CAPITULO 14

ADMINISTRACION DE PERIFERICOS

14.1. - **FUNCIONES**

Los objetivos principales de una administración de periféricos son:
1) Llevar el estado de los dispositivos, lo cual requiere de mecanismos especiales. Estos mecanismos requieren de Bloques de Control de Unidades (UCB) asociados a cada dispositivo.
2) Determinar políticas para la asignación y desasignación de dispositivos. Esto es ver quién obtiene el dispositivo, por cuánto tiempo y cuándo. Por ejemplo una política de mucho uso de dispositivo intenta coordinar los pedidos de los procesos con la velocidad de los dispositivos de E/S.

14.2. - **Tipos de Periféricos**

En función de su asignación los periféricos pueden ser clasificados en:
i) Dedicados: Un dispositivo asignado a un sólo proceso.
ii) Compartidos: Un dispositivo compartido por varios procesos.
iii) Virtuales: Un dispositivo físico (generalmente de tipo dedicado) es simulado sobre otro dispositivo físico (de tipo compartido).
Recordemos que una vía de comunicación la podemos esquematizar como se ve en la Fig. 14.1.

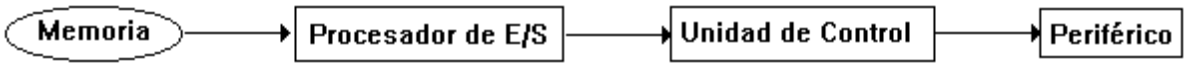


Fig. 14.1.

Un procesador de E/S controla la operación de E/S. Una unidad de control básicamente detecta errores de paridad en el envío de información y a veces es capaz de corregir esa información si tuviera una inteligencia suficiente. Además tiene buffers de sincronismo con los cuales compensa diferencias de velocidad entre los periféricos y el procesador.

En algunos sistemas se mantiene la estructura de la Fig. 14.2 que se suele denominar *String de Periféricos*. La cantidad de transferencias del string a memoria es una sola.

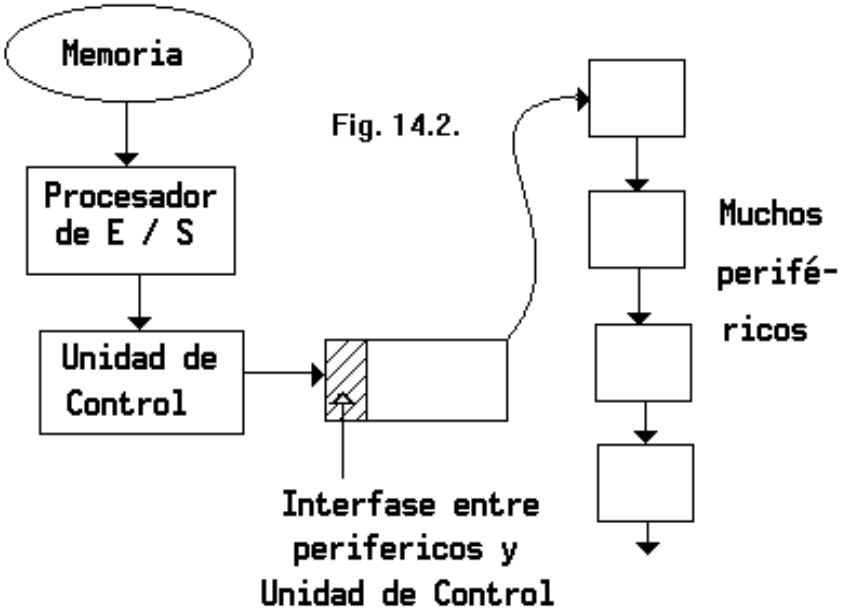


Fig. 14.2.

En todos los casos lo que se busca es:

a) confiabilidad en la transferencia y
b) velocidad.

Para ello se recurre a conectar los periféricos de distintas maneras como ser la que se ve en la Fig. 14.3.

Si deja de funcionar algún procesador de E/S todavía podremos acceder al periférico pasando por el otro. Esto se denomina comúnmente tener *Caminos Alternativos* para llegar a los periféricos.

Asimismo también es deseable proveer la posibilidad de transferencias simultáneas si el camino lo permite (Fig. 14.4).

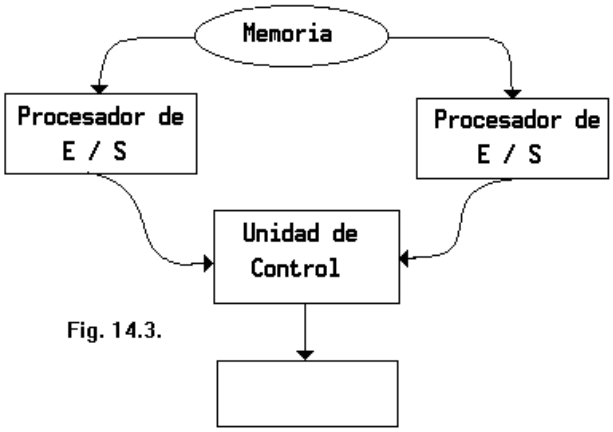


Fig. 14.3.

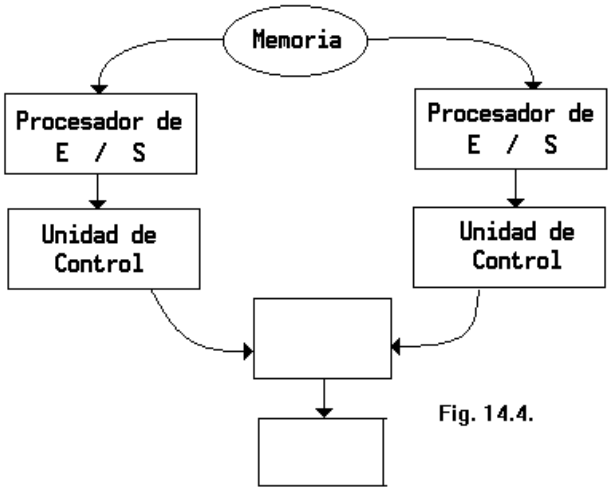


Fig. 14.4.

14.3. - **Canales y Unidades de Control**

Recordemos que un CANAL es un procesador especializado en operaciones de E/S. Las instrucciones que manejan son comandos de canal y sirven para dar órdenes al periférico y controlan la transferencia de la información. Al ser un procesador tiene su palabra de control, la cual puede esquematizarse como sigue (Fig. 14.5):

PC	Dirección	Cuenta residual (1)	Estado (2)

próxima instrucción del programa de canal

Fig. 14.5

(1) Indica cuántos bytes del último comando no han sido procesados. Usualmente su valor es 0 salvo una terminación de E/S anormal. En el comienzo contiene la longitud del buffer. A medida que se transfieren los bytes a memoria se decrementa. Cuando llega a 0 se produce una interrupción por fin de E/S.
(2) Indica si la operación se completó o si tuvo error.

Bajo un esquema de Procesador de E/S, Unidad de Control y Periférico, un periférico puede ser identificado por medio del camino a seguir por la información hasta llegar a él, por ejemplo Procesador de E/S 1, Unidad de Control 5, Periférico 3; luego ese periférico se identifica como "153".

Cuando un programa emite una instrucción de arranque (SIO 153 Start Input Output o IES de Inicialización de E/S) con Canal 1 - Unidad de Control 5 - Periférico 3, éste accede a una predeterminada dirección de memoria que tiene la dirección de la primera instrucción del programa del Procesador de E/S. El Procesador de E/S procesa estos comandos que tienen el formato de la Fig. 14.6.

Codop	Dirección del Buffer	Longitud (bytes)	Señales

Fig. 14.6.

codop: indica si son operaciones de transferencia, acciones al periférico sin transferencia, transferencia de control dentro del programa del procesador de E/S.
direcc.buffer: indica dónde se colocará o de dónde sacarán los datos de memoria.
long.bytes: indica la longitud.
señales: indican por ejemplo, si se concatenan las instrucciones del procesador de E/S, etc.

Con el campo de señales se logra la solución al problema del buffer a caballo entre 2 páginas (en administración de memoria paginada). Si se tiene un buffer cortado entre dos páginas, no se puede hacer otra cosa más que ejecutar dos instrucciones de procesador de E/S, una de las cuales tendrá la dirección de memoria inicial y la longitud hasta terminar la página, y concatenarla con otra instrucción que tendrá la nueva dirección y la longitud restante. Es decir el programa de transferencia contendrá instrucciones del siguiente tipo:

1ero) long.buffer = L1 + L2
2do) transferencia D1, L1
3ro) transferencia D2, L2
4to) señal fin

La interrupción de E/S se producirá al finalizar la ejecución de la segunda transferencia.

Cuando un procesador de E/S interrumpe al procesador debe informar si la operación de E/S finalizó correctamente, si hubo algún tipo de incidentes, etc.

