Sistemas Operativos I

Procesos

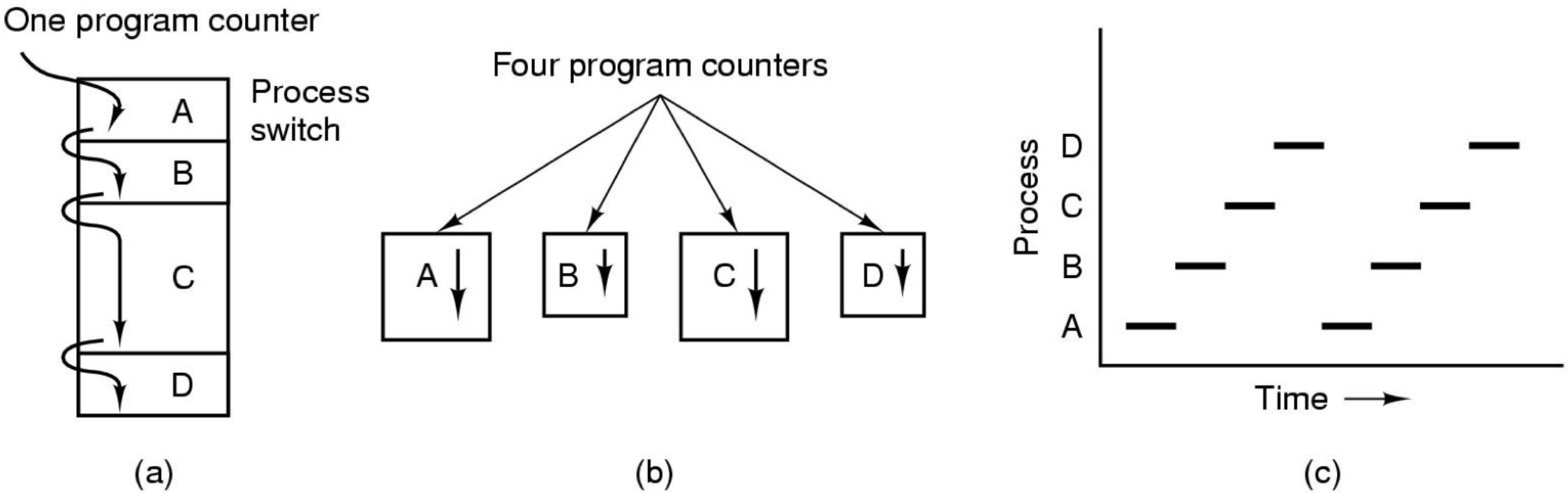
Introducción

- Los primeros sistemas de computadoras:
  - Permitían sólo ejecutar un programa a la vez
  - Este programa tenía control completo del sistema y sus recursos
- Hoy:
  - Múltiples programas pueden cargarse en memoria y ser ejecutados en forma concurrente
  - Requiere un control más estricto

Concepto de Proceso

- Los procesos en ejecución son:
  - Procesos del sistema operativo ejecutando código del sistema
  - Procesos del usuario ejecutando código del usuario
- Proceso – un programa en ejecución:
  - Programa (ejecutable) – entidad pasiva en almacenamiento secundario
  - Proceso – entidad activa
  - Los procesos en ejecución deben progresar en forma secuencial

Modelo de Ejecución



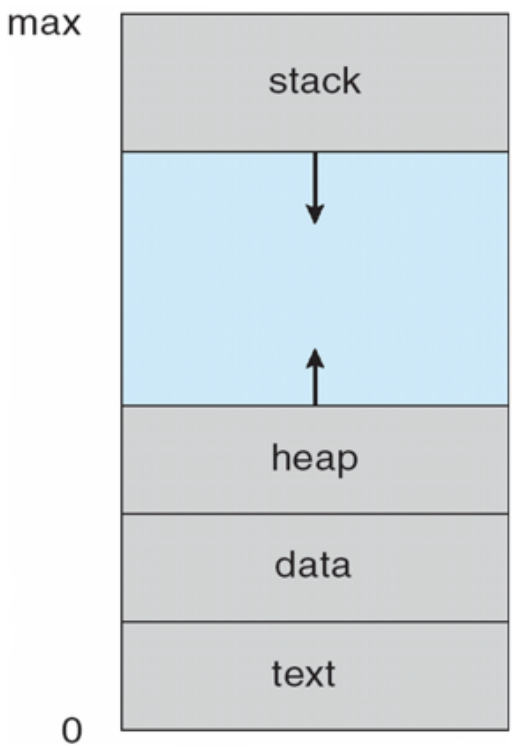
A. Multiprogramación de 4 procesos

B. Modelo conceptual de 4 procos independientes

C. Solo un proceso está activo en cada momento

Estructura de un Proceso

- Contador de programa (PC)
- Stack y heap
- Segmento de texto y datos



Proceso en Memoria

- **Stack**: Datos temportales, llamada a función, dir de retorno, variables locales
- **Heap**: Asignado dinámicamente durante la ejecución de un proceso
- **Data**: variables globales
- **Text**: código del programa

