Procesos y Sincronización

Proceso

Un proceso es un concepto manejado por el sistema operativo que consiste en el conjunto formado por:

- Las instrucciones de un programa destinadas a ser ejecutadas por el microprocesador.
- Su estado de ejecución en un momento dado, esto es, los valores de los registros de la CPU para dicho programa.
- Su memoria de trabajo, es decir, la memoria que ha reservado y sus contenidos.
- Otra información que permite al sistema operativo su planificación.

Los procesos son creados y destruidos por el sistema operativo, así como también este se debe hacer cargo de la comunicación entre procesos, pero lo hace a petición de otros procesos.

El mecanismo por el cual un proceso crea otro proceso se denomina bifurcación (*fork*).

Los nuevos procesos son independientes y no comparten memoria (es decir, información) con el proceso que los ha creado.

Políticas de Planificación

Se utilizan por el Sistema Operativo para tomar las decisiones, que cambian el estado de un proceso.

Planificaciones a Largo Plazo (‘PLP’)

- Decide que procesos llegan al estado ‘Listo’.
- Este tipo de política mide el grado de multiprogramación
- La PLP requiere un algoritmo muy complejo que se ejecuta cada cierto tiempo.

Planificaciones a Medio Plazo (‘PMP’)

- Controla el intercambio de procesos entre la memoria principal y la secundaria
- Su ejecución se realiza con más frecuencia que la ‘PLP’ por lo que su tiempo de ejecución será menor.

Planificaciones a Corto Plazo (‘PCP’)

- Controla cuando un proceso comienza su ejecución, y cuando debe finalizar.
- Este algoritmo debe ser muy simple, pues el proceso se ejecuta muy frecuentemente.

Planificaciones de un Procesador

- Para comparar los distintos algoritmos de planificación se deben establecer una serie de criterios que permitan esta comparación:

1) El Uso de la CPU:

- Mide el porcentaje de tiempo que el procesador pasa ejecutando los procesos

- Valores Adecuados [40% - 90%]

- Valores Imposibles [> 90%]

- Valores Catastróficos [< 40%]

2) La Productividad:

- Es el número de trabajos realizados por unidad de tiempo

3) El Tiempo de Retorno:

- El tiempo que el proceso pasa en el sistema
- Cuanto menos mejor.

4) El Tiempo de Espera:

- El tiempo que el proceso gasta en estado de espera (Sin hacer nada).

5) El Tiempo de Retorno Normalizado:

- TRN = Tretorno / Tservicio

- Permite realizar comparaciones absolutas
- Por lo que el ‘TRN’ es relativo.

6) El Tiempo de Respuesta Interactivo:

- Es el tiempo que pasa desde que el sistema interacciona con el usuario.
- Tiempo desde que el usuario ejecuta una aplicación, y el programa responde.

7) La Prioridad:

- El procesador muestra más prioridad en unos procesos que en otros.

FÓRMULAS:

- Tret = Tfin – Tinicio
- Tret = Tcpu + Tespera + Te/s
- Tservicio = Tcpu + Te/s

Algoritmos de Planificación

Existen dos categorías

Apropiativos

El Sistema Operativo puede expulsar del procesador un proceso en ejecución.

No Apropiativos

Estos procesos, no pueden ser expulsados por el Sistema Operativo.

Clases de algoritmos

Algoritmo de planificación FCFS(*first come, first served*)

Algoritmo de planificación SJF (*shortest job first)*

Algoritmo de planificación SRT - Shortest Remaining Time

Estados de un proceso

El principal trabajo del procesador es ejecutar las instrucciones de máquina que se encuentran en memoria principal. Estas instrucciones se encuentran en forma de programas. Para que un programa pueda ser ejecutado, el sistema operativo crea un nuevo proceso, y el procesador ejecuta una tras otra las instrucciones del mismo.

