CAPÍTULO II: GESTIÓN DE PROCESOS

ÍNDICE

Bibliografía

- Morera Pascual, Juan; Pérez Campanero, Juan A. *Teoría y diseño de los sistemas operativos*. Anaya Multimedia, 1995.
- Tanenbaum, Andrew S. *Sistemas operativos modernos*. Prentice-Hall, 2003.
- Stallings, William. *Sistemas Operativos*. Prentice-Hall 2001.
- Jesús Carretero, Félix García, Pedro De Miguel, Fernando Pérez. McGraw-Hill, 2001.
- A. Silberschatz, J. Peterson, P. Galvin. *Sistemas operativos*. Conceptos fundamentales. Addison-Wesley, 1999.

1. Introducción

Antes de ver como se ejecuta un proceso vamos a echarle un vistazo a cómo el procesador ejecuta las instrucciones.

1.1. Registros

La arquitectura de registros depende de la máquina.

Ofrecen el nivel de memoria más rápido y pequeño. Se dividen en dos grandes grupos:

- ***Registros visibles por el usuario***: están disponibles para el programador (el compilador los puede usar para optimizar). Se clasifican en:
    - *Registros de datos* (AX, BX,...): son de propósito general con restricciones (p. ej. sólo para operaciones en coma flotante).
    - *Registros de dirección*: contienen direcciones de memoria de datos e instrucciones. A veces están dedicados a un modo concreto de direccionamiento:
        - Registro Índice (SI,DI): para direccionamiento indexado (base + desplazamiento).
        - Puntero de segmento (CS,DS): para direccionamiento segmentado (CS,DS). La memoria está dividida en segmentos.
        - Puntero de pila(SP): contiene la dirección de la cima de la pila.
    - *Registro de código de condición(FLAGS)*: bits activados como resultado de determinadas operaciones. Se agrupan en uno o más registros.

    No todas las máquinas salvan estos registros al llamar a una subrutina de tratamiento de interrupciones.
- ***Registros de control y estado***: para la ejecución de instrucciones son esenciales los siguientes:
    - *Contador de programa (PC)*: la dirección de la próxima instrucción a leer.
    - *Registro de Instrucción (IR)*: la última instrucción leída.
    - *Palabra de estado del los elementos del sistema (PSW)*: es un conjunto de registros que contiene información de estado (códigos de condición y otra información). Entre otros: bits de signo, bits de cero, bits de acarreo, igualdad, desbordamiento, habilitar/deshabilitar interrupción, supervisor, ...
    - Otros registros de estado: registros con punteros a rutinas de tratamiento de interrupción, dirección de memoria con más información de estado, etc.

1.2. Ejecución de instrucciones

Un programa a ejecutar será un conjunto de instrucciones cargadas en memoria.

La tarea del procesador será leer las instrucciones del programa y ejecutarlas.

Vamos a considerar (de forma muy sencilla) que el procesamiento de instrucciones consta de dos pasos:

1. Trae la instrucción desde memoria, en IR se almacena el contenido indicado por PC.

2. Ejecución de la instrucción.

Estos pasos se repiten hasta acabar el programa:

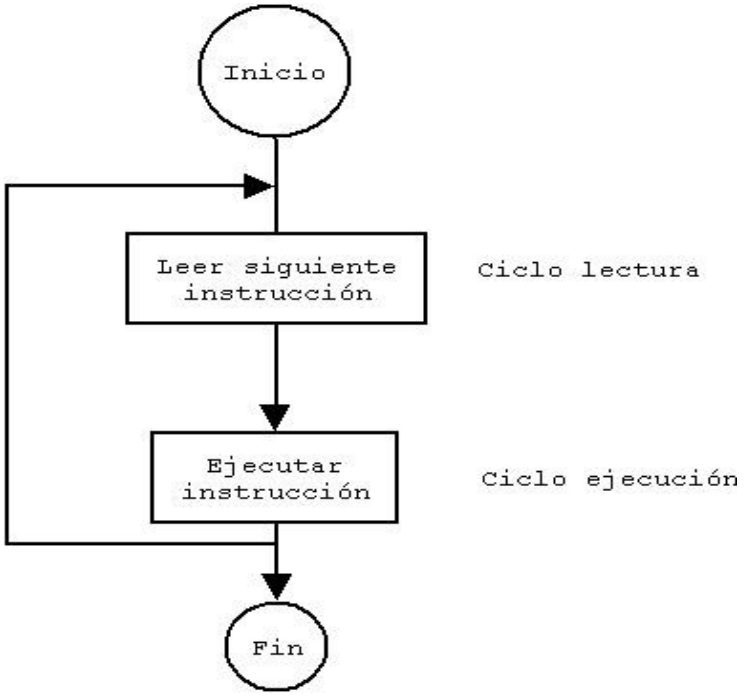


Figura II-1: Ciclo de instrucción

El procesamiento de una instrucción simple se llama **CICLO DE INSTRUCCIÓN**

El registro PC contiene la próxima instrucción a leer. Al comienzo de cada ciclo de instrucción el procesador lee una instrucción de memoria. A menos que indiquemos otra cosa , el PC siempre se incrementa después de leer una instrucción (con lo cual apunta a la siguiente instrucción de la secuencia, que es la ubicada en la posición superior de memoria).

La instrucción leída se carga en el registro IR. La instrucción contiene, en binario, la operación que el procesador debe ejecutar.

El procesador interpreta la instrucción y realiza la acción, que puede ser de las siguientes categorías:

- Procesador / memoria: transfiere datos de procesador a memoria y viceversa.

- Procesador- E/S: transfiere datos de o desde procesador a módulos de E/S (dispositivos).

- Tratamiento de datos: el procesador realiza alguna operación aritmética o lógica de los datos.
- Control: se altera la secuencia de ejecución (saltos, saltos condicionales ...). Esto se realiza ajustando el valor de PC.
- Otras instrucciones.

1.3. Niveles de ejecución

Existen dos niveles de ejecución del procesador:

- **Nivel de usuario**: en este modo se ejecutan los programas de usuario, tiene ciertas restricciones (subconjunto de instrucciones y registros, prohibidas E/S directamente, etc.).
- **Nivel del núcleo (modo supervisor)**: es el nivel en el que se ejecuta el núcleo y no tiene restricciones.

Por tanto la computadora presenta dos **modelos de programación**. Uno más restrictivo y otro más permisivo.

El cambio del nivel de ejecución se realiza modificando uno o varios bits del registro de estado.

1.4. Interrupción de reloj

Es una interrupción que se produce cada cierto intervalo de tiempo (20 ms) y provoca que el S.O tome el control mediante la R.S.I (Rutina de Servicio de Interrupción) del reloj.

Las R.S.I. forman parte del S.O.

1.5. Ejemplo de ejecución

Consideremos el siguiente proceso sencillo:

	4	12
Formato de Instrucción:	C. Operación	Dirección
Formato de un entero:	Signo	Magnitud
	1	15

Registros de la C.P.U:

Contador de programa: PC

Registro de Instrucción: IR

Acumulador: AC

Lista parcial de códigos de operación:

0001 --> Cargar de la memoria al AC

0010 --> Almacenar de AC a memoria

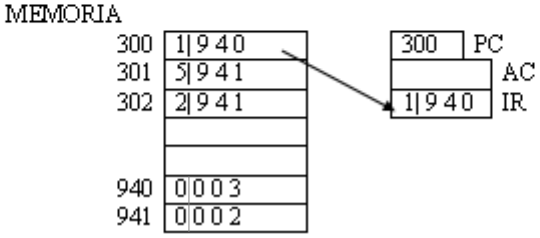
0101 --> Sumar al AC el contenido de la memoria

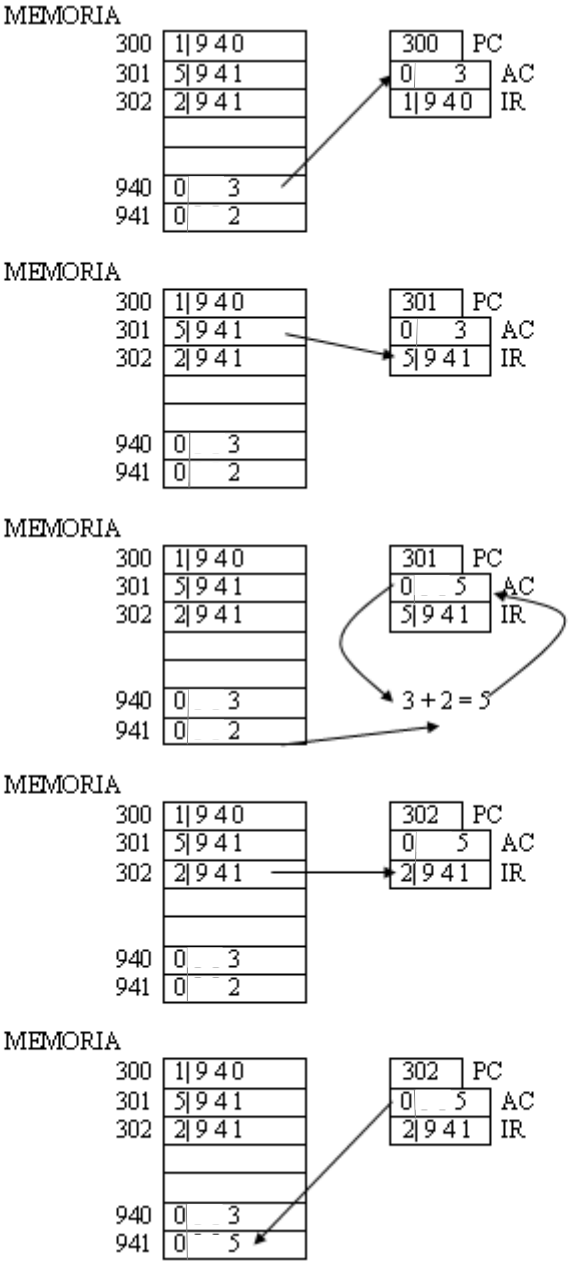
Los datos e instrucciones son de 16 bits.

A) ¿Cuántos códigos hay? 2^4

B) ¿Cuántas posiciones de memoria? 2^{12} (4K posiciones de memoria de 16 bits= 8K)

Inicialmente el PC contiene 300 y la memoria está de la siguiente forma:




MEMORIA
300 1|940
301 5|941
302 2|941
940 0 3
941 0 2
300 PC
0 3 AC
1|940 IR
MEMORIA
300 1|940
301 5|941
302 2|941
940 0 3
941 0 2
301 PC
0 3 AC
5|941 IR
MEMORIA
300 1|940
301 5|941
302 2|941
940 0 3
941 0 2
301 PC
0 5 AC
5|941 IR
3 + 2 = 5
MEMORIA
300 1|940
301 5|941
302 2|941
940 0 3
941 0 2
302 PC
0 5 AC
2|941 IR
MEMORIA
300 1|940
301 5|941
302 2|941
940 0 3
941 0 5
302 PC
0 5 AC
2|941 IR

2. Concepto de proceso

El concepto de proceso surge con la necesidad de que el computador esté ejecutando varios programas al “mismo tiempo”(concurrentemente).

Bajo este modelo todos los programas que se ejecutan en el ordenador (incluso a veces el S.O) se organizan en un conjunto de procesos.

¿Qué es un proceso?

- Un proceso es un programa en ejecución que comprende el valor actual del contador de programa, de los registros y de las variables, siendo además la unidad de propiedad de los recursos y la unidad de expedición.

- También podemos decir que un proceso es una instancia de un programa en ejecución, por tanto es una entidad dinámica, al contrario del concepto de programa que es una entidad estática.

- Normalmente el procesador está conmutando de un proceso a otro, pero el sistema es mucho más fácil de entender si lo consideramos como un conjunto de procesos que se ejecutan *quasi* en paralelo, olvidándose de cómo el procesador pasa de un proceso a otro.

3. Estados de un proceso

Nosotros vamos a considerar que los procesos se ejecutan en un S.O **multiusuario** y **multitarea**, que será el tipo de S.O. que tratemos de aquí en adelante.

La **multitarea** se basa en tres características diferentes:

- Paralelismo real entre E/S y procesador.

- Alternancia en los procesos de fases de E/S y de procesamiento.

- Memoria principal capaz de almacenar varios procesos.

Para poder realizar pues una multitarea con esas características necesitamos definir los posibles estados en los que puede estar un proceso.

Vamos a considerar que un proceso puede pasar por los siguientes **estados** a lo largo de su ciclo de vida:

- **Inactivo**. Consideraremos así a los programas, es decir, al proceso que aún no ha sido ejecutado, y por tanto no ha sido creado.

- **Preparado**. Proceso activo que posee todos los recursos necesarios para ejecutarse, pero al que le falta, precisamente, el procesador. Nada más crearse un proceso pasa al estado de preparado, y se queda esperando a que el S.O. le asigne el procesador. Existe un módulo en el S.O (el planificador) que se encarga de elegir a uno de los procesos de entre los que se encuentran en preparados.

- **Ejecución**. Proceso que está siendo ejecutado por el procesador. Con un solo procesador sólo puede haber un proceso en ejecución.

- **Bloqueado**. Proceso que además de no contar con el procesador requiere algún otro recurso del sistema. Generalmente un proceso queda bloqueado al

solicitar alguna operación de E/S o tener que esperar a que llegue alguna señal de sincronización.

En todo momento el conjunto de procesos activos está cambiando de estado. Los estados de todos los procesos activos y los recursos (tanto ocupados como libres) del sistema se considera el **ESTADO GLOBAL DEL SISTEMA**.

4. Transiciones de estado de los procesos

Cada vez que se ordena ejecutar un programa a un S.O, éste crea un proceso. Este proceso pasa de inactivo a preparado. En el estado preparado habrá generalmente muchos procesos, por lo que se tendrán que colocar al final de la cola. Poco a poco irán avanzando por la cola hasta ponerse al principio de la cola.

El primero de la cola, llegado el momento, pasará de preparado a ejecución. El módulo encargado de realizar esta actividad se denomina **dispatcher (despachador).**

Este proceso dispondrá de un determinado tiempo de ejecución llamado **quantum** (tiempo máximo de ocupación del procesador por parte de un proceso). Si el proceso no libera voluntariamente el procesador antes de que acabe ese tiempo, será el S.O. el que lo haga abandonar la C.P.U. para volver a colocarlo en la cola de preparados.

Si el proceso finaliza su ejecución antes de que acabe su tiempo, entonces, el S.O. destruirá el proceso, que volvería a un estado inactivo.

Por lo general el proceso requerirá alguna operación de E/S, antes de que finalice su tiempo, lo cual implica que él mismo libere el procesador y se bloquee esperando la finalización de la operación de E/S (o la llegada de alguna señal).

Cuando se produzca la operación de E/S, (o llegue la señal) el proceso volverá de nuevo a estar preparado y el S.O. lo colocará en la cola de procesos que se encuentran en ese estado.

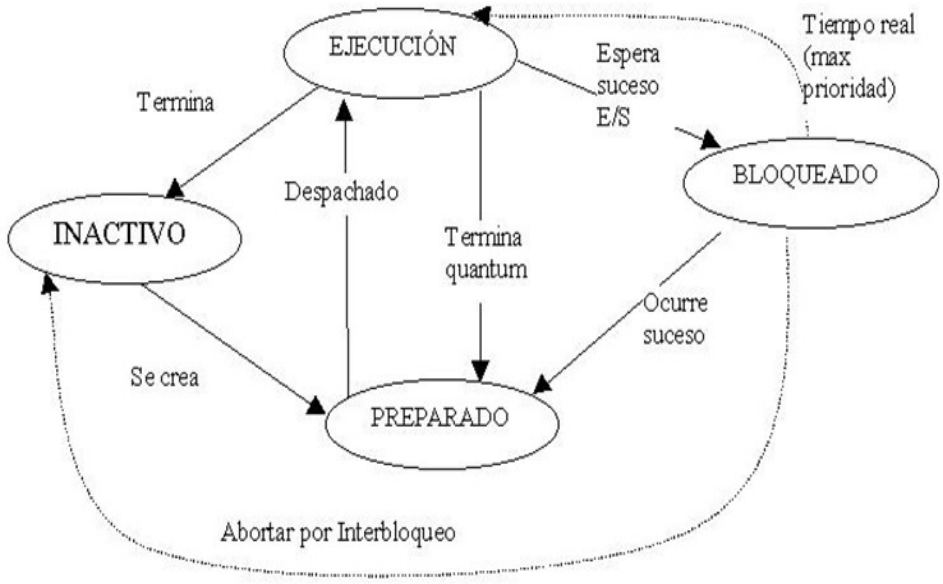


Figura II-2: Diagrama de Transición de Estados

Vemos que la única transición de estado iniciada por el proceso es el bloqueo, las otras 3 transiciones son iniciadas por entidades externas al proceso.