Programa

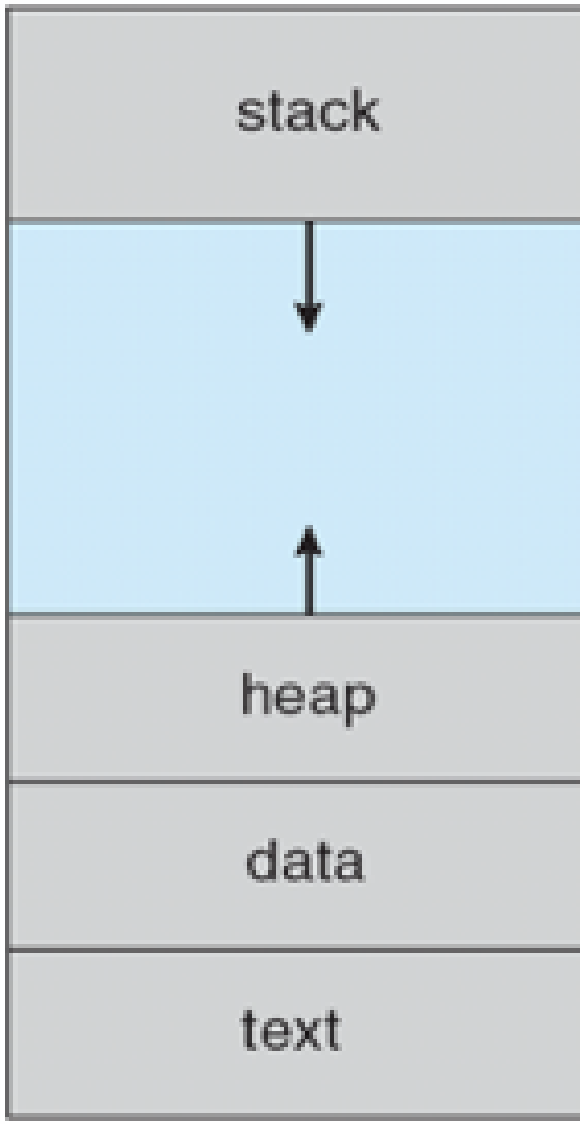
```
void foo(int param) // param se guarda en el stack
{
    int x; // x se guarda en el stack
    char *ptr = new char[255]; // 255 bytes se asignan en el stack
                               // y el puntero en el heap
    int array[255]; // 255 bytes se asigna en el stack
}
```

