería un ejemplo de ejecución en multitarea.

El número de procesos existentes en un sistema, es independiente de el número de procesadores; una muestra inmediata de tal independencia, la tenemos al trabajar con un solo procesador o unidad central de proceso (UCP)

en la modalidad de multiprogramación.

El administrador de procesos es un elemento del sistema operativo que controla la asignación y el uso del procesador (UCP) por parte de los procesos de cálculo. Implementa los mecanismos de selección y control y de comunicación y paso de información entre ellos. El problema de planificar trabajos, se incluye también como labor de este administrador.

B. Elementos del Administrador de procesos.

Los elementos más significativos del administrador de procesos son:

- Identificador del proceso: es una estructura de datos donde hay información del proceso: nombre, estado, prioridad, derechos, acceso, punteros, etc. Si imaginamos que el identificador es una estructura, el hecho de llenarla es equivalente a "crear un proceso".

- Listas de procesos: los identificadores de procesos se pueden encadenar (con punteros) para formar diferentes listas ordenadas (por ejemplo una cola); cada una de ellas representa una situación idéntica para los procesos encadenados a la misma lista (p.e. lista de

espera de un recurso determinado).

- Planificador de procesos: implementa la búsqueda de un nuevo proceso dentro de las diferentes listas de procesos. La ordenación de la lista, y por lo tanto la búsqueda, puede ser por prioridad, clase, prioridad dentro de la clase, balance entre tiempo de cálculo y de entrada/salida, primer, llegado primer salido, etc.

- Despachador de procesos: realiza la carga de los registros del procesador con la pareja de instrucciones y datos direccionables del proceso seleccionado por el planificador. También carga los registros de trabajo y otra información de control desde las áreas de trabajo.

C. Estados fundamentales y transiciones entre procesos.

Para cada proceso podemos definir tres estados básicos por los que va pasando, mientras se ejecuta:

1. Ejecución: los regitros del procesador han sido cargados con la pareja de instrucciones y datos direccionables del proceso y por lo tanto se ejecuta una secuencia de instrucciones con los datos del entorno correspondiente.

2. Espera: la pareja de instrucciones y datos direccionables del proceso ha sido descargado: no se ejecuta ninguna instrucción del proceso. La pareja de instruccciones, permanece guardada y el identificador del proceso queda encadenado a la lista de procesos en estado de espera de una ocurrencia determinada.

El concepto "esperar una ocurrencia" es completamente general: el proceso puede decidir esperar hasta que se cumpla una determinada condición:

i) final de entrada/salida
ii) señal de otro proceso
iii) final de un intervalo de tiempo

3. Preparado: El identificador del proceso permanece relacionado a una lista de procesos preparados a fin de ser seleccionados para pasar al estado de ejecución. En este estado, los procesos tienen asignados todos los recursos que necesitan, excepto el recurso UCP.

Los estados y las transacciones citadas se pueden representar con la figura siguiente:

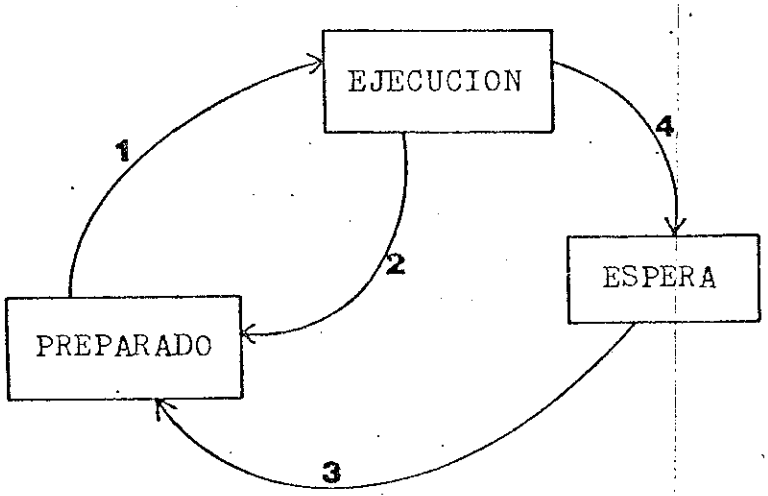


Fig. 4.1. Estados fundamentales y transiciones entre procesos.

Asi pues, entendiendo que en un programa los procesos de cálculo y de entrada/salida cooperan a través de llenar y usar respectivamente el recurso "área de entrada/salida", la secuencia clásica de ejecución: proceso de cálculo, procesos de entrada/salida, proceso de cálculo, proceso de entrada/salida..., es debido a que el proceso de cálculo espera hasta que el proceso de entrada/salida señale la teminación de su trabajo. Mientras tanto, el proceso de cálculo de un segundo programa puede estar en estado de ejecución: la multiprogramación es su resultado.

Generalizando este último concepto. Sobre un sistema pueden trabajar diferentes procesos independientes

o relacionados a través de compartir un recurso. En este último caso, es necesario establecer una sincronización entre ellos en el sentido de autorizar solo a uno de ellos a ceder al recurso; los demás han de esperar. El concepto de esperar en una cola hasta que el recurso quede libre es una consecuencia del razonamiento anterior. También lo es el avisar a los que esperan en cola de que el recurso ha quedado libre.

La transición 2 de la figura representa el hecho de "ponerse en cola y esperar", mientras que la transacción 3 representa que la condición ha sido satisfecha y el proceso vuelve a estar preparado para la ejecución.

Resumiendo las transiciones de la figura anterior son:

Transición 1: el planificador de procesos elige uno de la lista de procesos peoparados y el despachador asigna la pareja instrucciones - datos direccionables a los registros de la unidad central. La elección la realiza el planificador de procesos en base a la estrategia que implemente y que se indicará en la seccion C.1. El resultado es que la UCP

ejecutará las instrucciones del proceso.

Transición 2: El proceso de cálculo pide esperar. Esto fuerza a una transición 1 y por lo tanto a la acción del despachador y planificador de procesos.

Transición 3: El administrador del recurso que hacía esperar al proceso es el encargado de señalar que el recurso ya esta libre y de agregar el proceso en la lista de procesos preparados.

Transicion 4: Se produce cuando en función de su estrategia, el planificador de procesos decide asignar el recurso UCP a otro proceso provocando la correspondiente transición 1.

1. Estrategias de selección de procesos

Las estrategias de selección de procesos pueden ser diferentes según que el sistema haya sido pensando para dar uno u otro tipo de servicio. Según el sistema, el administrador de procesos puede escoger el proceso a ejecutar por:

- 'Round-robin': se asigna un quantum de tiempo 'q' de UCP a cada proceso de la lista de preparados. Una vez el proceso que tiene el procesador agota su quantum de tiempo vuelve a la lista de preparados (transicion 4 de la figura)y volvera a tener procesador cuando le toque.

- 'Round-robin' modificado: Parecido a la estrategia anterior pero teniendo en cuenta que si un proceso no ha agotado su quantum 'q' porque ha pasado al estado de espera por cualquier razón, cuando vuelva a tener el procesador (transiciones 3+1 de la figura) lo hará durante un tiempo q'+q.

- 'Feedback queue': Al llegar un nuevo proceso, recibe tantas veces 'q' tiempo como lo han recibido los que ya llevan rato bajo control del administrador de procesos. al llegar a nivel de los otros procesos el administrador asigna 'q' a cada proceso (Round robin) .

- Prioridad: Los procesos llevan asociada una prioridad. El adminisdor de procesos escoge entre los de la lista de preparados el que tiene prioridad más alta.

Prioridades dinámicas: Estrategia parecida a la anterior con la diferencia que cada vez que un proceso hace una transición 4 su prioridad es modificada por el

administrador de procesos.

Balanceo del sistema: El administrador de procesos pretende que haya una carga sesiblemente igual de procesos de cálculo y de procesos de entrada/salida para optimizar el uso global del sistema.

Hay otras estrategias más especializadas que se implementan por separado o formando combinaciones, de manera que el administrador puede responder a un conjunto más amplio de situaciones de forma más flexible y eficiente.

Los procesos del sistema operativo pueden ser tratados como procesos generales en competencia con los del usuario se pueden declarar privilegiados.

En este caso, el proceso tendrá la más alta prioridad del procesador y por lo tanto se ejecutará sin posibilidad de que otros procesos del usuario de entrada/salida le interrumpan.

D. Comunicación de Procesos.

Para aumentar la eficiencia, las distintas tareas comparten tanto el software, como el hardware del

sistema. Por ello, tendrán necesidad de intercambiar información e introducir mecanismos que permitan el intercambio.

Los problemas a resolver cuando existe concurrencia, podrían identificarse dentro de los siguientes puntos:

- Determinismo.
- Exclusión mutua.
- Sincronización.
- Deadlock (bloque mutuo entre tareas)

Cuando un conjunto de tareas pueden ser ejecutadas en paralelo, interesa a veces asegurar la unicidad de los resultados pese a las variaciones de velocidad de ejecución de cada una y su orden de ejecución. Es decir, que el sistema de tareas o procesos que satisfaga esta propiedad, se dice que es funcional, dependiendo de la velocidad de terminado.