Ejemplos de programas para Procesador de E/S son:

Para DISCO
Posicionamiento (cilindro, cabeza)

Búsqueda del bloque (sector)
Reintento
Transferencia (Dir, Long)
Para CINTA
Rebobinar
Saltear registro 1
Saltear registro 2
Escribir registro 3 de buffer 1
Escribir registro 4 de buffer 2
Escribir marca de cinta
Rebobinar y descargar

Dependiendo de la inteligencia de los elementos que componen la vía de comunicación (Unidad de control y periféricos), éstos pueden enviar órdenes y desconectarse (posicionamiento, rebobinar) aprovechando así tiempos "muertos".

Por otra parte las Unidades de Control tienen las siguientes funciones:

* Sincronizar la información (para ello cuenta con buffers de sincronismo para compensar diferencias de velocidad);
* En algunos casos serializa y des-serializa la información;
* Chequeo de paridad y detección de errores;
* Controlar movimiento del periférico (rebobinar, posicionamiento del brazo, etc.).

14.3.1. - **Tipos de Canales**

Selector: Desde que comienza a ejecutar un programa de canal para un periférico permanece conectado a él, aún cuando las operaciones no impliquen transferencia de información (Fig. 14.7).

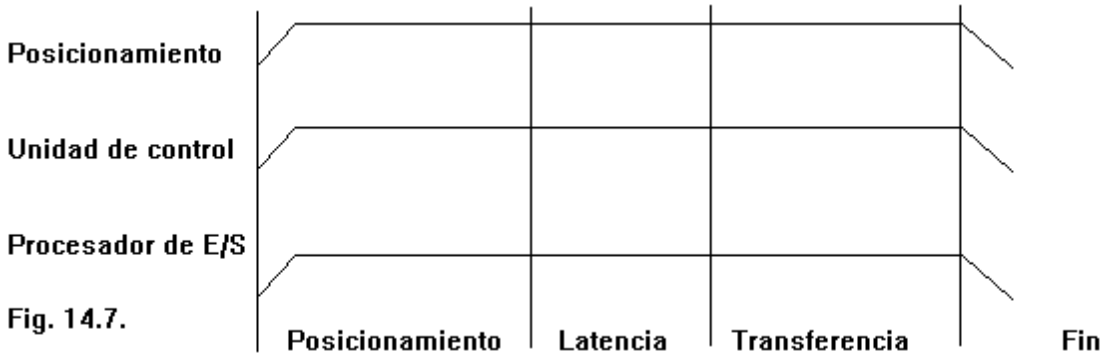


Fig. 14.7.

Solo sirve a un dispositivo a la vez. Normalmente se usa con dispositivos rápidos (cintas, discos y tambores), por lo tanto el pedido de E/S se completa rápido y se selecciona otro dispositivo.

Multiplexor: Se comparte entre varios programas de procesadores de E/S, o sea cuando da una orden de movimiento a un periférico, hasta que éste la complete, puede desconectarse del mismo y emitir otro comando del procesador de E/S para otro periférico. Debe estar conectado con el periférico siempre que exista transferencia de información. (Si el periférico es inteligente la unidad de control también puede desconectarse y atender a otro periférico).

Cuando el periférico está posicionado emite una señal con lo que vuelve a activar a la unidad de control y al procesador de E/S. Mientras dura la transferencia de información permanecen los tres activos (Fig. 14.8).

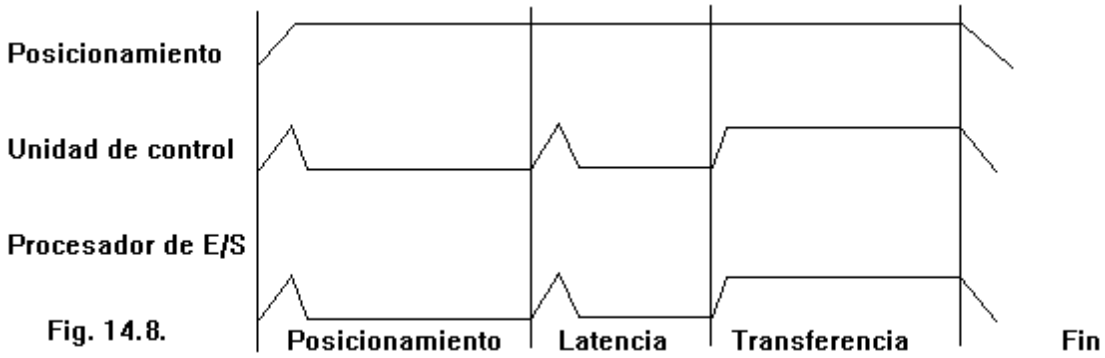


Fig. 14.8.

Puede llegar a ocurrir una desincronización: un periférico encuentra el sector, emite la señal pero el canal está ocupado con otro pedido. Entonces se pierde el sector y es necesario reintentar la búsqueda del mismo.

14.4. - **Tipos de Periféricos**

Existen distintas clasificaciones para periféricos pero lo que nos interesa es determinar si ese periférico debe ser de uso exclusivo para un proceso dadas sus características.

Por ejemplo una cinta no tiene sentido que se asigne a más de un proceso. Ese periférico va a estar dedicado solamente a uno.

Aquellos que pueden ser compartidos a lo largo de un lapso de tiempo, los llamaremos compartidos. El caso típico es el disco o el tambor magnético, que permiten acceso al azar. Si bien el periférico va a estar dedicado durante la transferencia no es necesario que lo esté a lo largo de todo el proceso.

La última clase que existe es la de periféricos virtuales: son periféricos que se los simula de alguna manera. Si tenemos periféricos dedicados, nos obligaría a ejecutar un solo proceso por vez que use esos periféricos. Si podemos simularlos, podríamos ejecutar más de un proceso de ese tipo simultáneamente. El ejemplo típico es el SPOOLing (Simultaneous Peripheral Operation On-line). En el apartado 14.11 trataremos más extensamente esta técnica.

14.5. - **Técnicas para la administración y asignación de periféricos**

Dependiendo del tipo de periférico que se trate existen diferentes técnicas de asignación del mismo. A saber:

1) dedicados: se asigna a un trabajo por el tiempo que dure el trabajo. Ej: lectora de tarjetas, cinta, impresora.

2) compartidos: pueden ser compartidos por varios procesos. La administración puede ser complicada. Se necesita una política para establecer que pedido se satisface primero basándose en :

   a) lista de prioridades,

   b) conseguir una mejor utilización de los recursos del sistema;

3) virtuales: Algunos dispositivos que deben ser dedicados (impresoras) pueden convertirse en compartidos a través de una técnica como el SPOOLing. Se hace una simulación de dispositivos dedicados en dispositivos compartidos. Por ejemplo un programa de SPOOLing puede imprimir todas las líneas de salida a un disco. Cuando un proceso escriba una línea, el programa de SPOOL intercepta el pedido y lo convierte a una escritura en disco. Dado que un disco es compartible, convertimos una impresora en varias impresoras "virtuales". En general se aplica esta técnica a periféricos dedicados lentos.

14.6. - **Políticas de asignación para periféricos Dedicados**

Para los periféricos de tipo dedicados se puede realizar su asignación al proceso en tres momentos diferentes :

* al comienzo del trabajo
* al comienzo de la etapa
* al realizarse la instrucción Open

Asignando al comienzo del trabajo estaremos seguros que cada vez que un proceso necesite el periférico, lo va a encontrar disponible. Pero si lo que tenemos es una larga secuencia de programas y sólo el programa que ejecuta en la última etapa es el que hace uso de ese periférico dedicado, hemos desperdiciado el periférico durante la ejecución de los programas anteriores.

Asignando al comienzo de la etapa intentamos entonces evitar la ociosidad de los periféricos y asignarlos sólo en el momento previo de las necesidades del programa. Puede ocurrir que en el momento de necesitar ejecutar esa etapa el periférico no se encuentra disponible y debamos tomar una decisión entre esperar a que el periférico se libere o cancelar el trabajo. A pesar de todo, todavía puede pasar un tiempo hasta que realmente se haga uso del dispositivo en esa etapa.

Asignarlo en el momento de la instrucción Open significa que ahora hacemos la asignación en el momento de querer usar realmente el dispositivo. Puede ocurrir, en este caso también, que no se encuentre disponible y habrá que tomar la decisión de esperar o cancelar. La desasignación generalmente se realiza en el momento de ejecutar un Close.

14.6.1. - **Asignación parcial y total de periféricos Dedicados**

Podemos hacer una asignación total o parcial del periférico.

En la asignación total el SO no entregará la totalidad de los periféricos dedicados que se necesitan hasta no tenerlos a todos disponibles. Si un proceso necesita 3 cintas, hasta que no haya 3 cintas libres no se le asignan, ni a nivel de etapa ni a nivel de trabajo.

En la asignación parcial a medida que el SO encuentra periféricos libres, los va asignando a los procesos, aunque no tenga la totalidad de los dispositivos que ese proceso requiere, esta política de asignación es la que se sigue cuando se asigna en el momento del Open.

Puede ocurrir en este caso un abrazo mortal (deadlock); ej. El proceso P1 pide 3 cintas, el proceso P2 pide 3 cintas y el sistema cuenta sólo con 4 cintas. Le asigna 2 a P1 y 2 a P2. P1 tiene 2 cintas y está esperando una más. Lo mismo ocurre con P2. Ninguno va a obtener el recurso que le falta pues lo tiene el otro. El caso de asignación parcial se debe controlar muy bien con el fin de evitar abrazos mortales, o evitar también el tener que matar algún proceso involucrado en un deadlock.