Diferentes en tamaño y contenido

Diferentes en tamaño y contenido

Diferente contenido

program counter



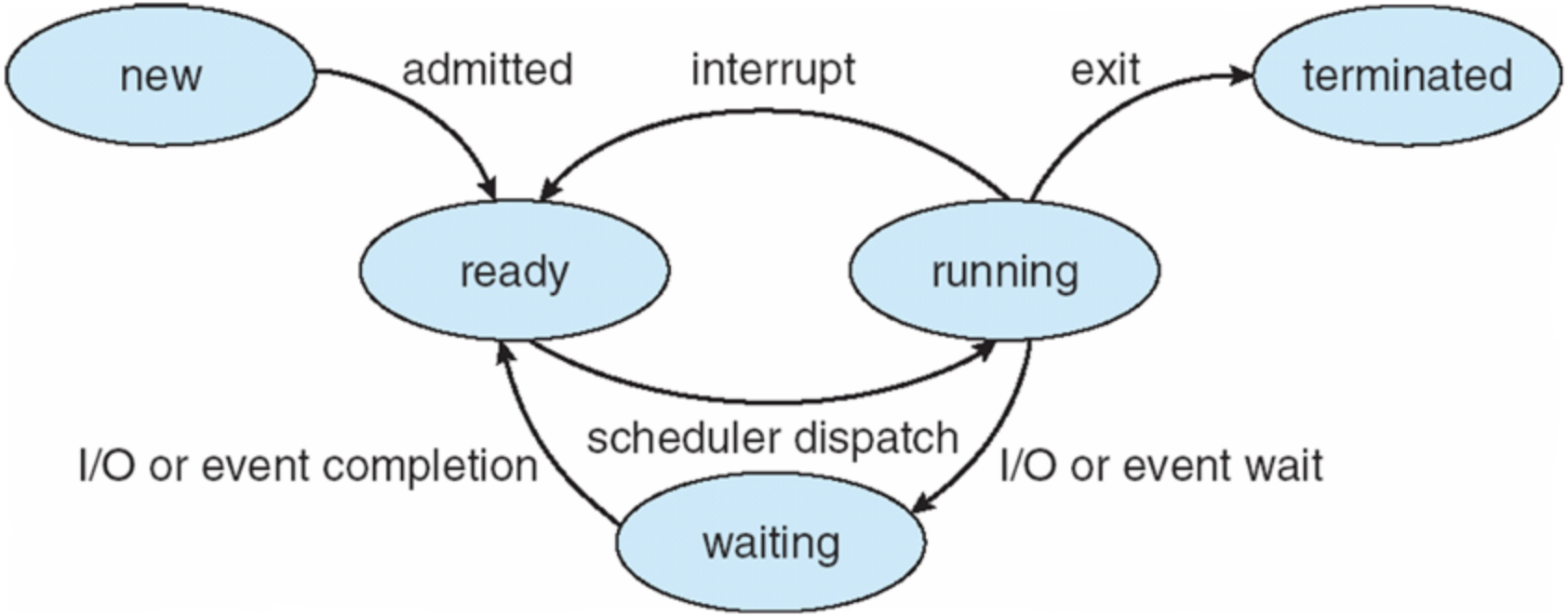
Programa y Proceso

- Dos procesos pueden estar asociados a un mismo programa:
  - 2 secuencias de ejecución diferente
  - La sección de texto son iguales, sin embargo, la sección de datos, heap y stack varían

Estados de un Proceso

- A medida que un proceso se ejecuta, cambia su estado
  - **new**: El proceso está siendo creado
  - **running**: Se están ejecutando instrucciones
  - **waiting**: El proceso está esperando por la ocurrencia de algún evento (p. ej., finalización de E/S)
  - **ready**: El proceso está esperando que se le asigne el procesador
  - **terminated**: El proceso ha finalizado la ejecución