Procesos concurrentes que operan sobre conjuntos disjuntos de variables, se llaman disjuntos o no - interactivos. La no - interactividad, es una condición suficiente para el comportamiento indepen - diente del tiempo de los procesos. Si hay interactividadad, cuando una variable es referenciada por una tarea

y existe una segunda que cambia su valor, el resultado de la primera dependerá del momento en que se produzca el cambio.

Si se cumplen determinados requisitos, los programas pueden ser incondicionalmente funcionales y para ellos, se cumple la siguiente propiedad: cualquier propiedad:cualquier interconexión de un número finito de programas incondicionalmente funcionales, es incondicionalmente funcional como un todo.

Esta conclusión, permitirá a un diseñador verificar que un programa largo es funcional, haciendo un análisis paso a paso de los pequeños programas componentes.

E. Exclusión mutua. El problema de las secciones críticas.

A pesar de que los recursos de un computador, pueden ser compartidos, en muchas ocasiones, sólo pueden ser utilizados por un proceso a la vez. En tal caso, se les llama recursos críticos y si varios procesos desean compartirlos, deben sincronizarse de forma que cuando uno los use, los demas esperen. Ejemplos de recursos críticos, son las cintas, lectoras, impresoras, etc. También, en ocasiones

los serán las variables o rutinas.que se comparten entre procesos.

Las regiones dentro de las tareas, desde las que se accesa a recursos críticos, pueden ser aisladas. Se les denomina regiones o secciones críticas y deben de tener la propiedad de exclusividad mutua.

Una solución realista al problema de la exclusión mutua debe satisfacer determinados criterios:

- El recurso exclusivo, podrá ser usado por un proceso a la vez como máximo.

- Cuando es solicitado por varios procesos simultáneamente, le ha de ser garantizado a uno de ellos en un tiempo finito.

- Si alguno lo adquiere, debe liberarlo en un tiempo también finito.

- Los procesos que esperan por un recurso, no deben consumir tiempo de CPU. Es decir, hay que evitar las esperas que ocupan el procesador.

Por lo tanto y como concecuencia, para las secciones

críticas, deben hacer los siguientes supuestos:

- Cuando un proceso quiera entrar en una sección crítica, debe poder hacerlo en un tiempo finito.

- Como máximo, un proceso a la vez puede estar en una sección crítica determinada.

- Un proceso estará en la citada región, solamente durante un tiempo finito.

Como estos supuestos, un proceso podrá entrar inmediatamente en una región crítica, si no hay otro dentro, y si lo hubiese, tendrá que esperar. Si existen varias tareas esperando, cuando la sección se desocupe, una de ellas ha de pasar a ocuparla y la desición de cual ha de ser, se ha de tomar en un tiempo basándose en una regla de palanificación equitativa.

La exclusión mutua, se implanta en el hardware haciendo las operaciones de acceso a memoria indivisibles. Esto significa que si dos tareas intentan almacenar un valor en una misma posición, el hardware hace arbitrario pemitindo un sólo acceso, mientras la otra espera.

Existen algoritmos (por ejemplo los de Dijkstra o Dekker),que resuelve el problema de las secciones críticas utilizando variables booleanas para indicar el estado de ocupación de una región y el turno de entrada de las tareas. La dificultad que surge al tratar de hallar una solución, es que estas variables introducidas, son compartidas y por tanto habrá que protegerlas a su vez, dentro de secciones críticas.

La depuración de errores en los casos de exclusividad mutua, es muy compleja.

F. Sincronización. Semáforos.

Para determinar problemas de sincronización en procesos interactivos como el estudiado en la sección precedente, las soluciones de software resultan sofisticadas y obscuras. Por ello, se introducen dos nuevas operaciones primitivas de bajo nivel, que han simplificado considerablemente la comunicación y sincronización de tareas. En forma abstracta, estas primitivas P y V, son operaciones indivisibles que operan sobre globales enteras, llamadas semáforos. Estas en forma más concreta son registros de hardware o posiciones de memoria que contienen enteros. Puesto que podrá ocurrir que varios procesos usen el mismo

semáforo, el sistema creará una cola asociada a él.

En otras palabras cuando dos procesos paralelos dependen uno del otro para la continuación de su operación (incluyendo probablemente la activación de otros procesos), alguna forma de comunicación elemental es necesaria entre ellos, para señalar cuando los eventos principales se han llevado a cabo. Esto puede ser implementado por el uso de una celda compartida de memoria, que es usada para indicar cuándo un evento ha ocurrido; esta celda es conocida como un 'semáforo'.

Por ejemplo, si un proceso 'A' no puede continuar su ejecución a un paso alfa, por un dato que tendría que ser provisto por un proceso 'B', del proceso A 'prende' el semáforo al valor negativo -1, indicando que la ejecución de A no puede continuar con el paso siguiente alfa. Cuando el proceso B obtiene el dato requerido, este agrega un 1 al semáforo, haciendo su valor cero y permitiendo entonces continuar la ejecución del proceso A.

V. CONCLUSIONES

El estudio de los sistemas operativos ayuda a englobar muchos conceptos vistos en una carrera de Ciencias de la Computación, permitiendo ahondar en campos más sofisticados.

Es conveniente que se enfatice en los contenidos de los cursos de sistemas operativos, pues el material que se abarca permite al estudiante afirmar las técnicas y algoritmos para manipulación de estructuras de datos e implementar nuevos, para cada una de las partes que conforman un sistema operativo.

En general, el estudio de los sistemas operativos, permite optimizar los recursos de la máquina que se tenga a disposición. De esta forma, se podría determinar si los dispositivos de entrada/salida están siendo asignados adecuadamente, o si el tiempo de espera de cada proceso se puede minimizar, o si el promedio de tamaño de los trabajos en memoria en un período dado de tiempo, es consecuente con el tamaño de las particiones utilizadas en un sistema de cómputo específico; como se trabaja en el apéndice A de este trabajo.

Las aplicaciones que automáticamente se derivan del estudio de los sistemas operativos son innumerables, siendo

todas ellas, una ventajosa ayuda para el operador del sistema.

Es así como esta ayuda se basa específicamente, en la manipulación adecuada de estructuras de datos y un conocimiento claro de cada componente de un sistema operativo, con el objeto de lograr la optimización en la administración de recursos.

GLOSARIO

ASSEMBLERS: Un assembler (contraccción para 'assembly program') es un programa que facilita la codificación de programas en lenguaje de máquina, tomando las representaciones simbólicas de palabras individuales y conviertiéndolas en una forma adecuada para la entrada a un encadenador (linker) o un cargador (loader).Un lenguaaje assembly es un lenguaje de programación en donde el conjunto básico de instrucciones incluye códigos de operación de la máquina y cuyas estructuras de datos accesan directamente a la memoria y registros de la máquina. Un assembler es un compilador del lenguaje assembly.

BUFFER: Dispositivo auxiliar de entrada/salida que contiene datos temporalmente y que puede realizar otras funciones en unión de varias máquinas de entrada/salida.

Es considerado también como un dispositivo de de almacenamiento usado para compensar la diferencia en la transferencia de datos, o el tiempo de ocurrencia de los eventos cuando se

transmiten datos, de un dispositivo a otro.

BUS: Un circuito sobre el que se trasmiten datos o energía. Es el que actúa como conección común entre un número de dispositivos.
La mayoría de computadores se comunican internamente a través de un bus de datos. La mayoría son bidireccionales, capaces de transmitir datos desde y hacia la UCP y y dispositivos o memoria periférica.

CALENDARIZADOR(Sheduler): El calendarizador se encarga de asignar y estudiar el tiempo operativo de todas las aactividades de procesamiento de datos, con el objeto de asegurar que el equipo de procesamiento de datos sea efectiva y eficientemente utilizado.

COMPUTADOR: Un dispositivo capaz de aceptar informacion realizar procesos predefinidos y proporcionar los resultados de estos procesos. Este generalmente consiste de dispositivos de entrada y salida, almacenamiento, unidades lógicas y aritméticas y una unidad de control.

DRIVER(BUFFER): Un driver o buffer es un circuito diseñado para aislar una entrada, de la fuente motriz; este está generalmente entre un chip microprocesador y las líneas de direccionamiento de memoria.

DRIVER(DISPOSITIVO): Generalmente se refiere a un módulo del sistema operativo que controla un periférico específico de entrada/salida. El driver es llamado por el ejecutivo del sistema en respuesta a una llamada de entrada/salida del programa del usuario. En la mayoría de sistemas cada tipo de periférico tiene un único driver dentro del sistema operativo.

ESTRUCTURAS DE DATOS: El término estructura es usado en muchos campos diferentes para denotar objetos que son construidos en una forma regular y característica a partir de sus componentes. Una estructura de datos es una estructura, cuyos componentes son objetos de datos.

HARDWARE: Es el equipo mecánico, eléctrico y usado para el procesamiento de datos, consistiendo en transistores alambres, motores, resistores, etc.

PROCESAMIENTO EN LINEA: La operación de terminales, archivos y otros equipos auxiliares bajo el control directo y absoluto del procesador, para eliminar la necesidad de intervención humana en cualquier etapa entre la entrada inicial y la salida del computador.

