SISTEMAS OPERATIVOS
Manejo de procesos

Amilcar Meneses Viveros
ameneses@computacion.cs.cinvestav.mx

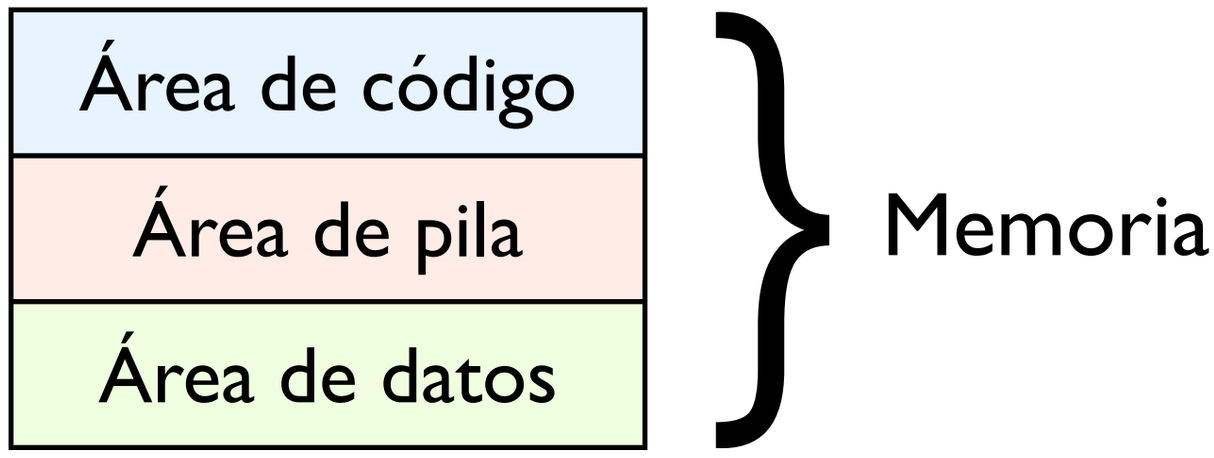
Universidad de Occidente

Presentación

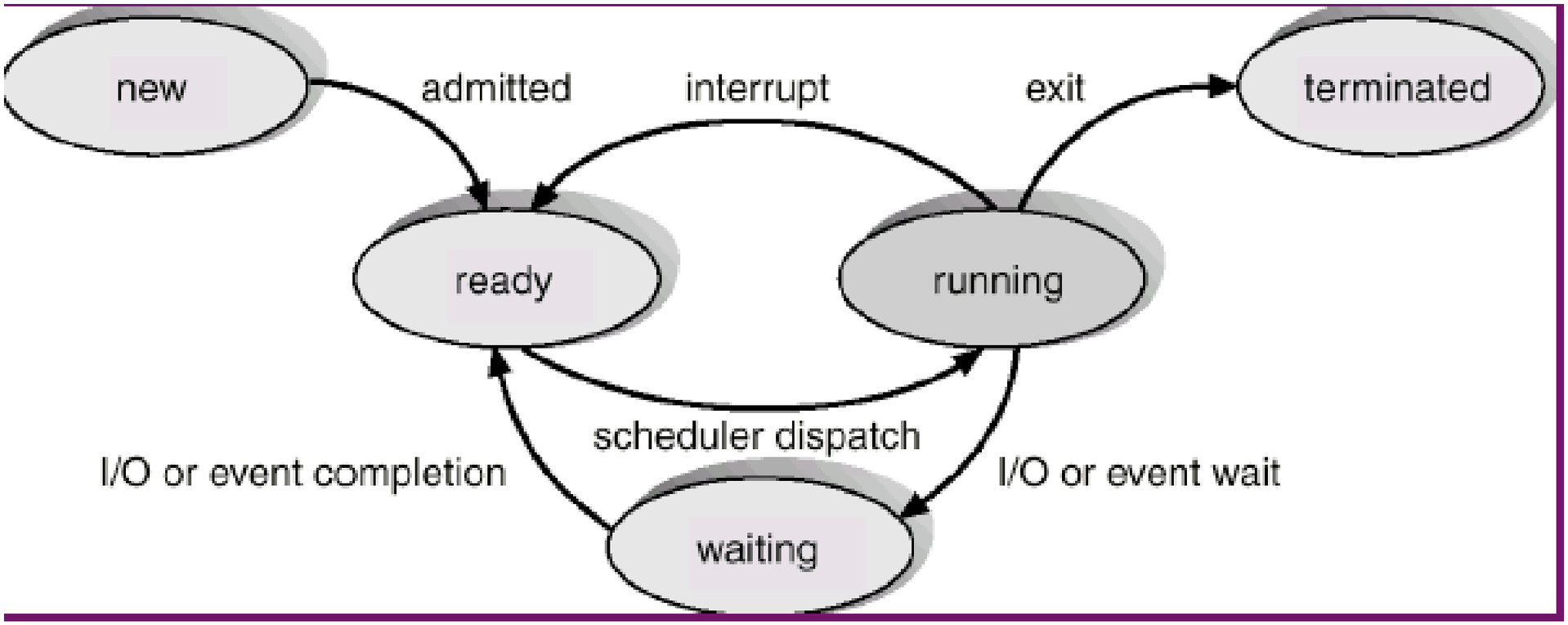
- Concepto de proceso
- Despacho de procesos
- Operaciones sobre procesos
- Procesos cooperativos
- Comunicación entre procesos
- Comunicación en sistemas cliente servidor

Concepto de proceso

- Un SO ejecuta una variedad de programas
  - Sistema por lotes: trabajo
  - Sistema de tiempo compartido: programas de usuario o tareas
- Proceso: programa en ejecución
- Partes de un proceso



