

4 PROCESOS

- 1 CONCEPTO DE PROCESO
- 2 ITINERACIÓN DE PROCESOS
- 3 OPERACIONES SOBRE PROCESOS
- 4 PROCESOS COOPERATIVOS
- 5 COORDINACIÓN ENTRE PROCESOS

4.1 CONCEPTO DE PROCESO

- 1 El Proceso
- 2 Estado del Proceso
- 3 Bloque de Control de Proceso

4.1.1 El Proceso

- Un sistema operativo ejecuta varios programas:
 - En Sistemas Batch : jobs
 - En Sistemas de tiempo compartido : programas de usuario o tareas
- Job y proceso se usan en forma equivalente.
- Proceso: un programa en ejecución; la ejecución del proceso debe progresar de manera secuencial.
- Un proceso incluye:
 - Contador de programa (program counter)
 - Stack
 - Sección de datos

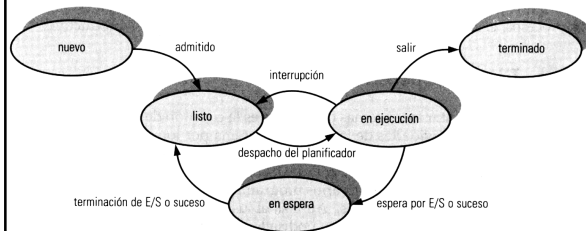
4.1.2 Estado del Proceso

A medida que un proceso se ejecuta, va cambiando de estado:

- **Nuevo** (New): El proceso está siendo creado.
- **Ejecución** (Running): Sus instrucciones están siendo ejecutadas.
- **Espera** (Waiting): El proceso está esperando a que un evento ocurra (generalmente la completación de una operación de E/S).
- **Listo** (Ready): El proceso está esperando que le sea asignado el procesador.
- **Terminado** (Terminated): El proceso ha finalizado su ejecución.

4.1.2 Estado del Proceso

Diagrama de Estados de un Proceso



4.1.3 Bloque de Control de Proceso

El **PCB** contiene información asociada con cada proceso.

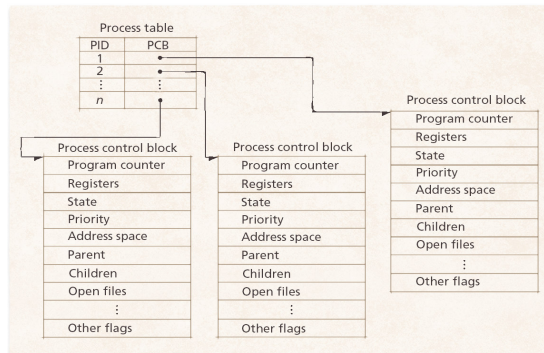
- Estado
- Contador de Programa
- Registros de la CPU
- Itineración de la CPU
- Administración de la Memoria
- Contabilidad (accounting)
- Estado de la E/S

4.1.3 Bloque de Control de Proceso

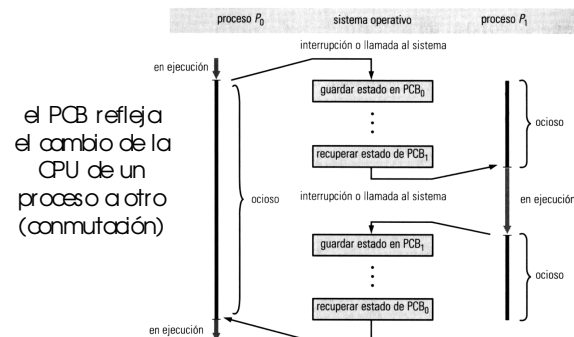
Bloque de Control de Proceso (PCB)

puntero	estado del proceso
número del proceso	
contador de programa	
registros	
límites de memoria	
lista de archivos abiertos	
•	
•	

4.1.3 Bloque de Control de Proceso



4.1.3 Bloque de Control de Proceso



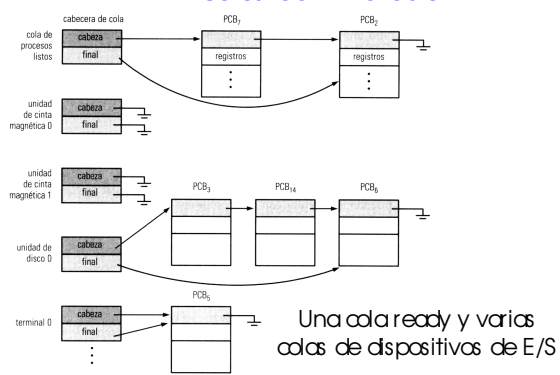
4.2 ITINERACIÓN DE PROCESOS

- 1 Colas de Itineración
- 2 Itineradores
- 3 Cambio de Contexto
- 4 Procesamiento de Interrupciones