PROCESAMIENTO EN TANDAS(LOTES)(BATCH PROCESS): Un proceso de procesamiento secuencial que utiliza una una acumulación o grupo de unidades; ésto es encontrarse al procesamiento en línea, durante el cual una unidad de datos o información es procesada en el mismo momento en que es ingresada.

SOFTWARE: El conjunto de programas necesitados para hacer más eficiente y facilitar el uso del equipo. El software incluye assemblers, generadores, compiladores, sistemas operativos, programas de librería y programas de aplicación en la industria.

TIEMPO COMPARTIDO: El uso de un dispositivo para dos o más propósitos durante el mismo intervalo de tiempo, realizado por la división del tiempo

entre las acciones de las componentes del del computador.

TIME SLICE:(TAJADA DE TIEMPO): Determinado intervalo de tiempo durante el trabajo; puede utilizar un recurso sin interrupción.

USUARIO: La persona o compañía que utiliza una terminal remota dentro de un sistema de cómputo de tiempo compartido o un computador simple, con el objeto de ejecutar un proceso.

BIBLIOGRAFIA

- Asociation for Computing Machinery. "Selected papers from
  1981 the seventh symposium on operating systems principples". Comunications of the ACM (U.S.A.); 23(2):67-71.

- Coffman, E. Jr. and P. Danning. Operating systems theory.
  1973 1st ed. New Jersey, Prentice - Hall, Inc. 331 pp.

- IEEE Computer Societty. "A comparison of dynamic and
  1981 static virtual memory allocation algorithms". Software Engineer (U.S.A.); 7(1):122-132.

- _____; "A mathematical model for compartion of static and dinamic memory allocation in a paged system". Software Engineering (U.S.A.); 7(4):403-417.

- _____; "Tutorial Series, operating systems ". Computer(U. S. A.); 14(6):69 -79.

- Nuevo diccionario Cuyás inglés-español y español-inglés.
  1972 Appleton. 5ta ed. New Jersey, Prentice - Hall, Inc. 589 pp.

- Ralston, A. Encyclopedia of computer science. 1st ed.
  1974 New York, Van Nostrano Reihold. 1523 pp.

- Shaw, AA. The logical design of operating systems.
  1974 1ra Ed. New Jersey, Prenteice - Hall, Inc. 306 pp.

- Sippl,C. and H. Sippl. Computer dictionary & handbook. 3ra ed.
  1980 Indianapolis, Sans & Co., Inc. 928 pp.

APENDICE A

Un sistema de Control de rendimiento

El objetivo de este proceso es el de determinar el tamaño promedio de los programas en memoria principal que están en estado uno (1) es decir calendarizados ('schedule'(sc)); y poder, con ello, obtener el tamaño óptimo de los bloques en la memoria física.

Como se trató en el capítulo de Técnicas de administración de memoria (capítulo III de este trabajo), la asignación paginada de memoria se introduce con el objeto de resolver en alguna forma, los problemas de desaprovechamiento de tiempo y del espacio de memoria. Este tipo de administración, se utiliza en sistemas operativos de los tipos MFP(multiprogramación con un número fijo de tareas) y MVT (multiprogramación con un número variable de tareas).

Uno de los principales problemas que podría tener una administración de memoria de este tipo, sería que el tamaño de página en memoria con la que debe trabajar el sistema operativo, no fuese de un tamaño adecuado al tamaño promedio de los trabajos que entran al sistema. Con el objeto de optimizar el manejo de la memoria en un sistema específico, se de-

sarrolló este sistema de control de rendimiento.

El sistema fue implementado en el computador HP 1000, propiedad de la Universidad del Valle, en el lenguaje FORTRAN IV. El programa principal, del que se incluye su codificación, estima el tamaño promedio de los trabajos que se ejecutan dentro del sistema, durante un período dado de tiempo.

Se utiliza una tabla propia del sistema que contiene un segmento de identificación para cada programa en estado de ejecución (estado 1). Cada segmento es de 33 palabras y contiene información estática y dinámica que define las propiedades de un trabajo en particular. La información estática es colocada en la tabla durante el tiempo de generación o cuando el programa es cargado al sistema. La información dinámica es mantenida por el ejecutivo del sistema operativo.

El número de segmentos de identificación contenidos en el sistema es directamente proporcional al número de programas que pueden estar en memoria principal en cualquier momento. Si todos los segmentos de identificación están en uso, ningún otro programa podrá ser agregado o puesto en ejecución, hasta que otro programa sea suspendido por las condiciones conocidas:

i) Finalización normal del trabajo.

ii) Detección de un error.

iii) Requerimiento de una operación de entrada/salida.

iv) Finalización de los fragmentos de tiempo.

Este programa, utiliza 3 de las 33 palabras de cada segmento:

PALABRA	BITS	CONTENIDO
1	0-15	encadenamiento
15	0- 3	estado
21	10-14	no. de páginas

y con ello, clasifica los programas en rangos de tamaño:

- promedio de programas utilizando entre 0 y 5 páginas.
- número máximo de programas que han utilizado entre 0 y 5 páginas.
- promedio de programas utilizando entre 6 y 10 páginas.
- número máximo de programas que han utilizado entre 6 y 10 páginas.
- promedio de programas utilizando entre 11 y 15 páginas.

- número máximo de programas que han utilizado entre 11 y 15 páginas.
- promedio de programas utilizando entre 16 y 20 páginas.
- número máximo de programas que han utilizado entre 16 y 20 páginas.

Cada uno de los rangos anteriores es guardado en una palabra de una estructura cuya unidad tiene una longitud de 8 palabras. Los rangos son tabulados y grabados en el archivo de datos estadísticos (ESTADI) cada 10 minutos. Al final del programa, el archivo ESTADI, contendrá la tabulación de los rangos para el período de tiempo que se haya ejecutado el programa.

Algoritmo:

1) Inicialización de variables.
2) Se obtiene el primer puntero al área de identificación de los programas. ('Program id segment').
   El primer puntero se encuentra en la posición 1657 en octal.
3) Se localizan los bits 0-3 de la palabra número 15 de cada segmento, donde se encontrará el estado en el que se encuentra el programa.
4) Si el estado es uno, obtiene la palabra 21, bits 10-14,

del segmento de identificación.

5) Aumenta los rangos de tabulación según el número de páginas.

6) Se encuentra el puntero al próximo segmento de identificación.

7) Se genera un record en el archivo ESTADI, si han transcurrido 10 minutos.

8) Regresa a 3.