14.7. - **Políticas de asignación para periféricos Compartidos**

En este caso la asignación se hace en el momento de usarlos, es decir, en el momento de lanzarse la operación de E/S. Se arman colas de espera de disponibilidad de periféricos, unidades de control y procesadores de E/S. Para ello es necesario el uso de tablas. Estas tablas se relacionan formando una estructura de árbol.

14.8. - **Base de Datos para Administración de periféricos**

La base de datos para manejar una administración de periféricos consta de tres tablas cuyo contenido se explicita a continuación (Fig. 14.9, 14.10 y 14.11).

En el **Bloque de Control de Periférico** tendremos la siguiente información:

Identificador del dispositivo
Estado del dispositivo
Lista de unidades de control a las que está conectado el dispositivo
Lista de procesos esperando este dispositivo

En el **Bloque de Control del Procesador de E/S** tendremos la siguiente información:

Identificador del canal
Estado del canal
Lista de unidades de control a las que está conectado el canal
Lista de procesos esperando el canal

En el **Bloque de Control de la Unidad de Control** tendremos la siguiente información:

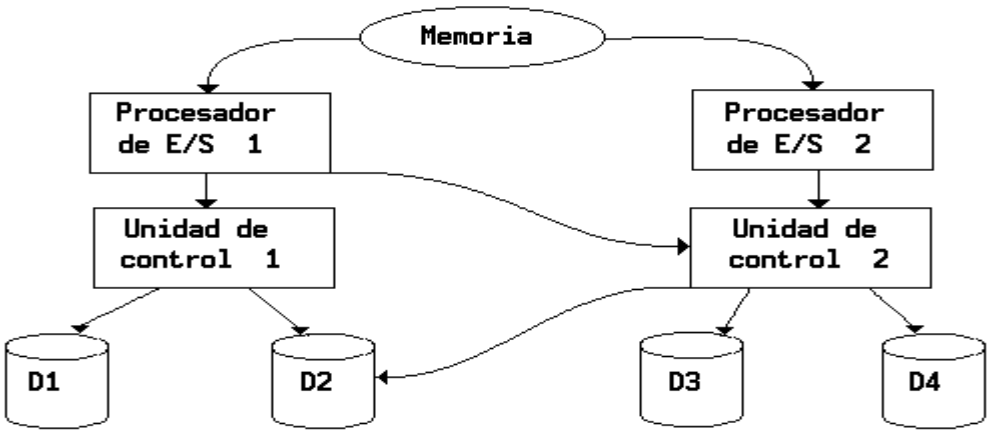
Identificador de la unidad de control
Estado de la unidad de control
Lista de dispositivos conectados a esta unidad de control
Lista de canales conectados a esta unidad de control
Lista de procesos esperando a esta unidad de control

14.9. - **Rutinas del Sistema**

Para estos dispositivos la asignación es dinámica, y participarán en este proceso los siguientes módulos:

Controlador de tráfico de E/S (Fig. 14.12)

- Lleva el estado de los dispositivos, unidades de control y procesadores de E/S por medio de bloques de



Distintos caminos para acceder al disco D2

Procesador de E/S 1 - U.C. 1 - D2

Procesador de E/S 2 - U.C. 2 - D2

Procesador de E/S 1 - U.C. 2 - D2

Fig 14.12.

control.

- Determina la posibilidad de una operación de E/S y la posibilidad de caminos alternativos

Cada bloque de control que maneja el controlador de tráfico de E/S servirá para poder determinar si es posible la operación y por qué camino. Cuando queremos hacer una operación de E/S, es necesario hallar toda la vía de comunicación. El proceso se encolará para que se le asigne un periférico, luego para que se le asigne una unidad de control y luego para que se le asigne un procesador de E/S. Sólo una vez que el proceso está a la cabeza de cada una de las tres colas, éste puede realizar su operación de E/S.

Planificador de E/S

- Decide qué proceso toma el recurso. Determina el orden en que se asignarán a los procesos los dispositivos, unidades de control y procesadores de E/S, ya sea por algoritmos o por alguna prioridad.

Una vez que hay un orden, es el controlador de tráfico de E/S el que determina que puede ser satisfecho el pedido. Las políticas a considerar son las mismas que usamos para procesos, salvo que una vez que un programa de procesador de E/S se lanza no se interrumpe sino hasta que termina.

Manipulador de periféricos

Existe uno por cada tipo de periférico que haya en el sistema.

- Arma el programa del procesador de E/S
- Emite la instrucción de arranque de E/S
- Procesa las interrupciones del dispositivo
- Maneja errores del dispositivo
- Realiza una planificación de acuerdo al periférico (estrategia de acceso).

14.9.1. – **Como interactúan las rutinas?**

Las rutinas mencionadas interactúan entre si para obtener que la E/S solicitada por un proceso se complete. Para ello es primero necesario obtener la ruta por la cual se realizará la transferencia de los datos desde o hacia el periférico hacia o desde la memoria central.

Las rutas se adquieren siempre comenzando desde el periférico y luego ascendiendo por la Unidad de Control y finalmente el canal. Esto obedece lógicamente a que la estructura jerárquica esta más poblada en las hojas (periféricos) que en su raíz (canal).

Una vez que la administración de la información ha determinado en qué lugar exacto del periférico se debe realizar la E/S, cuál es el periférico concretamente sobre el que se desea operar, le pasa esta información a la administración de periféricos.

La primera rutina que interviene es el manipulador de periféricos el cual con los datos que recibe construye el programa de canal ya que conoce cuál es el periférico concreto, de qué tipo de periférico se trata y sabe si la operación es de lectura o de grabación. Lo único que no conoce en este momento es qué ruta se utilizará para realizar la transferencia y, deja esta información aun sin completar.

Llama a continuación al Controlador de Tráfico de E/S para indicarle que este pedido va a esperar por el periférico en cuestión.

El Controlador de Tráfico de E/S coloca al pedido en el Bloque de Control del Periférico en la lista de procesos en espera del periférico.

A medida que el tiempo pasa y el periférico transfiere información de alguna E/S solicitada previamente (suponemos que el periférico esta ocupado) esta lista de procesos en espera del periférico se va poblando. Pero la lista no es de una capacidad infinita y el Controlador de Tráfico de E/S sabe cual es la cantidad de procesos que puede colocar en espera ya que es quién administra estas listas.

Una vez alcanzada una cota máxima de procesos en espera el Controlador de Tráfico invoca al Planificador de Procesos de E/S para que aplique su algoritmo y decida a cuál o cuáles de los procesos en espera se otorgará el uso del periférico cuando este se libere. Pueden llegar a ser más de un proceso ya que por características físicas del dispositivo puede convenir realizar dos o más E/S seguidas por cuestiones de optimización del uso del periférico.

Una vez obtenidos el/los ganadores de la selección hecha por el Planificador de Procesos de E/S éste invoca nuevamente al Controlador de Tráficos de E/S pidiéndole que ahora encole a estos procesos ganadores en la cola de procesos en espera de una Unidad de Control.

Hemos subido un escalón en la jerarquía de interconexión y ahora pasamos a competir para adquirir la Unidad de Control.

Nuevamente se repite el proceso mencionado anteriormente: se alcanza una cota, se invoca al Planificador de Procesos de E/S, se selecciona uno o más ganadores (aquí de la Unidad de Control) y se encola a los ganadores (por medio del Controlador de Tráfico de E/S) en la lista de procesos en espera por el Canal.

Una vez encolados en el Canal nuevamente repetimos el ciclo (cota, selección) y aquí ya hemos adquirido el escalón más alto en nuestra jerarquía de transferencia: el Canal y por ende toda la ruta Canal - Unidad de Control – Periférico.

Como ya obtuvimos la ruta entonces podemos invocar al Manipulador de Periféricos a efectos de que complete su programa de canal con la ruta obtenida. Se realiza finalmente (una vez que se libere el periférico) una última verificación de que la ruta está libre pero ahora en el sentido Canal → Unidad de Control → Periférico y si

estuviera libre entonces se lanza el comienzo de la E/S física, caso contrario habrá que esperar que se libere la porción de ruta que se encuentre ocupada.

Nótese que en ningún momento se optó por una ruta u otra aunque en realidad esta elección fue realizada por el Controlador de Tráfico de E/S ya que al encolar un pedido, su elección de hacerlo en una Unidad de Control u otra o un Canal u otro es la que implícitamente está determinando la ruta concreta que se utilizará finalmente para hacer la transferencia.

14.10. - **Algoritmos de planificación de E/S**

Bloqueo: es una técnica en la cual se agrupan muchos registros lógicos en un sólo registro físico. Esta técnica no sólo trae ventajas desde el punto de vista de nuestro programa, que va a estar menos tiempo demorado, sino que cada vez que haga una operación de READ va a tomar información que ya tiene en la memoria. Hay también un mejor aprovechamiento del periférico. Por ejemplo en una cinta entre registro y registro tengo un espacio entre registros desperdiciado (el IRG o Inter Record Gap) (Fig. 14.13).