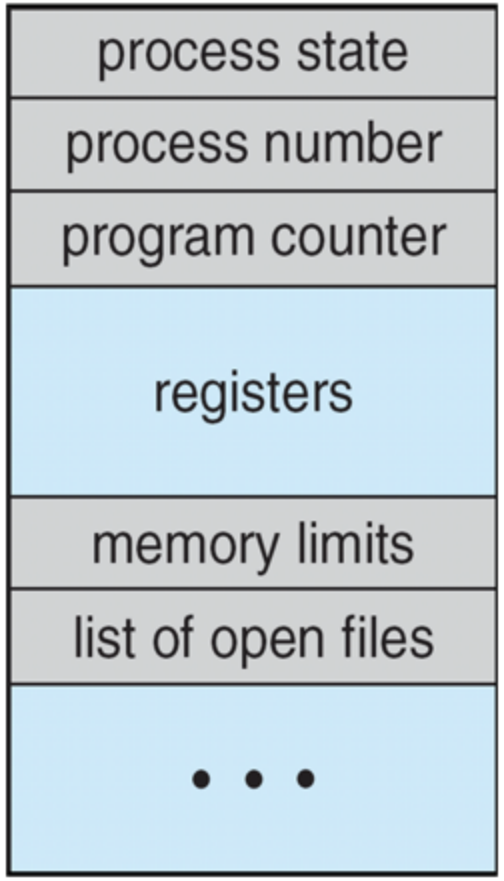
Diagrama de Estados



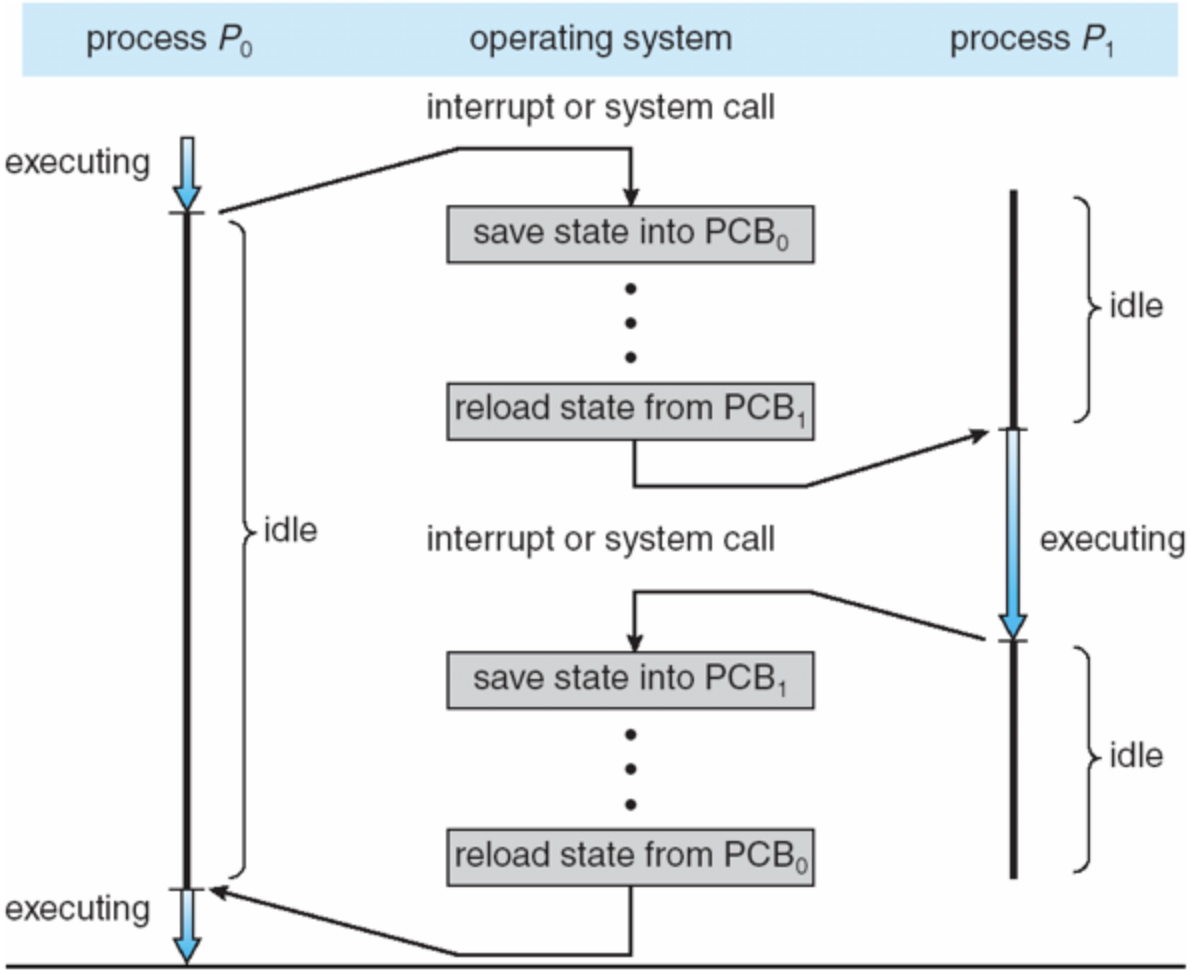
Bloque de Control de Proceso (PCB)

- Estado del proceso
- Contador del programa– dirección de la siguiente instrucción
- Registros de CPU
- Información de scheduling de CPU – p. ej., prioridad del proceso, punteros a colas de scheduling
- Información de administración de memoria – valores de los resgistros base y límite, tablas de página o segmento
- Información de auditoría
- Información de estado de E/S – lista de archivos abiertos, dispositivos de E/S asignados a procesos

PCB


process state
process number
program counter
registers
memory limits
list of open files
• • •