Este programa es un ejemplo de los controles de rendimiento que se pueden realizar dentro de las especificaciones de un sistema operativo dado. Lo que llevaría a realizar otras mediciones y posibles optimizaciones en la asignación de los recursos dentro de un sistema operativo.

```
0001  FTN4
0002  C      051281
0003       PROGRAM PROV3
0004  C*************************************************************************
0005  C*************************************************************************
0006  C**                                                                     **
0007  C**                                                                     **
0008  C**                                                                     **
0009  C**                                                                     **
0010  C**                                                                     **
0011  C**          "INVESTIGACION SOBRE EL TAMAÑO PROMEDIO DE                 **
0012  C**                      LOS PROGRAMAS EN MEMORIA"                      **
0013  C**                                                                     **
0014  C**                                                                     **
0015  C**                                                                     **
0016  C**                                                                     **
0017  C**                                                                     **
0018  C**                                                                     **
0019  C**                                                                     **
0020  C**                                                                     **
0021  C**     OBJETIVO DEL PROGRAMA:                                          **
0022  C**     ----------------------                                          **
0023  C**                                                                     **
0024  C** ESTE PROGRAMA TIENE POR OBJETO EL OBTENER EL TAMAÑO PROME-          **
0025  C** DIO DE LOS PROGRAMAS EN MEMORIA PRINCIPAL QUE ESTAN EN ESTADO       **
0026  C** UNO(1)+SCHEDULE (SC)                                               **
0027  C**                                                                     **
0028  C**                                                                     **
0029  C**     PROCEDIMIENTO UTILIZADO:                                        **
0030  C**     ------------------------                                        **
0031  C**                                                                     **
0032  C** ESTE PROGRAMA UTILIZA EL "PROGRAM ID SEGMENT", EL CUAL ES UNA       **
0033  C** TABLA FORMADA POR SEGMENTOS DE 33 PALABRAS, CADA PROGRAMA           **
0034  C** TIENE UN SEGMENTO DE IDENTIFICACION CONTENIENDO INFORMACION         **
0035  C** ESTATICA Y DINAMICA, DEFINIENDO LAS PROPIEDADES DEL PROGRAMA.       **
0036  C** LA INFORMACION ESTATICA ES PUESTA DURANTE EL TIEMPO DE GENE-        **
0037  C** RACION O CUANDO EL PROGRAMA ES CARGADO (ON-LINE), LA INFOR-         **
0038  C** MACION DINAMICA ES MANTENIDA POR EL EJECUTIVO DEL SISTEMA           **
```

```
0039  C**  OPERATIVO                                                    **
0040  C**  EL NUMERO DE SEGMENTOS DE IDENTIFICACION CONTENIDOS EN EL
0041  C**  SISTEMA ES ESTABLECIDO DURANTE LA GENERACION DEL SISTEMA, Y  **
0042  C**  ESTA DIRECTAMENTE RELACIONADO CON EL NUMERO DE PROGRAMAS QUE **
0043  C**  PUEDEN ESTAR EN MEMORIA PRINCIPAL EN CUALQUIER MOMENTO.      **
0044  C**  SI TODOS LOS SEGMENTOS DE IDENTIFICACION ESTAN EN USO, NIN-  **
0045  C**  GUN PROGRAMA PUEDE SER AGREGADO EN LINEA, HASTA QUE OTRO PRO **
0046  C**  GRAMA EXISTENTE SEA PUESTO EN "OFFED", ES DECIR QUE SEA REMO **
0047  C**  VIDO DEL SISTEMA PARA RECUPERAR EL SEGMENTO DE IDENTIFICA-   **
0048  C**  CION.                                                        **
0049  C**  ESTE PROGRAMA UTILIZA LOS SIGUIENTES BITS Y PALABRAS:        **
0050  C**                                                               **
0051  C**    PALABRA              BITS                SIGNIFICADO       **
0052  C**                                                               **
0053  C**       1                 0 - 15              encadenamiento    **
0054  C**       15                0 - 3               status            **
0055  C**       21                10- 14              no. de páginas    **
0056  C**                                                               **
0057  C**                                                               **
0058  C**     ALGORITMO UTILIZADO:                                      **
0059  C**  ----------------------                                       **
0060  C**                                                               **
0061  C**  1. Inicialización de variables                               **
0062  C**     Aquí se inicializa la estructura de datos utilizada       **
0063  C**     en Sistem Common.                                         **
0064  C**     Estructura de Datos en Sistem Common:                     **
0065  C**     Longitud: 8 palabras.                                     **
0066  C**     Tipo: entero                                              **
0067  C**     Contenido:                                                **
0068  C**               SC(1).....PROMEDIO DE PROGRAMAS                 **
0069  C**                         UTILIZANDO ENTRE 0 y 5                **
0070  C**                         PAGINAS.                              **
0071  C**               SC(2).....NUMERO MAXIMO DE PROGRAMAS            **
0072  C**                         QUE HAN UTILIZADO ENTRE               **
0073  C**                         0 y 5 PAGINAS                         **
0074  C**               SC(3).....PROMEDIO DE PROGRAMAS                 **
0075  C**                         UTILIZANDO ENTRE 6 y 10               **
0076  C**                         PAGINAS.                              **
```

```
0077  C**              SC(4).....NUMERO MAXIMO DE PROGRAMAS      **
0078  C**                        QUE HAN UTILIZADO ENTRE         **
0079  C**                        6 y 10 PAGINAS.                 **
0080  C**              SC(5).....PROMEDIO DE PROGRAMAS           **
0081  C**                        UTILIZANDO ENTRE 11 y 15        **
0082  C**                        PAGINAS.                        **
0083  C**              SC(6).....NUMERO MAXIMO DE PROGRAMAS      **
0084  C**                        QUE HAN UTILIZADO ENTRE         **
0085  C**                        11 y 15 PAGINAS                 **
0086  C**              SC(7).....PROMEDIO DE PROGRAMAS           **
0087  C**                        UTILIZANDO ENTRE 16 y 20        **
0088  C**                        PAGINAS.                        **
0089  C**              SC(8).....NUMERO MAXIMO DE PROGRAMAS      **
0090  C**                        QUE HAN UTILIZADO ENTRE         **
0091  C**                        16 y 20 PAGINAS.                **
0092  C**   ALGORITMO UTILIZADO:
0093  C**   1. Inicialización de variables.
0094  C**   2. SE OBTIENE EL PRIMER POINTER AL AREA DE IDENTIFICA-
0095  C**      CION DE LOS PROGRAMAS.
0096  C**      EL PRIMER POINTER SE ENCUENTRA EN LA POSICION 1657
0097  C**      OCTAL.
0098  C**   3. SE ACCESA LA PALABRA NUMERO 15 DE CADA IDENTIFICACION
0099  C**      DE LOS PROGRAMAS.
0100  C**      EN ESTA PALABRA SE ENCUENTRA EL STATUS EN LOS BITS 0-3.
0101  C**   4. SI EL STATUS ES UNO, OBTIENE LA PALABRA 21,
0102  C**      DEL BLOCK DE IDENTIFICACION,
0103  C**      EN PROGRAMA.
0104  C**   5. DISTRIBUCION SEGUN EL NUMERO DE
0105  C**      PAGINAS QUE CONTENGA LA IDENTIFICACION
0106  C**
0107  C*************************************************************************
0108  C*************************************************************************
0109  C-----------------------
0110  C     INICIALIZACION DE VARIABLES
0111  C-----------------------
0112        INTEGER SC(10), NAME(3),IRES(5),IDCB(144)
```

```
0113      DIMENSION INAME(6),ITIME(5),IYEAR(1),IBUFR(3),IBUF(4)
0114      DATA IBUFR/2HES,2HTA,2HDI/
0115      ICCNT=0
0116    C---------------------
0117    C  SE OBTIENE EL PRIMER POINTER AL AREA
0118    C  DE IDENTIFICACIONES DE LOS PROGRAMAS.
0119    C---------------------
0120      IPTR=26214B
0121      I=IGET(IPTR)
0122    C---------------------
0123    C  SE ACCESA LA PALABRA NUMERO 11 DE
0124    C  CADA IDENTIFICACION DE SEGMENTOS.
0125    C---------------------
0126    10 J=I+11
0127      DO 15 K=1,3
0128    15 NAME(K)=IGET(J+K)
0129      NAME(3)=NAME(3)/256
0130      J=I+15
0131      INOW=IGET(J)
0132    C---------------------
0133    C  SE ACCESA EL STATUS Y SE VERIFICA
0134    C  SI ESTA EN ESTADO 1 (SC).
0135    C  SABIENDO QUE EL STATUS OCUPA LOS
0136    C  BITS 0-3 DE LA 15a. PALABRA.
0137    C---------------------
0138      IVAR=IAND(INOW,17B)
0139      IF (IVAR.NE.1) GO TO 70
0140    C---------------------
0141    C  SI EL STATUS ES 1:
0142    C  OBTIENE LA 21a. PALABRA DEL BLOCK
0143    C  DE IDENTIFICACION.
0144    C---------------------
0145      J=I+21
0146      ILAST=IGET(J)
0147    C---------------------
0148    C  OBTIENE EL NUMERO DE PAGINAS DEL
0149    C  PROGRAMA DE ESA IDENTIFICACION.
```

```
0150  C  SABIENDO QUE EL No. DE PAGINAS SE
0151  C  ENCUENTRA EN LOS BITS 10-14 DE LA
0152  C  21a. PALABRA.
0153  C-------------------------
0154     IVAR=ILAST/1024
0155     IVAR=IAND(IVAR,31)
0156  C-------------------------
0157  C  DISTRIBUCION SEGUN EL NUMERO DE PAGI-
0158  C  NAS QUE CONTENGA LA IDENTIFICACION DEL
0159  C  SEMENTO.
0160  C-------------------------
0161     IF(IVAR.GE.6)GO TO 30
0162  C-------------------------
0163  C  ***DE 0 A 5 PAGINAS***
0164  C-------------------------
0165     SC(1)=SC(1)+IVAR
0166     SC(2)=SC(2)+1
0167     GO TO 70
0168  C-------------------------
0169  C  ***DE 6 A 10 PAGINAS***
0170  C-------------------------
0171  30 IF(IVAR.GE.11) GO TO 40
0172     SC(3)=SC(3)+IVAR
0173     SC(4)=SC(4)+1
0174     GO TO 70
0175  40 IF(IVAR.GE.16) GO TO 50
0176  C-------------------------
0177  C  ***DE 11 A 15 PAGINAS***
0178  C-------------------------
0179     SC(5)=SC(5)+IVAR
0180     SC(6)=SC(6)+1
0181     GO TO 70
0182  50 IF(IVAR.GE.21) GO TO 55
0183  C-------------------------
0184  C  ***DE 16 A 20 PAGINAS***
0185  C-------------------------
```

```
0186      SC(7)=SC(7)+IVAR
0187      SC(8)=SC(8)+1
0188      GO TO 70
0189 55   IF (IVAR,GE,26) GO TO 60
0190 C-----------------------------
0191 C    ***DE 21 A 25 PAGINAS***
0192 C-----------------------------
0193      SC(9)=SC(9)+1VAR
0194      SC(10)=SC(10)+1
0195      GO TO 70
0196 C-----------------------------
0197 C    MENSAJE DE ERROR POR MAS DE
0198 C    25 PAGINAS;
0199 C-----------------------------
0200 60   WRITE(1,500)
0201 C-----------------------------
0202 C    SE ENCUENTRA EL NUEVO POINTER A LA
0203 C    PROXIMA IDENTIFICACION HASTA QUE EL
0204 C    POINTER PROXIMO SEA CERO (0).
0205 C-----------------------------
0206 70   I=I+41B
0207 95   FORMAT(2X,"POINTER' ",I9)
0208      ICONT=ICONT+1
0209      IF(ICONT.NE.73) GO TO 10
0210 C-----------------------------
0211 C    SE OBTIENE EL NUMERO PROMEDIO DE
0212 C    PAGINAS POR DIVISION.
0213 C-----------------------------
0214      IF (SC(2).EQ.0) GO TO 71
0215      IRES(1)=XS(1)/SC(2)