Sabemos que el procesador nunca está inactivo, por tanto, ¿cómo podemos mantener esta estructura de funcionamiento cuando no hay ningún proceso en ejecución, o cuando el proceso que está en ejecución está realizando una operación de E/S?. Para solucionar este problema tenemos un proceso llamado **proceso nulo**, que siempre está en el sistema y que ejecuta un bucle infinito, de tal forma que al tener la prioridad más baja de todos, sólo se ejecuta cuando no hay ningún otro proceso para hacerlo. Este proceso abandonará el procesador al llegar un proceso de prioridad mayor (cualquiera).

4.1. Estados suspendidos

Para disminuir el grado de multiprogramación el S.O. puede suspender algunos procesos, lo que implica que se le quita la memoria que tiene asignada, dejándolos enteramente en la zona de intercambio. El objetivo es dejar memoria suficiente a los procesos no suspendidos para que 'el conjunto residente' tenga el tamaño adecuado y evitar la 'hiperpaginación (Thrasing)'.

Si el proceso que se suspende está en la cola de preparados irá al estado de preparado suspendido, pero si está en la cola de bloqueados se pasará al estado de suspendido bloqueado.

Los procesos batch que entren al sistema lo podrán hacer pasando al estado de preparado o suspendido preparado.

Para poder hacer esta suspensión se necesita trabajar con 'intercambio', y no todos los sistemas operativos lo permiten

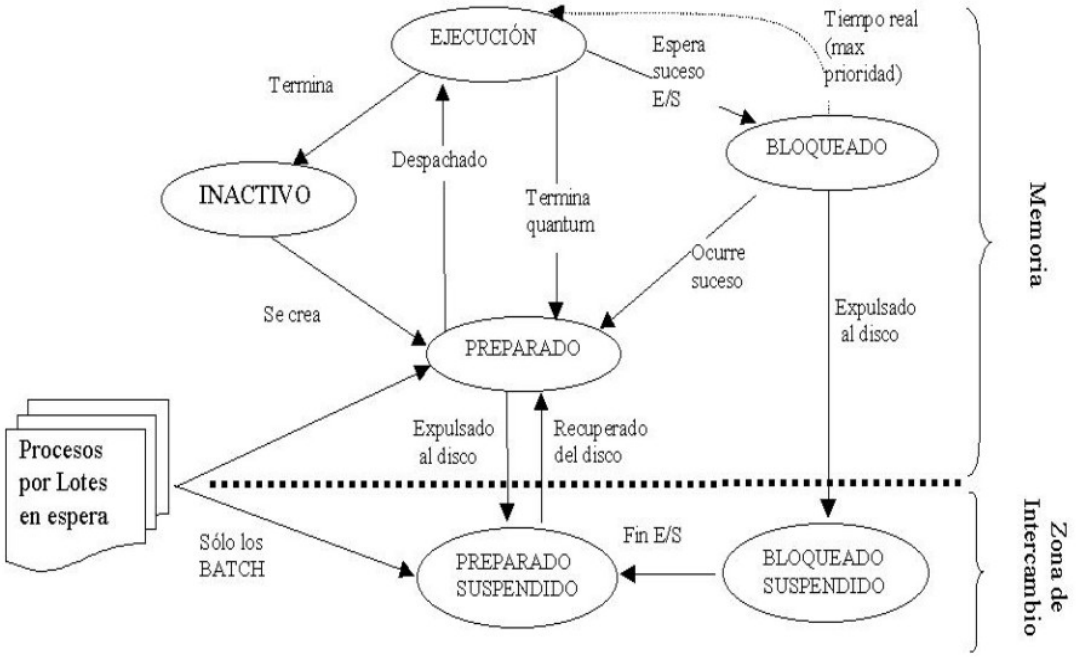


Figura II-3: Diagrama de Transición de Estados con los Estados Suspendidos

5. El Bloque de Control de Procesos (PCB)

El S.O. necesita saber en todo momento el *estado global del sistema*, pero ¿cómo consigue llevar el control de los procesos para poder tener un estado global en cada momento?.

A continuación vamos a ver las estructuras de datos que se necesitan para llevar a cabo todo esto.

El S.O. necesita una serie de estructuras en las que mantener la situación del sistema en todo momento. La estructura que usa el S.O. para controlar sus datos y la ejecución de los programas de los usuarios se denomina *Bloque de Control del Sistema (SCB)* y básicamente contiene la siguiente información:

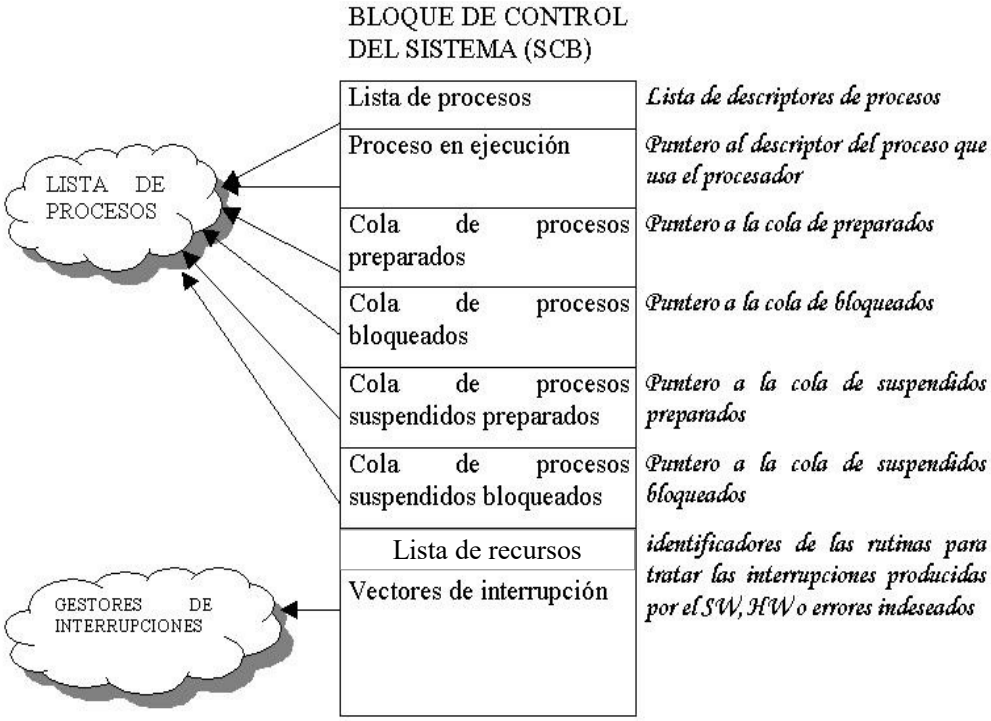


Figura II-4: Bloque de Control de Sistema

Ahora bien, todos esos apuntadores a procesos deberán apuntar a unas estructuras que definan inequívocamente al proceso. En dicha estructura deberemos poder encontrar toda la información referente a dicho proceso. Esa estructura, que en la gráfica anterior se denominan descriptores de procesos, se llama *Bloque de Control de Proceso*, y vamos a verla a continuación.

5.1. El Bloque de Control de Procesos (PCB)

Cada proceso se representa en el S.O. por un conjunto de datos, que incluyen, toda la información necesaria para definirlo: el estado, recursos usados, valores de los registros , etc.

Esta estructura representa el concepto de proceso para el S.O. y se denomina Bloque de Control de Proceso (PCB).

El PCB debe estar en Memoria Principal, para poder realizar el acceso de forma rápida, aunque parte de la información, que no sea imprescindible, puede estar descargada en disco.

La información que se incluye en el PCB es:

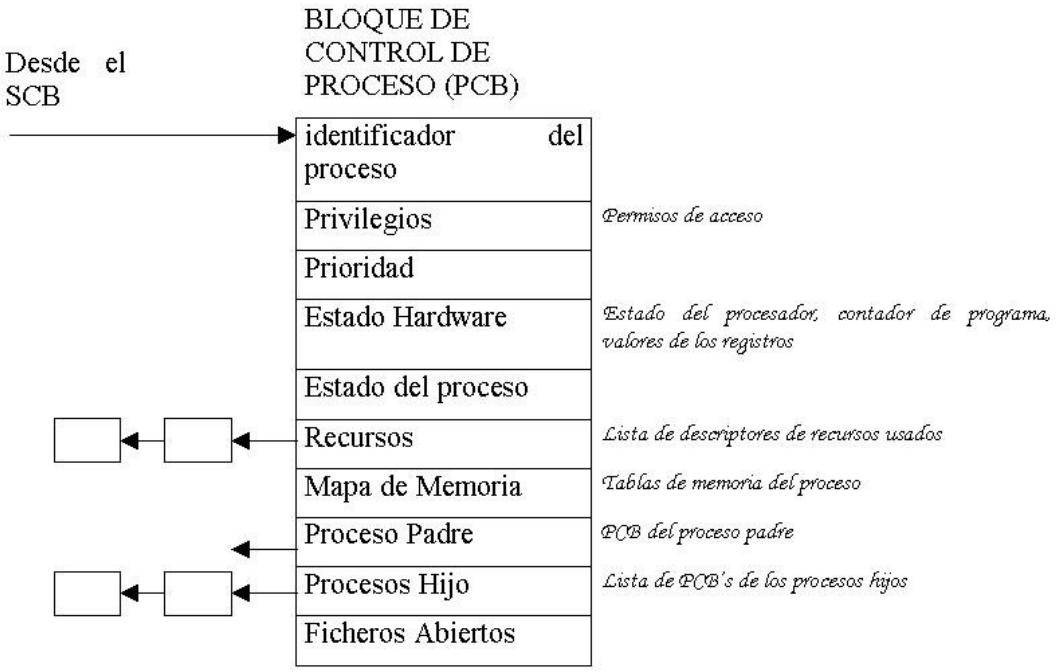


Figura II-5: Bloque de Control de Proceso

La información de cada proceso se encuentra 'parcialmente' en el PCB y parcialmente fuera de él(como vemos en la figura). La decisión de donde colocarla dependerá de:

- **Eficiencia**. La tabla de PCBs suele ser una estructura estática, donde los PCBs tienen un tamaño fijo. Así pues, la información que puede crecer debe estar fuera del PCB (por ejemplo, la tabla de páginas de cada proceso tendrá un tamaño diferente, en su lugar tenemos un apuntador a dicha tabla. Si es segmentación tenemos en el PCB una descripción del segmento, donde se incluye el puntero a la subtabla de páginas).

- **Compartir información**. Cuando la información es compartida por varios procesos, esta no se incluye en el PCB. Un ejemplo son los punteros de posición de los archivos abiertos (al ser compartidos requieren que no estén en el PCB), así si un proceso escribe en el fichero y actualiza el puntero al escribir el otro proceso lo hará a continuación, y no encima de lo que ha escrito el anterior, como ocurriría si cada uno tiene su propio puntero de posición.

5.2. Objetivos

Con el PCB se persiguen dos objetivos fundamentales:

- Que el S.O pueda localizar fácilmente la información

- Preservar los datos del proceso en caso de que tenga que ser desalojado temporalmente de su ejecución.

5.3. Prioridades

Normalmente, todos los procesos no tienen las mismas exigencias de tiempo en su ejecución. Hay procesos que requieren una atención inmediata, mientras que otros pueden ser demorados más tiempo en su ejecución sin ningún perjuicio.

En estos casos hay procesos que deben acceder más veces al procesador y antes que otros, y esto se consigue definiendo prioridades de ejecución.

De esta forma los procesos más prioritarios se ven más favorecidos en la ejecución que los procesos menos prioritarios. La prioridad normalmente viene dada por un número. Dependiendo del S.O. un número más alto indicará mayor prioridad o al revés.

La prioridad de un proceso se puede modificar en ejecución.

5.4. Estado Global del Sistema

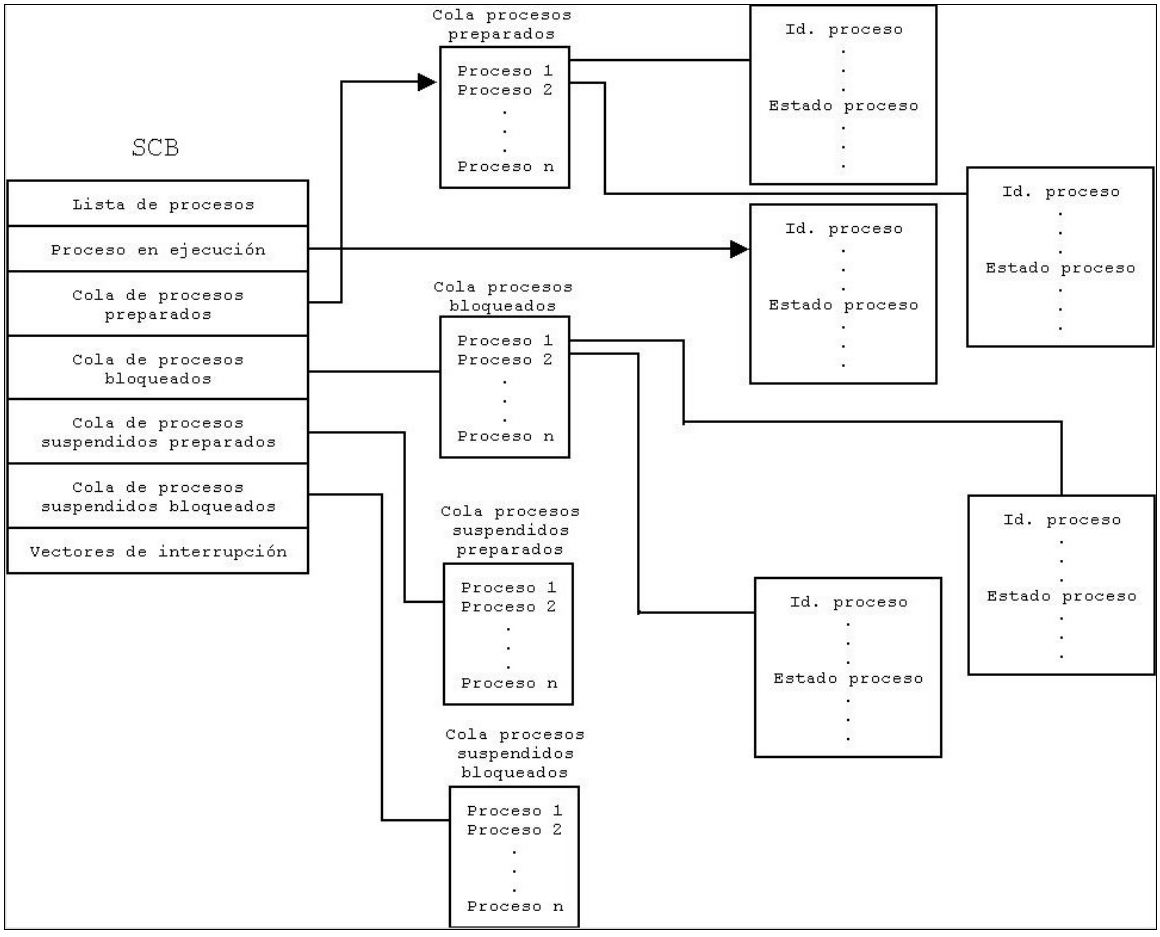


Figura II-6: Estado Global del Sistema

6. Operaciones sobre procesos

Las operaciones más habituales realizadas con los procesos son:

Crear: como argumentos requiere argumentos como el nombre y la prioridad.

¿Que acciones se realizan?