PCB y Asignación de CPU



Colas de Planificación

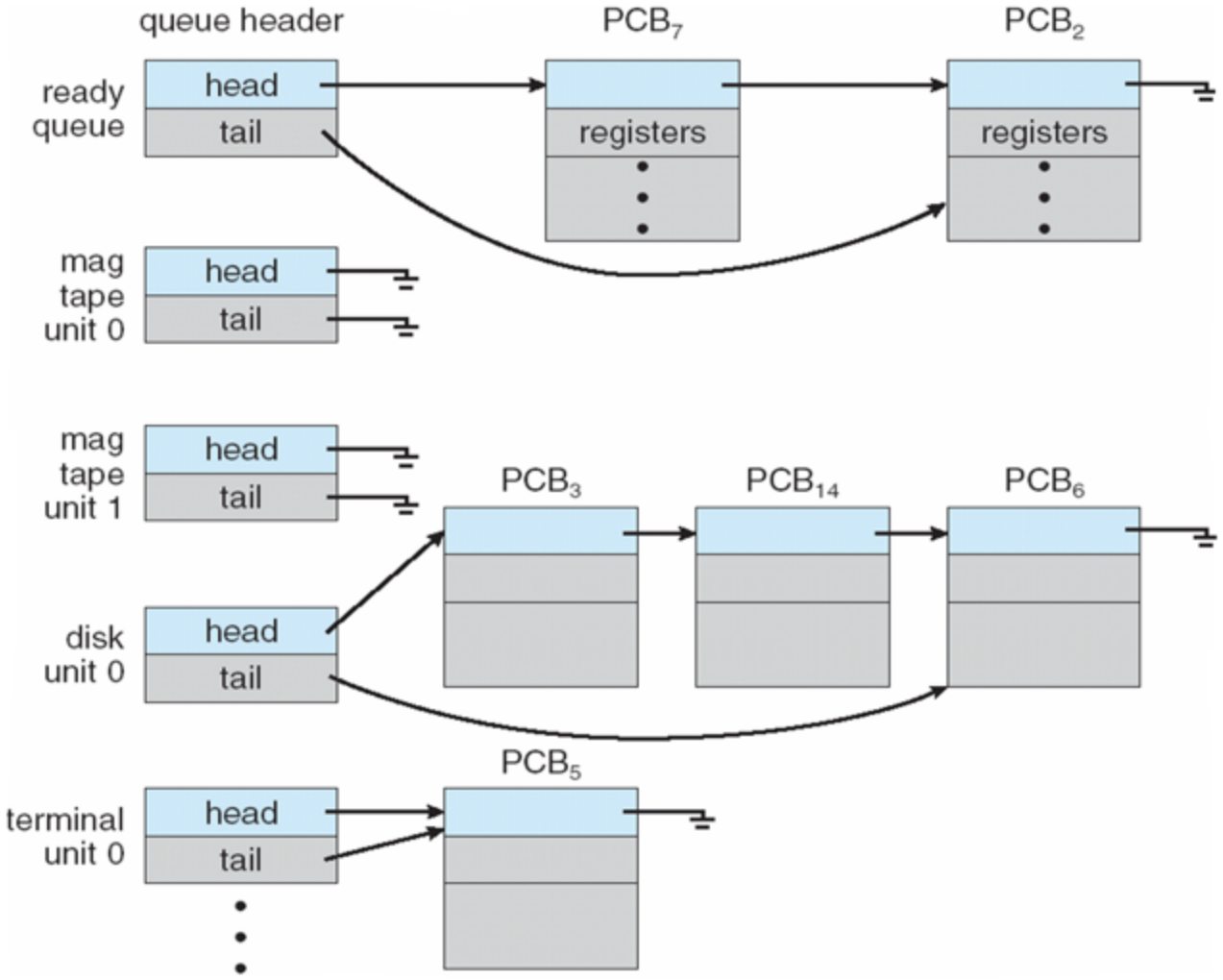
- **Objetivo** – tener procesos corriendo todo el tiempo para maximizar el uso de CPU