0216 71   IF(SC(4).EQ.0) GO TO 72
0217      IRES(2)=SC(3)/SC(4)
0218 72   IF(SC(6).EQ.0) GO TO 73
0219      IRES(3)=SC(5)/SC(6)
0220 73   IF(SC(8).EQ.0) GO TO 74
0221      IRES(4)=SC(7)/SC(8)
```

```
0222 74   IF(SC(10).EQ.0) GO TO 75
0223      IRES(5)=SC(9)/SC(10)
0224 C-----------------------------
0225 C    OBTIENE LA HORA DE TRABAJO PARA
0226 C    ASIGNAR POSICION EN ARCHIVO DE
0227 C    ESTADISTICA.
0228 C    ITIME(4)=HORA
0229 C-----------------------------
0230 75   ICODE=11
0231      CALL EXEC(ICODE,ITIME,IYEAR)
0232 C-----------------------------
0233 C    SE LLENA EL ARCHIVO DE ESTADISTICA
0234 C    CON LOS DATOS OBTENIDOS EN ESTOS
0235 C    10 MINUTOS.
0236 C-----------------------------
0237      IHORA=ITIME(4)
0238      IF(IHORA.NE.0) GO TO 96
0239      IHORA=1
0240      GO TO 97
0241 96   IHORA=(IHORA*5)+1
0242 97   IFIN=IHORA+4
0243      J=0
0244      K=0
0245 C-----------------------------
0246 C    APERTURA DEL ARCHIVO ESTADI
0247 C-----------------------------
0248      CALL OPEN(IDCB,IERR,IBUFR)
0249      IF (IERR.GE.0) GO TO 1000
0250 C-----------------------------
0251 C    MENSAJE DE ERROR.
0252 C-----------------------------
0253      WRITE(1,800)
0254      GO TO 101
0255 C-----------------------------
0256 C    SE ACTUALIZA EL ARCHIVO ESTADI EN
0257 C    LAS 5 SUBDIVISIONES DE CADA HORA.
0258 C-----------------------------
```

```
0259  1000 DO 99 NUM=IHORA,IFIN
0260       CALL READF(IDCB,IERR,IBUF,4,LEN,NUM)
0261       J=J+1
0262       K=K+2
0263       ITEMP=IBUF(1)
0264       IBUF(1)=IBUF(1)+1
0265       IF(ITEMP,EQ,0) GO TO 98
0266       IBUF(2)=((ITEMP/IBUF(2))+SC(K))/IBUF(1)
0267       IF(IBUF(3).GT.SC(K)) GO TO 99
0268       IBUF(3)=SC(K)
0269       GO TO 99
0270  98   IBUF(2)=SC(K)
0271       IBUF(3)=SC(K)
0272  99   CALL WRITEF(IDCB,IERR,IBUF,4,NUM)
0273  C-----------------------------------
0274  C    SE CIERRA EL ARCHIVO ESTADI
0275  C-----------------------------------
0276  101  CALL CLOSE (IDCB)
0277  C**********************************
0278  C**********************************
0279  C******FORMATOS DE IMPRESION********
0280  C**********************************
0281  C**********************************
0282  100  FORMAT(2X,"LINEA DE STATUS:",I9)
0283  200  FORMAT(2X,"STATUS: ",I9)
0284  300  FORMAT(2X,"LINEA DE PAGINAS: ",I9)
0285  400  FORMAT(2X,"PAGINAS: ",I9)
0286  500  FORMAT(2X,"EXISTE UNA PARTICION CON MAS DE 25 PAGIN
0287  600  FORMAT(2X,"NAME: ",3A2)
0288  700  FORMAT(2X,"HORA: ",I4)
0289  800  FORMAT(2X,"ERROR AL ABRIR EL ARCHIVO ESTADI")
0290  900  FORMAT(2X,"POSICION DEL SC: ",I8)
0291  950  FORMAT(2X,"POSICION DE PROMEDIOS: ",I9)
0292       END
0293       END$
```

```
0001   FTN4
0002   C     101281
0003         PROGRAM GRAFI
0004   C*****************************************************
0005   C*****************************************************
0006   C**                                                  **
0007   C**                                                  **
0008   C**                                                  **
0009   C**           GRAFICA ESTADISTICA                    **
0010   C**           ======================                 **
0011   C**                                                  **
0012   C**     PROYECTO DESARROLLADO POR LISSETTE GALVEZ     **
0013   C**     ========================================     **
0014   C**                                                  **
0015   C**                     1983                         **
0016   C**                                                  **
0017   C**     UNIVERSIDAD DEL VALLE DE GUATEMALA.          **
0018   C**     1983.                                        **
0019   C**     PROGRAMA DE UTILIDAD .                       **
0020   C**                                                  **
0021   C**                                                  **
0022   C**     DESCRIPCION DEL PROGRAMA:                    **
0023   C**     =========================                    **
0024   C**     ESTE PROGRAMA DA UNA GRAFICA ESTADISTICA EN LA CUAL **
0025   C**     SE PUEDE APRECIAR EL NUMERO MAXIMO DE PROGRAMAS QUE **
0026   C**     HAN ESTADO EN MEMORIA PRINCIPAL DURANTE LAS ULTIMAS **
0027   C**     HORAS QUE HA CORRIDO EL PROGRAMA PROV3. LA MEDICION **
0028   C**     SE HACE CADA 10 MINUTOS.                     **
0029   C**                                                  **
0030   C**     ESTE PROGRAMA UTILIZA LA UNIDAD NUMERO 6----(IMPRE- **
0031   C**     SORA) PRESENTANDO EL INFORME A TRAVES DE UNA GRAFICA **
0032   C**     EN LA CUAL EL EJE DE LAS "y" SIGNIFICA EL TIEMPO Y **
0033   C**     EL NUMERO DE PAGINAS Y EL EJE DE LAS "x" SIGNIFICA **
0034   C**     EL NUMERO MAXIMO DE PROGRAMAS, OBTENIENDO ASI EL NU **
0035   C**     MERO MAXIMO DE PROGRAMAS QUE SE HAN DADO EN CADA HO **
0036   C**     RA Y PARA CADA TAMAÑO DE PARTICION.          **
0037   C**                                                  **
0038   C**                                                  **
```

```
0039  C**                                                                **
0040  C**                                                                **
0041  C**                                                                **
0042  C**                                                                **
0043  C**                                                                **
0044  C**                                                                **
0045  C**                                                                **
0046  C**                                                                **
0047  C**                                                                **
0048  C**                                                                **
0049  C********************************************************************
0050  C********************************************************************
0051  C-----------------------------
0052  C INICIALIZACION DE VARIABLES
0053  C-----------------------------
0054        INTEGER IDCB(144)
0055        DIMENSION IBUFR(3),IBUF(4)
0056        DATA IBUFR/2HES,2HTA,2HDI/
0057        M=0
0058        I=0
0059        J=1
0060        K=1
0061        L=5
0062  C-----------------------------
0063  C APERTURA DEL ARCHIVO ESTADI.
0064  C-----------------------------
0065        CALL OPEN(IDCB,IERR,IBUFR)
0066        IF(IERR.GE.0) GO TO 19
0067  C-----------------------------
0068  C MENSAJE DE ERROR POR APERTURA
0069  C ERRONEA DEL ARCHIVO ESTADI.
0070  C-----------------------------
0071        WRITE(1,90)
0072        GO TO 85
0073  C-----------------------------
```

```
0074  C     ROTULOS Y ENCABEZADOS DE LAS
0075  C     ORDENADAS DE LA GRAFICA.
0076  C-------------------------------
0077  19    WRITE(6,100)
0078  C-------------------------------
0079  C LECTURA DEL ARCHIVO ESTADI E INVES-
0080  C TIGACION DEL NUMERO MAXIMO DE PRO-
0081  C GRAMAS POR DIVISION.
0082  C-------------------------------
0083        DO 80 NUM=1,120
0084        WRITE(6,101)
0085        CALL READF(IDCB,IERR,IBUF,4,LEN,NUM)
0086        IFLAG=IBUF(3)
0087        M=M+1
0088        IF(M.NE.1) GO TO 21
0089        WRITE(6,102)I,J
0090        I=I+1
0091        J=J+1
0092  21    IF(IFRAG.NE.0) GO TO 30
0093        WRITE(6,16)K,L
0094        GO TO 79
0095  30    IF(IFRAG.GT.1) GO TO 31
0096        WRITE(6,1)K,L
0097        GO TO 79
0098  31    IF(IFLAG.GT.2) GO TO 32
0099        WRITE(6,2)K,L
0100        GO TO 79
0101  32    IF(IFRAG.GT.3) GO TO 33
0102        WRITE(6,3)K,L
0103        GO TO 79
0104  33    IF(IFLAG.GT.4) GO TO 34
0105        WRITE(6,4)K,L
0106        GO TO 79
0107  34    IF(IFLAG.GT.5) GO TO 35
0108        WRITE(6,5)K,L
```

```
0109          GO TO 79
0110    35    IF(IFLAG.GT.6) GO TO 36
0111          WRITE(6,6)K,L
0112          GO TO 79
0113    36    IF(IFLAG.GT.7) GO TO 37
0114          WRITE(6,7)K,L
0115          GO TO 79
0116    37    IF(IFLAG.GT.8) GO TO 38
0117          WRITE(6,8)K,L
0118          GO TO 79
0119    38    IF(IFLAG.GT.9) GO TO 39
0120          WRITE(6,9)K,L
0121          GO TO 79
0122    39    IF(IFLAG.GT.10) GO TO 40
0123          WRITE(6,10)K,L
0124          GO TO 79
0125    40    IF(IFLAG.GT.11) GO TO 41
0126          WRITE(6,11)K,L
0127          GO TO 79
0128    41    IF(IFLAG.GT.12) GO TO 42
0129          WRITE(6,12)K,L
0130          GO TO 79
0131    42    IF(IFLAG,GT,13) GO TO 43
0132          WRITE(6,13)K,L
0133          GO TO 79
0134    43    IF(IFLAG.GT.14) GO TO 44
0135          WRITE(6,14)K,L
0136          GO TO 79
0137    44    IF(IFLAG.GT.15) GO TO 45
0138          WRITE(6,15)K,L
0139          GO TO 79
0140    C-------------------------
0141    C UNA DIVISION CON MAS DE 15 PROGRAMAS
0142    C-------------------------
0143    45    GO TO 89
0144    79    IF(M.NE.5) GO TO 49
```

```
0145          K=1
0146          L=5
0147          M=0
0148          GO TO 80
0149    49    K=K+5
0150          L=L+5
0151    80    CONTINUE
0152          WRITE(6,103)
0153          WRITE(6,104)
0154          WRITE(6,105)
0155          WRITE(6,106)
0156    C-------------------------
0157    C SE CIERRA EL ARCHIVO ESTADI.
0158    C-------------------------
0159          CALL CLOSE(IDCB)
0160    85    TO TO 1000
0161    89    WRITE(1,200)
0162    C**************************************************
0163    C**************************************************
0164    C***********FORMATOS DE IMPRESION*****************
0165    C**************************************************
0166    C**************************************************
0167    1     FORMAT(10X,I2,"-",I2,5X,"*",5,"-")
0168    2     FORMAT(10X,I2,"-",I2,5X,"*",10,"-")
0169    3     FORMAT(10X,I2,"-",I2,5X,"8",15,"-")
0170    4     FORMAT(10X,I2,"-".I2,5X,"*",20,"-")
0171    5     FORMAT(10X,I2,"-",I2,5X,"*",25,"-")
0172    6     FORMAT(10X,I2,"-",I2,5X,"*",30,"-")
0173    7     FORMAT(10X,I2,"-",I2,5X,"*",35,"-")
0174    8     FORMAT(10X,I2,"-",I2,5X,"*",40,"-")
0175    9     FORMAT(10X,I2,"-",I2,5X,"*",45,"-")
0176   10     FORMAT(10X,I2,"-",I2,5X,"*",59,"-")
0177   11     FORMAT(10X,I2,"-",I2,5X,"*",55,"-")
0178   12     FORMAT(10X,I2,"-",I2,5X,"*",60,"-")
```

```
0179   13  FORMAT(20X,I2,"-",I2,5X,"*",65,"-")
0180   14  FORMAT(10X,I2,"-",I2,5X,"*",70,"-")
0181   15  FORMAT(10X,I2,"-",I2,5X,"*",75,"-")
0182   16  FORMAT(10X,I2,"-",I2,5X,"*")
0183  100  FORMAT(2X,"TIME   SIZE    *")
0184  101  FORMAT(20X,"*")
0185  102  FORMAT(2X,I2,"-",I2)
0186  103  FORMAT(20X,80,"*")
0187  104  FORMAT(25X,I5(")",4X))
0188  105  FORMAT(25X,"1",4X,"2",4X,"3",4X,"4",4X,"5",4X,"6",4X,"7",4X,
0189       +"8",4X,"9",4X,"10",3X,"11",3X,"12",3X,"13",3X,"14",3X,"15")
0190  106  FORMAT(30X,"****NUMERO MAXIMO DE PROGRAMAS DADOS****")
0191   90  FORMAT(2X,"ERROR EN APERTURA DE ARCHIVO ESTADI")
0192  200  FORMAT(2X,"DIVISION CON MAS DE 15 PROGRAMAS")
0193 1000  END
0194       END$
```

```
0001    FIN4
0002    C         091281
0003          PROGRAM SCHED
0004    C---------------------------------
0005    C INICIALIZACION DE VARIABLES
0006    C---------------------------------
0007          DIMENSION IPROG(3)
0008          IPROG(1)=2HPR
0009          IPROG(2)=2HOV
0010          IPROG(3)=2H3
0011          ICODE=12
0012          IRESL=3
0013          MTPLE=10
0014          IOFST=1
0015          CALL EXEC(ICODE,IPROG,IRESL,MTPLE,IOFST)
0016          END
0017          END$
```