- Dar un identificador al proceso.
- Buscar un hueco en la lista de PCBs del sistema.
- Crear el PCB del proceso con toda la información necesaria.
- Asignarle los recursos que necesite.
- Insertar el proceso en la lista de procesos preparados.

La creación de procesos puede ser:

- Jerárquica: el proceso que se crea es hijo del proceso creador y hereda el entorno de ejecución de su padre:

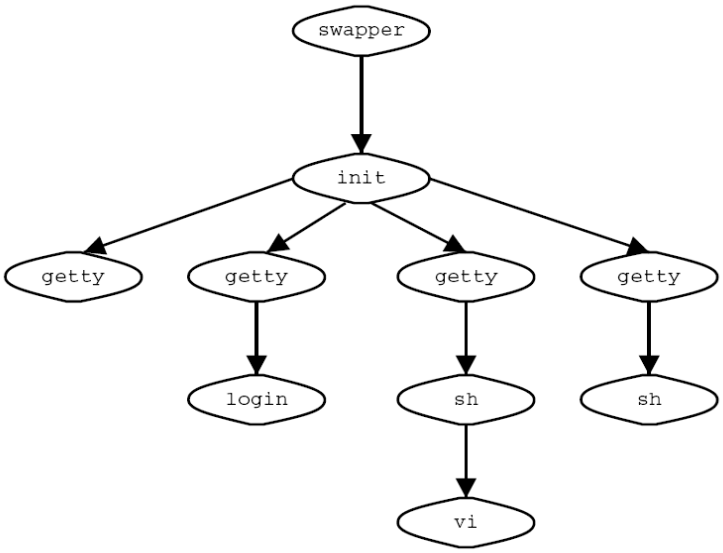


Figura II-7: Creación de procesos jerárquica

- No jerárquica: los procesos nuevos son independientes de su creador. Windows NT no mantiene de forma explícita la estructura jerárquica de procesos.

Finalizar: se destruye su PCB y se libera el hueco de la lista de PCB's del sistema. En una jerarquía de proceso hay que decidir si se destruye a los hijos al destruir el proceso o no.

Despachar un proceso: poner en ejecución a uno de los procesos que hay en la cola de preparados.

Bloquear un proceso: Llevar a un proceso de ejecución a la cola de bloqueados.

Desbloquear un proceso : ocurre el evento que esperaba el proceso, éste pasará de la cola de procesos bloqueados a preparados.

Suspender un proceso: se lleva al proceso a la cola de procesos suspendidos preparados o suspendidos bloqueados, según sea su estado.

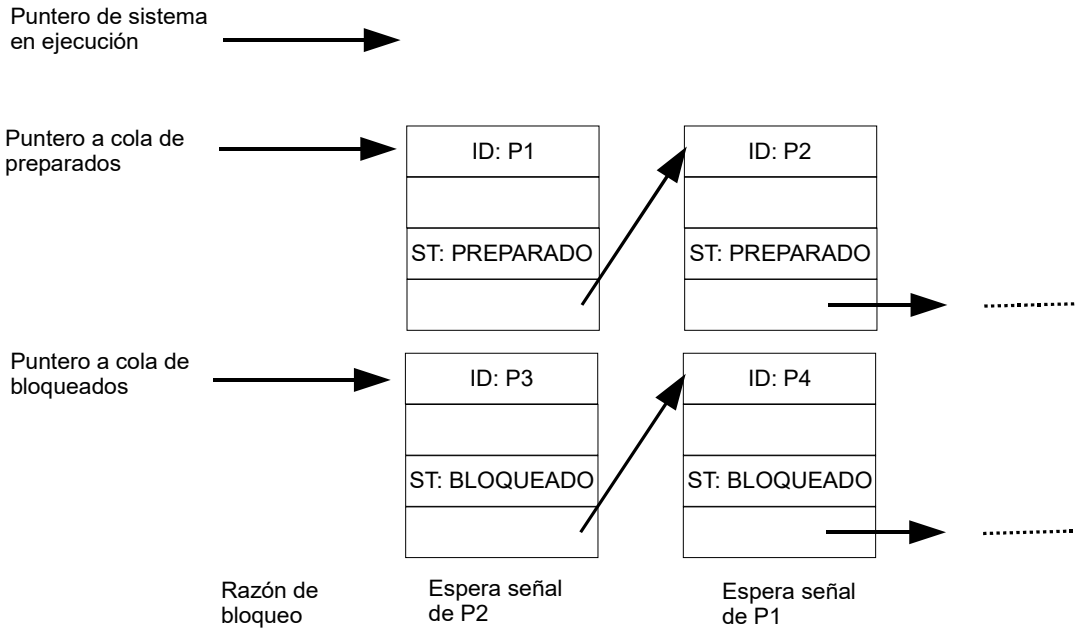
Recuperar un proceso suspendido. Se lleva el proceso desde la cola de suspendidos a preparados.

Abortar. Eliminar un proceso de la cola de bloqueados o de preparados.

Cambiar la prioridad. Modifica la prioridad del proceso.

6.1. Ejemplo

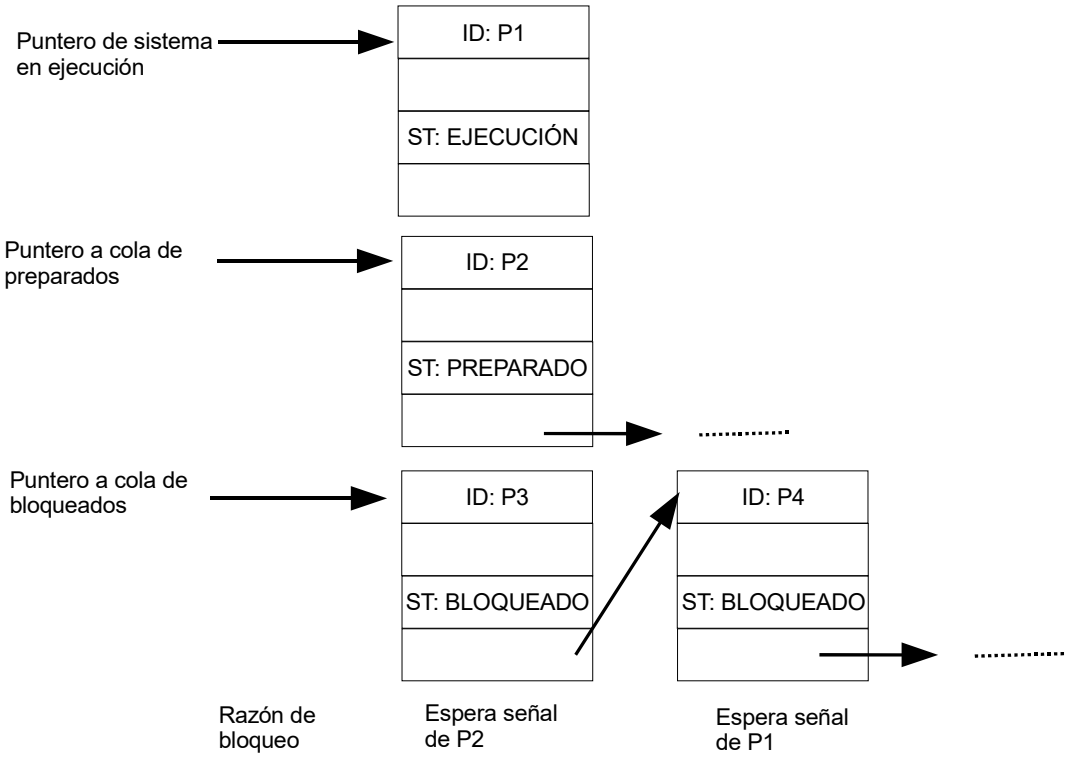
Partiendo de la figura siguiente:



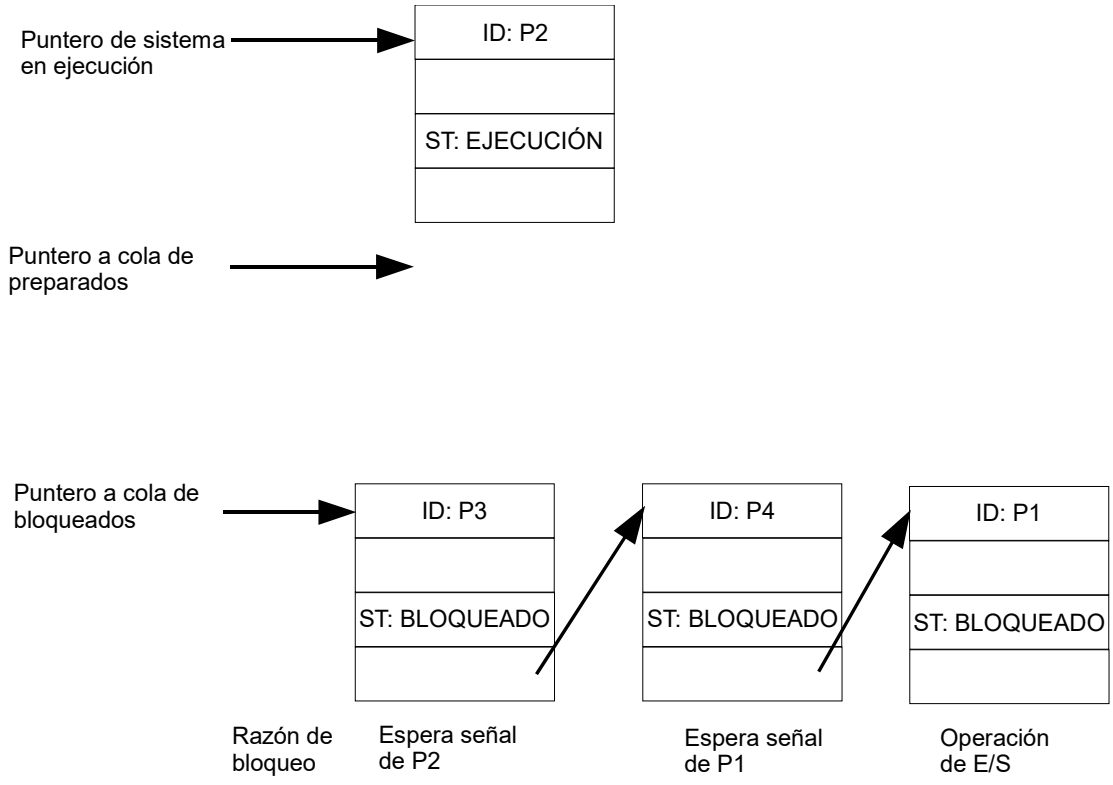
Reflejar las siguientes operaciones:

1. P1 pasa a ejecución.
2. P1 realiza una petición de E/S de disco.
3. Finaliza la E/S de P1.
4. Finalizar el quantum de P2.

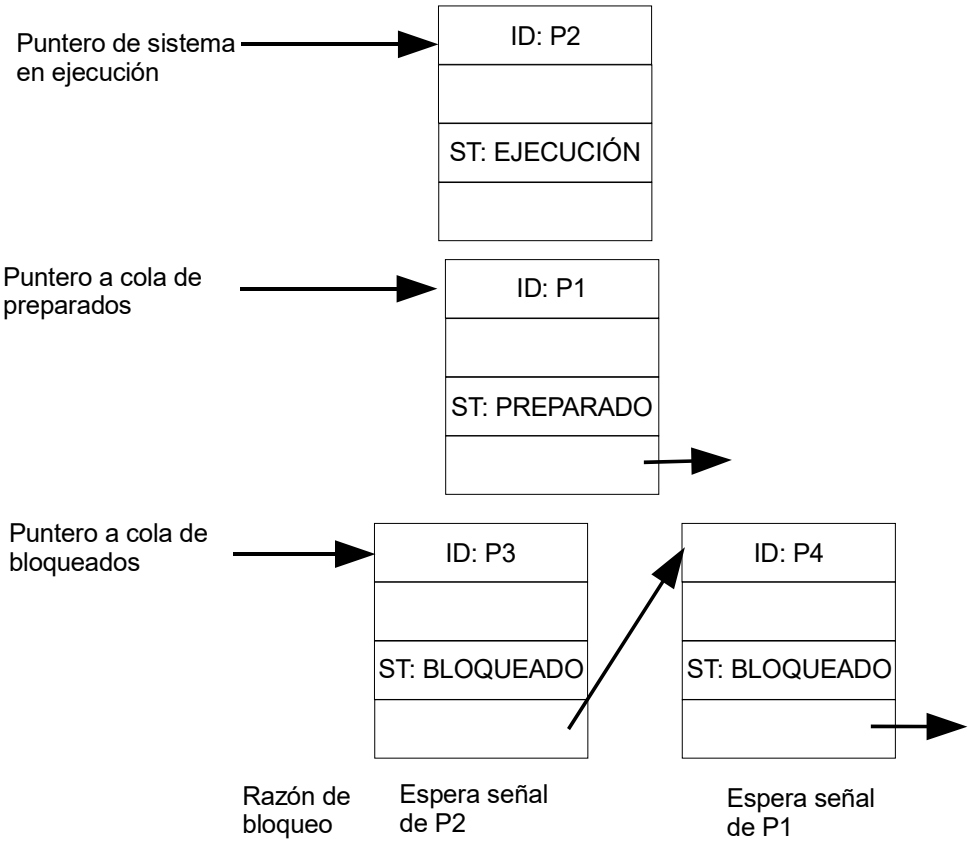
1. El proceso P1 pasa a ejecución:



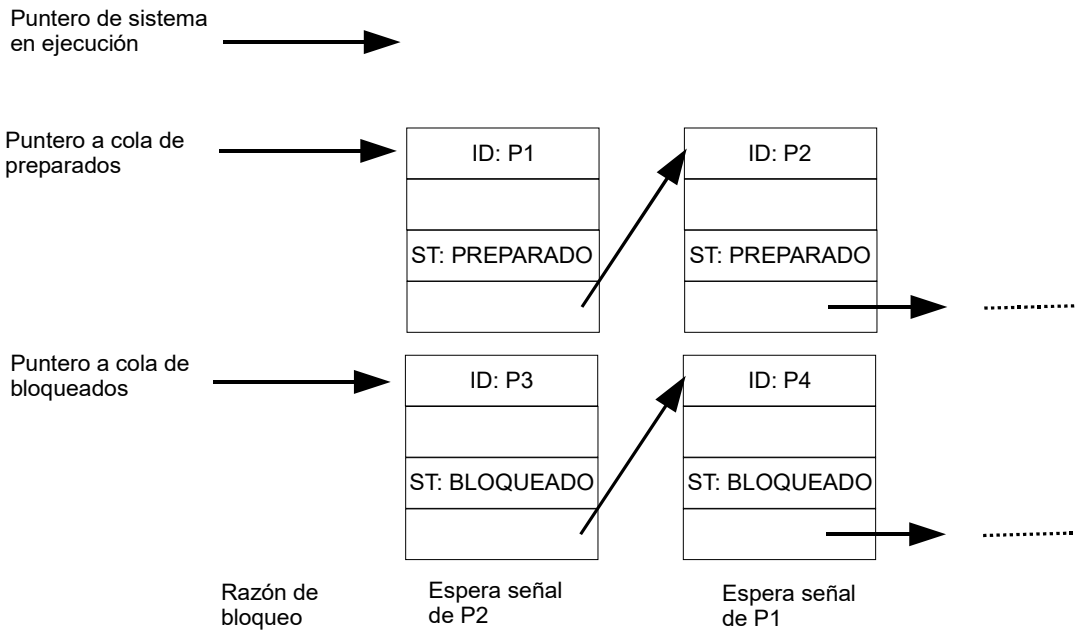
2. Se realiza una petición de E/S de disco por P1:



3. El proceso P1 termina la E/S de disco:



4. El quantum del proceso P2 finaliza:



7. Procesos e hilos

Todo proceso se puede ver como:

- Unidad de propiedad de los recursos (los recursos se asignan al proceso).
- Unidad de expedición: el proceso es unidad de ejecución y planificación.

Estas características se pueden diferenciar para que sean tratadas de forma independiente por el S.O.:

- Unidad de propiedad de los recursos: proceso o tarea.
- Unidad de expedición: Hilo.

A los hilos también se le llaman procesos ligeros, hebras o thread.