> Para cumplir el objetivo se usa un scheduler de procesos

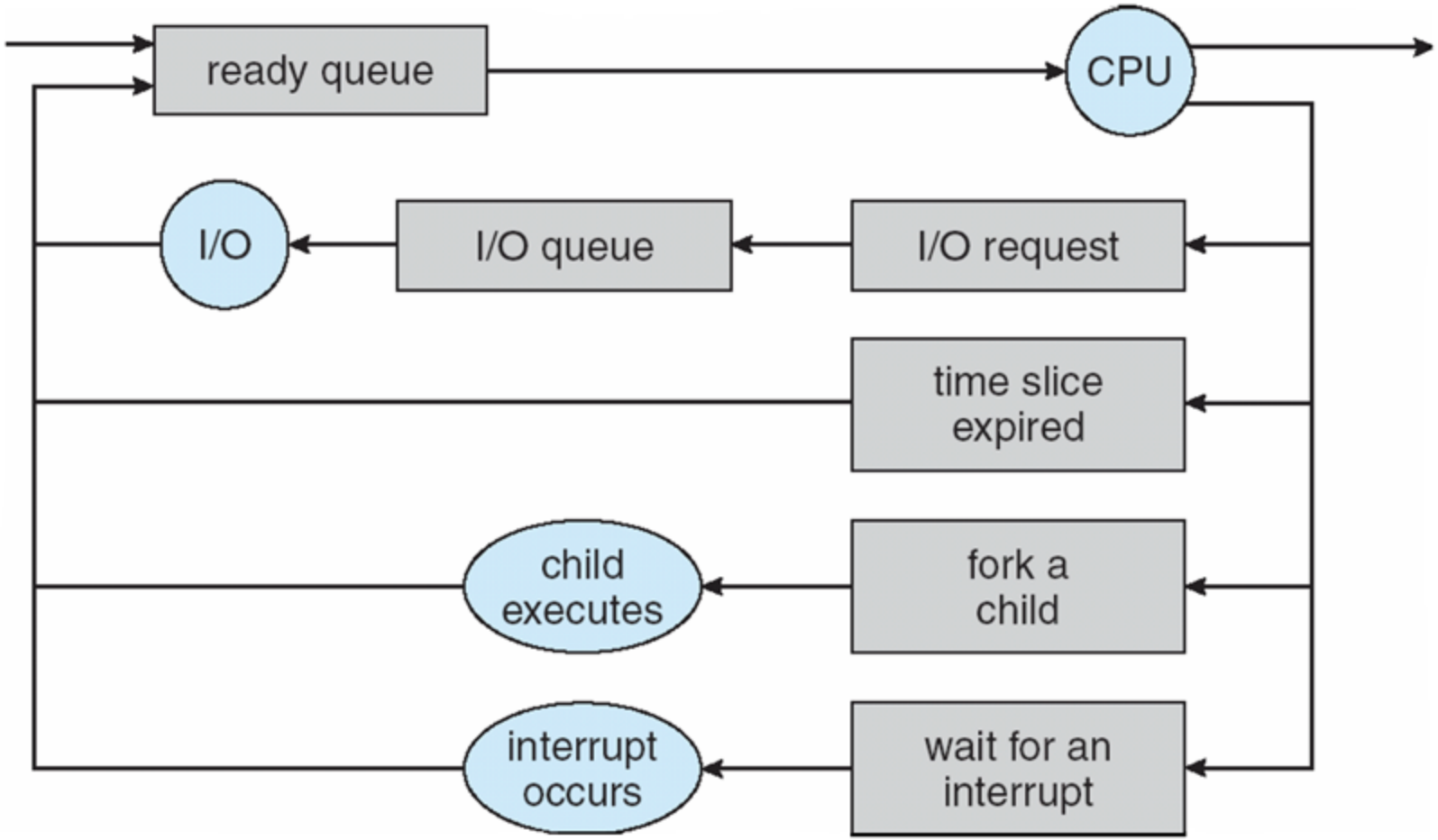
Cola de tareas – conjunto de todos los procesos del sistema
Cola de listos – conjunto de todos los procesos en memoria, listos y esperando para ejecutar
Colas de dispositivos – conjunto de todos los procesos esperando por un dispositivo de E/S

> Los procesos migran entre las distintas colas

Ejemplo: cola de Listos y E/S



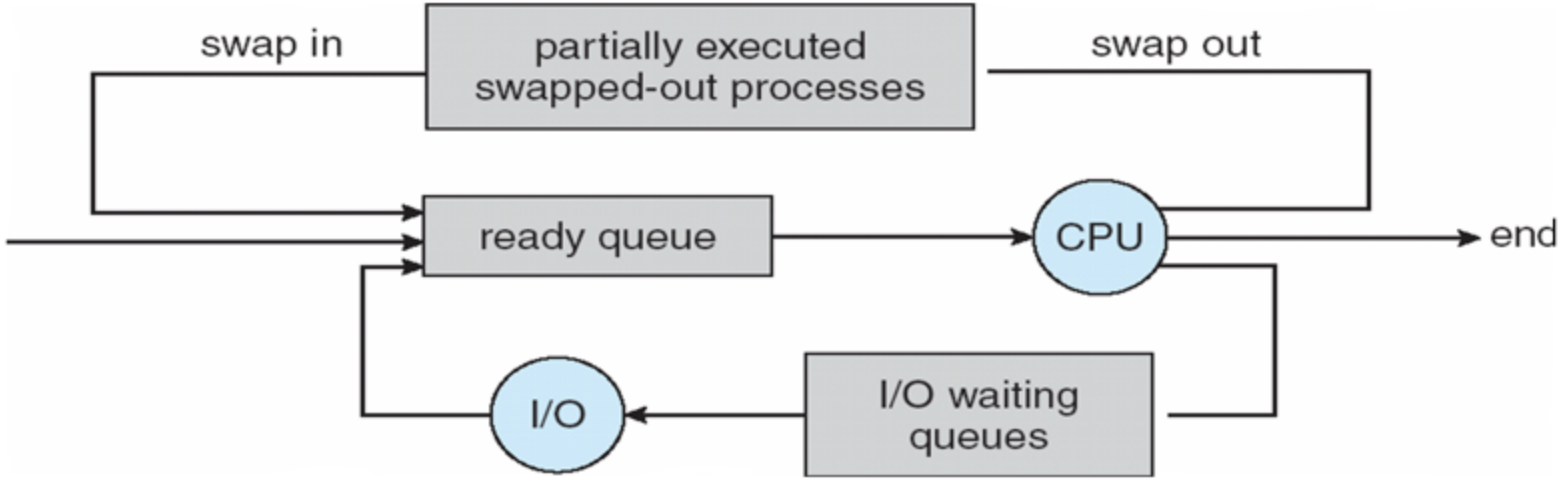
Planificación



> Por qué empiezan en *ready*?