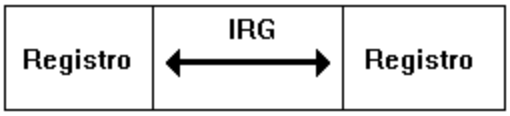


Fig. 14.13.

Este espacio da el tiempo necesario para el arranque y la parada de la cinta. Existe un tiempo para frenar y otro para arrancar y adquirir la velocidad de transferencia que tiene el dispositivo. El IRG depende de la densidad de grabación. Para minimizar el Gap se utiliza la técnica de bloqueo.

Las ventajas de bloqueo son:

1) Menor cantidad de operaciones de E/S dado que cada operación de E/S lee o graba múltiples registros lógicos a la vez.

2) Menor espacio desperdiciado, dado que la longitud del registro físico es mayor que el IRG.

3) Menor cantidad de espacio en cinta cubierto cuando se leen varios registros.

Sus desventajas en cambio son:

1) Overhead de software y rutinas que se necesitan para hacer bloqueo, desbloqueo y mantenimiento de registros.

2) Desperdicio de espacio en buffers.

3) Mayor probabilidad de errores de cinta dado que se leen registros largos.

Buffering: Si tenemos un dispositivo o unidad de control con buffer podemos igualar velocidades con el procesador.

Por ej. Para dispositivos de muy baja velocidad se puede ir realizando una lectura adelantada del próximo registro a efectos de cuando el proceso lo requiera se transfieran varios registros desde el periférico a una mayor velocidad y se pueda liberar de esta forma el Canal de E/S. Se pueden tener también dos buffers y se va leyendo alternadamente de cada uno de ellos. El proceso no se vería detenido nunca, su pasaje a bloqueado en espera de E/S es prácticamente inexistente aunque no nulo (Fig. 14.14).

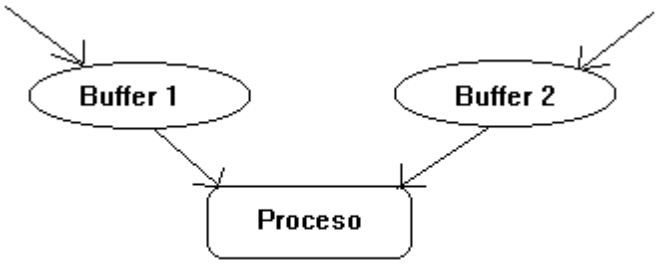


Fig. 14.14.

14.11. - **Algoritmos para encolar pedidos (en disco)**

El manipulador de periféricos puede utilizar técnicas de agrupación de pedidos sobre un mismo dispositivo para disminuir los tiempos mecánicos del mismo, algunas de ellas son :

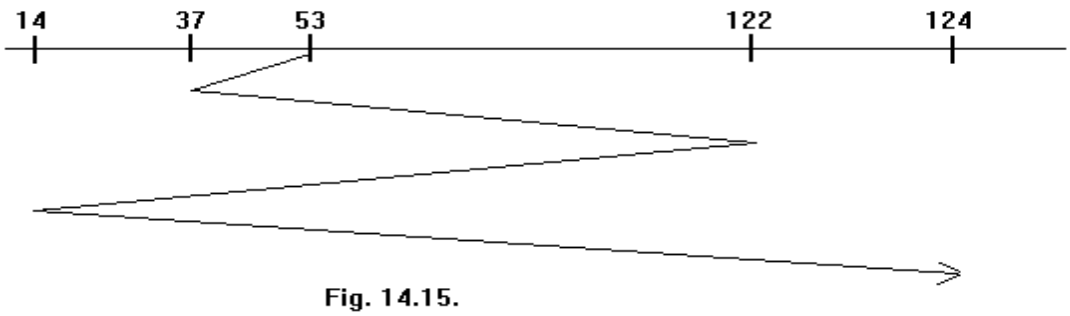


Fig. 14.15.

- **FIFO**: el primero que llega, es el primero en ser servido (FCFS - First Come First Served). Es el más simple. Sin embargo no da el mejor servicio promedio. Consideremos una cola que pide accesos a disco a las siguientes pistas: 37, 122, 14, 124 (estando inicialmente en la 53, Fig. 14.15).

Se recorren muchas pistas, podría haber sido más conveniente servir 122 y 124 juntos y luego la 14 y 37. El recorrido de la cabeza es de 319 pistas en total.

- **Más corto primero**: Se selecciona el pedido con el menor recorrido de búsqueda desde la posición actual de la cabeza, o sea, a la pista que está más cerca. El recorrido es de 149 pistas. Se debe conocer la posición actual de la cabeza (Fig. 14.16).

Fig. 14.16.

Pero esto puede ocasionar inanición (un pedido no es atendido nunca). Por lo tanto no es óptimo, aunque es mejor que el FIFO. Una solución a la inanición es encerrar a los pedidos en grupos.

- **Barrido Ascensor (SCAN)**: Aquí, la cabeza lectora/grabadora comienza en un extremo del disco y se va moviendo hacia el otro, atendiendo los pedidos a medida que llega a cada pista. Cuando llega al otro extremo vuelve. Para ello debemos conocer no sólo la posición de la cabeza sino también la dirección (Fig. 14.17).

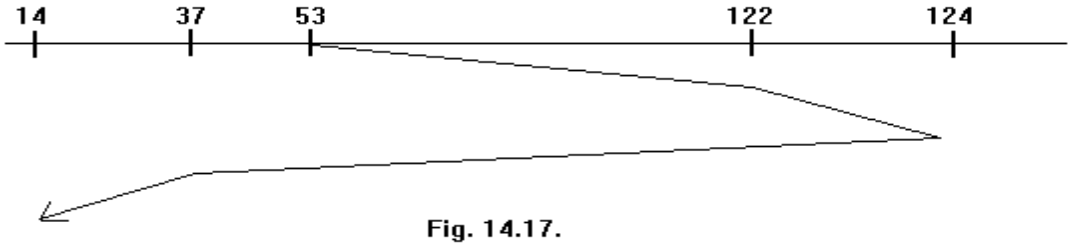


Fig. 14.17.

- **Barrido Circular (C-SCAN, circular SCAN)**: Está diseñado para atender los pedidos con un tiempo de espera uniforme. Mueve la cabeza lectora/grabadora de un extremo al otro del disco. Cuando llega al otro extremo, vuelve inmediatamente al comienzo del disco sin atender ningún pedido. Trata al disco como si fuese circular (Fig. 14.18).

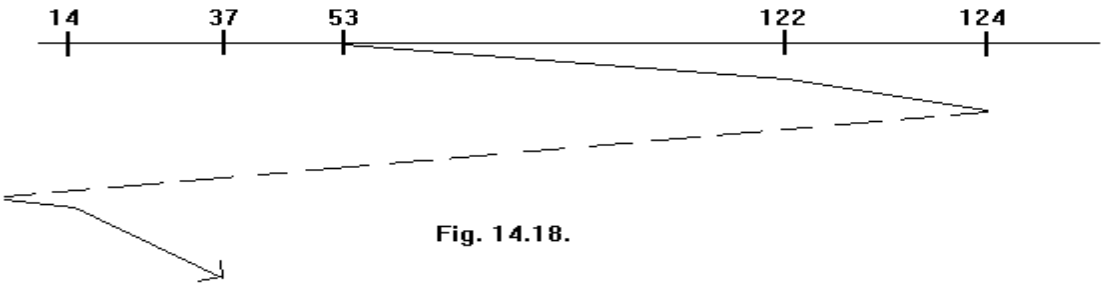


Fig. 14.18.

En la práctica ni SCAN ni C-SCAN mueven la cabeza de un extremo al otro. La cabeza se mueve hasta la posición del último pedido en esa dirección. Cuando no existen más pedidos en esa dirección se cambia la dirección del movimiento de la cabeza.

Estas versiones de SCAN y C-SCAN se llaman LOOK y C-LOOK.

Hay que tener en cuenta la ubicación de los archivos (directorios, archivos muy usados) en un dispositivo compartido. Conviene colocarlos en el medio del disco para reducir el movimiento de la cabeza. Si tengo archivos secuenciales, conviene que estén en pistas correspondientes a un mismo cilindro, de tal manera que no exista tiempo de posicionamiento para acceder a ese archivo.

- **Posición angular (sector queueing)** : Es un algoritmo para planificar dispositivos de cabezas fijas. Se divide cada una de las pistas en un número fijo de bloques, llamados sectores. El pedido ahora especifica pista y sector.

Se atiende un pedido por cada sector que pasa por debajo de la cabeza, aún cuando el pedido no está primero en la cola de espera. Cada sector tiene una cola.

Cuando un pedido para el sector i llega, se coloca en la cola del sector i. Cuando el sector i pase por debajo de la cabeza lectora/grabadora el primer pedido de la cola de ese sector es atendido.

Se puede aplicar también a discos de cabezas móviles, si hay más de un pedido en un mismo cilindro o pista. Cuando la cabeza se mueve a un cilindro en particular, todos los pedidos para ese cilindro pueden ser atendidos sin más movimiento de la cabeza.