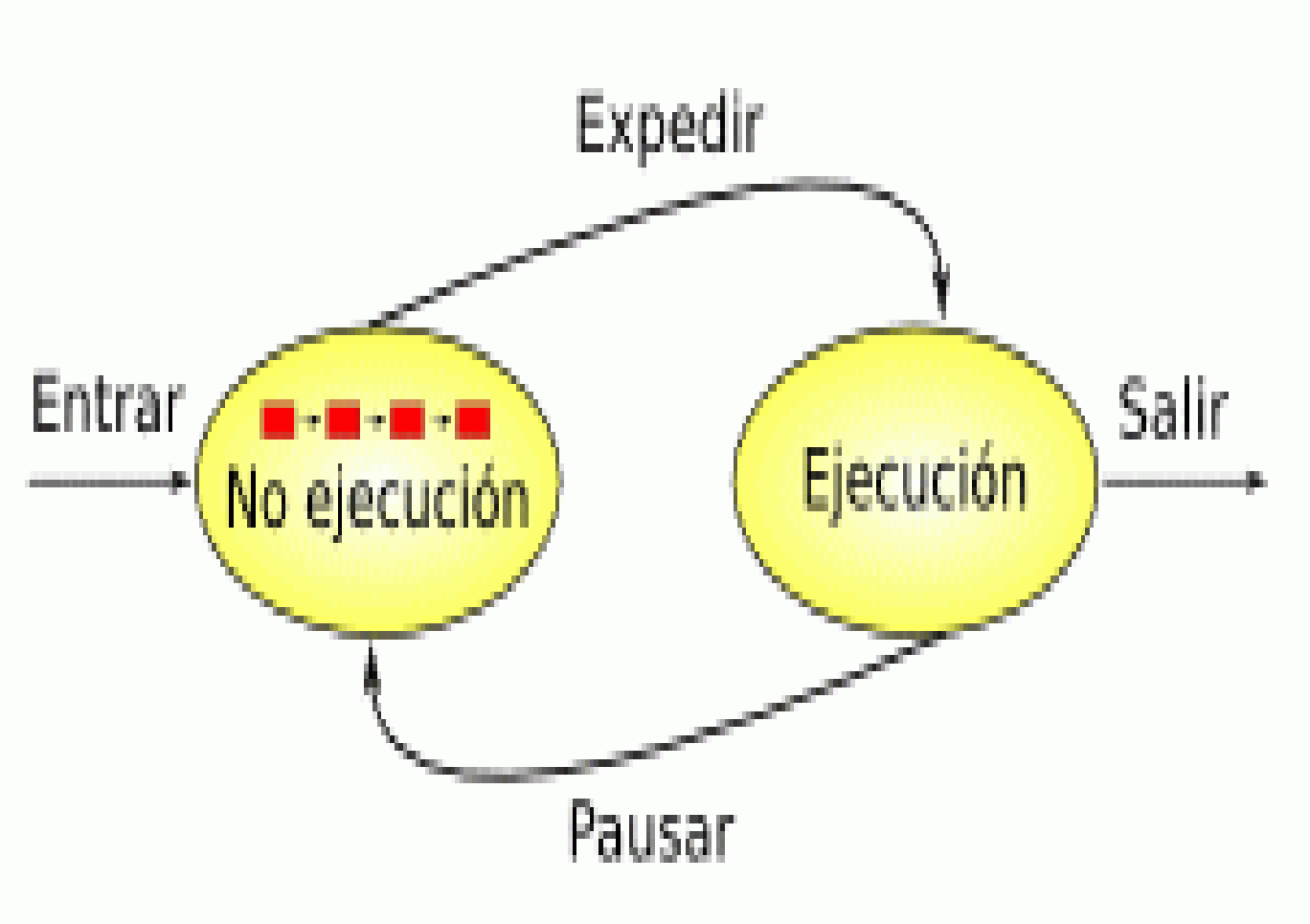
En un entorno de multiprogramación, el procesador intercalará la ejecución de instrucciones de varios programas que se encuentran en memoria. El sistema operativo es el responsable de determinar las pautas de intercalado y asignación de recursos a cada proceso.

Modelo de dos estados

El modelo de estados más simple es el de dos estados. En este modelo, un proceso puede estar ejecutándose o no. Cuando se crea un nuevo proceso, se pone en estado de *No ejecución*. En algún momento el proceso que se está ejecutando pasará al estado *No ejecución* y otro proceso se elegirá de la lista de procesos listos para ejecutar para ponerlo en estado *Ejecución*.