Tipos de Planificadores

- **Scheduler de Largo Plazo** (o scheduler de tareas) – selecciona que proceso puede incluirse en la cola de procesos listos – carga los procesos de disco a memoria
- **Scheduler de Corto Plazo** (o scheduler CPU) – selecciona que proceso debe ejecutar a continuación (de la cola de listos) y asigna la CPU
- **Scheduler de Mediano Plazo** - selecciona que procesos que fueron supendidos van a ser incorporados en a la cola de listos



Caracterísicas de los Planificadores

- El scheduler de corto plazo se invoca muy frecuentemente (milisegundos) ⇒ debe ser rápido, de otra forma crea un overhead sustancial
- El planificador de largo plazo se invoca muy infrecuentemente ⇒ puede ser lento
- EL planificador de largo plazo controla el grado de multiprogramación (número de procesos en memoria)
- Los procesos se pueden clasificar como:
    - **Limitados por E/S** – pasan más tiempo haciendo E/S que computaciones, muchas ráfagas de CPU cortas (p. ej., navegando la web, copiando archivos grandes)
    - **Limitados por CPU** – pasa la mayor parte haciendo computaciones; pocas ráfagas de CPU largas (p. ej., cálculo de Pi)

Planificador de Largo Plazo

- Es importante para el scheduler de largo plazo seleccionar una buena mezcla de procesos limitados por E/S y limitados por CPU. Qué sucede con las colas de listos y E/S:
    - Si todos los procesos son limitados por E/S?
    - Si todos los a son limitados por CPU?
    - Que sucede si no tenemos planificador de largo plazo?