Concepto de proceso



Estados de un proceso

Concepto de proceso

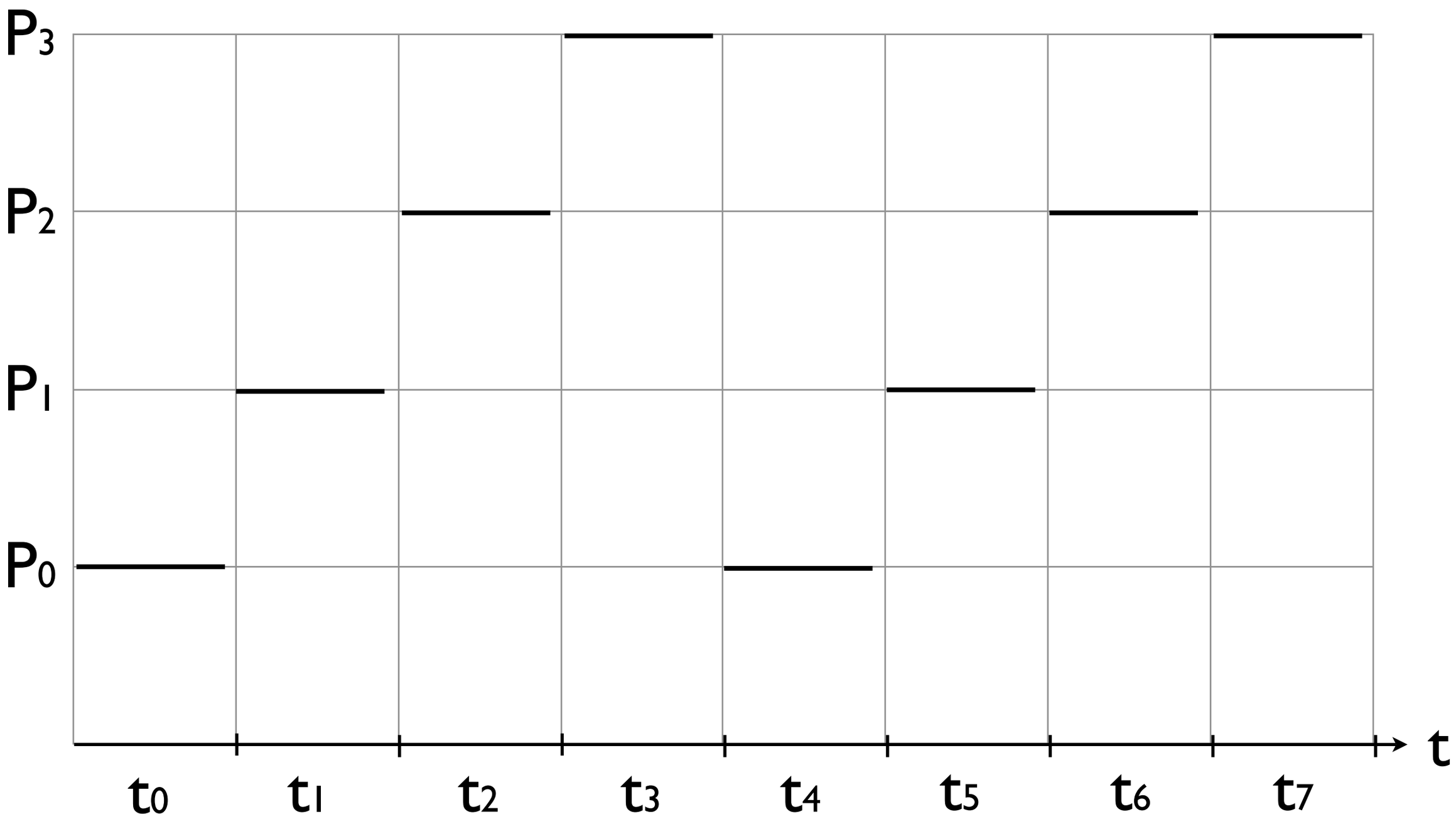
Bloque de control de procesos (PCB)

- Estado del proceso
- Contador de programa (PC/IP)
- Registros del CPU
- Información de despacho
- Información del manejo de memoria
- Información de conteo
- Información de estado de E/S