Un hilo es un programa en ejecución (flujo de ejecución), que comparte la imagen de memoria y otras informaciones con otros hilos.

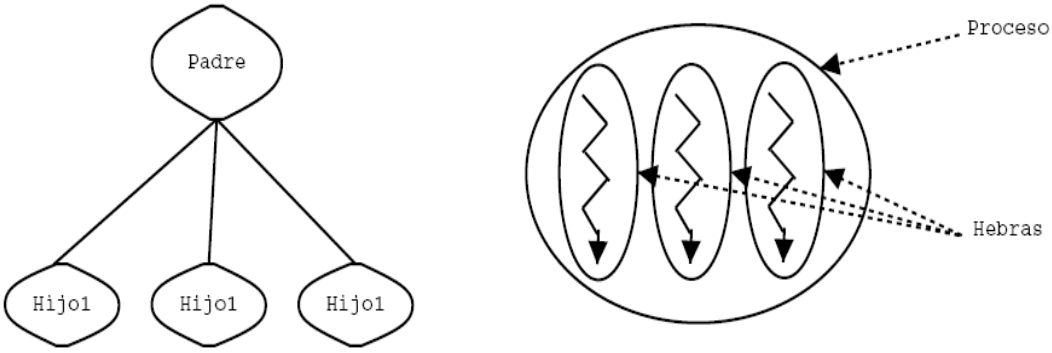


Figura II-8: (a) Jerarquía de procesos (b) Hebras

Los procesos tradicionales (o procesos pesados) se pueden ver como procesos que sólo tienen una hebra.

Utilidad: puede existir más de un hilo dentro del mismo proceso ⇒ mejora concurrencia y compartición de recursos. Podemos hacer que las aplicaciones sean 'paralelizables' y conseguimos que el proceso avance más rápidamente (el cambio de contexto entre hebras es mínimo).

Cambio de contexto = Salvar PCB + CP a la dirección de la rutina de tratamiento de interrupción + cambio modo + ejecución rutina de tratamiento de interrupción.

Las hebras de un proceso no son tan independientes (no hay protección) como los procesos distintos ya que:

- comparten espacio de direcciones ⇒ uso de mecanismos de sincronización.
- comparten archivos, procesos hijos, recursos E/S, etc.

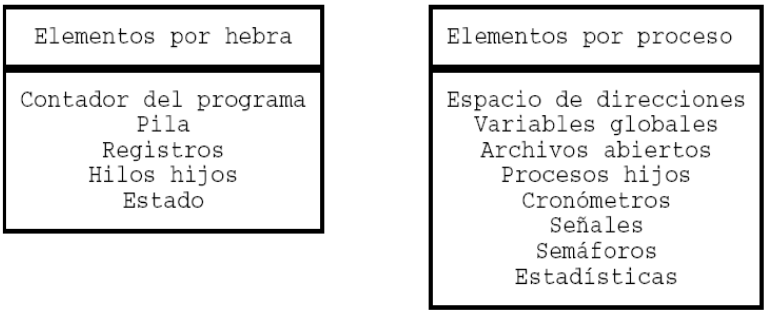


Figura II-9:Elementos de los procesos y de los hilos

Ventajas:

- Es más rápido crear un hilo en un proceso existente que crear un proceso nuevo.
- Es más rápido terminar el hilo que el proceso.
- Es más rápido cambiar entre dos hilos de un mismo proceso.

La comunicación entre hebras se hace a través de la memoria compartida.

Cada hilo sólo pertenece a un proceso y no puede existir fuera de él.

Los **estados de los hilos** serán:

- Preparados
- Bloqueados
- En ejecución

El **estado del proceso** será:

- En ejecución, si hay al menos un hilo en ejecución.
- Preparado, si hay al menos un hilo preparado y ninguno en ejecución.
- Bloqueado, si todos los hilos lo están.

La finalización de un proceso supone la finalización de todos los hilos.

Con los hilos no tiene sentido hablar de estados suspendidos, puesto que si el proceso está suspendido, todos sus hilos lo estarán.

A partir de Windows NT, los hilos constituyen la unidad ejecutable en los sistemas Windows.

8. Interrupciones

Para entender cómo funcionan las interrupciones nos podemos fijar en cualquier ejemplo de la vida real. Por ejemplo, si estamos leyendo un libro en casa y suena el timbre de la puerta lo que haremos será marcar el lugar por donde estamos leyendo el libro, a continuación abriremos la puerta para ver quién es. Suponiendo que es el cartero con una carta certificada la recogeremos, firmaremos, leeremos la carta, y a continuación volveremos al libro **por el lugar donde lo dejamos**. Eso será una interrupción y su tratamiento.

Por tanto, una **interrupción** es la presencia de un evento o señal que obliga al S.O. a tomar el control del procesador para estudiarla y tratarla. Normalmente esto ocurre por dos tipos de eventos: las interrupciones y las excepciones.

De cara al usuario es una interrupción de la secuencia de ejecución, no necesita añadir ningún código especial:

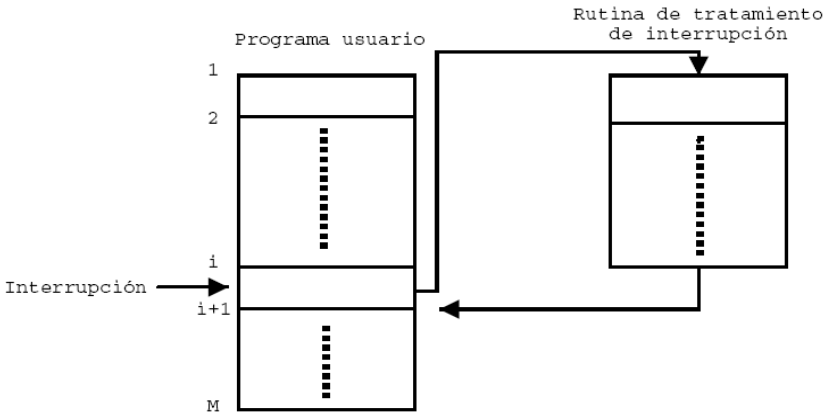


Figura II-10: Funcionamiento de interrupción

Para poder atenderlas en el SCB están las direcciones de las rutinas que deben activarse (vector de interrupciones o excepciones):

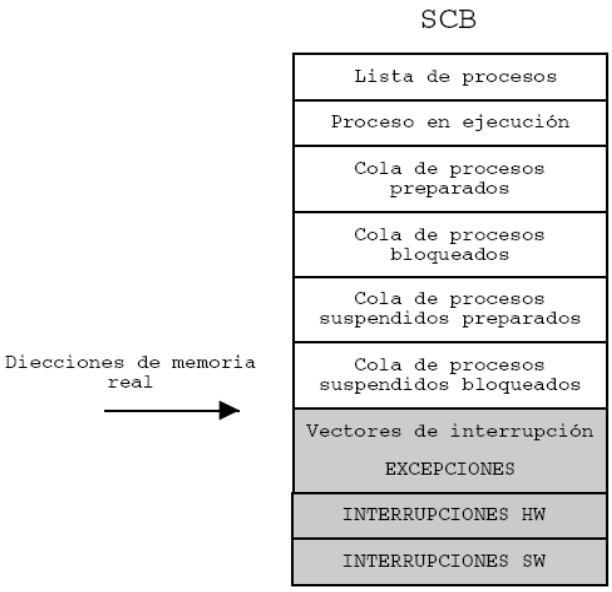


Figura II-11: Vector de Interrupciones

Estas rutinas, llamadas R.S.I. (Rutinas de Servicio de Interrupción) forman parte del S.O.

8.1. Utilidades

En multiprogramación permiten al S.O. tomar el control ante un error del HW o del programa en ejecución.

Permiten simultanear el uso de la E/S y el procesador (el dispositivo avisa con una interrupción).

Permite reparto de tiempos (tiempo compartido).

Permiten la posibilidad de reconocer eventos externos que deba controlar el sistema.

8.2. Tipos de interrupciones

Interrupciones HW. Las interrupciones hardware ocurren cuando un dispositivo necesita atención del procesador y genera una señal eléctrica en la línea IRQ (Interrupt ReQuest) que tiene asignada.

Interrupciones SW: Los procesadores Intel de la gama x86 y compatibles, disponen de una instrucción INT que permite generar por software cualquiera de los tipos de interrupción hardware. El proceso seguido es exactamente el mismo que si se recibe la interrupción hardware correspondiente.

Excepciones: Durante el funcionamiento del procesador pueden ocurrir circunstancias excepcionales; es usual citar como ejemplo el caso de una división por cero. En estos casos, el procesador genera una excepción, que es tratada como si fuese una interrupción, con la diferencia de que el número de interrupción asociado depende del tipo de excepción.

8.3. Gestión de las interrupciones.

Cuando ocurre una interrupción el S.O. toma el control del procesador, suspendiendo el proceso en ejecución.

Dependiendo del tipo de interrupción lleva a cabo su tratamiento mediante una rutina cuya dirección de comienzo está en el vector de interrupciones.

Cada tipo de interrupción lleva asociada una acción distinta.

Una interrupción desencadena una serie de sucesos hardware y software:

1. El dispositivo emite una señal de interrupción al procesador.

2. El procesador finaliza la instrucción en curso antes de responder a la interrupción

3. El procesador pregunta por la interrupción, comprueba que existe y envía una señal de reconocimiento al dispositivo que la generó.

4. El procesador necesita prepararse para transferir el control a la rutina de tratamiento de la interrupción. Para ello debe guardar la información del proceso en curso. Lo mínimo que necesita es el valor del PC y el PSW. Esto se puede guardar en la pila de control del sistema.

5. El procesador carga el PC con la dirección de comienzo de la rutina de tratamiento de la interrupción.

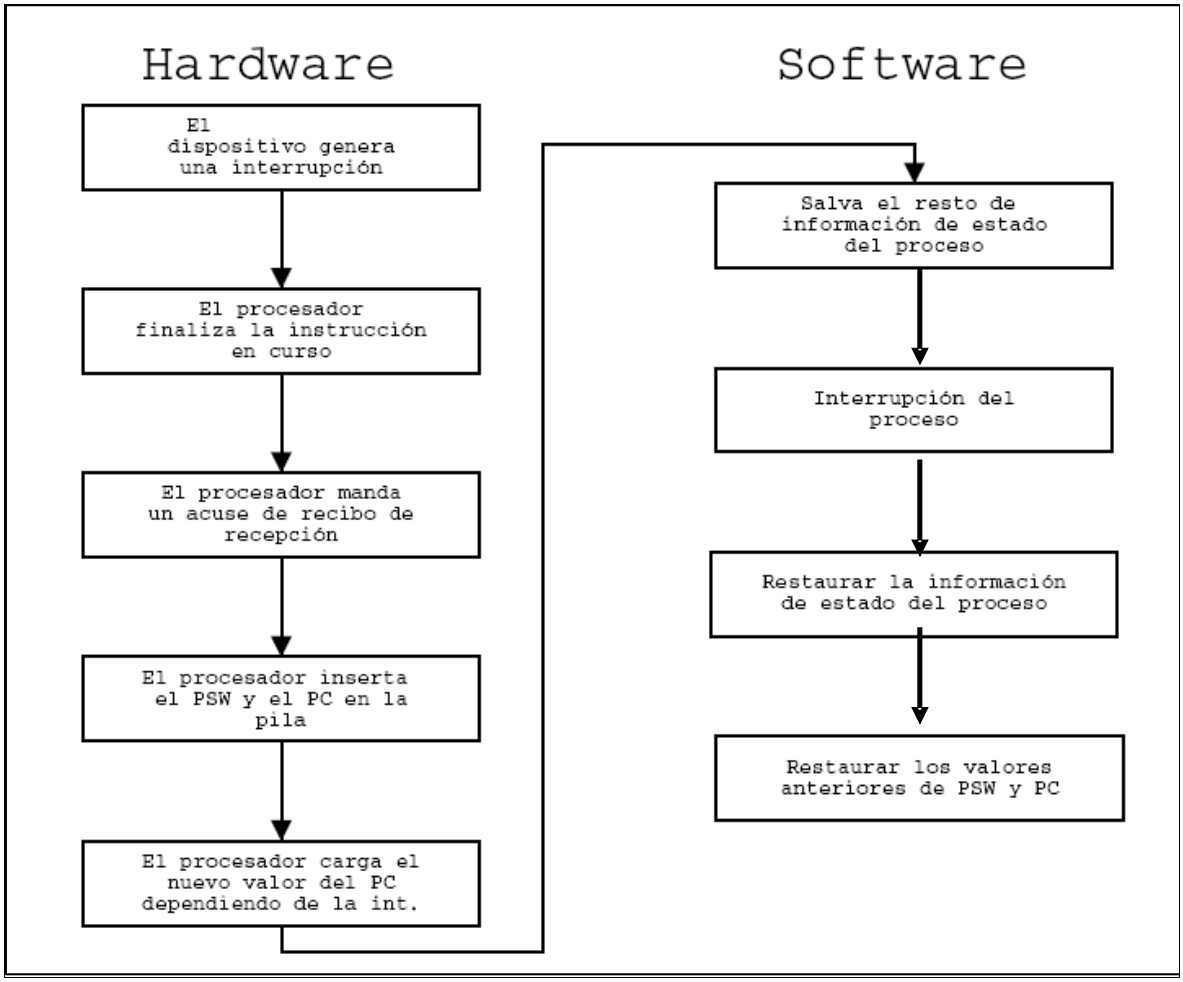


Figure II-12: Sucesos desencadenados por una interrupción

Hasta este momento todo lo ha realizado el hardware, y a partir de ahora comienza el próximo ciclo de instrucción, donde el contador de programa PC está cargado con la dirección de la rutina de tratamiento de la interrupción:

6. La información del proceso que ha sido salvada antes de transferir el control a la rutina de atención a la interrupción de la interrupción ha sido mínima, de tal forma que lo primero que hace la R.S.I. es salvar el resto de valores de todos los registros que tienen todavía información del proceso que estaba en ejecución colocándolos en la pila del sistema.

7. Ahora se trata la interrupción, lo que puede provocar comunicación con el dispositivo de E/S.

8. Ahora se restauran los valores de los registros del proceso que estaba en ejecución desde la pila.

9. Finalmente se colocan nuevamente el PSW y el PC del proceso que fue interrumpido, con lo cual la siguiente instrucción que se ejecuta es la del proceso, que continúa su ejecución normalmente.

Como vemos el programa de usuario no dispone de ningún código especial, sino que son el procesador y el S.O. los responsables de la atención de la interrupción, suspensión del programa de usuario y su posterior reanudación.

Pero en el ciclo de instrucción que hemos estudiado, no hay posibilidad de realizar todas estas operaciones. Para poder llevar a cabo estas operaciones el ciclo de instrucción debe poder reconocer las interrupciones, quedando de la siguiente forma:

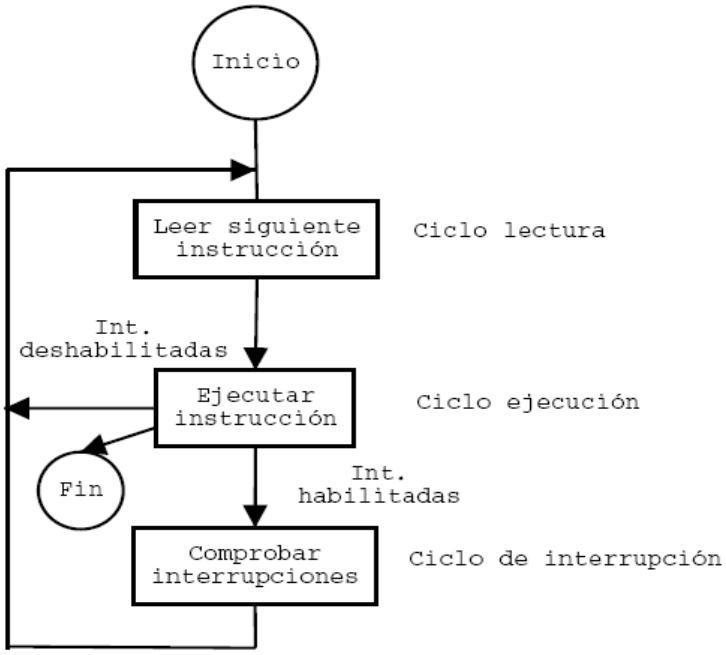


Figura II-13: Ciclo de instrucción con tratamiento de interrupciones

Se añade un ciclo de interrupción. En este ciclo:

- Comprobamos si hay interrupciones pendientes.
- Si no las hay continuamos la ejecución.
- Si hay pendientes se suspende la ejecución y se ejecuta la rutina de tratamiento.