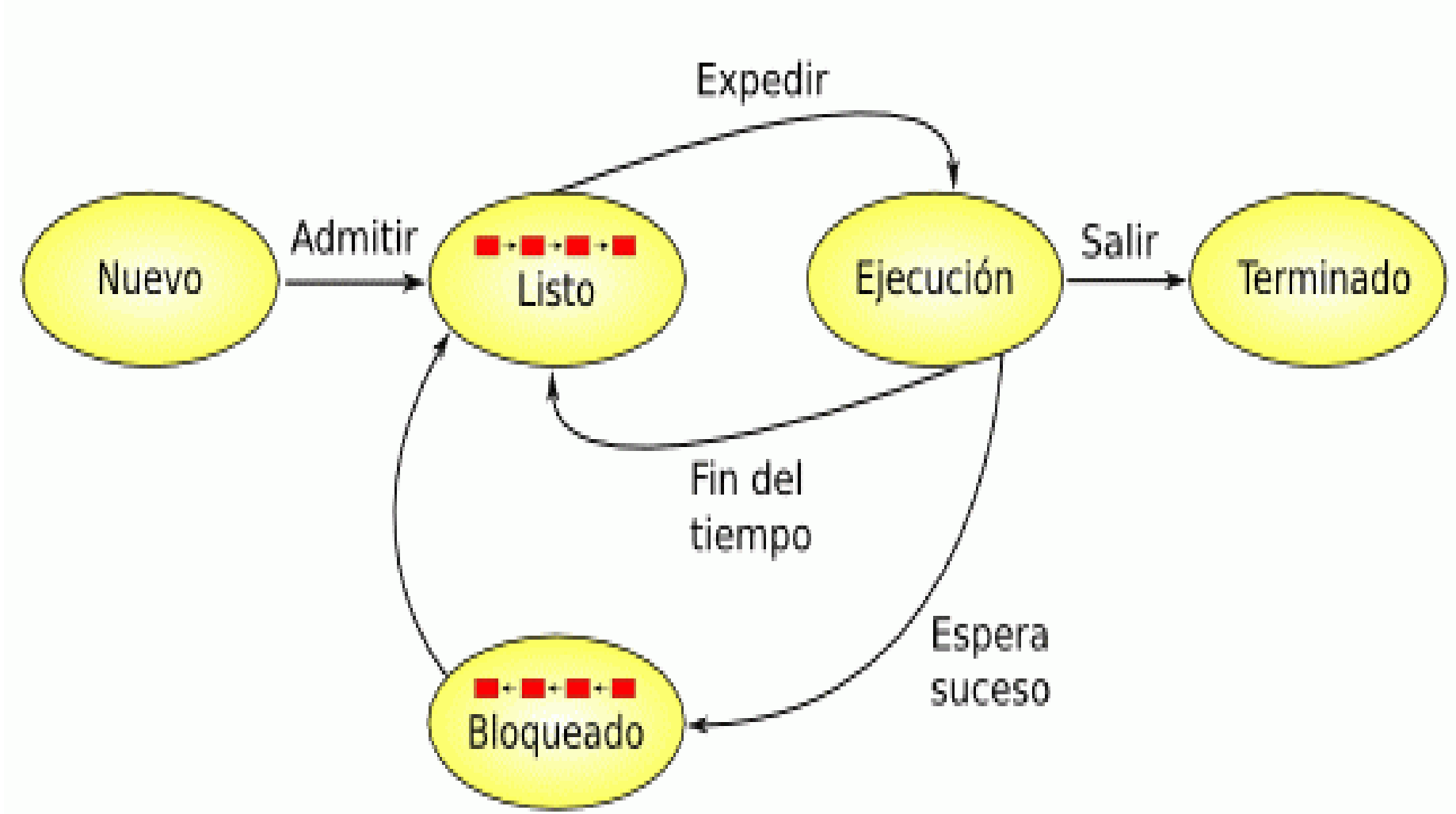
Es necesario que el sistema operativo conozca el estado de los procesos y el lugar que ocupa en memoria.

Además los procesos que no se están ejecutando deben guardarse en algún tipo de cola mientras esperan su turno para ejecutar.


Expedir
Entrar
No ejecución
Ejecución
Salir
Pausar

Modelo de cinco estados

El modelo anterior de dos estados funcionaría bien con una cola FIFO y planificación por turno rotatorio para los procesos que no están en ejecución, si los procesos estuvieran siempre listos para ejecutar. En la realidad, los procesos utilizan datos para operar con ellos, y puede suceder que no se encuentren listos, o que se deba esperar algún suceso antes de continuar, como una operación de Entrada/Salida. Es por esto que se necesita un estado donde los procesos permanezcan bloqueados esperando hasta que puedan proseguir. Se divide entonces al estado *No ejecución* en dos estados: *Listo* y *Bloqueado*. Se agregan además un estado *Nuevo* y otro *Terminado*.


Expedir
Nuevo
Admitir
Listo
Ejecución
Salir
Terminado
Fin del
tiempo
Espera
suceso
Bloqueado

Los cinco estados de este diagrama son los siguientes:

- **Ejecución:** el proceso está actualmente en ejecución.