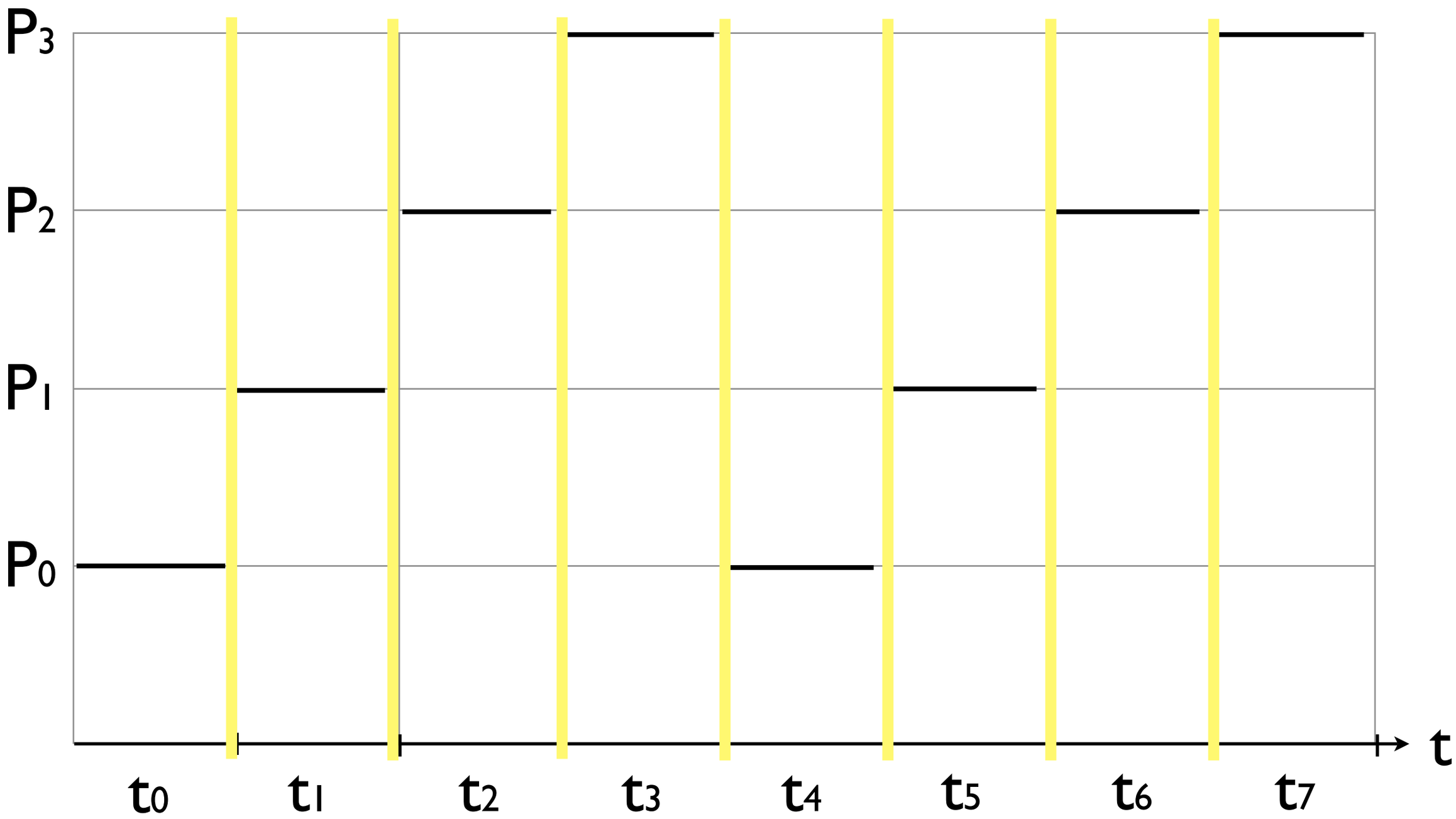
Despacho de proceso

- Sistemas multitareas
- Sistemas multiusuarios

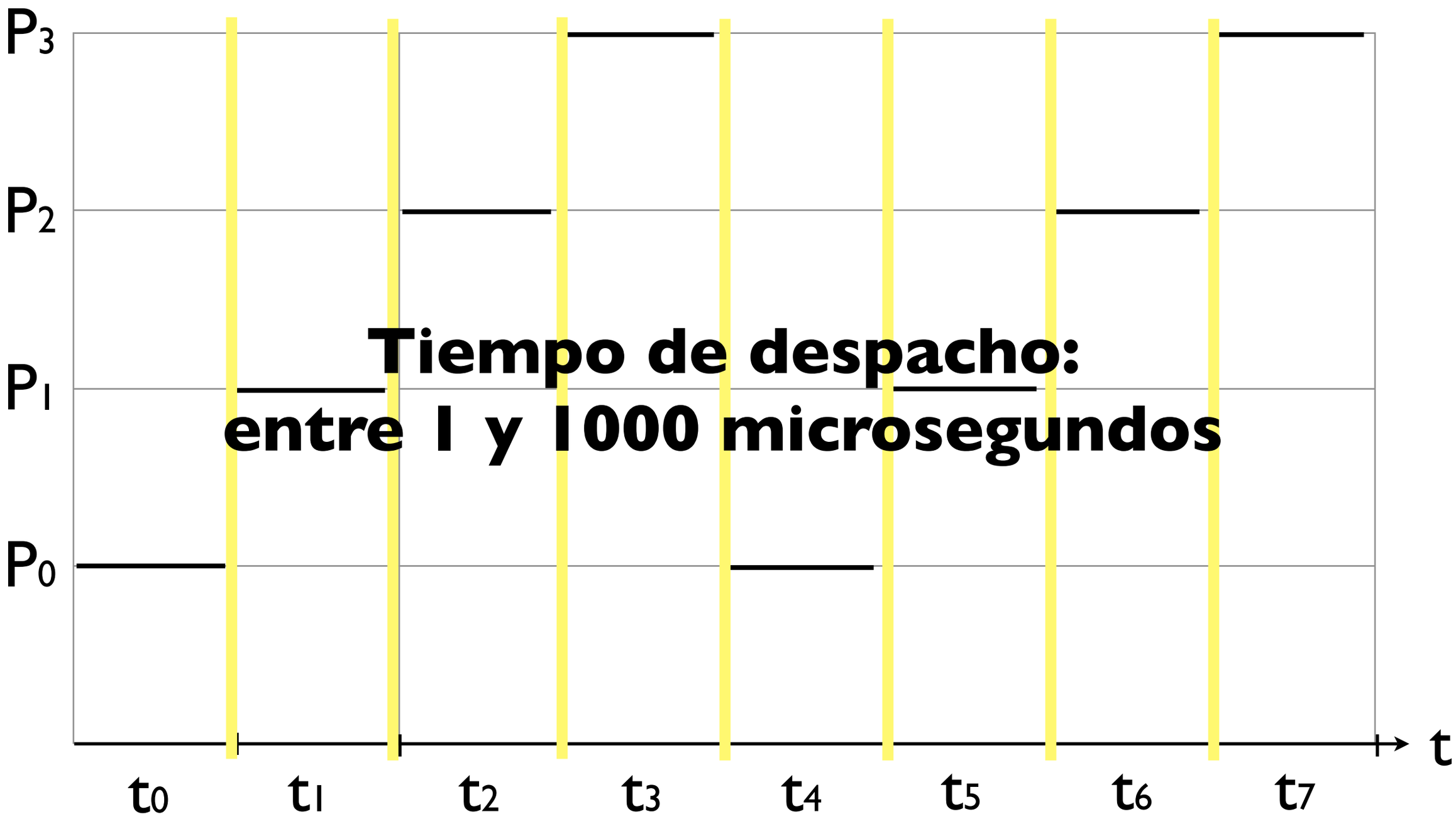
Despacho de proceso



Despacho de proceso

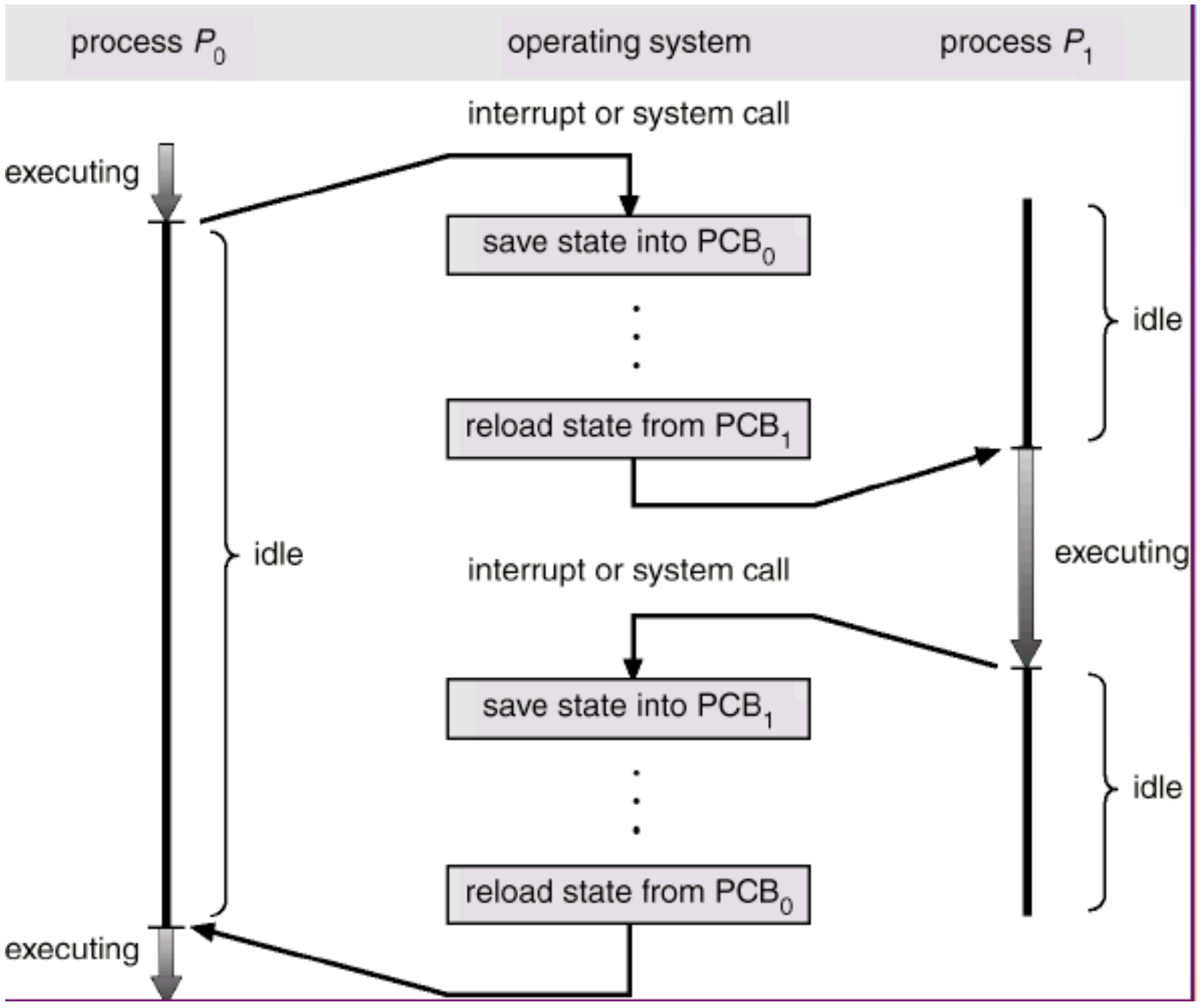


Despacho de proceso



Tiempo de despacho:
entre 1 y 1000 microsegundos

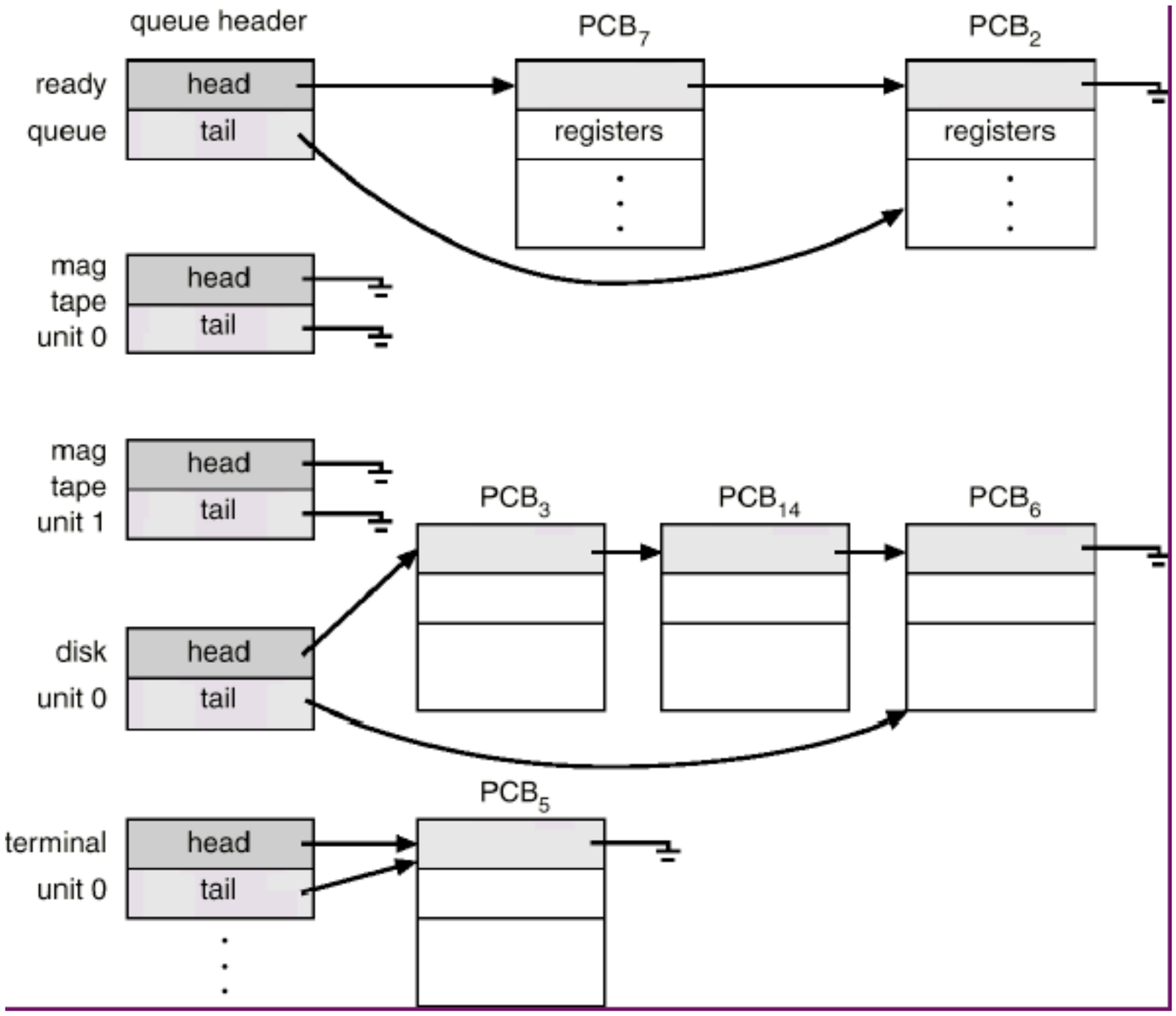
Despacho de proceso



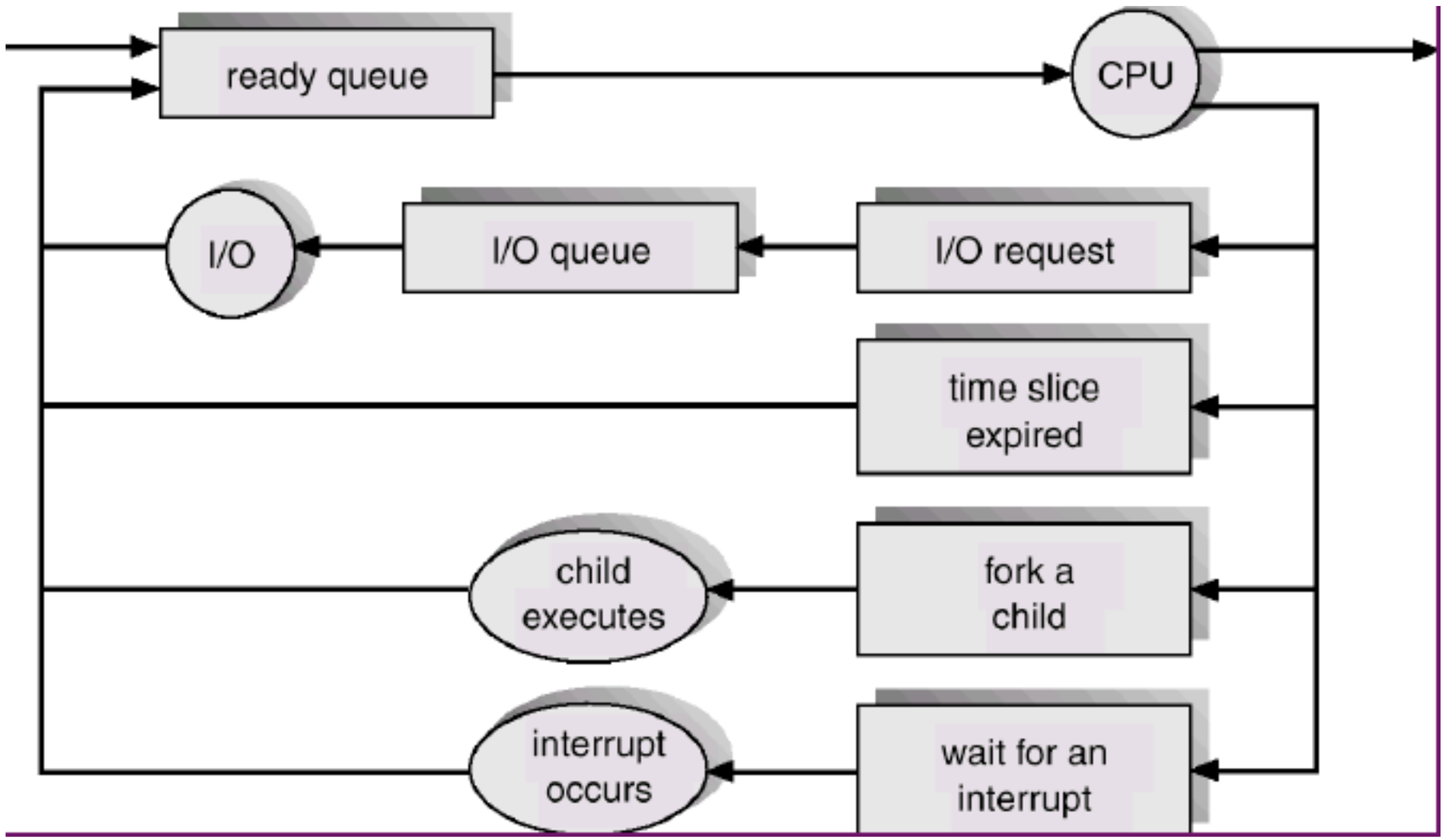
Despacho de proceso