El nuevo ciclo añade una pequeña sobrecarga de proceso en comparación con el tiempo que se gana al usar interrupciones.

8.4. Cambios de contexto

Entenderemos por contexto, toda aquella información que puede verse alterada, por la ejecución de una rutina de interrupción, y que será necesaria para continuar el proceso que ha sido interrumpido (registros del procesador, zonas de memoria, etc.).

Un cambio de contexto ocurre, por tanto, cuando la información referente al proceso en curso debe ser almacenada, para dar paso a la ejecución de otra rutina o proceso, que utiliza su propia información. Normalmente, esto lleva asociado además un cambio en el modo de trabajo del procesador.

Los cambios de contexto se deben a dos causas:

1. El proceso que está en ejecución produce una llamada al sistema operativo.

2. Llega una interrupción.

En ambos casos, la información del proceso en curso debe ser almacenada, para dar lugar a la ejecución de una rutina del sistema operativo. Además, esto conlleva un cambio de modo en el funcionamiento del procesador, pasando a modo supervisor. Dichas tareas constituyen un cambio de contexto.

Una vez que la llamada al sistema finaliza (o la interrupción ha sido atendida), se debe restaurar la información del proceso que estaba en ejecución, y volver a poner el procesador en modo usuario. Esto constituirá también un cambio de contexto.

Por tanto, una llamada al sistema o la atención de una interrupción suponen realizar dos cambios de contexto.

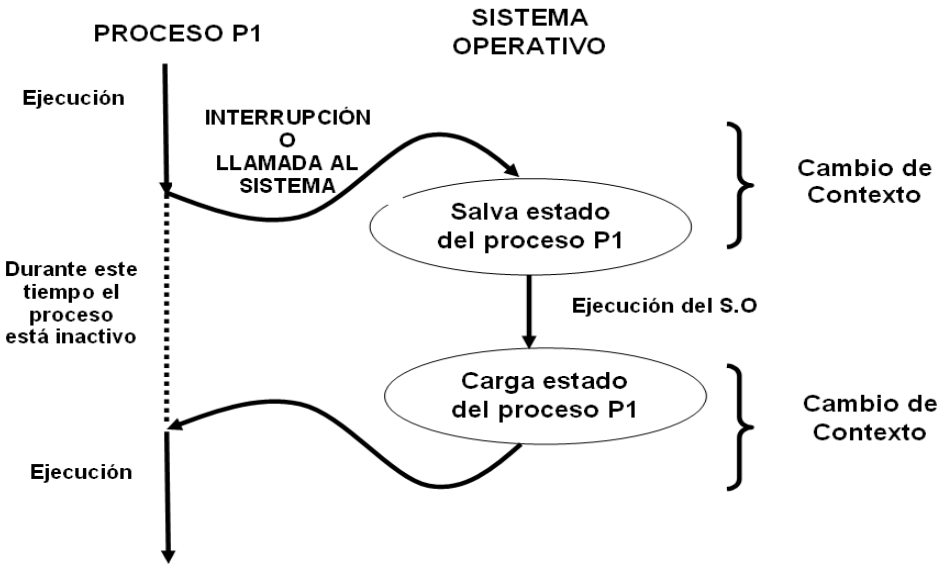


Figura II-14: Cambios de Contexto

Hay ocasiones en las que tras la llamada al sistema o la llegada de la interrupción, el proceso que debe ser ejecutado es distinto al que se estaba ejecutando.

En este caso, el primer cambio de contexto se realizará para salvaguardar el contexto del proceso en ejecución, y tras la llamada al sistema o la interrupción, el siguiente cambio de contexto se producirá para restaurar el contexto de otro de los procesos preparados.

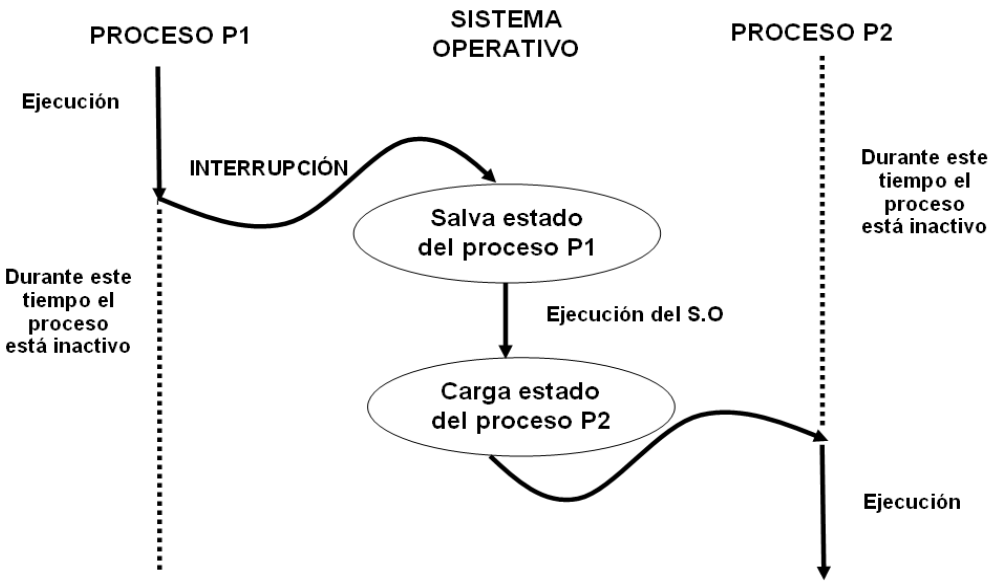


Figura II-15: Cambios de contexto durante los cambios de proceso

8.5. Interrupciones Múltiples

Es posible que lleguen varias interrupciones a la vez.

Dos enfoques para resolverlo:

- Inhabilitar interrupciones mientras se trata una, se ignorarían las señales de interrupción que quedaran pendientes. Se tratan las interrupciones secuencialmente, no tiene en cuenta las necesidades críticas de tiempo:

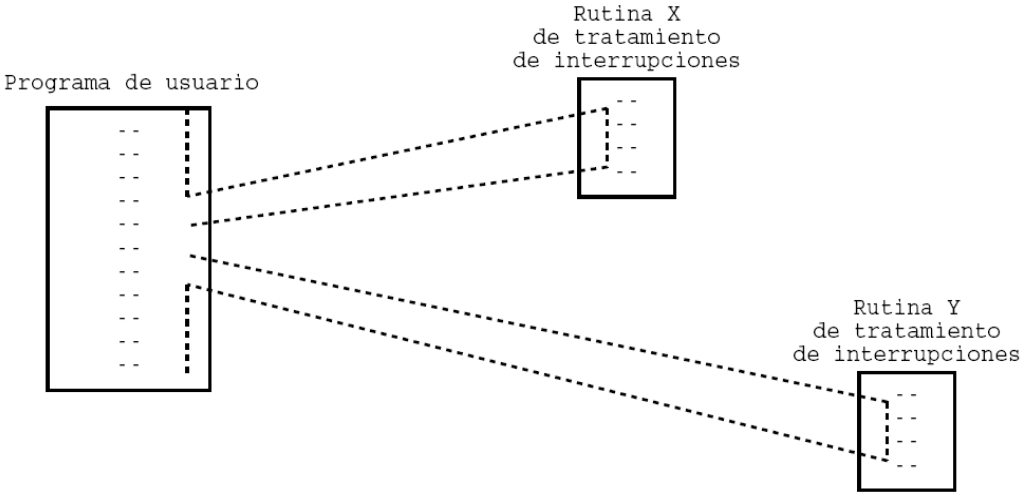


Figura II-16: Interrupciones múltiples deshabilitadas

- Definir prioridades, una interrupción de alta prioridad puede interrumpir a otra de menos prioridad.

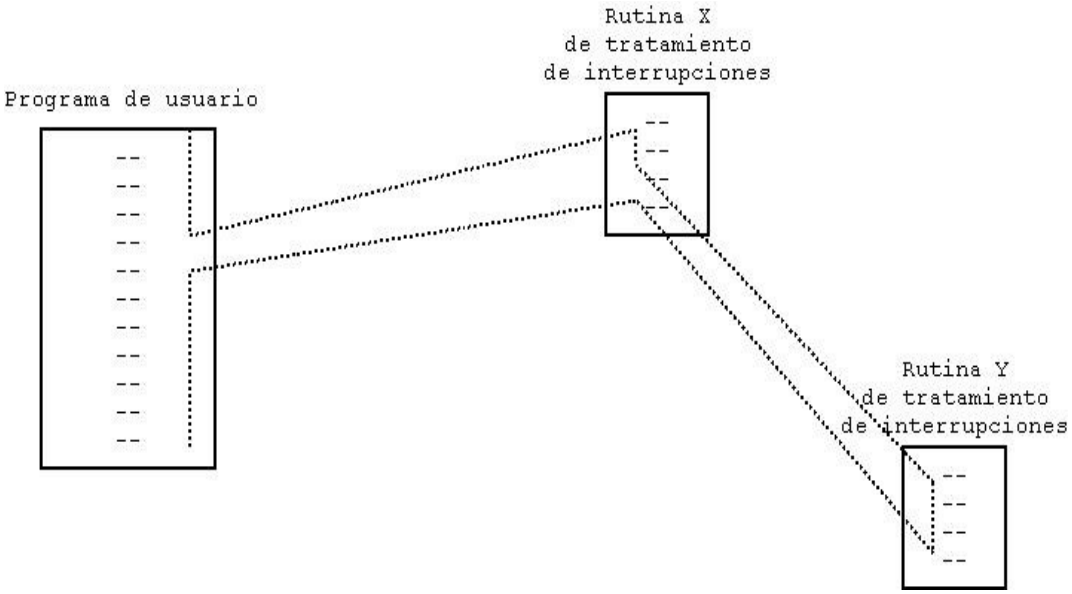


Figura II-17: Interrupciones múltiples habilitadas

9. Planificación del procesador

En un sistema multiprogramado tenemos varios procesos que se alternan en el uso del procesador y en esperar a realizar operaciones E/S u otro suceso.

La clave de la eficiencia de la multitarea es la planificación.

La planificación consiste en asignar procesos al procesador para que sean ejecutados a lo largo del tiempo cumpliendo una serie de objetivos.

9.1. Niveles de planificación

La actividad de planificación se divide en tres funciones o niveles independientes:

El planificador a largo plazo (planificador de trabajos)

- Determina cuáles son los programas admitidos al sistema, controla el grado de multiprogramación.
- Crea los procesos y los coloca en cola, sacándoles de la misma cuando puedan ser cargados en memoria
- Sólo existe cuando el sistema permite procesos por lotes
- Decide quién se ejecuta basándose en criterios de disponibilidad y necesidad de recursos.
- Su frecuencia de actividad es de varios minutos.
- Su tarea es pasar los procesos por lotes de inactivos a preparados, y deciden el grado de multiprogramación

El planificador a medio plazo (planificador de procesos suspendidos)

- Decide si el proceso debe ser suspendido para reducir el grado de multiprogramación y devolverlo después a memoria cuando sea necesario.
- Existe en sistemas de tiempo compartido y en aquellos que tengan gestión de memoria virtual o procesos intercambiables.
- Su frecuencia de actividad es de varios segundos.

El planificador a corto plazo (planificación del procesador)

- Tiene la responsabilidad de decidir cómo y cuándo un proceso que está preparado pasa a ejecutarse, realiza las funciones de multiprogramación.
- Siempre está en memoria, se suele ejecutar con mucha frecuencia (milisegundos) y es muy rápido.
- Es en este nivel donde nos preocupamos de dar un buen servicio a los procesos interactivos.

El planificador a corto plazo es el que vamos a estudiar.

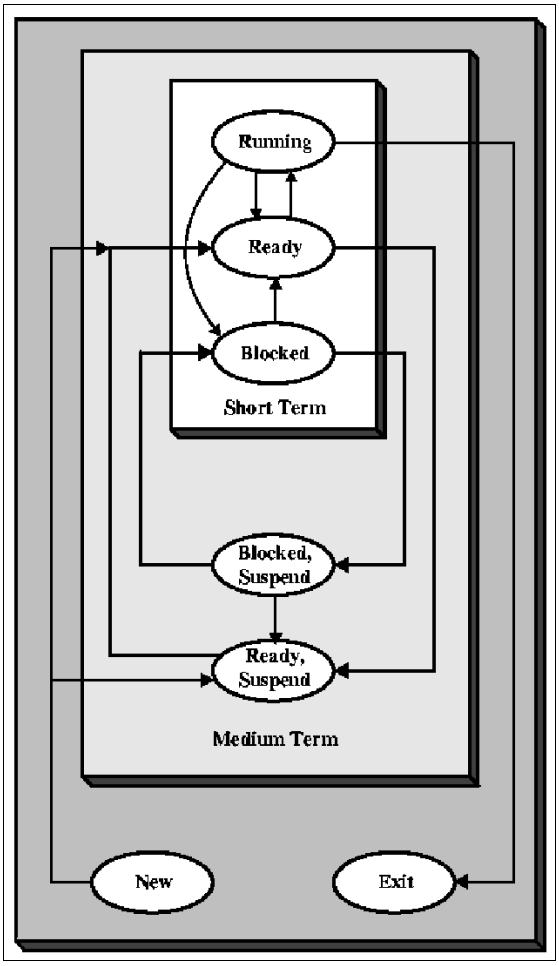


Figura II-18: Niveles de Planificación

9.2. Objetivos de la planificación

Cada planificador intenta alcanzar los siguientes objetivos:

- **Justicia**: reparto equitativo del procesador.
- **Máxima capacidad de ejecución**: que los trabajos se ejecuten lo más rápidamente posible (reducir el tiempo de respuesta), dando un servicio aceptable a todos ⇒ disminuir el número de cambios de proceso.
- **Máximo número de usuarios interactivos**: es primordial en sistemas de tiempo compartido. Requiere mucha memoria, un procesador rápido y una planificación ágil.
- **Minimizar la sobrecarga**: incide en el rendimiento del ordenador. Menos cambios de contexto ⇒ los procesos gozarán de mayor tiempo del procesador.
- **Equilibrio del uso de recursos**: se intenta que no haya bloqueos indeseados, consiguiendo que los recursos soporten una ocupación similar al mayor tiempo posible.
- **Seguridad de las prioridades**: favorecer a los procesos que tienen mayor prioridad.

Muchos de estos objetivos son contradictorios hay que llegar a un equilibrio.

Resumen: el S.O. debe ofrecer un buen rendimiento y un buen servicio.

9.3. Criterios de planificación

El diseñador estudiará el comportamiento del algoritmo según los criterios siguientes con objeto de observar si se alcanzan los objetivos fijados:

A) **Tiempo de respuesta**: velocidad con que el ordenador empieza a dar respuesta a una petición de usuario. No se tiene en cuenta porque depende de la E/S y otros factores. Es sólo un factor psicológico.

B) **Tiempo de servicio o retorno** : tiempo que transcurre entre que el usuario da la orden de cargar hasta que finaliza su ejecución.

$$T_{servicio}=T_{carga}+T_{preparado}+T_{tprocesador}+T_{bloqueado}$$

C) **Tiempo de ejecución**: representa el tiempo teórico que un proceso necesita para ejecutarse si estuviera solo.

$$T_{ejecucion}=T_{carga}+T_{tprocesador}+T_{bloqueado}$$

D) **Tiempo de procesador**: tiempo que el proceso hace uso del procesador.

E) **Tiempo de espera**: tiempos de estancia en las colas de bloqueados y preparados.

9.4. Medidas

Sea ***t* el tiempo de procesador**, t_i el momento de carga y t_f el momento de finalización de un proceso *p*.