14.12. - **DISPOSITIVOS VIRTUALES**

Dispositivos como las lectoras de tarjetas, perforadoras, impresoras, plotters, pantallas gráficas, presentan dos problemas que dificultan su utilización:

- Estos dispositivos trabajan bien cuando existe un flujo continuo y estable de pedidos a una tasa que se aproxime a sus características físicas (ej.: una tasa de impresión de 1000 líneas por minuto, etc.). Si un trabajo trata de generar pedidos más rápidamente que la tasa de performance del dispositivo, el trabajo debe esperar una cantidad significativa de tiempo. Por otro lado, si el trabajo genera pedidos a una tasa inferior, el dispositivo se mantiene ocioso gran parte del tiempo.
- Estos dispositivos se deben dedicar a un único trabajo a la vez.

Resumiendo, necesitamos una tasa de pedidos estable, mientras que los programas generan pedidos altamente irregulares (un programa con una tasa alta de cálculo puede imprimir muy pocas líneas, mientras que un programa con tasa alta de E/S puede imprimir miles de líneas en unos pocos segundos de CPU). La multiprogramación y el buffering reducen estos problemas, pero no lo hacen por completo.

14.12.1. - **Soluciones Históricas**

Estos problemas desaparecerían (o disminuirían substancialmente) si tuviéramos la posibilidad de utilizar algún dispositivo compartible (como ser un disco) para todas las entradas y salidas. Estos dispositivos pueden utilizarse simultáneamente para leer y/o escribir datos por varios trabajos, como mostramos en la Fig. 14.19.

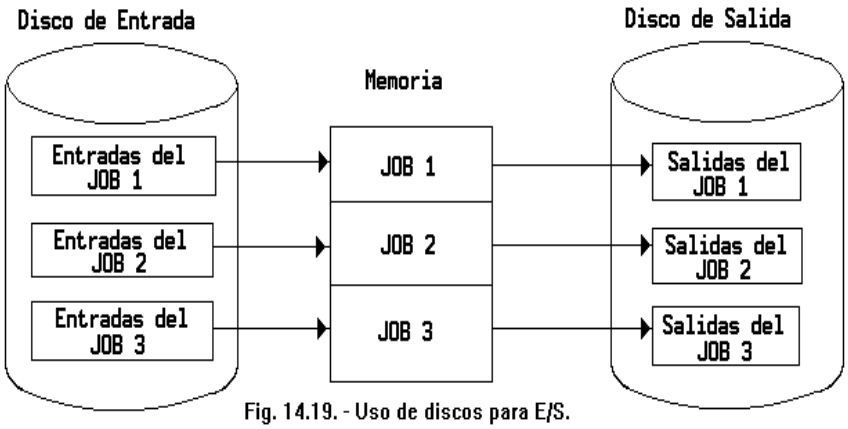


Fig. 14.19. - Uso de discos para E/S.

Además estos dispositivos tienen muy alta performance, especialmente si se hace bloqueo de datos, disminuyendo el tiempo de espera de los trabajos que requieren mucha E/S.

Pero esta especulación es imposible de llevar a la práctica, puesto que la utilización de dispositivos como las impresoras es necesaria para la mayoría de las aplicaciones.

14.12.2. - **Operaciones periféricas fuera de línea**

Una solución a este dilema puede encontrarse en el proceso de tres pasos ilustrado en la Fig. 14.20. En un primer paso, usamos una computadora separada cuya única función es leer tarjetas a máxima velocidad y almacenar la información correspondiente en un disco. Se pueden usar varias lectoras simultáneamente.

Como segundo paso, el disco conteniendo los datos de entrada ingresados en la computadora 1 se mueve a la computadora 2, encargada del procesamiento principal. Este paso es el correspondiente al mostrado anteriormente en la Fig. 14.19.

Puede haber muchos trabajos en ejecución, cada uno leyendo sus datos de entrada del disco; y escribiendo sus resultados en otro disco. Notar que en nuestro ejemplo es posible multiprogramar tres trabajos aunque existen sólo dos lectoras.

Finalmente, como tercer paso, se mueve el disco a una tercera computadora que lee del mismo los datos de salida, imprimiéndolos a alta velocidad e imprimiendo la información en las impresoras.

Este proceso se llama PROCESAMIENTO PERIFERICO FUERA DE LINEA, y las computadoras son conocidas con el nombre de COMPUTADORES PERIFERICOS, porque no hacen cálculo alguno sino que simplemente transfieren información de un dispositivo periférico a otro.

Podemos hacer varias observaciones a este esquema.

- Como las computadoras periféricas tienen tareas simples, pueden ser computadoras sencillas, lentas y baratas.
- Originalmente se utilizaban cintas en lugar de discos móviles.
- Aunque se solucionan los problemas planteados anteriormente, se introducen nuevos problemas, relaciona-

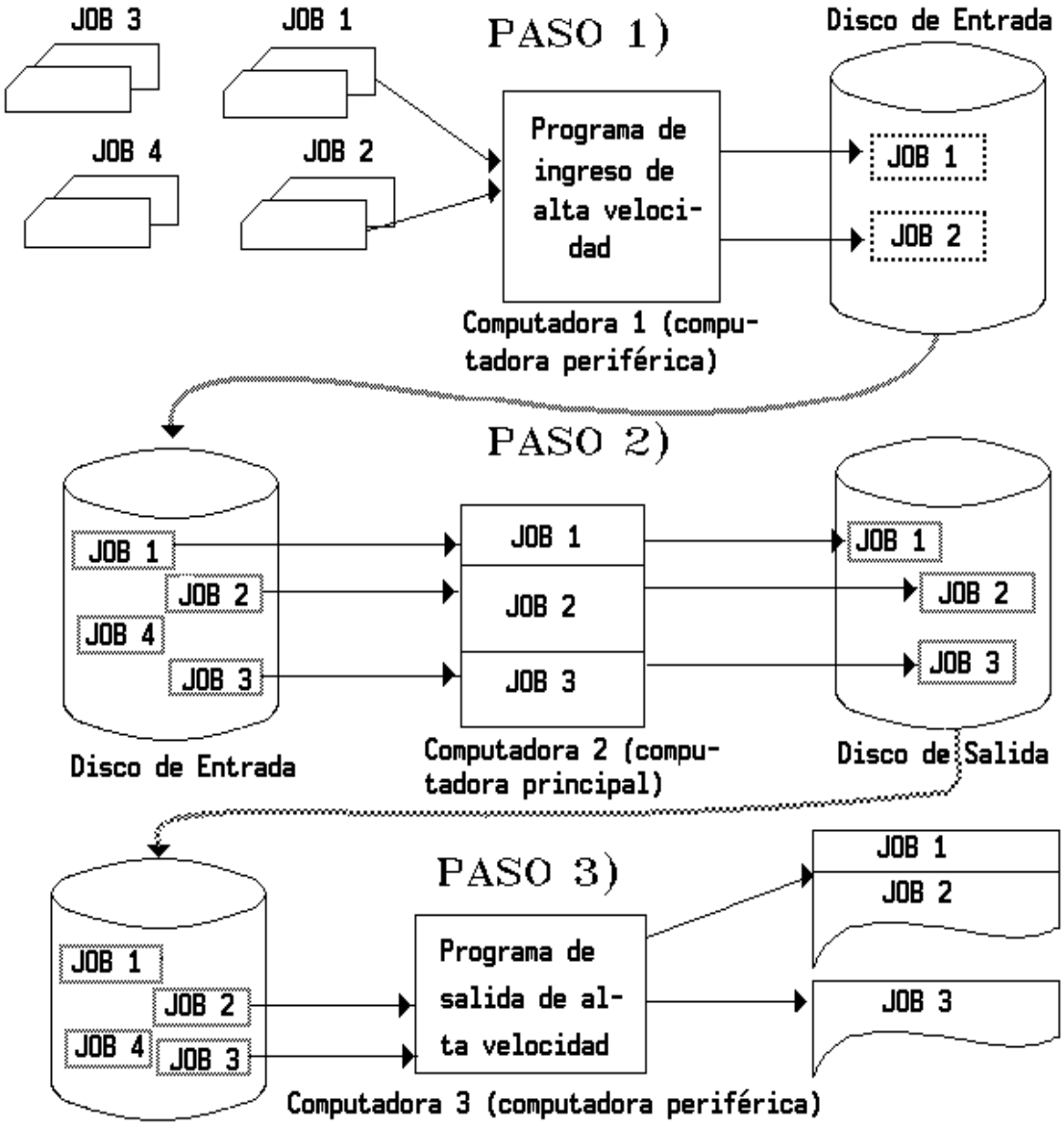


Fig. 14.20. - Operaciones periféricas off-line.

dos con:

* Intervención humana: es necesario que alguien mueva los discos desde las computadoras periféricas hasta la computadora principal. Esto tiene como resultado la introducción de errores humanos y de ineficiencia.
* Tiempo de turnaround: como consecuencia inmediata de lo anterior, aumenta el tiempo entre la entrada y la salida de un trabajo del sistema. Otro motivo de aumento de este tiempo es la necesidad de que un disco se encuentre lleno antes de poder trasladarlo a la computadora principal.
* Es difícil proveer algún tipo de administración del procesador por prioridades. Sólo podemos usar PROCESAMIENTO BATCH.