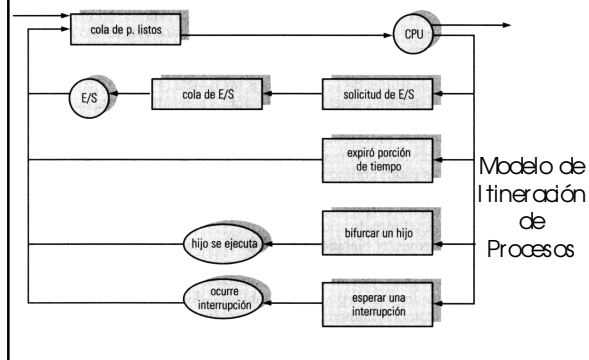
4.2.1 Colas de Itineración

- **Colas de jobs:** conjunto de todos los procesos en el sistema.
- **Cola Ready:** conjunto de todos los procesos que residen en memoria principal, listos para ejecutarse.
- **Colas de Dispositivos:** conjunto de todos los procesos que están esperando por un dispositivo de E/S.
- Migración de procesos entre las distintas colas.

4.2.1 Colas de Itineración

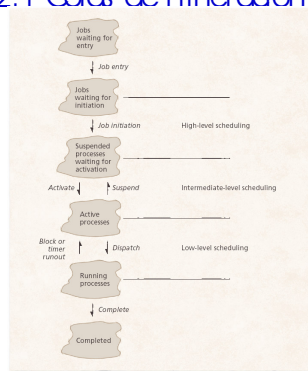


4.2.1 Colas de Itineración



4.2.1 Colas de Itineración

Niveles de Itineración



4.2.2 Itineradores

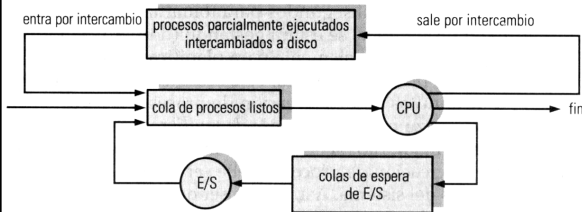
- Itinerador de **largo plazo** (o itinerador de jobs): selecciona qué procesos serán traídos a la cola ready.
- Itinerador de **corto plazo** (o itinerador de la CPU): selecciona qué proceso (de la cola ready) será ejecutado a continuación y le asigna la CPU.

4.2.2 Itineradores

- El itinerador de corto plazo es invocado muy frecuentemente (miliseg) $\Rightarrow$ (debe ser rápido).
- El itinerador de largo plazo es invocado poco frecuentemente (seg, min) $\Rightarrow$ (puede ser lento).
- El itinerador de largo plazo controla el grado de multiprogramación.
- Los procesos pueden ser descritos ya sea
 - Limitados por la E/S: ocupan más el tiempo haciendo E/S que cálculos; tienen muchas y muy pequeñas ráfagas de CPU (CPU Bursts).
 - Limitados por la CPU: ocupan más el tiempo haciendo cálculos; tienen pocas y extensas ráfagas de CPU.

4.2.2 Itineradores

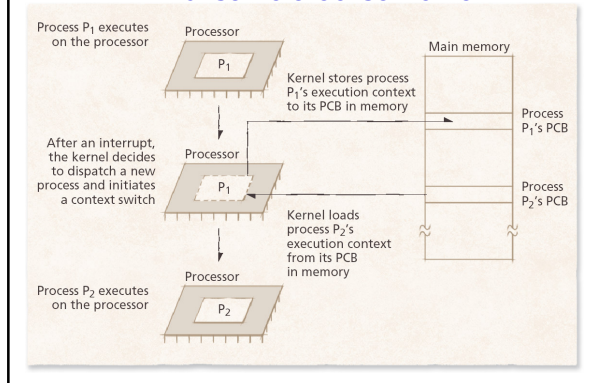
Agregación de un Itinerador de Mediano Plazo



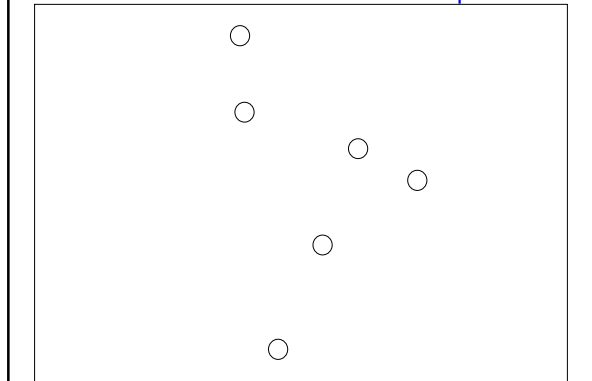
4.2.3 Cambio de Contexto

- Cuando la CPU cambia a otro proceso, el sistema debe salvar el estado del antiguo proceso y cargar el estado (previamente salvado) del nuevo proceso.
- El tiempo de cambio de contexto es *overhead*; el sistema no hace trabajo útil mientras hace el cambio.
- El tiempo depende del soporte de hardware.