Definimos las siguientes medidas:

- **Tiempo de Servicio**: $T_s = t_f - t_i$
- **Tiempo de Espera**: $T_e = T_s - t$
- **Índice de Servicio**: $I = t/T_s$. Es el porcentaje de tiempo que el proceso está usando el procesador con respecto a su tiempo de vida. Mide el rendimiento. Es adimensional y se suele expresar en %. Interpretación:
  - Para $I=1$ el proceso siempre estará ejecutándose (raro porque suelen necesitar E/S).
  - Para $I=0$ el proceso nunca se ejecuta.
  - Cuanto más alto sea I menos desperdicio se produce.
  - Si I está próximo a 1 decimos que el proceso está limitado por el procesador.
  - Si I está próximo a 0 decimos que el proceso está limitado por E/S.
- Otras medidas: tiempo para salvar el contexto, tiempo de reposición del contexto, tiempo de planificación, tiempo de cambio de proceso...

Para sistema multitarea con n procesos, usaremos los promedios:

- Tiempo medio de servicio. $Tms = \sum Ts / n$
- Tiempo medio de espera. $Tme = \sum Te / n$
- Índice medio de servicio $\mathrm{Im} = \sum I / n$

Siendo n el número de procesos.

9.5. Organización de la planificación

La tarea de pasar un proceso de preparado a ejecución (realizada por el núcleo) se divide en:

- Planificación de procesos (planificador): elige qué proceso entre los que están en la cola de preparados pasa a ejecución.
- Despachar un proceso (activador o despachador): entrega al procesador, extrayéndolo de la cola de procesos preparados, cambiando su estado y reponiendo el estado del procesador.

9.6. Algoritmos de planificación

Existen en los sistemas comerciales diversas políticas de planificación que son puestas en práctica por diversos algoritmos que son los que estudiaremos.

Nos basamos en un grupo de procesos que se deben ejecutar en el ordenador en los instantes dados en la siguiente tabla:

Proceso	t_i	$t_{ejecucion}$	Cola
A	0	3	1
B	1	5	1
C	4	2	2
D	5	6	3
E	8	4	3

De esta forma evaluando los diferentes algoritmos de planificación bajo la misma carga podemos comparar la 'bondad' de dichos algoritmos.

Suponemos que no se realiza E/S, con lo cual el tiempo de ejecución coincide con el tiempo de procesador

La unidad de tiempo es virtual, con lo cual no nos importa su valor, aunque el valor más representativo del comportamiento de la política se mide en % y es adimensional (Índice de Servicio).

Vamos a tener dos tipos de política de planificación:

- Apropiativas: Interrumpen la ejecución de un proceso con el fin de estudiar si puede seguir siendo ejecutado o por el contrario debe ser otro el que lo haga.
- No apropiativas: Un proceso nunca abandona el procesador una vez comenzada su ejecución.

9.6.1. Algoritmo primero en llegar, primero en servirse (FCFS – First Come First Served – FIFO – First In First Out)

Es una política **no apropiativa**, el proceso se ejecuta hasta que termine. Se basa única y exclusivamente en el orden de llegada de los procesos. Tras una E/S, el proceso se colocará al final de la cola de preparados.

Ejemplo:

- Cronograma o diagrama de ocupación:

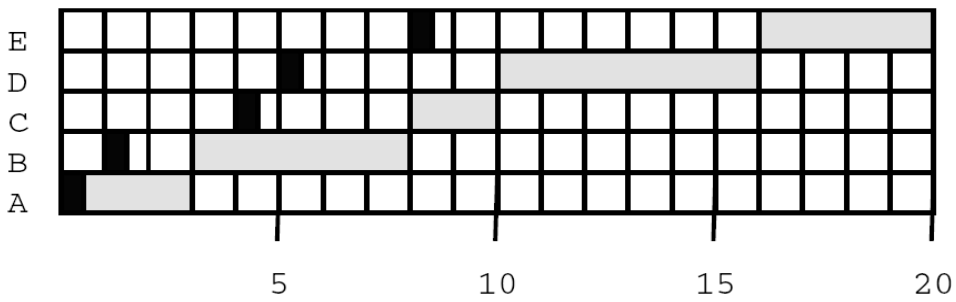


Figura II-19: Planificación FIFO

- Tabla de sucesos: como superíndice indicamos lo que le queda de ráfaga, el tiempo que usara la C.P.U hasta que realice la siguiente operación de E/S. Los instantes marcados en negrita indican que el planificador se activa.

t	Finalizados	Preparados	Ejecución
0		A^3	A^3
1		B^5	A^2
2		B^5	A^1
3	A	B^5	B^5
4	A	C^2	B^4
5	A	C^2, D^6	B^3
6	A	C^2, D^6	B^2
7	A	C^2, D^6	B^1
8	A, B	C^2, D^6, E^4	C^2
9	A, B	D^6, E^4	C^1
10	A, B, C	E^4	D^6
11	A, B, C	E^4	D^5
12	A, B, C	E^4	D^4
13	A, B, C	E^4	D^3
14	A, B, C	E^4	D^2
15	A, B, C	E^4	D^1
16	A, B, C, D	E^4	E^4
17	A, B, C, D		E^3
18	A, B, C, D		E^2
19	A, B, C, D		E^1
20	A, B, C, D, E		

Tabla de tiempos:

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.33
D	5	6	16	11	5	0.54
E	8	4	20	12	8	0.33
			Media	7.8	3.8	0.58

Ejemplo 2: con entrada/salida. Se considerará que si coinciden la llegada de un proceso y la finalización de la E/S de otro, la llegada del proceso ocurrirá antes.

Proceso	t_i	Ejecución	Cola
A	0	1 + E/S + 1	1
B	1	2 + 2 E/S + 1	1
C	4	2	2
D	5	3 + 2 E/S + 1	3
E	8	1 + 1 E/S + 2	3

- Cronograma o diagrama de ocupación:

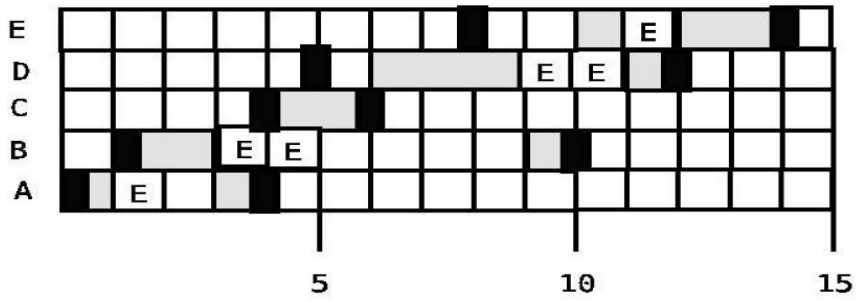


Figura II-20: Planificación FIFO con E/S

- Tabla de sucesos: como superindice indicamos lo que le queda de rafaga de C.P.U o E/S, como subindice lo que le queda para finalizar. Los instantes marcados en negrita indican que el planificador se activa.

t	Finalizados	Preparados	Bloqueados	Ejecución
0		A_3^1		A_3^1
1		B_5^2	A_2^1	B_5^2
2		A_1^1		B_4^1
3		A_1^1	B_3^2	A_1^1
4	A	C_2^2	B_2^1	C_2^2
5	A	D_6^3, B_1^1		C_1^1
6	A, C	D_6^3, B_1^1		D_6^3
7	A, C	B_1^1		D_5^2
8	A, C	B_1^1, E_4^1		D_4^1
9	A, C	B_1^1, E_4^1	D_3^2	B_1^1
10	A, C, B	E_4^1	D_2^1	E_4^1
11	A, C, B	D_1^1	E_3^1	D_1^1
12	A, C, B, D	E_2^2		E_2^2
13	A, C, B, D	E_2^2		E_1^1
14	A, C, B, D, E			E_1^1

- Tabla de tiempos:

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	2	4	4	2	2/4 =0,5
B	1	3	10	9	6	3/9 = 0,33
C	4	2	6	2	0	2/2 = 1
D	5	4	12	7	3	4/7 = 0,57
E	8	3	14	6	3	3/6 = 0,5
			Media	5,6	2,8	0,58

Ventajas:

- Es fácil de llevar a la práctica.
- Es justa.
- Es predecible.

Desventajas:

- Penaliza los procesos cortos, sobre todo cuando llegan detrás de uno largo. Las esperas siempre dependen de los procesos que estén en la cola, T_s no suele ser el deseado.

9.6.2. Algoritmo Round Robin (RR)

Se traduce como rueda o asignación cíclica (Round Robin).

Trata de ser más justa que la FIFO para procesos cortos.

Es apropiativa, concede el procesador a un proceso durante un quantum 'q' de tiempo (10-100 msg.), antes de devolverlo a la cola de procesos preparados.

La cola de procesos preparados se gestiona mediante una FIFO (aunque a veces se hace por prioridades).

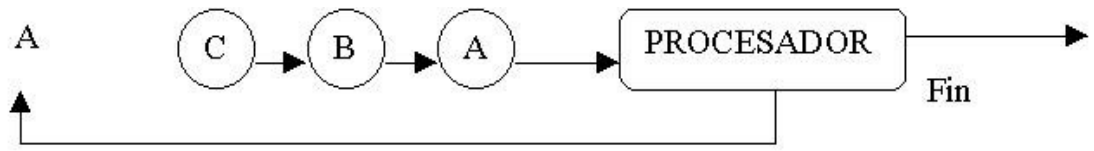


Figura II-21: Planificación Round Robin

¿Influye 'q' en el comportamiento?

- q grande ⇒ FIFO.
- q pequeño ⇒ sobrecarga.

Condiciones a tener en cuenta:

- Si un proceso finaliza en su quantum se le concede el procesador a otro por un quantum completo.
- Los procesos nuevos entran en la cola al final de preparados.
- Si un proceso llega al sistema justo al final de un quantum consideraremos que lo hace justo antes de acabar el quantum.

Ejemplo 1: q=1

- Cronograma o diagrama de ocupación:

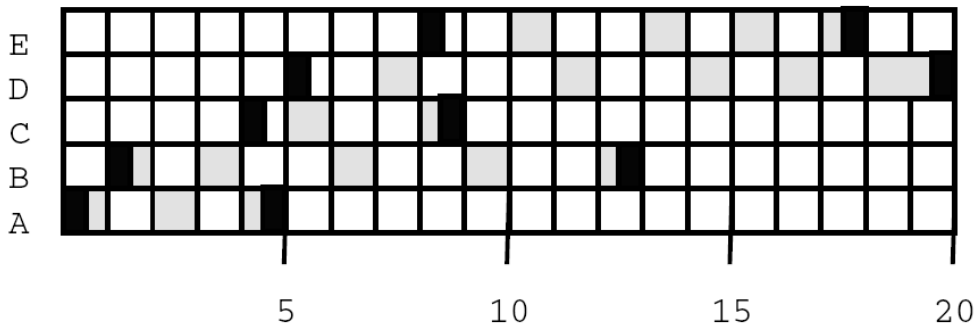


Figura II-22: Planificación Round Robin con q=1

- Tablas de sucesos: como superíndice indicamos el quantum que le queda, como subíndice lo que le queda por ejecutarse .

t	Finalizados	Preparados	Ejecución
0		A_3^1	A_3^1
1		B_5^1, A_2^1	B_5^1
2		A_2^1, B_4^1	A_2^1
3		B_4^1, A_1^1	B_4^1
4		A_1^1, C_2^1, B_3^1	A_1^1
5	A	C_2^1, B_3^1, D_6^1	C_2^1
6	A	$B_3^1, D_6^1, C_1^1,$	B_3^1
7	A	D_6^1, C_1^1, B_2^1	D_6^1
8	A	$C_1^1, B_2^1, E_4^1 D_5^1,$	C_1^1
9	A, C	$B_2^1, E_4^1 D_5^1$	B_2^1
10		E_4^1, D_5^1, B_1^1	E_4^1
11		D_5^1, B_1^1, E_3^1	D_5^1
12		B_1^1, E_3^1, D_4^1	B_1^1
13	A,C,B	E_3^1, D_4^1	E_3^1

t	Finalizados	Preparados	Ejecución
14	A,C,B	D_4^1, E_2^1	D_4^1
15	A,C,B	E_2^1 , D_3^1	E_2^1,
16	A,C,B	D_3^1, E_1^1	D_3^1
17	A,C,B	E_1^1, D_2^1	E_1^1
18	A, C, B, E	D_2^1	D_2^1
19	A,C,B,E	D_1^1	D_1^1
20	A,C,B,E,D		

- Tabla de tiempos:

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	5	5	2	0.6
B	1	5	13	12	7	0.42
C	4	2	9	5	3	0.40
D	5	6	20	15	9	0.40
E	8	4	18	10	6	0.40
			Media	9.5	5.4	0.44

Ejemplo 2: q=3

Proceso	t_i	t	Cola
A	0	3	1
B	1	5	1
C	4	2	2
D	5	6	3
E	8	4	3

- Cronograma o diagrama de ocupación :

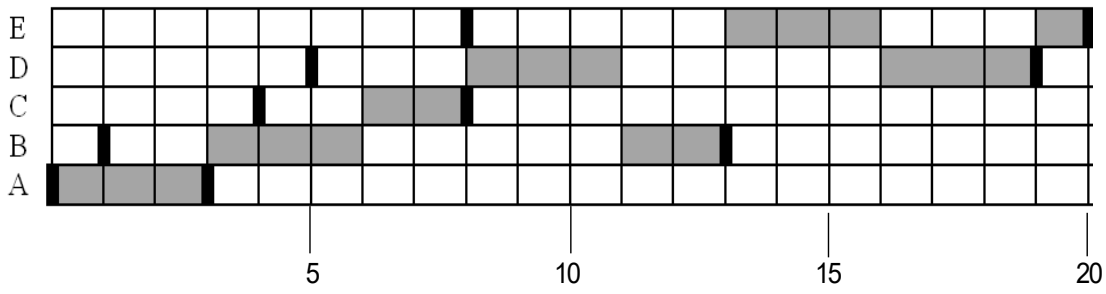


Figura II-23: Planificación Round Robin con q=3

- Tablas de sucesos, como superíndice indicamos el quantum que le queda, como subíndice lo que le queda por ejecutarse.

t	Finalizados	Preparados	Ejecución
0		A_3^3	A_3^3
1		B_5^3	A_2^2
2		B_5^3	A_1^1
3	A	B_5^3	B_5^3
4	A	C_2^3	B_4^2
5	A	C_2^3, D_6^3	B_3^1
6	A	C_2^3, D_6^3, B_2^3	C_2^3
7	A	D_6^3, B_2^3	C_1^2
8	A, C	D_6^3, B_2^3, E_4^3	D_6^3
9	A,C	B_2^3, E_4^3	D_5^2
10	A,C	B_2^3, E_4^3	D_4^1
11	A,C	B_2^3, E_4^3,D_3^3	B_2^3
12	A,C	E_4^3,D_3^3	B_1^2
13	A,C,B	E_4^3,D_3^3	E_4^3
14	A,C,B	D_3^3	E_3^2
15	A,C,B	D_3^3	E_2^1
16	A,C,B	D_3^3 , E_1^3	D_3^3
17		E_1^3	D_2^2
18		E_1^3	D_1^1
19	A,C,B,D	E_1^3	E_1^3
20	A,B,C,D,E		

- Tabla de tiempos:

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	13	12	7	0.42
C	4	2	8	4	2	0.50
D	5	6	19	14	8	0.43
E	8	4	20	12	8	0.33
			Media	9	5	0.54

Ejemplo 3: q=3 con entrada/salida

Proceso	t_i	Ejecución	Cola
A	0	1 + E/S + 1	1
B	1	2 + 2 E/S + 1	1
C	4	2	2
D	5	3 + 2 E/S + 1	3
E	8	1 + 1 E/S + 2	3