14.12.3. - **SISTEMAS DIRECTAMENTE ACOPLADOS**

La mayor desventaja de la aproximación anterior era la necesidad de mover físicamente los discos entre el computador principal y las computadoras periféricos. Existen diversas soluciones a este problema. En la Fig. 14.21 mostramos una configuración en la cual los discos de entrada y salida están conectados físicamente al computador periférico y al computador principal.

Esta configuración se conoce como Sistema Directamente Acoplado. Normalmente se utiliza una única computadora periférica lo suficientemente poderosa, acoplada al computador principal.

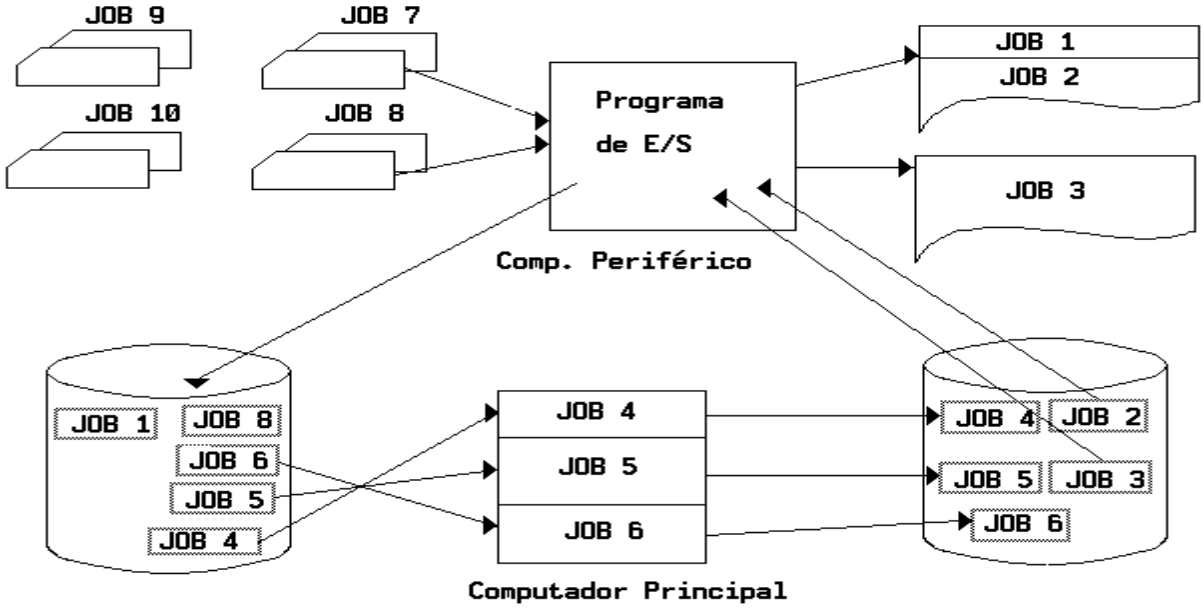


Fig. 14.21. - Configuración de un sistema directamente acoplado.

Esta aproximación elimina la mayoría de los problemas del procesamiento periférico fuera de línea. No es necesaria la intervención humana, y como los trabajos pueden procesarse tan pronto como la computadora periférica los coloca en el disco de entrada, no existe pérdida de tiempo de turnaround ni restricciones de administración del procesador.

Se puede utilizar un único disco como entrada y salida; pero el uso de un disco compartido de esta forma debe coordinarse cuidadosamente, porque si existen accesos frecuentes, puede haber conflictos que reduzcan la performance convirtiendo al disco un cuello de botella crítico.

14.12.4. - **PROCESADOR ADOSADO DE SOPORTE**

Otra variación de esta idea consiste de conectar directamente el periférico y la computadora principal vía una conexión de alta velocidad, como se ilustra en la Fig. 14.22.

En esta configuración, el computador periférico es un Procesador adosado de soporte.

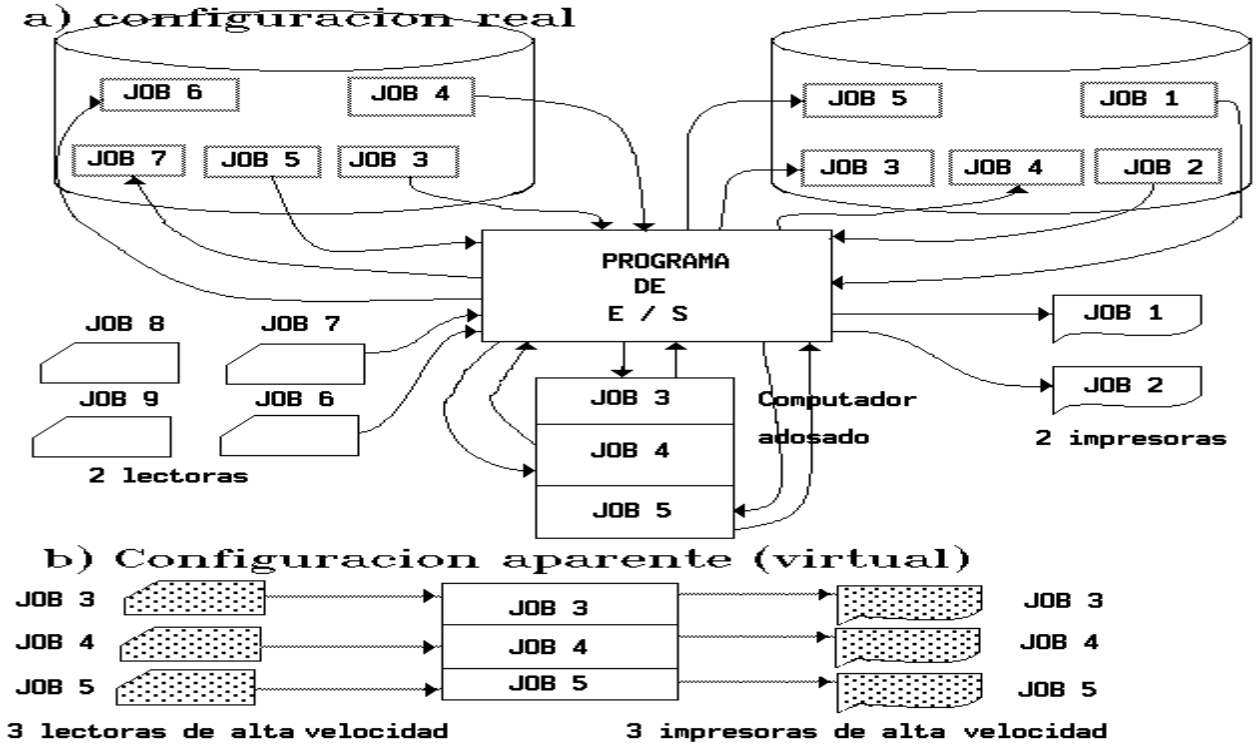


Fig. 14.22. - Configuración de Procesador Adosado de Soporte.

El procesador de soporte asume toda la responsabilidad de controlar los periféricos de E/S, así como los discos de E/S. También hace buffering y bloqueo, simplificando el trabajo del computador principal. En este modo, el procesador adosado tiene la apariencia de la existencia de múltiples lectoras e impresoras de muy alta velocidad (como vemos en la Fig. 14.22). Por analogía con el concepto de memoria virtual, el procesador adosado produce el efecto de dispositivos virtuales (es decir, muchos dispositivos y más rápidos que los que en realidad existen).

Las principales desventajas de este sistema son que son necesarios dos o más procesadores, y es posible que alguno de los procesadores esté ocioso o poco utilizado, mientras que el otro no puede manejar su carga. Este sistema, por lo tanto, no usa todos los recursos eficientemente.

14.12.5. - **SISTEMA VIRTUAL**

En todas las soluciones anteriores asumimos que existía una o más computadoras especiales, dedicadas al manejo de funciones de procesamiento de E/S. Considerando las facilidades de procesamiento disponibles en canales de E/S convencionales, así como las capacidades de multiprogramación, es necesario usar una computadora separada para manejar la E/S?.
En un sistema de SPOOLing, la computadora efectúa operaciones simultáneas de periféricos en línea (Simultaneous Peripheral Operations On-line). La idea se muestra en la Fig. 14.23.