APENDICE B

Lineamientos de un sistema operativo de tiempo compartido

Con el objeto de visualizar la asignación simbólica de recursos ante las demandas provenientes de diferentes dispositivos, se simula un sistema operativo de tiempo compartido.

El sistema que se propone es un sistema de 5 pantallas y 3 impresoras. Cada terminal del sistema propuesto, puede requerir un conjunto de programas, dispuestos en una 'tabla menu' que le permita al usuario escoger un trabajo (task), entre las alternativas posibles para cada terminal. El administrador de procesos, el que además de chequear prioridades y procesos en reserva para el dispositivo solicitante, genera los parámetros a enviar al Administrador de recursos. Este último es implementado mediante un controlador de archivos, un controlador de impresoras y un controlador de pantallas.

Se hace énfasis que los procesos implementados en este

apéndice tienen como objetivo único, el dar una idea de lo que podría ser un sistema operativo de tiempo compartido.

Los programas fueron implementados en el computador HP 1000, propiedad de la Universidad del Valle, en lenguaje FORTRAN IV y haciendo uso de subrutinas del sistema, referenciadas por 'CALL EXECS'. (Si se desea mayor información sobre 'CALL EXECS', se sugiere leer el manual del sistema). La simulación se hizo sobre el sistema de hardware existente. En esta parte se incluye únicamente el listado del proceso principal del sistema, el administrador de procesos o trabajos.

Fases desarrolladas y funcionamiento del programa:

1. Opción y control de ingreso de algún comando de cualquiera de las cinco pantallas que forman el sistema.

Esto lo efectúa mediante un proceso repetitivo según sea el número máximo de pantallas y de trabajos, con el objeto de proporcionar alguna respuesta a cualquiera de ellos. Para esto utiliza los comandos 'CALL EXECS' en modo 17 ('READ'), lo que provoca la lectura de algún comando ingresado. Esta lectura se implementa en modo de 'no espera' ('NO WAIT'), lo que implica que el programa se seguirá ejecutando, aún si no se ingresa ningún comando desde alguna terminal o si no se de-

tecta ningún requerimiento de algún trabajo.

El resultado de esto es una rotación entre las pantallas o fuentes de requerimiento de trabajos, para poder aceptar un comando en cualquier momento.

En esta fase se pueden aceptar varios tipos de requerimientos:

a. Requerimiento de una tabla de menú ('MENU TABLE') propia para cada pantalla. Las tablas de menú proporcionan la lista de las actividades o procesos permitidos para cada terminal en particular.

b. Requerimiento de un programa determinado. Esto lo realizará el operador, digitando cualquiera de las opciones que se le presentan.

El comando es aceptado por el sistema, mediante un comando 'CALL EXEC' en modo 21 ('GET') con modalidad de espera ('WITH WAIT'). Esto permite que el programa pase a un estado pasivo y permanezca en él hasta que algún comando específico lo active.

2. Verificación de la legalidad y determinación del tipo de comando ingresado. El operador ingresa el comando que necesi-

ta según los siguientes formatos:

a. MENUX.....Este comando permite la visualización del menú correspondiente a la pantalla X.

b. XY.....Se podrá requerir un programa Y del sistema X.

Si el comando ingresado es el requerimiento de un programa, el programa procede a legalizar el comando a la subrutina llamada PROVE. Para el funcionamiento de esta subrutina, se utilizan 5 archivos internos que contienen las 5 posibles tablas menú, una para cada pantalla que se considera en el sistema propuesto.