- Cronograma o diagrama de ocupación,

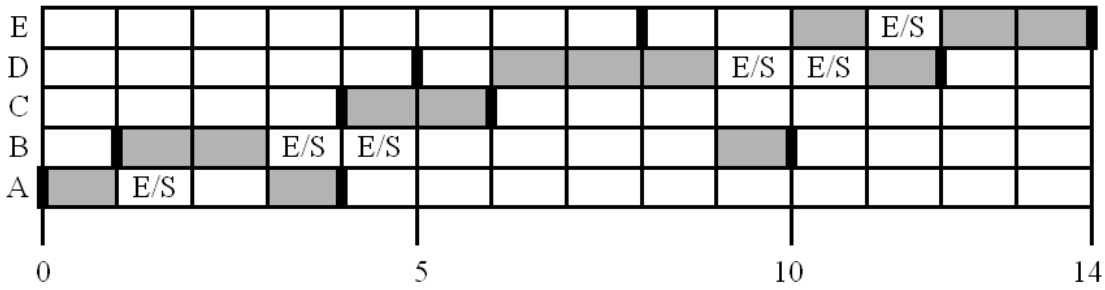


Figura II-24: Planificación Round Robin con q=3 y E/S

- Tabla de sucesos, como superíndice indicamos el quantum que le queda y la ráfaga, como subíndice lo que le queda por ejecutarse:

t	Finalizados	Preparados	Bloqueados	Ejecución
0		$A_3^{3,1}$		$A_3^{3,1}$
1		$B_5^{3,2}$	A_2^{1}	$B_5^{3,2}$
2		$A_1^{3,1}$		$B_4^{2,1}$
3		$A_1^{3,1}$	B_3^{2}	$A_1^{3,1}$
4	A	$C_2^{3,2}$	B_2^{1}	$C_2^{3,2}$
5	A	$D_6^{3,3}$, $B_1^{3,1}$		$C_1^{2,1}$
6	A, C	$D_6^{3,3}$, $B_1^{3,1}$		$D_6^{3,3}$
7	A, C			$D_5^{2,2}$
8	A, C	$B_1^{3,1}$, $E_4^{3,1}$		$D_4^{1,1}$
9	A, C	$B_1^{3,1}$, $E_4^{3,1}$	D_3^{2}	$B_1^{3,1}$
10	A, C, B	$E_4^{3,1}$	D_2^{1}	$E_4^{3,1}$
11	A, C, B	$D_1^{3,1}$	E_3^{1}	$D_1^{3,1}$
12	A, C, B, D	$E_2^{3,2}$		$E_2^{3,2}$
13	A, C, B, D			$E_1^{3,1}$
14	A, C, B, D, E			

- Tabla de tiempos:

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	2	4	4	2	2/4 = 0,5
B	1	3	10	9	6	3/9 = 0,33
C	4	2	6	2	0	2/2 = 1
D	5	4	12	7	3	4/7 = 0,57
E	8	3	14	6	3	3/6 = 0,5
			Media	5,6	2,8	0,58

Conclusiones:

- Con q=1 el tiempo de servicio permanece prácticamente constante.
- Con q=3 el tiempo de espera crece con la longitud de los procesos.

Propiedades de la política:

- Baja sobrecarga si el cambio de contexto es eficiente y los procesos están en memoria principal.
- Es la más usada para tiempo compartido.
- Ofrece un índice de servicio uniforme para todos los procesos.

9.6.3. El siguiente proceso el más corto (SJF – Shortest Job First)

Es una política **no apropiativa**.

Se escoge de la cola de procesos preparados el de menor **tiempo de ejecución**. En caso de igualdad por orden de llegada. Tras una entrada salida, el proceso que vuelve a la cola de preparados tiene la misma consideración que uno nuevo (se vuelve a calcular el tiempo restante).

¿Cómo conocer a priori el tiempo de ejecución de los procesos?

- Lo proporciona el usuario, poco fiable.
- Cada proceso se auto describe, difícil de conseguir ya que los procesos no siempre se ejecutan igual.
- Tomar la decisión de forma Heurística, basándose en información recabada estadísticamente. Sólo es posible en ambientes de ejecución muy repetitivos.

Ejemplo:

- Cronograma o diagrama de ocupación:

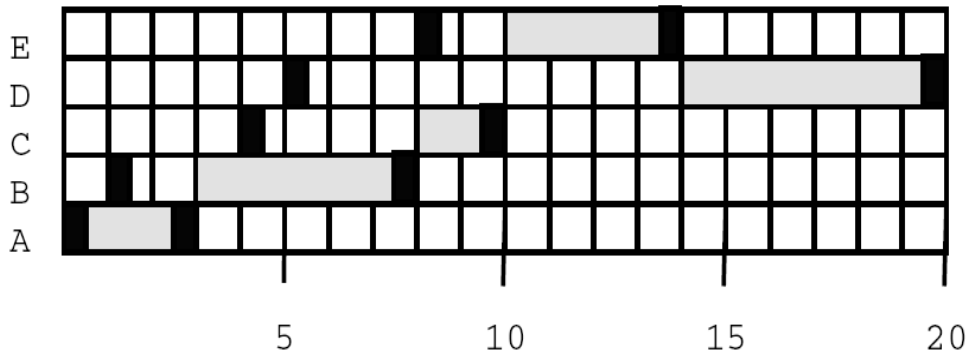


Figura II-25: Planificación SJF

- Tablas de sucesos, como superíndice indicamos el tiempo de ejecución que resta

t	Finalizados	Preparados	Ejecución
0		A^3	A^3
1		B^5	A^2
2		B^5	A^1
3	A	B^5	B^5
4	A	C^2	B^4
5	A	C^2, D^6	B^3
6	A	C^2, D^6	B^2
7	A	C^2, D^6	B^1
8	A, B	C^2, D^6, E^4	C^2
9	A, B	D^6, E^4	C^1
10	A, B, C	D^6, E^4	E^4
11	A, B, C, E	D^6	E^3
12	A, B, C, E	D^6	E^2
13	A, B, C, E	D^6	E^1
14	A, B, C, E	D^6	D^6
15	A, B, C, E		D^5
16	A, B, C, E		D^4
17	A, B, C, E		D^3
18	A, B, C, E		D^2
19	A, B, C, E		D^1
20	A, B, C, E, D		

- Tabla de tiempos:

Proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.33
D	5	6	20	15	9	0.40
E	8	4	14	6	2	0.67
			Media	7.4	3.4	0.62

Ejemplo 2 : con entrada/salida

Proceso	t_i	Ejecucion	Cola
A	0	1 + E/S + 1	1
B	1	2 + 2 E/S + 1	1
C	4	2	2
D	5	3 + 2 E/S + 1	3
E	8	1 + 1 E/S + 2	3

- Cronograma o diagrama de ocupación:

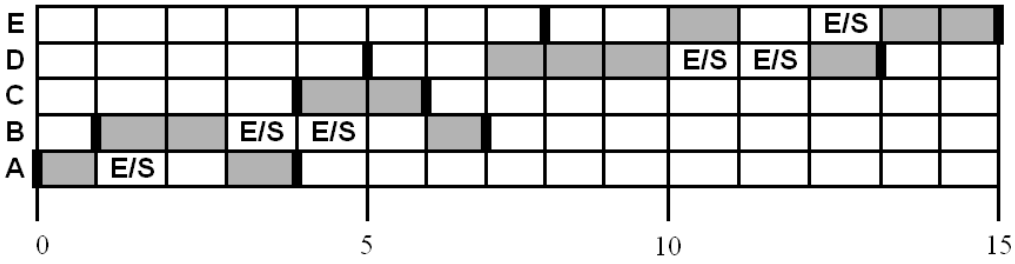


Figura II-26: Planificación SJF con E/S

- Tablas de sucesos, como superindice indicamos la ráfaga de C.P.U o E/S, como subindice el tiempo total y el que le queda.

t	Finalizados	Preparado	Bloqueados	Ejecución
0		$A_{3,3}^{1}$		$A_{3,3}^{1}$
1		$B_{5,5}^{2}$	$A_{3,2}^{1}$	$B_{5,5}^{2}$
2		$A_{3,1}^{1}$		$B_{5,4}^{1}$
3		$A_{3,1}^{1}$	$B_{5,3}^{2}$	$A_{3,1}^{1}$
4	A	$C_{2,2}^{2}$	$B_{5,2}^{1}$	$C_{2,2}^{2}$
5	A	$D_{6,6}^{3}$, $B_{5,1}^{1}$		$C_{2,1}^{1}$
6	A, C	$D_{6,6}^{3}$, $B_{5,1}^{1}$		$B_{5,1}^{1}$
7	A, C, B	$D_{6,6}^{3}$		$D_{6,6}^{3}$
8	A, C, B	$E_{4,4}^{1}$		$D_{6,5}^{2}$
9	A, C, B	$E_{4,4}^{1}$		$D_{6,4}^{1}$
10	A, C, B	$E_{4,4}^{1}$	$D_{6,3}^{2}$	$E_{4,4}^{1}$
11	A, C, B		$D_{6,2}^{1}$, $E_{4,3}^{1}$	NULO
12	A, C, B	$D_{6,1}^{1}$	$E_{4,3}^{1}$	$D_{6,1}^{1}$
13	A, C, B, D	$E_{4,2}^{2}$		$E_{4,2}^{2}$
14	A, C, B, D			$E_{4,2}^{2}$
15	A, C, B, D, E			

- Tabla de tiempos:

Proceso	t_i	t	t_f	T_s	T_e	I
A	0	2	4	4	2	2/4 = 0,5
B	1	3	7	6	3	3/6 = 0,5
C	4	2	6	2	0	2/2 = 1
D	5	4	13	8	4	4/8 = 0,5

E	8	3	15	7	4	3/7 = 0,43
			Media	5,4	2,6	0,59

Propiedades

- El tiempo de espera aumenta con la longitud de los procesos.
- Es poco predecible
- No es justa, favorece a los procesos cortos.
- Tiene un buen T_s (sobre todo para procesos cortos) e I.
- Es difícil de poner en práctica por la dificultad de conocer los tiempos de ejecución a priori.

9.6.4. Próximo proceso el de tiempo restante más corto (SRT – Shortest Remaining Time)

Es una política **apropiativa**.

Trata de conjugar la ventaja de la RR (apropiación) con la SJF (información del tiempo de los procesos)

Cambia el proceso que está en ejecución cuando un proceso entra en la cola de preparados con una exigencia de **tiempo de ejecución** menor que le queda al proceso que está en curso.

Ejemplo:

- Cronograma o diagrama de ocupación:

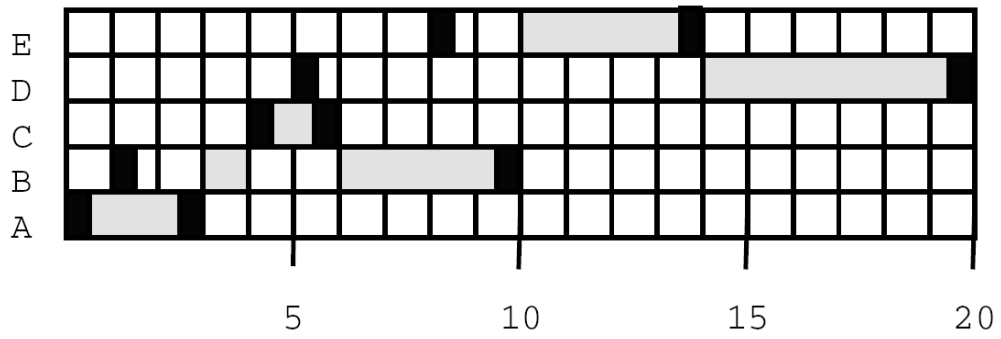


Figura II-27: Planificación SRT

- Tablas de sucesos

t	Finalizados	Preparados	Ejecución
0		A_3	A_3
1		B_5, A_2	A_2

2		B_5	A_1
3	A	B_5	B_5
4	*	C_2, B_4	C_2
5	*	B_4, D_6,C_1	C_1
6	A, C	B_4, D_6	B_4
7	A, C	D_6	B_3
8	A, C	D_6, E_4, B_2	B_2
9	A, C	D_6,E_4	B_1
10	A, C, B	D_6,E_4	E_4
11	A, C, B	D_6	E_3
12	A, C, B	D_6	E_2
13	A, C, B	D_6	E_1
14	A, C, B, E	D_6	D_6
15	A, C, B, E		D_5
16	A, C, B, E		D_4
17	A, C, B, E		D_3
18	A, C, B, E		D_2
19	A, C, B, E		D_1
20	A, C, B, E, D		

- Tabla de tiempos

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	10	9	4	0.55
C	4	2	6	2	0	1
D	5	6	20	15	9	0.4
E	8	4	14	6	2	0.67
			Media	7	3	0.72

Ejemplo 2: Con entrada salida

Proceso	t_i	Ejecucion	Cola
A	0	1 + E/S + 1	1
B	1	2 + 2 E/S + 1	1
C	4	2	2
D	5	3 + 2 E/S + 1	3
E	8	1 + 1 E/S + 2	3

- Cronograma o diagrama de ocupación:

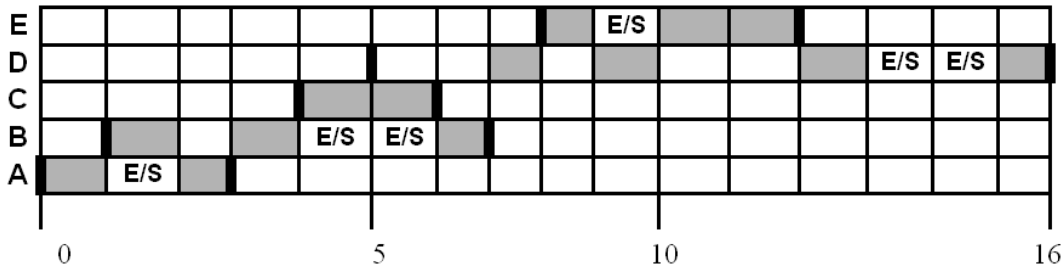


Figura II-28: Planificación SRT con E/S

- Tablas de sucesos. Como subíndice indicamos el tiempo de ejecución restante.

t	Finalizados	Preparados	Bloqueados	Ejecución
0		A_3		A_3
1		B_5	A_2	B_5
2		A_1 , B_4		A_1
3	A	B_4		B_4
4	A	C_2	B_3	C_2
5	A	D_6	B_2	C_1
6	A,C	D_6 ,B_1		B_1
7	A,C,B	D_6		D_6
8	A,C,B	D_5 ,E_4		E_4
9	A,C,B	D_5	E_3	D_5
10	A,C,B	E_2 ,D_4		E_2
11	A,C,B	D_4		E_1
12	A,C,B,E	D_4		D_4
13	A,C,B,E		D_3	NULO
14	A,C,B,E		D_2	NULO
15	A,C,B,E	D_1		D_1
16	A,C,B,E,D			

- Tabla de tiempos

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	2	3	3	1	2/3 = 0,66
B	1	3	7	6	3	3/6= 0,5
C	4	2	6	2	0	2/2 = 1
D	5	4	16	11	7	4/11 = 0,37
E	8	3	12	4	1	3 /4 = 0,75
			Media	5.2	2,4	0,66

Las **características** de la política son:

- Excelente I y un T_e muy bajo (la cola de preparados es lo más corta posible ⇒alto rendimiento.).
- Bajo Tiempo de Espera para la mayoría de los procesos.
- Puede ser injusta, incluso causar 'inanición'.
- Gran sobrecarga ya que hay que almacenar el tiempo de ejecución consumido por cada proceso.

9.6.5. Próximo el de más alto índice de respuesta (HRN – Highest Response Ratio Next)

Es una política **no apropiativa**.

Trata de corregir la desventaja de los procesos largos en el SJF.

Calcula la expresión de la prioridad interna (P) como:

$$P=\frac{w+t_e}{t_e}$$

donde w es el tiempo de espera en la cola de preparados y t_e es el tiempo de ejecución, cada vez que sea necesario planificar. Tras una E/S se considerará el tiempo de ejecución total, no el restante, y el tiempo de espera total en preparados.

Favorece a los procesos que más tiempo lleven en el sistema y a los más cortos.