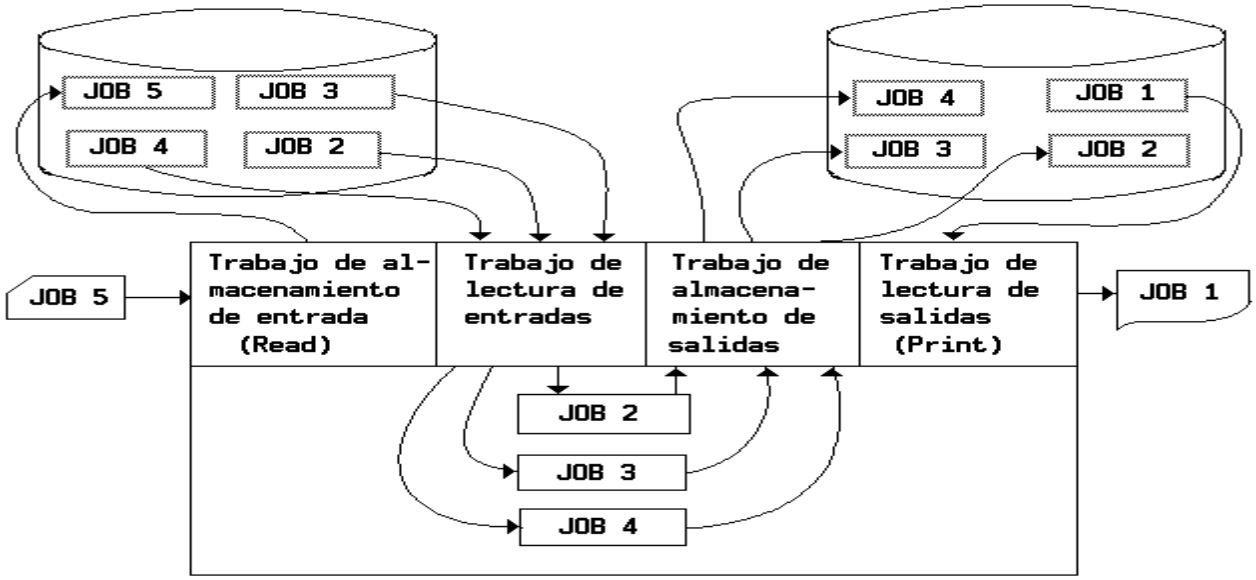


Fig. 14.23. - Sistema de Spooling.

En este sistema, determinados trabajos tienen las funciones del Procesador Adosado de Soporte, pero residen permanentemente en la computadora principal, y comparten el uso del procesador con trabajos normales vía multiprogramación. Muchas veces estos trabajos se incorporan al área de memoria del Sistema Operativo. Como debe haber coordinación y comunicación entre estos trabajos, se deben usar diversas facilidades de administración del procesador. Es necesario mantener información de los datos de entrada y salida que fueron almacenados en el disco. Finalmente, se debe lograr bloqueo, buffering y control de E/S, para lograr una buena performance.

La sencillez de implementar un mecanismo de Spool, depende directamente de las facilidades específicas provistas por la Administración de Memoria, Administración de Información, Administración del Procesador, y los componentes de Administración de Dispositivos del Sistema Operativo.

En un Sistema Operativo limitado, los programas de Spool pueden tener que hacer muchas funciones operativas; mientras que en un Sistema más complejo, el rol del Spool puede ser menor, dejándose el "trabajo duro" al Sistema Operativo. De hecho, los sistemas de Spooling mejoran tanto el sistema que se suelen proveer como parte de algunos sistemas operativos simples que no soportan multiprogramación.

14.12.5.1. - **DISEÑO DE UN SISTEMA DE SPOOLING**

Asumiremos que los programas de Spool son parte del Sistema Operativo, y que usan una Administración de Información propia y especializada. Estas son suposiciones típicas de sistemas operativos de pequeña y mediana escala. En estos sistemas hay dos BSV especiales:

READNEXT(BUFFER) : Lee la próxima tarjeta de entrada.
PRINTNEXT(BUFFER) : Imprime la próxima línea de salida.

En realidad, la tarjeta no es leída ni la línea impresa, ya que se están utilizando dispositivos virtuales; pero los trabajos que solicitan estos pedidos no lo saben.

Un sistema general de Spool de E/S puede subdividirse en cuatro componentes, como se ilustra en la Fig. 14.24. Estos componentes pueden agruparse de varias formas: por función (entrada o salida) o por el método de obtener el control (llamada al supervisor o interrupción).

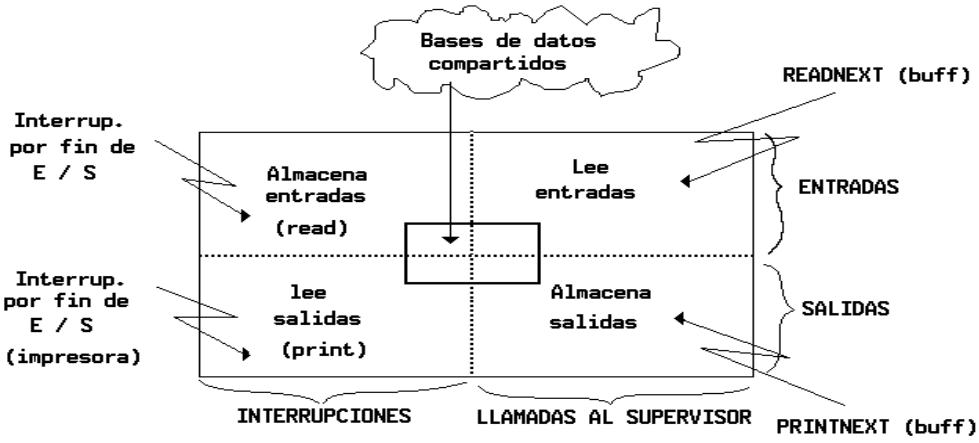


Fig. 14.24. - Estructura general del sistema de Spool.

14.12.5.2. - **SPOOL DE SALIDA**

Analicemos la función de Spool de salida (la del spool de entrada se maneja de una forma similar aunque esta técnica ha caído en desuso al ir desapareciendo la mayoría de los periféricos de entrada de baja velocidad).

El Spool debe efectuar dos operaciones principales:

- Interceptar la llamada a impresión por parte del proceso y almacenar físicamente cada línea impresa en un disco
- Recuperar cada línea impresa en el disco y transferirla a la impresora (impresión física).

Cuándo se efectúa realmente la operación de impresión? Debe hacerse independientemente del trabajo ya que, de hecho, *debe completarse luego de que finalice el mismo*.

Hay varias razones que justifican el hecho de que la impresión se realice luego de finalizado el trabajo o proceso que la generó:

- No es posible prever de antemano con qué frecuencia el proceso imprime. Si la impresión se comienza a realizar mientras el proceso aún está en ejecución puede ocurrir que se hayan impreso todas las líneas almacenadas en el spool y el proceso demora en generar más impresiones con lo cual la impresora está asignada (recordar que es un periférico dedicado) y sin embargo está ociosa.
- El contar con varios trabajos finalizados pendientes de impresión permite la planificación de los mismos para obtener el mejor tiempo de respuesta (por ejemplo: Mas corto primero –el de menor cantidad de líneas impresas-).
- Si un proceso genera una cantidad de líneas impresas excesiva es posible impedir que se imprima al observar que su spool en disco ha crecido desmesuradamente.

El sistema de Spool no es ni más ni menos un proceso más dentro del sistema con la característica que pertenece a las rutinas del sistema operativo y su función básica es interceptar los llamados a impresora por parte de los procesos y en lugar de realizar la impresión almacenar la línea impresa en un periférico compartido y luego de la finalización del proceso, transferir las líneas impresas almacenadas en el disco hacia la impresora.

Como cualquier proceso cuenta con su propia información, es decir, va llevando un registro de todos los procesos que inician impresión y mantiene los datos del lugar físico en el disco (que por extensión suele denominarse, disco de spool) en el que se encuentra almacenada la información impresa del proceso en cuestión.

Mantiene información sobre si el proceso aún está en uso de la impresora, o sea, su impresión aún no ha finalizado y sobre si el proceso ya finalizó su uso de la impresora y en consecuencia debe transferir la impresión al dispositivo físico, la impresora.

En un sistema que cuenta con spool para un proceso que pasa al estado de Terminado se agrega ahora un procesamiento especial por parte del sistema operativo que será la liberación del disco de spool (si el proceso generó impresión). Por lo tanto la rutina de spool interviene en las transiciones o estados de :

- Ejecutando a Bloqueado, interceptando el pedido de impresión ya que ahora se realizará una E/S al disco de spool en lugar de la impresora.
- Terminado, para transferir las líneas impresas desde el disco de spool hacia la impresora.

En la transición de Bloqueado a Listo, cuando finalice la E/S del disco de spool la rutina que atiende esa interrupción colocará al proceso nuevamente en estado de Listo

14.12.5.2.1. – **Un ejemplo**

Veamos con detenimiento un ejemplo concreto con un gráfico de tiempos.

Supongamos que contamos con un proceso que realiza durante toda su ejecución solamente dos impresiones y que además es el único proceso presente en el sistema.

Estado Proceso	E	B	B	E	E	B	B	E	E	E	T	T	T	T	T	T
CPU proceso	X			X	X			X	X	X						
CPU Spool		X				X					X			X		
Disco spool			Graba				Graba					Lee			Lee	
Impresora													X			X
Tiempo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

En nuestro gráfico hemos despreciado el tiempo de las rutinas del sistema operativo excepto los de la rutina de spool (por ejemplo, la rutina de atención de interrupciones por fin de E/S, el planificador de procesos, etc.).

Veamos que ocurre en cada momento:

Tiempo 1: El proceso comienza a ejecutar y lanza una E/S sobre la impresora

Tiempo 2: La rutina de spool intercepta el pedido, genera información en sus tablas internas para este proceso, busca espacio en el disco para almacenar la línea impresa y finalmente lanza la E/S sobre el disco de spool.