- **Listo:** el proceso está listo para ser ejecutado, sólo está esperando que el planificador así lo disponga.
- **Bloqueado:** el proceso no puede ejecutar hasta que no se produzca cierto suceso, como una operación de Entrada/Salida.
- **Nuevo:** El proceso recién fue creado y todavía no fue admitido por el sistema operativo. En general los procesos que se encuentran en este estado todavía no fueron cargados en la memoria principal.
- **Terminado:** El proceso fue expulsado del grupo de procesos ejecutables, ya sea porque terminó o por algún fallo, como un error de protección, aritmético, etc.

Los nuevos estados *Nuevo* y *Terminado* son útiles para la gestión de procesos. En este modelo los estados *Bloqueado* y *Listo* tienen ambos una cola de espera. Cuando un nuevo proceso es admitido por el sistema operativo, se sitúa en la cola de listos. A falta de un esquema de prioridades ésta puede ser una cola FIFO. Los procesos suspendidos son mantenidos en una cola de bloqueados. Cuando se da un suceso se pasan a la cola de listos los procesos que esperaban por ese suceso.

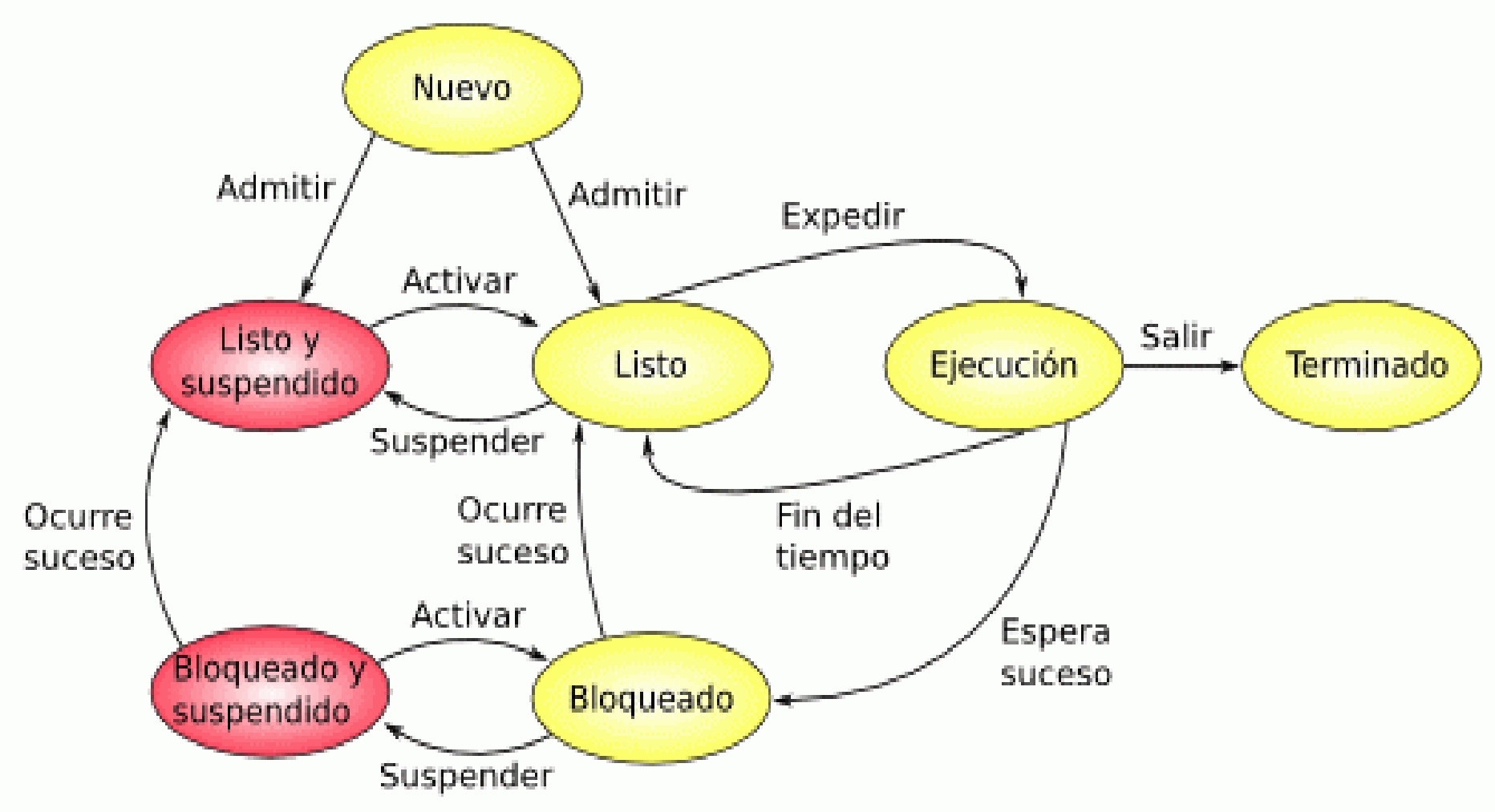
Si existe un esquema con diferentes niveles de prioridad de procesos es conveniente mantener varias colas de procesos listos, una para cada nivel de prioridad, lo que ayuda a determinar cuál es el proceso que más conviene ejecutar a continuación.