- Colas de procesos: todos los procesos en el sistema
- Cola de procesos listos: todos los procesos residentes en memoria principal, listos y esperando ejecutarse
- Colas de dispositivos: todos los procesos esperando por una operación de E/S
- Procesos migran entre disferentes colas

Despacho de proceso



Despacho de proceso

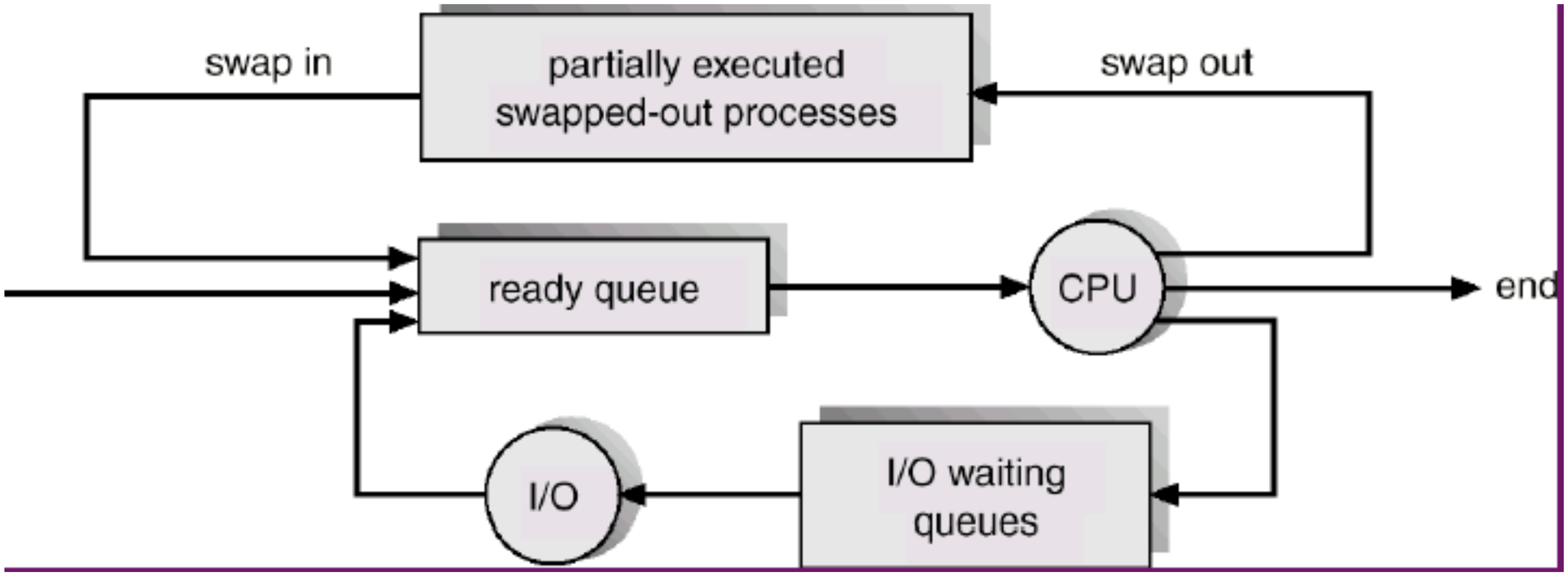


Migración de procesos entre las distintas colas

Despacho de proceso

- Planificador a largo plazo (despachador de trabajos): selecciona cual proceso debera colocarse en la cola de procesos listos (sistemas por lotes)
- Planificador a corto plazo (despachador de procesos): selecciona el siguiente proceso a ejecutarse y le asigna tiempo de CPU
- Planificador de mediano plazo: selecciona un proceso para guardarlo temporalmente en un dispositivo de almacenamiento secundario

Despacho de proceso



Planificador de mediano plazo

Despacho de proceso

- Planificador a corto plazo: se invoca frecuentemente (milisegundos) -> (debe ser rápido)

- Planificador a largo plazo no se invoca frecuentemente (segundos, minutos) -> (puede ser lento)

- Planificador a largo plazo controla el grado de multiprogramación

- Procesos pueden describirse como

  - Procesos de E/S - gastan más tiempo haciendo operaciones de E/S que computaciones

  - Procesos de CPU - gastan más tiempo haciendo computaciones utilizan más ciclos de CPU.