Para localizar estos archivos se utiliza la técnica de índices, asignando al archivo el número de la pantalla para la cual están destinados. De esta forma, la subrutina PROVE buscará el archivo indexado, con el número de la terminal que ha respondido (LUTEM). Al encontrar el archivo correspondiente hace una lectura secuencial, hasta encontrar el comando, busca el nombre del programa, imprime el mensaje mediante un 'CALL EXEC' a la pantalla y regresa sin condición de error:

RETRN = 0

Si en la búsqueda secuencial del archivo no encuentra el

comando ingresado, regresa con dondición de error (RETRN = 5) e imprime 'ERROR EN INPUT' en la pantalla.

3. Interpretación del comando ingresado.

Por medio de la subrutina SPRIN se busca, abre e imprime el archivo que se requiere. Si es un menú específico, el operador habrá ingresado MENUX y se abrirá el archivo MENU(X), con su posterior lectura e impresión en la pantalla que lo necesita.

Si por el contrario, se requiere un programa, se asigna el programa solicitado para ejecución. Esto se realiza después de la verificación de validez del comando.

4. Control de mensajes y archivos a imprimir en las pantallas o impresoras del sistema propuesto.

El sistema controla el envío de mensajes y comunicación entre pantallas mediante el uso de CALL EXECS. La subrutina SPRIN se encarga de localizar, leer y trasladar la información a imprimir. Si el texto se debe imprimir en una de las impresoras disponibles, se transfiere la longitud y localización del archivo de impresión, así como la identificación de la impresora, al programa PPRIN. Este último es una simulación de un canal multiplexor para los dispostitivos de sali-

da. La restauración de parámetros en los programas auxiliares se realiza mediante 'CALL EXEC,S' modo 10.

Subrutinas que utiliza:

a. SPRIN Subrutina para imprimir un archivo en una pantalla determinada.

b. PROVE Subrutina para verificar legalidad y tipo de un comando ingresado.

Variables principales utilizadas:

MENU Variable que contiene la palabra MENU para poder detectar el comando MENUX de una pantalla en particular.

LONG Longitud del archivo a imprimir por la subrutina SPRIN. También es la longitud máxima de los archivos utilizados.

IBUFR Memoria compensadora para contener el comando ingresado de la pantalla y posteriormente el nombre del programa a asignar.

IBUFL Longitud del IBUFR.

LUTYS Tabla que contiene las identificaciones de cada terminal o números de las pantallas del sistema con que se trabaja.

ERROR Contiene el mensaje fijo de error:

"ERROR EN INPUT"

LUTEM Variable que contiene el número o identificación de la terminal que responde.

ICLAS Número de clase con el que trabaja el sistema.

RETRN Variable de retorno de la subrutina PROVE. Indica condición de error si es igual a 5.

```
&TESG1 T=00004 IS ON CR06015 USING 00013 BLKS R=0000

0001  FTN4
0002  C       041281
0003        PROGRAM CONTR
0004  C*****************************************************************
0005  C*****************************************************************
0006  C**                                                             **
0007  C**                                                             **
0008  C**                                                             **
0009  C**                    PROYECTO          DE 1981                **
0010  C**          SIMULACION DE UN SISTEMA DE TIEMPO COMPARTIDO       **
0011  C**                                                             **
0012  C**                                                             **
0013  C**                                                             **
0014  C**                                                             **
0015  C**    PROYECTO DESARROLLADO POR: LISSETTE GALVEZ P.             **
0016  C**    UNIVERSIDAD DEL VALLE DE GUATEMALA                       **
0017  C**    CARNET NO.:       77030                                  **
0018  C**                                                             **
0019  C**                                                             **
0020  C**                                                             **
0021  C**    NOMBRE DEL PROGRAMA: CONTRL                              **
0022  C**    NOMBRE DEL ARCHIVO:  &CONTR                              **
0023  C**                                                             **
0024  C**                                                             **
0025  C**                                                             **
0026  C**    DESCRIPCION GENERAL DEL PROGRAMA:                        **
0027  C**                                                             **
0028  C**                                                             **
0029  C**    ESTE PROGRAMA ES EL PROGRAM CONTROLLER DE UN SISTEMA DE   **
0030  C**    TIEMPO COMPARTIDO, SIMULADO PARA UNA MEDIANA EMPRESA,    **
0031  C**    SU FUNCION ES EL DE MODERAR EL FUNCIONAMIENTO DEL SISTEMA **
0032  C**    EN LO QUE RESPECTA A LAS PANTALLAS DEL SISTEMA PROVISTO,  **
0033  C**                                                             **
0034  C**    FASES A CONTROLAR:                                       **
0035  C**                        1)OPCION Y CONTROL DE INGRESO DE ALGUN **
0036  C**                          COMANDO DE CUALQUIERA DE LAS 5 PANTALLAS **
0037  C**                          DEL SISTEMA,                       **
0038  C**                        2)VERIFICACION DE LA LEGALIDAD Y DETERMINA- **
0039  C**                          CION DEL TIPO DEL COMANDO INGRESADO, **
0040  C**                        3)INTERPRETACION DEL COMANDO INGRESADO, **
0041  C**                        - DISPLAY DEL MENU CORRESPONDIENTE A LA **
0042  C**                          PANTALLA QUE LO REQUIERE,          **
0043  C**                        - SCHEDUL DEL PROGRAMA O TASK SOLICITADO **
0044  C**                        - DISPLAY DE ALGUN MENSAJE DE ERROR,  **/
0045  C**                        4)CONTROL DE LOS MENSAJES POSIBLES QUE LOS **
0046  C**                          DIFERENTES PROGRAMAS PUEDEN MANDAR A IMPRI- **
0047  C**                          MIR A LAS PANTALLAS,               **
0048  C**                                                             **
0049  C**                                                             **
0050  C**    SUBRUTINAS QUE UTILIZA:                                  **
0051  C**                                                             **
0052  C**            1) SPRIN: SUBRUTINA PARA IMPRIMIR UN ARCHIVO     **
0053  C**                      EN UNA PANTALLA DETERMINADA,           **
0054  C**            2) PROVE: SUBRUTINA PARA VERIFICAR LEGALIDAD     **
0055  C**                      Y TIPO,                                **
0056  C**                                                             **
0057  C**                                                             **
0058  C**    VARIABLES PRINCIPALES UTILIZADAS:                        **
```

```
0059  C**    ================================                          **
0060  C**                                                              **
0061  C**    -MENU, VARIABLE QUE CONTIENE LA PALABRA MENU PARA PODER    **
0062  C**           DETECTAR EL COMANDO MENUX DE UNA PANTALLA.         **
0063  C**    -LONG, LONGITUD DEL ARCHIVO A IMPRIMIR POR LA SUBRUTINA    **
0064  C**           SPRIN.  TAMBIEN ES LA LONGITUD MAXIMA DE LOS       **
0065  C**           ARCHIVOS DE MENU.                                  **
0066  C**    -IBUFR:BUFFER PARA CONTENER EL COMANDO INGRESADO DE LA     **
0067  C**           PANTALLA Y POSTERIORMENTE EL NOMBRE DEL PROGRAMA A  **
0068  C**           SCHEDULAR.                                         **
0069  C**    -IBUFL:LONGITUD DEL IBUFR.                                **
0070  C**    -LUTYS:ARREGLO QUE CONTIENE LAS IDENTIFICACIONES DE CADA   **
0071  C**           TERMINAL O NUMEROS DE LAS PANTALLAS.               **
0072  C**    -ERROR:CONTIENE UN MENSAJE FIJO DE ERROR:                 **
0073  C**              "ERROR EN INPUT"                                **
0074  C**    -LUTEM:VARIABLE QUE CONTIENE EL NUMERO O IDENTIFICACION DE **
0075  C**           LA TERMINAL QUE RESPONDE.                          **
0076  C**    -ICLAS:NUMERO DE CLASE CON EL QUE TRABAJA EL SISTEMA.      **
0077  C**    -RETRN:VARIABLE DE RETORNO DE LA SUBRUTINA "PROVE" INDICA  **
0078  C**           CONDICION DE ERROR SI ES IGUAL A 5.                **
0079  C**                                                              **
0080  C**                                                              **
0081  C**                                                              **
0082  C**                                                              **
0083  C**                                                              **
0084  C**************************************************************
0085  C**************************************************************
0086  C-------------------------------------------
0087  C INICIALIZACION DE VARIABLES
0088  C-------------------------------------------
0089        INTEGER MENU(4),LONG,IFIN(1),RETRN
0090        DIMENSION IBUFR(6),IRANE(3),LUTYS(6),ERROR(16)
0091        DATA LUTYS /23/
0092        DATA MENU /2HME,2HNU/
0093        DATA IFIN /2HOF/
0094        DATA ERROR /2HER,2HRO,2HR ,2HEN,2H I,2HNP,2HUT/
0095        LUTEM=0
0096        IBUFL=-6
0097        ICLAS=0
0098        LERR=-8
0099        RETRN=0
0100        LONG=40
0101  C-------------------------------------------
0102  C PROCESO PARA ADMITIR UN COMANDO DE
0103  C CUALQUIER TERMINAL.
0104  C-------------------------------------------
0105        WRITE (1,31)
0106        I=1
0107  10    CALL EXEC (17,LUTYS(I)+400B,IBUFR,IBUFL,LUTYS(1),0,ICLAS)
0108        ICLAS=ION(ICLAS,000600)
0109  C-------------------------------------------
0110  C GET WITH WAIT DEL ICLAS, EN CUALQUIER
0111  C          TERMINAL O TASK.
0112  C-------------------------------------------
0113        WRITE (1,31)
0114  20    CALL EXEC (21,ICLAS,IBUFR,IBUFL,LUTEM)
0115  C-------------------------------------------
0116  C     ES RESPUESTA DE TASK?
0117  C-------------------------------------------
0118        WRITE (1,21)ICLAS
```

```
0119  21    FORMAT(2X,"EL NUMERO DE CLASE ES: ",I9)
0120        WRITE (1,32)
0121  32    FORMAT (2X,"ANTES DE VER SI ES DE TASK")
0122        IF (LUTEM.GE.35) GO TO 30
0123        IF (LUTEM.GE.35) GO TO 30
0124        WRITE (1,33)
0125  33    FORMAT(2X,'AHORA VERA SI ES MENU")
0126  C-----------------------------------------
0127  C     ES REQUERIMIENTO DE MENU?
0128  C-----------------------------------------
0129        IF (IBUFR(1).EQ.IFIN+1) GO TO 60
0130        IF (IBUFR(1).EQ.MENU(1)) GO TO 30
0131        WRITE (1,34)
0132  34    FORMAT(2X,'NO ES MENU")
0133  C-----------------------------------------
0134  C LLAMADA A LA SUBRUTINA DE VERIFICACION
0135  C     DE LEGALIDAD Y TIPO.
0136  C-----------------------------------------
0137        CALL PROVE (IBUFR,LUTEM,RETRN)
0138  C-----------------------------------------
0139  C  SI ESTA CORRECTO REGRESA EN:
0140  C  RETRN=0
0141  C  IBUFR.........NOMBRE DEL PROGRAMA
0142  C                  A SCHEDULAR.
0143  C  SI ESTA INCORRECTO REGRESA EN:
0144  C  RETR=5
0145  C  IBUFR.........NOTHING
0146  C-----------------------------------------
0147        WRITE(1,100)
0148  100   FORMAT(2X,"REGRESO DE LA SUBRUTINA PROVE")
0149        WRITE(1,101)RETRN
0150  101   FORMAT (2X,"RETRN= ",I4)
0151        IF (RETRN.EQ.5) GO TO 40
0152        WRITE(1,102)
0153  102   FORMAT(2X,"ANTES DE SCHEDULAR PROGRAMA PEDRO")
0154        CALL EXEC (10,IBUFR,LUTEM,ICLAS)
0155        GO TO 50
0156  C-----------------------------------------
0157  C  SUBRUTINA PARA IMPRIMIR EL ARCHIVO
0158  C  CORRESPONDIENTE EN PANTALLA CORRES-
0159  C  PONDIENTE.
0160  C-----------------------------------------
0161  30    CALL SPRIN (IBUFR,LUTEM,LONG,ICLAS)
0162        GO TO 50
0163  40    CALL EXEC (18,LUTEM,ERROR,LERR,LUTEM,0,ICLAS)
0164  C-----------------------------------------
0165  C DA DIRD CLASS READ EN LA TERMINAL
0166  C-----------------------------------------
0167  50    CALL EXEC (17,LUTEM+400B,IBUFR,IBUFL,LUTEM,0,ICLAS)
0168        GO TO 20
0169  60    WRITE (1,37)
0170  37    FORMAT (2X,"FIN DEL PROGRAMA")
0171        ICLAS=IAND(ICLAS,77777B)
0172        CALL EXEC(2,,ICLAS)
0173  31    FORMAT (2X,"INICIO DEL SISTEMA")
0174        END
0175  C*****************************************************************
0176  C*****************************************************************
0177  C                    SUBRUTINA PARA IMPRIMIR
0178  C               UN ARCHIVO EN UNA PANTALLA DETERMINADA
```