Procesos suspendidos

Una de las razones para implementar el estado *Bloqueado* era poder hacer que los procesos se puedan mantener esperando algún suceso, por ejemplo una Entrada/Salida. Sin embargo, al ser mucho más lentas estas operaciones, puede suceder en nuestro modelo de cinco estados todos los procesos en memoria estén esperando en el estado *Bloqueado* y que no haya más memoria disponible para nuevos procesos. Podría conseguirse más memoria, aunque es probable que esto sólo permita procesos más grandes y no necesariamente nuevos procesos. Además hay un costo asociado a la memoria y de cualquier forma es probable que se llegaría al mismo estado con el tiempo.

Otra solución es el **intercambio**. El intercambio se lleva a cabo moviendo una parte de un proceso o un proceso completo desde la memoria principal al disco, quedando en el estado *Suspendido*. Después del intercambio, se puede aceptar un nuevo proceso o traer a memoria un proceso suspendido anteriormente.

El problema que se presenta ahora es que puede ser que si se decide traer a memoria un proceso que está en el estado *Suspendido*, el mismo todavía se encuentre bloqueado. Sólo convendría traerlo cuando ya está listo para ejecutar, esto implica que ya aconteció el suceso que estaba esperando cuando se bloqueó. Para tener esta diferenciación entre procesos suspendidos, ya sean listos como bloqueados, se utilizan cuatro estados: *Listo*, *Bloqueado*, *Bloqueado y suspendido* y *Listo y suspendido*.


Nuevo
Admitir
Admitir
Expedir
Activar
Listo y suspendido
Listo
Ejecución
Salir
Terminado
Suspender
Ocurre suceso
Ocurre suceso
Fin del tiempo
Activar
Espera suceso
Bloqueado y suspendido
Bloqueado
Suspender

Procesos en espera

Dos o más procesos pueden cooperar mediante señales de forma que uno obliga a detenerse a los otros hasta que reciban una señal para continuar.

- Se usa una variable llamada semáforo para intercambiar señales.
- Si un proceso esta esperando una señal, se suspende (WAIT) hasta que la señal se envíe (SIGNAL).
- Se mantiene una cola de procesos en ESPERA en el semáforo.
- La forma de elegir los procesos de la cola en ESPERA es mediante una política FIFO.

La sincronización explícita entre procesos es un caso particular del estado "bloqueado". En este caso, el suceso que permite desbloquear un proceso no es una operación de entrada/salida, sino una señal generada a propósito por el programador desde otro proceso.

Operaciones sobre procesos

Los sistemas operativos son responsables de la "gestión de procesos y memoria", por lo que están encargados de realizar una serie de actividades, tales como la planificación o itineración de procesos, la operación sobre procesos y la comunicación entre procesos. Para **operar sobre un proceso**, los sistemas operativos actuales suministran ciertas funciones, que pueden ser ejecutadas ya sea desde el mismo proceso o desde el intérprete de comandos, si es un usuario el que solicita algún servicio de dicho sistema.

Entre las operaciones sobre procesos que con mayor frecuencia ofrecen los sistemas operativos se encuentran la creación, terminación o destrucción, suspensión y reanudación de procesos. Actualmente, en la mayoría de los S.O., los procesos pueden ejecutarse de forma concurrente, pudiéndose crear y eliminar de forma dinámica, por lo que es necesario que estos sistemas brinden un mecanismo para la creación y terminación de procesos.

Creación de procesos

Durante la ejecución de un proceso, éste puede crear otros procesos. Si lo hace, el proceso creador es llamado **proceso padre**, y el creado, **proceso hijo**. Estos último pueden, a su vez, crear otros procesos, construyendo así un árbol de procesos.