Despacho de proceso

- Cuando el CPU cambia a otro proceso, el sistema debe guardar el estado del proceso viejo, y carga el estado almacenado para el nuevo proceso
- El tiempo de cambio de contexto es elevado, el sistema no hace trabajo útil mientras se realiza el cambio de contexto
- El tiempo del cambio depende en el soporte de hardware

Operaciones con procesos

- Creación de procesos
- Terminación de procesos

Operaciones con procesos

Creación de procesos

- Proceso padre crea procesos hijos, el cual, a su vez, crea otros procesos, formando un arbol de procesos
- Recursos compartidos
  - Padre e hijos comparten todos los recursos
  - Padre e hijos comparten algunos de los recursos
  - Padre e hijos no comparten recursos

Operaciones con procesos

Creación de procesos

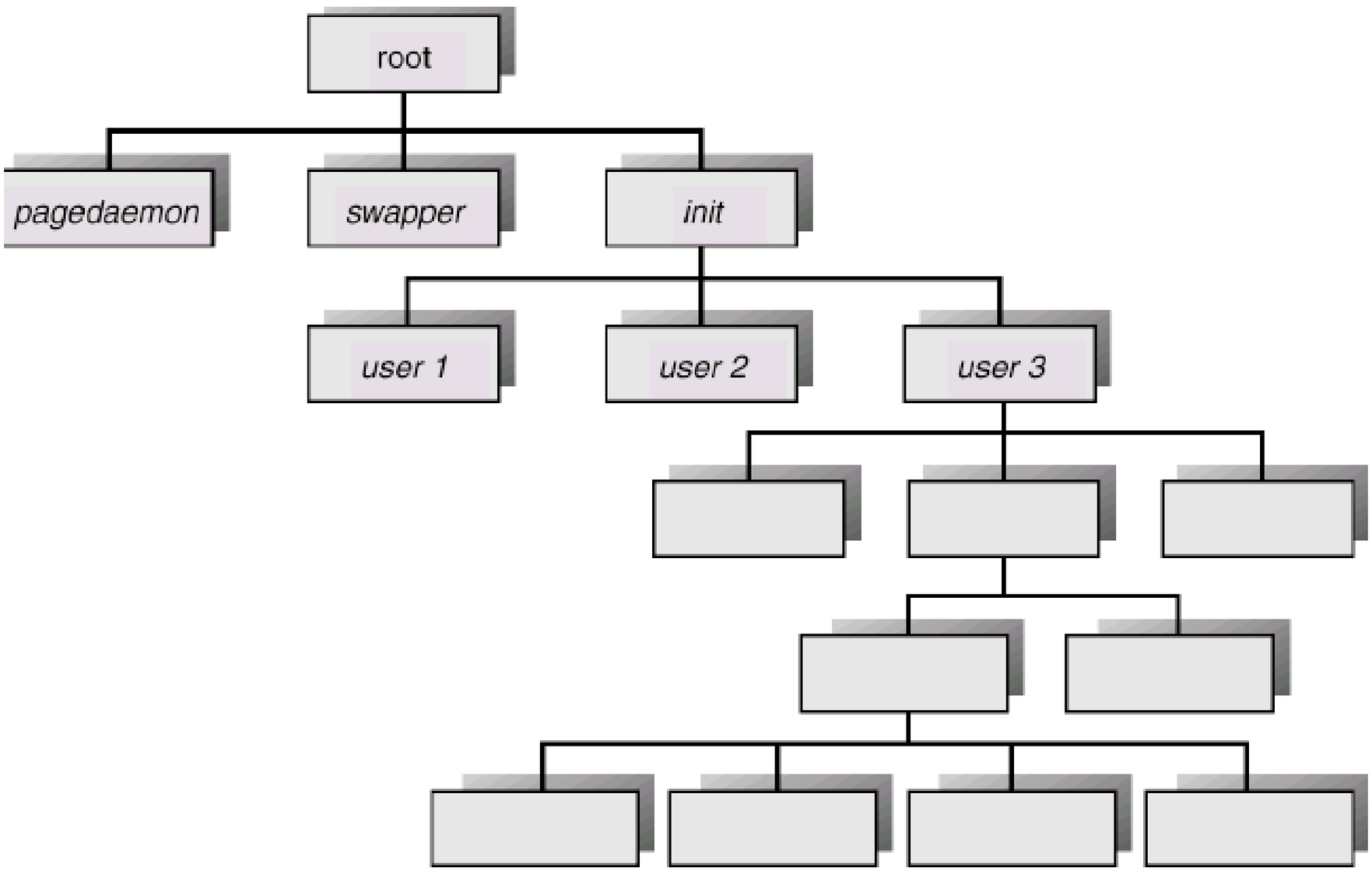
- Ejecución
  - Padre e hijo se ejecutan concurrentemente
  - Padre espera hasta que el hijo termine
- Espacio de direccionamiento
  - El hijo duplica al padre
  - El hijo carga un programa a su espacio de direccionamiento

Operaciones con procesos

Creación de procesos

- Ejemplo UNIX:
  - llamado al sistema **fork** crea nuevo proceso
  - llamado al sistema **exec** se utiliza después de **fork** para reemplazar el espacio de memoria del proceso con un nuevo programa