Planificación de CPU

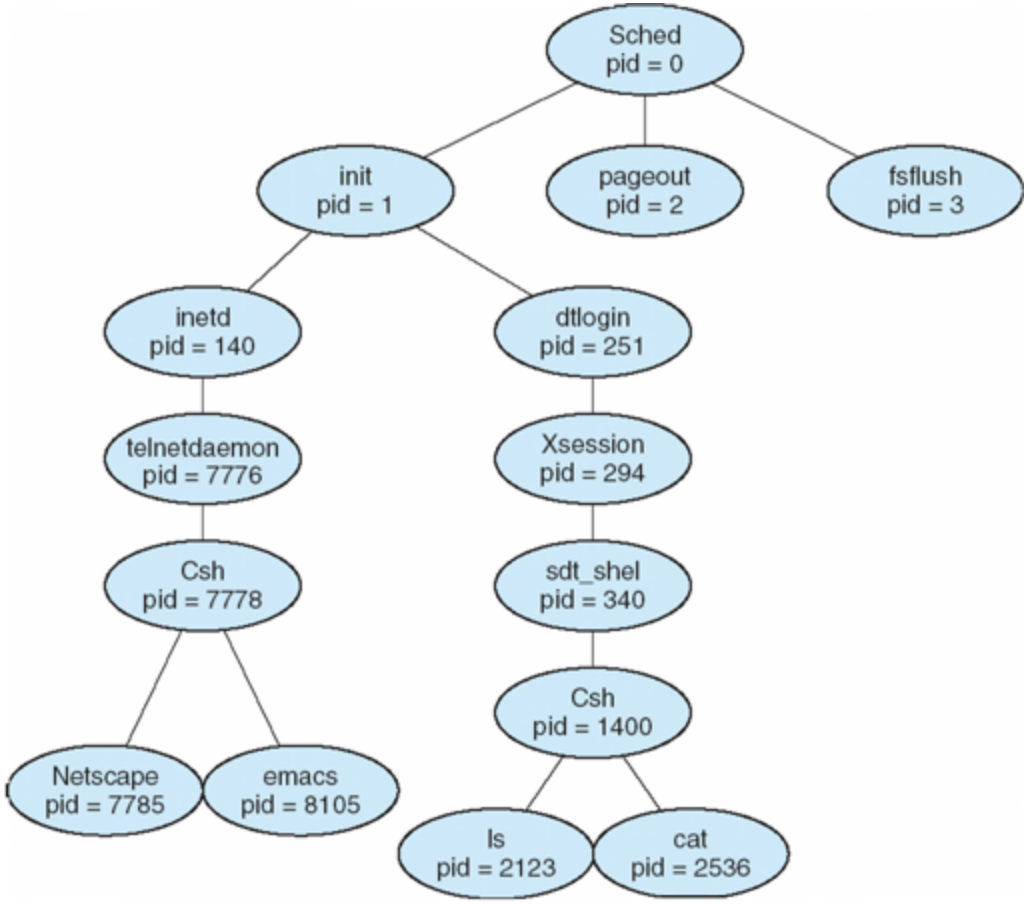
Cambio de Contexto

- Cuando la CPU cambia de un proceso a otro, el sistema debe guardar el estado del proceso viejo y cargar el estado guardado del nuevo proceso vía un cambio de contexto
- El contexto de un proceso es representado por el PCB
- El tiempo de cambio de contexto es un overhead; el sistema no está realizando ninguna tarea útil durante el cambio
- Dependiente del soporte de hardware
    - P. ej., la Sun UltraSPARC provee múltiples juegos de registros

Creacióm de Procesos

- Creación de procesos – se utiliza la llamada al sistema de creación de procesos (`fork()`)
- El proceso padre crea procesos hijos, los cuales, a su vez crean otros proceso, formando un árbol de procesos
- Generalmente, los procesos son identificados y manejados a través de un identificador de proceso (pid)

Arbol de Procesos



> `pstree`

Formas de Creación de Procesos

- Recursos compartidos (tiempo de CPU, memoria, dispositivos de E/S)
  - Padre e hijo comparten todos los recursos
  - El hijo comparte un subconjunto de los recursos del padre
  - Padre e hijo no comparten recursos: un proceso puede sobrecargar el sistema creando muchos procesos
- La información de inicialización se pasa de padre a hijo en la creación

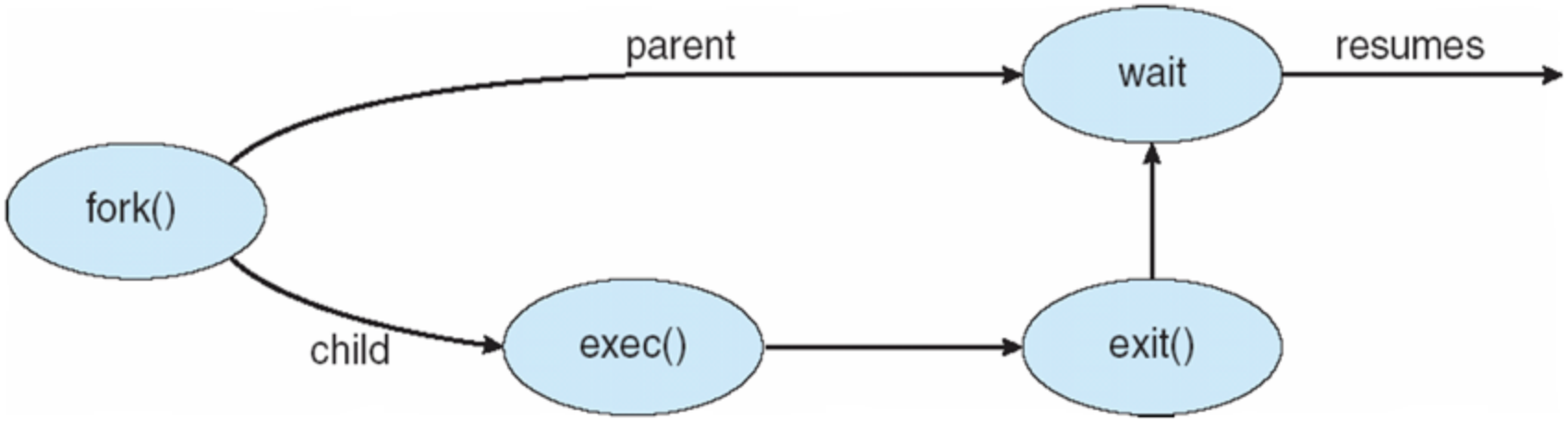
Creación de Procesos: UNIX

- La llamada al sistema `fork` crea un nuevo proceso
  - El espacio de memoria del padre es duplicado
  - mbos procesos continúan la ejecución en la instrucción posterior al fork()
- La llamada al sistema `exec` es usada luego de un fork para reemplazar el espacio de memoria del proceso con un nuevo programa
- `wait` saca un proceso de la cola de listos hasta que el hijo termine

Formas de Ejecución y Asignación

- Ejecución
  - Padre e hijo ejecutan concurrentemente
  - El padre espera que el hijo termine
- Espacio de memoria
  - El hijo duplica el espacio de memoria del padre – mismo código y datos
  - El hijo carga un nuevo programa

Ejemplo: Creación de un Proceso



IPC

Comunicación entre Procesos