Hay 4 eventos comunes que conducen a la creación de procesos:

- En un ambiente *batch*, un proceso es creado en respuesta al sometimiento a ejecución de un trabajo (job). El S.O. tomará el próximo trabajo a ser ejecutado y creará el respectivo proceso.
- En un ambiente interactivo, un proceso es creado cuando un nuevo usuario entra al sistema (log on). El S.O crea un proceso *shell* que espera las órdenes del usuario.
- El S.O. puede crear un proceso para que realice una función en respuesta a una petición de un programa usuario, sin que el usuario tenga que esperar. Por ejemplo, si un usuario quiere que se imprima un archivo, el S.O. puede crear un proceso que maneje esta impresión, mientras que el proceso que realizó la petición puede continuar independientemente del tiempo requerido para completar la impresión.

Para procesos de modularidad o para explotar paralelismo, donde un programa usuario ya existente puede crear un conjunto de procesos hijos.

Además, crear un proceso implica varias operaciones, entre ellas:

- Buscarle un ***identificador***: cada proceso (padre e hijo) es identificado por un id o PID (process identifier) distinto para cada uno.
- Determinar la ***prioridad*** inicial del proceso.
- Crear el **Process Control Block (PCB)**: el S.O. busca un hueco libre en la lista de PCB, y cuando lo encuentra crea el PCB del proceso.
- Asignar ***recursos*** iniciales al proceso: por ejemplo, establecer su contexto de memoria (espacio de direcciones) y de CPU (registros), los archivos a usar y los dispositivos de E/S.
- Insertarlo en la **cola de procesos**.

Terminación de procesos

Un proceso acaba cuando termina de ejecutar su último enunciado y le pide al S.O. que lo elimine. Destruir o terminar un proceso implica eliminarlo del sistema: se le borra de las tablas o listas de procesos, sus recursos reutilizables se devuelven al sistema y su PCB se borra, es decir, el espacio de memoria ocupado por su PCB se devuelve al espacio de memoria disponible. Como es de esperar, el término de un proceso es más difícil cuando éste ha creado otros procesos. En algunos sistemas un proceso hijo se destruye automáticamente cuando su padre ha terminado, creando **terminaciones en cascada**, que son iniciadas normalmente por el sistema operativo. En otros sistemas, los procesos creados son independientes de sus padres, por lo que la destrucción de éste no afecta sobre sus hijos.

Algunas de las razones por la que se termina un proceso son:

- Un proceso hijo sobrepasa el límite de recursos que le fueron asignados.

- Las tareas de un proceso ya no son necesarias.
- El padre finaliza, por lo que su hijo también.
- La ocurrencia de un error irrecuperable o no controlado.
- Excede el tiempo límite.
- Errores aritméticos.
- Fallas de E/S.
- Instrucciones inválidas.

Suspensión y reanudación de procesos

Un proceso suspendido o bloqueado no puede proseguir sino hasta ser reanudado por otro proceso. Generalmente, es el S.O. el que se encarga de eliminar temporalmente ciertos procesos con el fin de reducir la carga del sistema durante una situación crítica. Normalmente, los procesos son suspendidos durante periodos pequeños para disminuir la exigencia al sistema, pero también hay casos en que ocurren suspensiones más largas, en las que es posible liberar los recursos del proceso para poder utilizarlos durante ésta. Según la naturaleza del recurso, se toma la decisión de liberarlo o no: la memoria principal debe ser liberada de inmediato cuando se suspenda un proceso; una unidad de cinta puede ser retenida brevemente por un proceso suspendido, pero debe ser liberada si el proceso se suspende por un periodo largo o indefinido. Reanudar (o activar) un proceso implica reiniciarlo a partir del punto en el que se suspendió.

La suspensión y reanudación de procesos son importantes y han sido puestas en práctica en diferentes sistemas. Estas operaciones pueden ser de gran necesidad en casos como:

- Si un sistema está funcionando mal, y es probable que falle, se puede suspender los procesos activos para reanudarlos cuando se haya corregido el problema.
- Un usuario que desconfíe de los resultados parciales de un proceso puede suspenderlo (en lugar de abortarlo) hasta que verifique si el proceso funciona correctamente o no.

Algunos procesos se pueden suspender como respuesta a las fluctuaciones a corto plazo de la carga del sistema, y reanudarse cuando las cargas vuelvan a niveles normales.

Investigue que son y cómo se resuelven

Problema del barbero durmiente

Problema de la cena de los filósofos