4.2.3 Cambio de Contexto



4.2.4 Procesamiento de Interrupciones



4.3 OPERACIONES SOBRE PROCESOS

- 1 Creación
- 2 Terminación

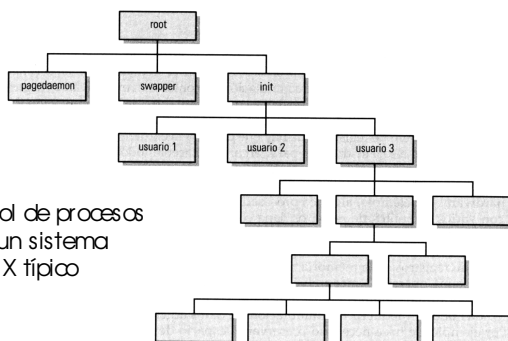
4.3.1 Creación de un Proceso

- Un proceso padre crea procesos hijos, los cuales, a su vez, crean otros procesos, formando un árbol de procesos.
- La compartición de recursos puede ser :
 - Padre e hijos comparten todos los recursos.
 - Los hijos comparten un subconjunto de los recursos del padre.
 - Padre e hijos no comparten los recursos.
- La ejecución puede ser :
 - Padre e hijo se ejecutan concurrentemente.
 - El padre espera a que los hijos terminen.

4.3.1 Creación de un Proceso

- En cuanto al espacio de direcciones:
 - El hijo es una duplicación del padre.
 - El hijo tiene su propio espacio de direcciones.
- Ejemplos de UNIX
 - La llamada al sistema *fork* crea un nuevo proceso.
 - La llamada al sistema *execute* es usada después de *fork* para reemplazar el espacio de memoria del proceso con un nuevo programa.

4.3.1 Creación de un Proceso



Árbol de procesos en un sistema UNIX típico

4.3.2 Término de un Proceso

- El proceso ejecuta su última sentencia y solicita al SO ser eliminado (salida normal o exit).
 - Envía los datos desde el tipo al padre (vía wait).
 - Los recursos del proceso son desasignados por el sistema operativo.
- Un padre puede terminar la ejecución de los procesos hijos (término anormal o abort), porque:
 - El hijo ha excedido los recursos asignados.
 - La tarea asignada al hijo ya no es requerida.
 - El padre está terminando.
 - * El sistema operativo no permite que un hijo continúe si su padre termina.
 - * Terminación en cascada.

4.4 PROCESOS COOPERATIVOS

- Un proceso **independiente** no puede afectar o ser afectado por la ejecución de otro proceso.
- Un proceso **cooperativo** puede afectar o ser afectado por la ejecución de otro proceso.
- Ventajas de los procesos cooperativos:
 - Compartir información
 - Aumento en la velocidad de computación
 - Modularidad
 - Conveniencia
- En el paradigma de los procesos cooperativos; un proceso *productor* produce información, la cual es consumida por un proceso *consumidor*.

4.5 HEBRAS (THREADS)

- Una hebra (o *proceso liviano*) es una unidad básica de utilización de CPU; consiste de:
 - program counter
 - registros
 - stack
- Una hebra comparte con sus pares:
 - sección de código
 - sección de datos
 - recursos del sistema operativoEs conocido colectivamente como tarea (*task*).
- Un proceso tradicional o *pesado* es igual a una tarea con una sola hebra.

4.5 HEBRAS (T HREADS)

- En una tarea con múltiples hebras, mientras una está bloqueada, otra de la misma tarea puede ejecutarse.
- La cooperación de múltiples hebras en el mismo job da un throughput y rendimiento mayor.
- Las hebras proveen un mecanismo que permite a procesos secuenciales hacer llamadas al sistema con bloqueo y también lograr paralelismo.
- Hebras soportadas por kernel (Mach y OS/2).
- Hebras de nivel de usuario; por encima del kernel, via llamadas a biblioteca a nivel de usuario (Andrew).
- Híbrido que implementa ambos tipos (Solaris).

4.5 HEBRAS (T HREADS)

Múltiples
hebras dentro
de una tarea