Ejemplo:

- Cronograma o diagrama de ocupación:

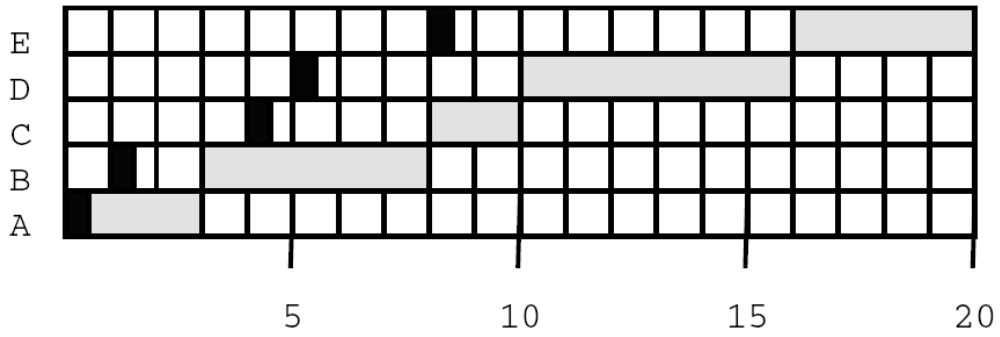


Figura II-29: Planificación HRN

Cálculo de la prioridad interna				
	C	D	E	Seleccionado
t=8	P=6/2=3	P=9/6=1.5	P=4/4=1	C
t=10		P=11/6=1.8	P=6/4=1.5	D

- Tablas de sucesos:

t	Finalizados	Preparados	Ejecución
0		A	A
1		B	A
3	A	B(1.4)	B
4	A	C	B
5	A	C,D	B
8	A,B	C(3),D(1.5), E(1)	C
10	A,B,C	D(1.8), E(1.5)	D
16	A,B,C,D	E(3)	E
20	A,C,B,E,D		

- Tabla de tiempos:

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	3	3	0	1
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.33
D	5	6	16	11	5	0.54
E	8	4	20	12	8	0.33
			Media	7.8	3.8	0.58

Propiedades:

- Un proceso corto tras uno largo espera mucho sólo si el largo ha comenzando su ejecución.
- Es justa.
- Es muy costosa al tener que calcular la prioridad interna P, lo que produce mucha sobrecarga

9.6.6. Colas Múltiples (MQ)

Se divide la cola de procesos preparados en varias colas, los procesos se asignan de forma **permanente** a una determinada.

Cada cola tendrá asociado un algoritmo de planificación.

¿Cuándo se pasa el control a cada cola? ⇒ algoritmo de planificación entre colas.

Ejemplo:

Tres colas con algoritmo RR de quantum = 1, 2 y 3 respectivamente.

El algoritmo entre colas es de prioridades apropiativo, la cola 1 es la más prioritaria

Proceso	t_i	t	Cola
A	0	3	1
B	1	5	1
C	4	2	2
D	5	6	3
E	8	4	3

- Cronograma o diagrama de ocupación:

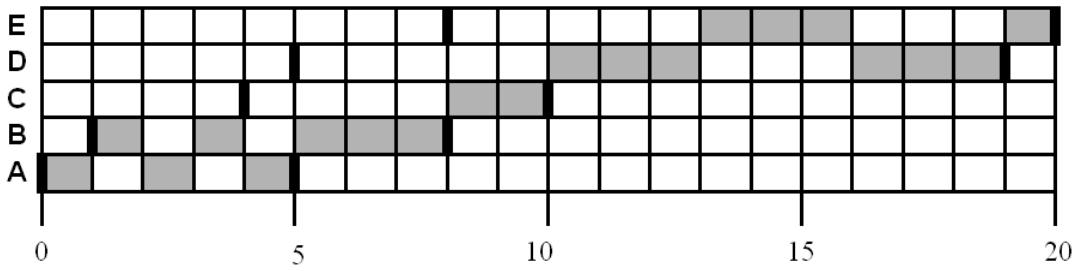


Figura II-30: Planificación con Colas Múltiples

- Tablas de sucesos:

t	Finalizados	Preparados			Ejecución
		C1	C2	C3	
0		A			A
1		B,A			B
2		A,B			A
3		B,A			B
4		A,B	C		A
5	A	B	C	D	B
6	A	B	C	D	B
7	A	B	C	D	B
8	A,B		C	D,E	C
10	A,B,C			D,E	D
13	A,B,C			E,D	E
16	A,B,C			D,E	D
19	A,B,C,D			E	E
20	A,B,C,D,E				

- Tabla de tiempos:

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	5	5	2	0.6
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.3
D	5	6	19	14	8	0.43
E	8	4	20	12	8	0.33
			Media	8.8	4.8	0.47

9.6.7. Realimentación de colas múltiples (FB)

Si queremos tratar de formas justa a los procesos necesitamos conocer su longitud, memoria, si está limitado por proceso o por E/S, etc...

Nos conformamos con asumir las siguientes reglas:

- Favorecer a los procesos cortos.
- Favorecer a los procesos limitados por E/S, porque así ocupan los recursos y dejan libres el procesador para procesos limitados por proceso.
- Determinar la naturaleza del trabajo a realizar para poder realizar su planificación.

Se divide la cola de procesos preparados en varias, de la forma cola1, cola2, cola3, ... de forma que la cola con número mayor tiene asignada una prioridad menor.

Para cada cola se le concede al proceso un determinado tiempo de procesador, de forma que si excede su tiempo se pasa a la cola de nivel inmediatamente inferior (con esto vamos disminuyendo su prioridad mientras más tiempo está en el sistema)

El tiempo que cada proceso está en una cola de un determinado nivel vendrá dado como parámetro de la política.

El tiempo de la última fila será infinito.

NOTA: en colas múltiples, si un proceso de una cola menos prioritaria, es interrumpido por la llegada de un proceso a una cola de prioridad mayor, tomaremos como criterio, que se volverá a evaluar la situación de la cola menos prioritaria aplicando el criterio de la cola.

Parámetros:

- Número de colas a usar.
- Algoritmo de planificación de cada cola.
- Método de paso de una cola a otra.
- Cola en la que comienza cada proceso.
- Algoritmo de planificación entre las distintas colas.

Se intenta dar un trato justo, separando los procesos por categoría.

Los procesos limitados por procesador terminarán en las colas del nivel más bajo. Dejando en las de mayor prioridad los procesos más interactivos.

Características de la política:

- Soporta bien cargas de procesos altas. Los procesos se reparten entre las colas.
- Es adaptable a las necesidades del sistema (Según gestionemos cada cola)
- Es apropiativa entre colas.

Ejemplo:

Existen tantas colas como sean necesarias de tipo RR con q=1.

Cada proceso puede consumir un quantum en cada nivel.

Todos los procesos empiezan por la cola de más alta prioridad.

- Cronograma o diagrama de ocupación:

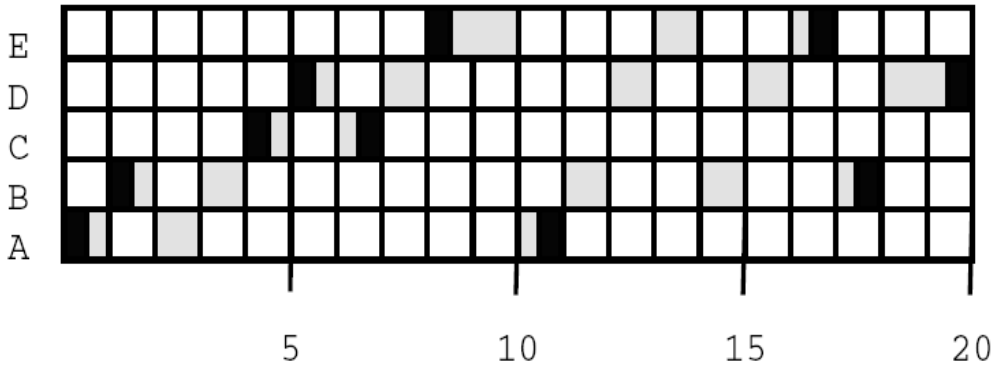


Figura II-31: Planificación con Realimentación de Colas Múltiples

- Tabla de sucesos:

t	Finalizados	Preparados						Ejecución
		C1	C2	C3	C4	C5	C6	
0		A						A
1		B	A					B
2			A, B					A
3			B	A				B
4		C		A, B				C
5		D	C	A, B				D
6			C, D	A, B				C
7	C		D	A, B				D
8	C	E		A, B, D				E
9	C		E	A, B, D				E
10	C			A, B, D, E				A
11	C, A			B, D, E				B
12	C, A			D, E	B			D
13	C, A			E	B, D			E
14	C, A				B, D, E			B
15					D, E	B		D
16	C, A				E	B, D		E
17	C, A, E					B, D		B
18	C, A, E, B					D		D
19	C, A, E, B						D	D
20	C, A, E, B, D							

- Tabla de tiempos:

Nombre proceso	t_i	t	t_f	T_s	T_e	I
A	0	3	11	11	8	0.27
B	1	5	18	17	12	0.29
C	4	2	7	3	1	0.67
D	5	6	20	15	9	0.40
E	8	4	17	9	5	0.44
			Media	11	7	0.42

9.6.8. La prioridad y los algoritmos de planificación

Hasta ahora no hemos tenido en cuenta las prioridades externas.

Normalmente los sistemas desdoblan las colas en tantas como prioridades puedan alcanzar los procesos (de la forma en que lo hacen las colas múltiples)

El problema de las prioridades es la postergación indefinida o inanición que pueden sufrir ciertos procesos. Para evitarlo se usa la técnica del 'envejecimiento de prioridades', mediante la cual el proceso va ganando prioridad conforme más tiempo está en el sistema.

Estos algoritmos de envejecimiento de prioridades pueden ser además apropiativos o no apropiativos, según las acciones que tomen con el proceso que está en ejecución en un momento dado.

9.7. Planificación en POSIX

Especifica una serie de políticas de planificación aplicable a procesos y procesos ligeros.

Cada unidad ejecutable lleva asociada una política y una prioridad, que pueden ser modificados.

Cada política de planificación lleva asociada un rango de prioridades (al menos 32 niveles). Se elegirá siempre al proceso o hilo con la prioridad más alta.

Las políticas disponibles en POSIX son:

- FIFO modificada:
  - El proceso que es expulsado por otro de mayor prioridad pasa a ser el primero de su cola.
  - Un desbloqueo ⇒ último de su cola.
  - Cambio de prioridad o política ⇒ replanificación. Si tras esta el proceso resulta expulsado ⇒ último de su cola.
- Cíclica (Round-Robin).
  - Fin quantum ⇒ final de la cola de su prioridad.

- El proceso que es expulsado por otro de mayor prioridad pasa a ser el primero de su cola pero con el resto de quantum que aún no había consumido.

- Otra: Esta tercera política es opcional y dependerá de la implementación que se realice de la norma POSIX.

Implementación LINUX del estándar POSIX

Soporta tres tipos de planificación, dos algoritmos de TR (tiempo real) y un algoritmo de TC (tiempo compartido).

Las prioridades de los procesos en TR son siempre mayores que las de TC.

Las políticas de planificación son:

- Las de los procesos en TR son las definidas en el estándar POSIX con prioridades estáticas. Los niveles de prioridad van desde 1 a 99.

- La política de TC es la tercera política de planificación del estándar POSIX, y está implementada mediante una RR con prioridades dinámicas. La prioridad estática es 0. Su funcionamiento es el siguiente:

  - Todo proceso tiene asignada una prioridad base (dinámica).

  - Cada vez que se produce la interrupción del reloj se resta una unidad a la prioridad del proceso en ejecución.

  - Se elige al proceso más prioritario de la cola.

  - Si llega un momento en que todos los procesos preparados tienen una prioridad igual a 0 se produce un reajuste de las prioridades.

  - La nueva prioridad se calcula dividiendo por 2 la actual (redondeando por exceso) y sumando la prioridad base ⇒ procesos preparados vuelven a su prioridad base y bloqueados aumentan (prioridad mayor a la de los que ya estaban)

  - ¿Qué ocurre cuando un proceso del tipo 'tiempo compartido' es expulsado cuando llega un proceso en tiempo real? No está documentado.

9.8. Planificación Windows

Se utiliza un algoritmo de planificación Round Robin apropiativo basado en prioridades.

La unidad de planificación son las hebras. Siempre se ejecuta la hebra más prioritaria. Si llega a preparado una hebra más prioritaria que la que está en ejecución, ésta es desalojada.

Una hebra en ejecución sólo abandona ese estado si:

- Es desalojada por una hebra de mayor prioridad

- Termina

- Concluye su quantum de tiempo

- Se bloquea

La parte del kernel de Windows que gestiona la planificación se llama despachador. El despachador usa un esquema de prioridades de 32 niveles.

Las prioridades se dividen en dos clases:

- clase variable: prioridades de 1 a 15 (cambian arriba o abajo pero nunca a prioridades mayores de 15)
- clase de tiempo real: prioridades de 16 a 31 (son fijas, nunca cambian)
- Existe una hebra (usada para gestión de memoria) con prioridad 0

Hay una cola para cada prioridad.

Si no hay ninguna hebra preparada se ejecuta una hebra especial llamada hebra inactiva.

La prioridad inicial de un hilo de la clase de prioridad variable se determina en base a dos cantidades:

- la prioridad base del proceso (0-15)
- la prioridad base del hilo (prioridad base del proceso $\pm$ 2)

Ejemplo: Un proceso con prioridad base 4 y uno de sus hilos con prioridad base –1 quiere decir que el hilo tiene prioridad inicial de 3. Esta prioridad es dinámica y puede fluctuar dentro de unos límites (Nunca por debajo del rango inferior de la prioridad base del hilo (2) y nunca por encima de 15.

Si un hilo agota su quantum se baja su prioridad.

Si un hilo se interrumpe para esperar por una E/S se sube su prioridad.

Los hilos limitados por procesador tienden a prioridades menores y los hilos limitados por E/S tienden a prioridades mayores.

Dentro de los hilos limitados por E/S se sube más la prioridad a los que realizan esperas interactivas (teclado, pantalla) que a los que realizan otro tipo de esperas de E/S (disco), por lo que los hilos interactivos tienden a tener mayores prioridades.

9.9. Planificación clásica en UNIX

Está diseñado para entornos de TC aunque SVR4 también cubres las necesidades del TR.

Emplea realimentación multinivel usando RR con, q = 1 sg., en cada una de las colas de prioridad.

¿Cómo se calcula la prioridad?

$$CPU_j(i) = CPU_j \frac{(i-1)}{2}$$

$$P_j(i) = Base_j + CPU_j(i) + Nice_j$$

donde $CPU_j(i)$es la media ponderada de la utilización de la C.P.U del proceso j en el intervalo i, $P_j(i)$ es la prioridad del proceso j al principio del intervalo i, $Base_j$ es la prioridad base del proceso j y $nice_j$ es un factor de ajuste controlado por el usuario.

El propósito de la prioridad base es dividir los procesos en bandas fijas de prioridad.

Los valores de *C.P.U* y *nice* están restringidos para impedir que el proceso salga de la banda asignada.

Las bandas que se definen, en orden decreciente de prioridad son:

- Intercambio
- Control de dispositivos de E/S de bloques.
- Gestión de archivos.
- Control de dispositivos de E/S de caracteres.
- Procesos de usuario.

Esta jerarquía garantiza la utilización eficiente de los dispositivos de E/S.

Dentro de la banda de usuario se trata de penalizar a los procesos con carga en el procesador a costa de los procesos con carga de E/S.

10. Comunicación y sincronización entre procesos

Los procesos interaccionan entre si.

Normalmente son controlados por los programadores para beneficiarse de la concurrencia.

Los procesos deben sincronizarse cuando se van a usar recursos compartidos.

Los procesos deben ser ellos mismos los que se encarguen de sincronizar sus operaciones, aunque como ayuda a los usuarios los S.O. multitarea y algunos lenguajes de programación proporcionan una colección de primitivas de sincronización entre procesos

Hay tres formas de interacción entre procesos:

- Sincronización entre procesos: protocolos y mecanismos usados para preservar la integridad y consistencia del sistema.
- Señalización entre procesos: intercambio de señales de sincronización usados para coordinar su progreso colectivo
- Comunicación entre procesos: los procesos se comunican con propósitos tales como intercambiar datos, acumular resultados colectivos, ... (Si lo hacen mediante memoria compartida deben sincronizar sus accesos a dicha memoria compartida).

La concurrencia da lugar a un aumento de la productividad si se implementa bien, pero puede degradar la fiabilidad cuando el sistema se contamina con sincronizaciones indeseadas.