0760

```
0177  C*****************************************************************
0178  C*****************************************************************
0181        SUBROUTINE SPRIN (IBUFR,LUTEM,LONG,ICLAS)
0182  C-------------------------------------
0183  C  DEFINICION E INICIALIZACION DE
0184  C           VARIABLES.
0185  C-------------------------------------
0186        INTEGER IDCB(144),LONG
0187        DIMENSION IBUFO(75),IBUFR(6)
0188  C-------------------------------------
0189  C  LLAMADA PARA ABRIR EL ARCHIVO
0190  C-------------------------------------
0191        CALL OPEN (IDCB,IERR,IBUFR)
0192        IF (IERR.LT.0) GO TO 20
0193  C-------------------------------------
0194  C LECTURA E IMPRESION DEL ARCHIVO.
0195  C-------------------------------------
0196        WRITE (1,9)LONG
0197  9     FORMAT (2X,"LONGITUD DEL ARCHIVO A IMPRIMIR: ",I2)
0198        DO 10 I=1,LONG
0199        DO 12 J=1,75
0200  12    IBUFO(J)=2H
0201        CALL READF (IDCB,IERR,IBUFO,75)
0202        IF (IERR.LT.0) GO TO 20
0203        CALL EXEC (18,LUTEM,IBUFO,75,LUTEM,0,ICLAS)
0204        CALL EXEC (2,ICLAS,IBUFO,0)
0205  10    CONTINUE
0206        GO TO 25
0207  20    WRITE (LUTEM,30)IERR,(IBUFR(I),I=1,6)
0208  25    CALL CLOSE (IDCB)
0209  30    FORMAT (2X," ERROR : ",I6,"EN ARCHIVO : ",6A2)
0210        RETURN
0211        END
0212  C*****************************************************************
0213  C*****************************************************************
0214  C              SUBRUTINA PARA VERIFICAR LEGALIDAD Y TIPO
0215  C*****************************************************************
0216  C*****************************************************************
0217        SUBROUTINE PROVE (IBUFR,LUTEM,RETRN)
0218  C-------------------------------------
0219  C DEFINICION E INICIALIZACION DE VARIABLES
0220  C-------------------------------------
0221        INTEGER IDCB(144),IFIN(2),LUTEM
0222        DIMENSION IRECD(6),ICOD(2),ITASK(6),MENS(7),IBUFR(6),LUTIM(3)
0223        EQUIVALENCE (IRECD(1),ICOD(1)),(IRECD(2),ITASK(1)),(IRECD(5),MENS(1))
0224        DATA IFIN /2H0 /
0225        LUTIM(2)=2H
0226        LUTIM(3)=2H
0227        LUTIM(1)=2H&0
0228        LMEN=-7
0229        IERR=0
0230        WRITE(1,50)LUTIM(1)
0231  C-------------------------------------
0232  C LLAMADA PARA ABRIR EL ARCHIVO(LUTEM)
0233  C        ******TABLA DE MENU******
0234  C-------------------------------------
0235        LUTIM(1)=LUTIM(1)+(LUTEM-21)
0236        WRITE(1,50)LUTIM(1)
0237        WRITE(1,50)LUTIM(2)
0238        WRITE(1,50)LUTIM(3)
```

```
0239  50    FORMAT(2X,"LUTIM LECTURA DEL ARCHIVO:" A2)
0240  53    FORMAT(2X,"LUTIM ES ASCII",I5)
0241        WRITE(1,53)LUTIM(1)
0242        WRITE(1,51)IERR
0243  51    FORMAT(2X,"ERROR DE REGRESO DE LECTURA:",I5)
0244        CALL OPEN (IDCB,IERR,LUTIM)
0245        WRITE(1,51)IERR
0246        IF (IERR.LT.0) GO TO 40
0247  C-----------------------------------------
0248  C  LECTURA Y BUSQUEDA DEL COMANDO
0249  C    IGUAL EN EL ARCHIVO LUTEM.
0250  C-----------------------------------------
0251        WRITE(1,54)
0252  54    FORMAT(2X,"PROCEDE A LEER EL ARCHIVO $1 SIN ERROR")
0253        DO 10 I=1,60
0254        CALL READF (IDCB,IERR,IRECD,8)
0255        WRITE(1,51)IERR
0256        IF (IERR.LT.0) GO TO 40
0257        IF (IRECD(1).EQ.IFIN(1)) GO TO 40
0258        IF (IRECD(1).EQ.IBUFR(1)) GO TO 20
0259  10    CONTINUE
0260  20    DO 30 J=1,3
0261  30    IBUFR(J)=ITAEN(J)
0262        WRITE(1,55)
0263  55    FORMAT(2X,"ANTES DE IMPRIMIR RUNNING Y RETURN")
0264        WRITE(1,56)(IBUFR(J),J=1,3)
0265  56    FORMAT(2X,"NOMBRE DEL PROGRAMA A CORRER",A2)
0266        CALL EXEC (10,LUTEM,NEAS,LTEM,0,0,ICLAS)
0267        RETURN
0268  C-----------------------------------------
0269  C RETORNO CON ERROR
0270  C-----------------------------------------
0271  40    RETRN=5
0272        RETURN
0273        END
0274        END$
```