Operaciones con procesos



Operaciones con procesos
Finalización

- El proceso ejecuta su última declaración e informa al SO que ha decidido terminar (**exit**)
  - Se envía un dato de salida del proceso hijo al proceso padre (via **wait**)
  - Los recursos del proceso se liberan por el SO
- El proceso padre puede terminar la ejecución de los procesos hijos (**abort**)
  - El hijo a excedido los recursos asignados
  - La tarea asignada al proceso hijo ya no se requiere
  - El proceso padre termina
    - El SO no permite que los procesos hijos continuen si el padre termina
    - Finalización en cascada

Procesos cooperativos

- Procesos independientes: no pueden afectar o ser afectados por la ejecución de otros procesos
- Procesos cooperativos pueden afectar o ser afectados por la ejecución de otros procesos
- Ventajas de la cooperación de procesos:
  - Compartir información
  - Incremento en el rendimiento de una aplicación
  - Modularidad
  - Conveniencia

Procesos cooperativos

Problema: productor-consumidor

- El problema productor consumidor es un paradigma para procesos cooperativos. El productor genera información que es tomada por el consumidor.
  - *Buffer ilimitado*: no contiene un límite práctico en el tamaño del buffer
  - *Buffer limitado*: asume que existe un tamaño de buffer fijo

Procesos cooperativos

Problema: productor-consumidor

- Dato compartido

```
#define BUFFER_SIZE 10
typedef struct _element {
  ...
} element;
element buffer[BUFFER_SIZE];
int in=0;
int out=0;
```

- Buena solución.... pero está limitada al tamaño de buffer.

Procesos cooperativos

Problema: productor-consumidor

- Productor

```
element nextElement;

while(TRUE) {
   ...
   nextElement = getElement();
   ...
   while (((in+1)%BUFFER_SIZE)==out) ; //buffer lleno
   ...
   buffer[in] = nextElement;
   in = (in+1)%BUFFER_SIZE;
}
```

Procesos cooperativos

Problema: productor-consumidor

- Consumidor

```
element nextElementConsumed;

while(TRUE) {
    ...
    while (in==out) ;      // buffer vacio
    nextElementConsumed = buffer[out];
    out = (out+1)%BUFFER_SIZE;
}
```

Hilos

- Un proceso se puede descomponer en una parte estática y una dinámica
  - estática (tarea): memoria, identificadores de archivos...
  - dinámica (hilos): PC, registros del CPU, pila,...

PROCESO=TAREA+HILO

Hilos

- Hilo: procesos de carga ligera, compuesto del PC, registros y pila.
- Una tarea puede tener asociados uno o más hilos.
- Los hilos comparten variables.
- Ejemplos: C-Threads, pthreadss

Comunicación entre procesos (IPC)

- Mecanismos de comunicación y sincronización
- Sistema de mensajes: los procesos se comunican sin compartir variables
- IPC proporciona dos operaciones
  1. `send(message)`
  2. `receive(message)`

Comunicación entre procesos (IPC)

- Si los procesos **P** y **Q** desean comunicarse, necesitan:
  - establecer una liga de comunicación entre ellos
    - intercambiar mensajes vía `send/receive`
- Implantación de la liga de comunicación
  - física (memoria compartida, bus, ...)
  - lógica (sockets, puertos,...)

Comunicación entre procesos (IPC)

1. ¿Cómo se establecen las ligas?
2. ¿Una liga puede asociarse a más de dos procesos?
3. ¿Cuántas ligas pueden establecerse entre dos procesos que se comunican?
4. ¿Cuál es la capacidad de una liga?
5. ¿El tamaño del mensaje es fijo o variable?
6. ¿La liga es unidireccional o bidireccional?

Comunicación entre procesos (IPC)

- Comunicación directa
- Comunicación indirecta

IPC
Comunicación directa

- Los procesos deben nombrar al proceso destinatario u origen
  `send (P,message);`
  `receive (Q, message);`

- Propiedades de la liga de comunicación:
  Las ligas se establecen automaticamente
  Una liga se establece sólo con un par de procesos
  Entre cada par de procesos existe sólo una liga
  La liga puede ser unidireccionar, pero generalmente es bidireccional

IPC
Comunicación indirecta

- Los mensajes se envian y reciben a traves de buzones o puertos.
  Cada puerto tiene un único identificador
  Los procesos se pueden cominicar si comparten el puerto

- Propiedades de la liga de comunicación
  La liga se establece sólo si los procesos comparten el puerto
  Una liga puede asociarse a varios procesos
  Un par de procesos puede compartir varias ligas de comunicación
  La liga puede ser unidireccional o bidireccional

IPC
Comunicación indirecta

- Operaciones
  - crear un nuevo puerto
  - enviar y recibir mensajes a través del puerto
  - destruir el puerto

- Primitivas

```
send(A, message)
receive(A,message)
```

IPC
Comunicación indirecta

- Compartir puerto
  - P1, P2 y P3 comparten el puerto A
  - P1 envia un mensaje; los procesos P2 y P3 reciben
  - ¿Quién toma el mensaje?

- Soluciones
  - Permitir una liga asociada a mas de dos procesos
  - Permitir que un solo proceso a la vez ejecute la operacion recibir
  - Permitir que el sistema seleccione arbitrariamente al receptor. El origen es notificado de quien fue el receptor

IPC
Sincronización

- El paso de mensaje puede ser con bloqueo o sin bloqueo
- El bloqueo se considera síncrono
- El no-bloqueo se considera asíncrono
- Las primitivas `send` y `receive` puede ser por bloqueo o sin bloqueo

IPC
Uso de buffers

- Capacidad cero
- Capacidad limitada
- Capacidad ilimitada