Tiempo 3: Se comienza a realizar la E/S sobre el disco de spool para almacenar la línea impresa

Tiempo 4: El proceso retoma su ejecución una vez finalizada la E/S

Tiempos 5 a 7: se repite el proceso de generación de la línea impresa en el disco de spool

Tiempos 8 a 10: El proceso continúa su ejecución y finaliza en el tiempo 10

Tiempo 11: La rutina de spool luego de finalizado el proceso recupera la información del mismo del disco de spool para enviar la impresión. En este caso se recupera la primer línea impresa y se despacha a la impresora (de estar libre)

Tiempo 12: Se lee el disco de spool para obtener la primer línea impresa del proceso.

Tiempo 13: La impresora realiza la impresión física de la primer línea del proceso

Tiempo 14: La rutina de spool recupera la segunda línea del proceso

Tiempo 15: Se lee el disco de spool

Tiempo 16: Se imprime la segunda línea del proceso.

Como nuestro proceso es el único en el sistema no tenemos interferencia por parte de otros procesos, es decir, la impresora está libre y pudo asignarse ni bien terminó el proceso para imprimir sus líneas. Además ni bien finalizó la E/S del disco de spool nuestro proceso pudo recomenzar su ejecución y no debió esperar que otro proceso abandone el control de la CPU.

El gráfico no está hecho a una escala correcta. Lo correcto sería que el tiempo de la rutina de spool fuera muy pequeño (aunque no nulo) ya que el spool como rutina del sistema operativo está optimizada para que su ejecución tenga un bajo impacto en la performance del sistema.

Además el tiempo de la E/S de impresión debería ser mucho mayor comparado con el tiempo de la E/S del disco de spool.

14.12.5.2.2. – **Multiprogramación con y sin spool y Monoprogramación**

Nótese que comparativamente respecto de un sistema que no cuenta con spool ahora se agregan a los tiempos de cualquier proceso el tiempo de ejecución de la rutina de spool que es pequeño, pero no nulo y los tiempos del disco de spool que son tiempos bastante importantes por ser tiempos de E/S físicas a un disco.

En conclusión si se compara el tiempo de ejecución de un solo proceso, solo en el sistema, respecto del mismo proceso en un sistema sin spool y en un sistema de monoprogramación puede establecerse que el tiempo total del proceso, entendiéndose por tal desde que ingresa al sistema hasta que finaliza todo su procesamiento (impresión incluida) será:

Multiprogramación con spool >> Multiprogramación sin spool > Monoprogramación

En la multiprogramación con spool se agregan los tiempos de la rutina de spool más los tiempos del disco de spool que deben computarse dos veces (grabar el disco y luego leerlo para mandar la impresión) mientras que en la multiprogramación sin spool respecto de monoprogramación solo se agrega, principalmente, la rutina del planificador de procesos (eventualmente las rutinas de atención de interrupción por reloj en los esquemas de administración del procesador que poseen reloj de intervalos).

14.12.5. – **Beneficios del Spool**

Luego de lo antedicho en el párrafo anterior uno podría preguntarse, entonces cuál es el beneficio concreto de agregar Spool en un sistema?

Hay varios beneficios, a saber:

1) Mediante la técnica de spool es posible simular más periféricos de los que existen en la instalación. No sería posible ejecutar un proceso que utiliza dos impresoras simultáneamente para emitir dos listados si la instalación tiene una sola impresora y no tiene spool.

2) Mediante la técnica de spool es posible simular periféricos que no existen en la instalación. Si la impresora se rompe es posible ejecutar igualmente los procesos y luego llevarse el disco de spool a otra instalación y hacer las impresiones.
3) Los procesos, en un sistema que cuenta con spool, están menos tiempo en el circuito de estados Ejecución-Listo-Bloqueado. Como las impresiones se realizan en realidad sobre un disco el tiempo que el proceso permanece en estado Bloqueado es menor y por lo tanto vuelve pronto al estado Listo. Esto permite un mayor grado de multiprogramación en la instalación.
4) Los periféricos manejados por el spool, la impresora en nuestro caso, pueden funcionar a su máxima velocidad lo que redunda en un mejor aprovechamiento del mismo. Nótese que como el spool en el momento de la impresión física solamente está realizando una transferencia del disco a la impresora no existe ningún procesamiento de los datos y por ende la transferencia se puede realizar a la máxima velocidad soportada por el periférico.
5) La planificación de los trabajos de impresión que se realiza una vez finalizados los procesos redunda también en un mejor aprovechamiento del uso de las impresoras.

Hay que tener presente que el beneficio de la implementación del spool es un beneficio *para el sistema en su conjunto* y no para un proceso en particular.

14.13. – **TECNOLOGIA RAID**

La tecnología de discos RAID (Redundant Array of Independent Disks) fue originalmente concebida en 1987 por ingenieros en la Universidad de California en Berkeley. Ellos definieron originalmente los niveles 0 a 5 de Raid y luego se agregaron más niveles.

La idea subyacente en esta tecnología es proveer a través de hardware un mecanismo de tolerancia a fallas directamente implementado sobre un conjunto de discos en donde se almacenan los datos. La lógica de funcionamiento está implementada íntegramente por hardware

Esto suele formar parte de lo que se suele denominar "**almacenamiento estable**", ya que el mismo es tolerante a fallas.

Hay que indicar que el nivel 6 no es mejor que el nivel 0 y los niveles solo se utilizan para indicar la forma en que la información se almacena. La diferencia es, por supuesto, el costo el cual deberá evaluarse según la criticidad de los datos que se deseen almacenar.

14.13.1. – **Niveles de RAID**

Nivel 0

La información se reparte entre diferentes ejes de discos que son considerados en su conjunto como una sola unidad. Es lineal y no se mantiene ninguna información de paridad.

Nivel 1

Este es uno de los más conocidos ya que se trata de un mirror (espejo) de discos. Un disco es copia fiel del contenido de otro disco, o sea, se duplica el espacio en disco siendo uno de los discos backup del otro.

Nivel 2

En este nivel la información se reparte entre diversos discos a nivel de bit y cuenta con información de paridad.

Nivel 3

En este nivel la información se divide a nivel de byte y un disco del conjunto esta dedicado a almacenar la información de paridad (códigos de Hamming).

Nivel 4

Es similar a RAID 3, pero la división de la información se realiza en grandes pedazos.

Nivel 5

La información se reparte en bloques secuenciales a través de los discos junto con la información de paridad.

Nivel 6

Es igual que RAID 5, pero agrega controladores de discos redundantes, buses extra, etc.

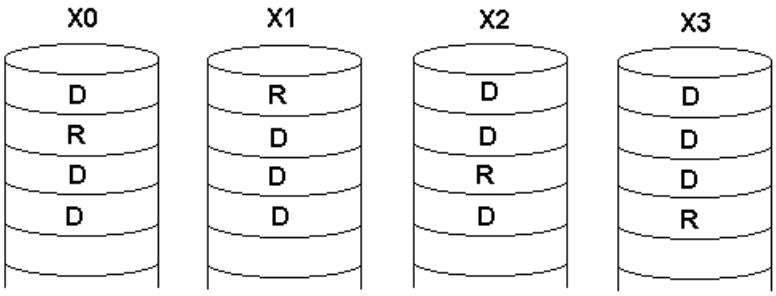
Nivel 7

En este nivel los buses de datos de E/S son asincrónicos. La jerarquía de dispositivos y buses de datos también es asincrónica.

Incluye un sistema operativo incorporado, orientado a los procesos de E/S, para manejar todas las transferencias de E/S a través de los discos.

14.13.2. – **Forma de almacenamiento de los datos**

Tomaremos como ejemplo un sistema con 4 discos implementados con tecnología RAID.



El dato original está compuesto por X0, X1 y X2 partes. En tanto que X3 se construye con una operación que resulta de las partes del dato original.

Digamos por ejemplo, que la operación aplicada es un XOR la que denotaremos con el símbolo $\oplus$.

Luego, se tiene :

$$X3 = X0 \oplus X1 \oplus X2$$

Supongamos ahora que el dato cambia en su parte de X1, luego :

$$X3' = X0 \oplus X1' \oplus X2$$

Ahora bien, como se sabe $A \oplus A = 0$, entonces podemos agregar lo siguiente:

$$X3' = X0 \oplus X1' \oplus X2 \oplus X1 \oplus X1$$

Pero $X0 \oplus X1 \oplus X2$ es igual a X3 con lo que queda

$$X3' = X3 \oplus X1' \oplus X1$$

Y esto ahorra el tener que leer todos los Xi de los otros discos.

Supongamos ahora que falla el disco que contiene X1, entonces se agrega a ambos miembros de la igualdad :

$$X3 \oplus X3 \oplus X1 = X0 \oplus X1 \oplus X2 \oplus X3 \oplus X1$$

Y eliminando los miembros nulos se obtiene:

$$X1 = X0 \oplus X2 \oplus X3$$

Con lo cual el dato original de X1 se recupera con los otros datos almacenados en los otros ejes.