

# **FUNCIONES PRINCIPALES DE UN SISTEMA OPERATIVO**

- PROCESOS:
  - Necesidad y Definición
  - Estados de procesos / Cambios
  - Jerarquía de procesos
  - Creación/Terminación de procesos
  - Implementación de procesos en Linux
- SEÑALES
  - Distintos tipos de Señales
  - Manejador
  - Implementación de señales en Linux

# Procesos

¿Necesidad de multiprocesamiento?

- Velocidad Procesador
- Velocidad Entrada Salida

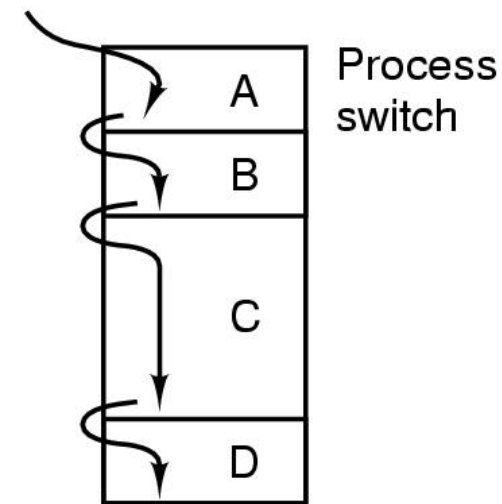
- Pseudoparalelismo

- Un solo procesador
- Una instrucción por vez
- ¿Cómo lo logro?

# Procesos

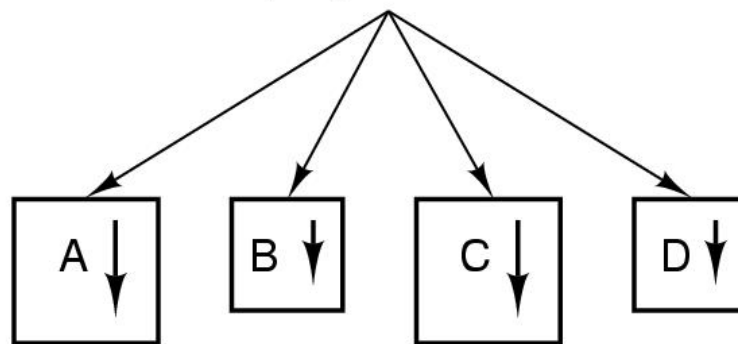
## Pseudoparalelismo

One program counter

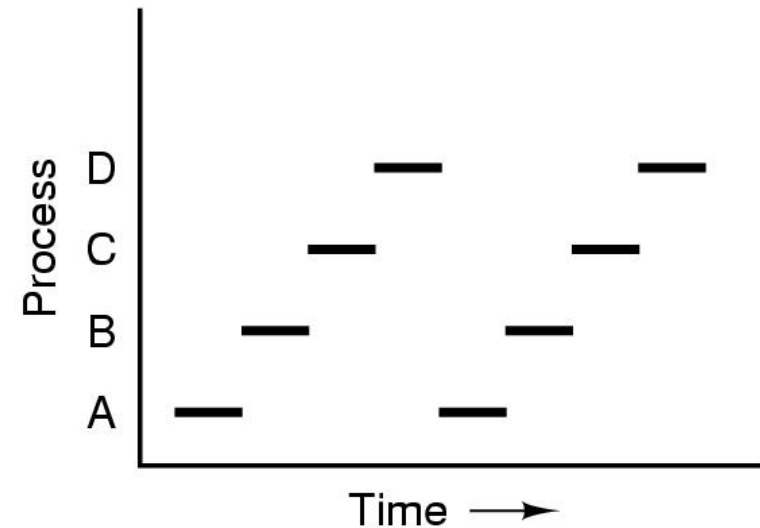


(a)

Four program counters



(b)



(c)

Programas DEBEN estar en memoria

Asignación de tiempos (Problemas con Real Time)

No programar basandose en supuestos de tiempo

Si se ejecuta nuevamente probablemente terminarán en distinto orden.

# Procesos cont.

¿Qué es un proceso?

Programa (entidad pasiva) en ejecución

+ valor del contador de programa

+ valor de los registros

+ pila del proceso (parámetros de funciones, direcciones de retorno, etc.)

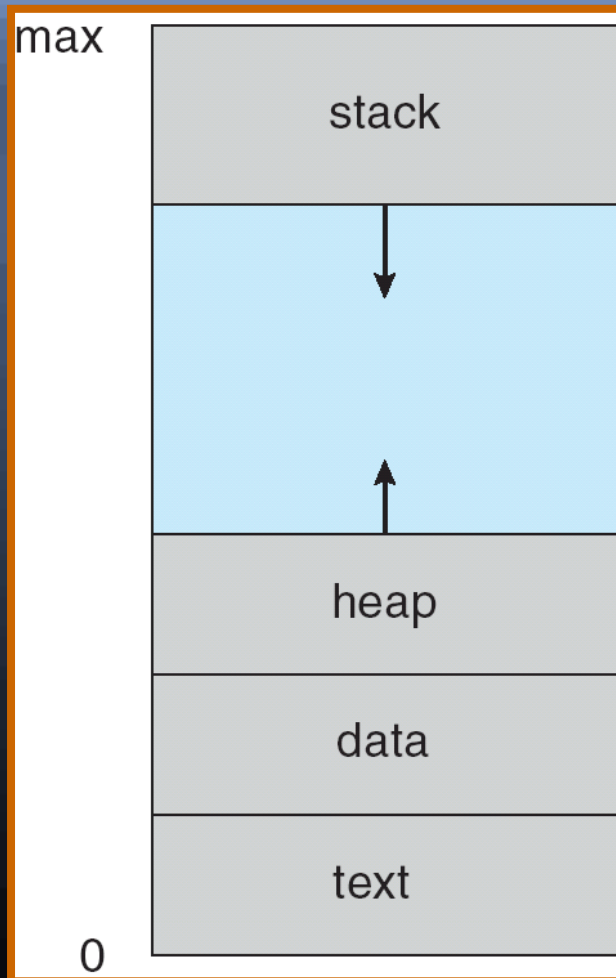
-----

**PROCESO (entidad activa)**

– Analogía

# Procesos cont.

Ubicación del proceso en memoria:



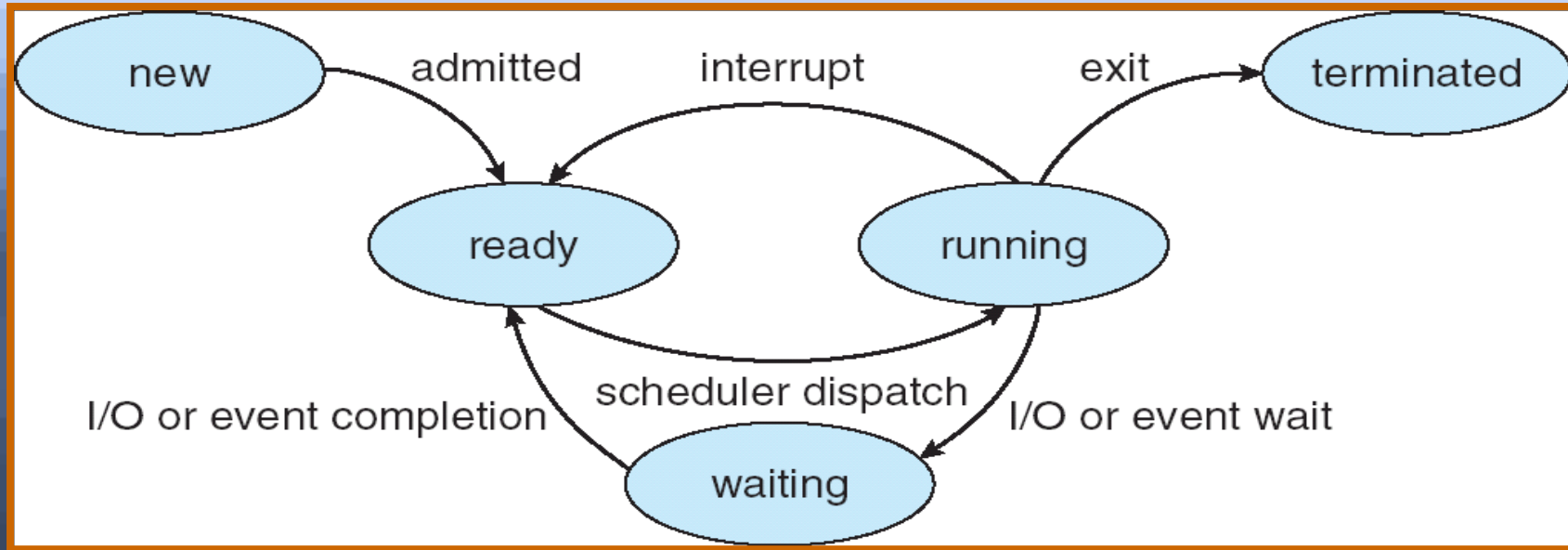
Stack: argumentos para la función, direcciones de retorno

Heap: espacio para asignación dinámica de memoria (malloc())

Data: Variables globales, variables inicializadas.

Text : código (binario) del programa

# Estado de los Procesos



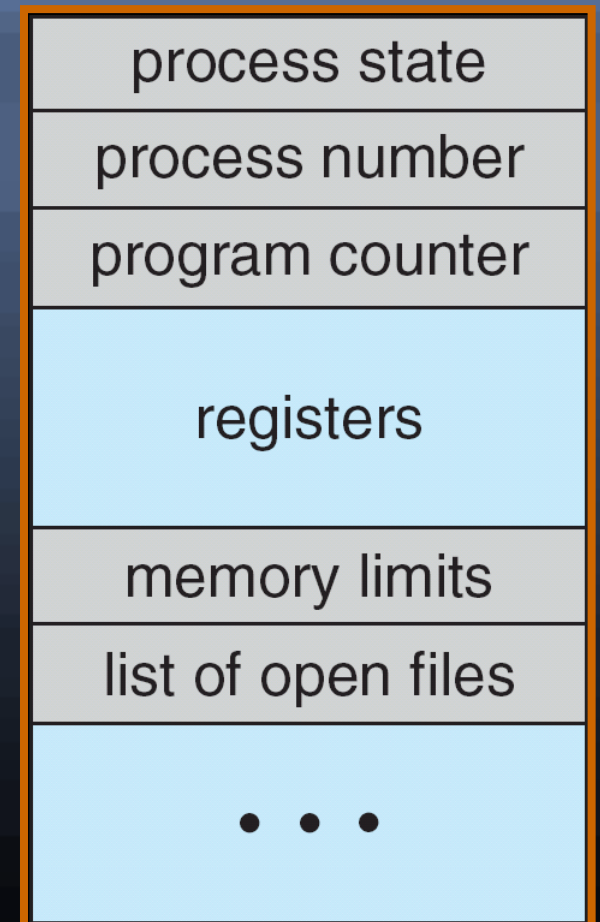
Diferencia entre los estados:

- ¿Cómo cambian de estado?
  - Scheduler .....que??
  - Interrupciones
  - Usuario puede forzar cambio de estado

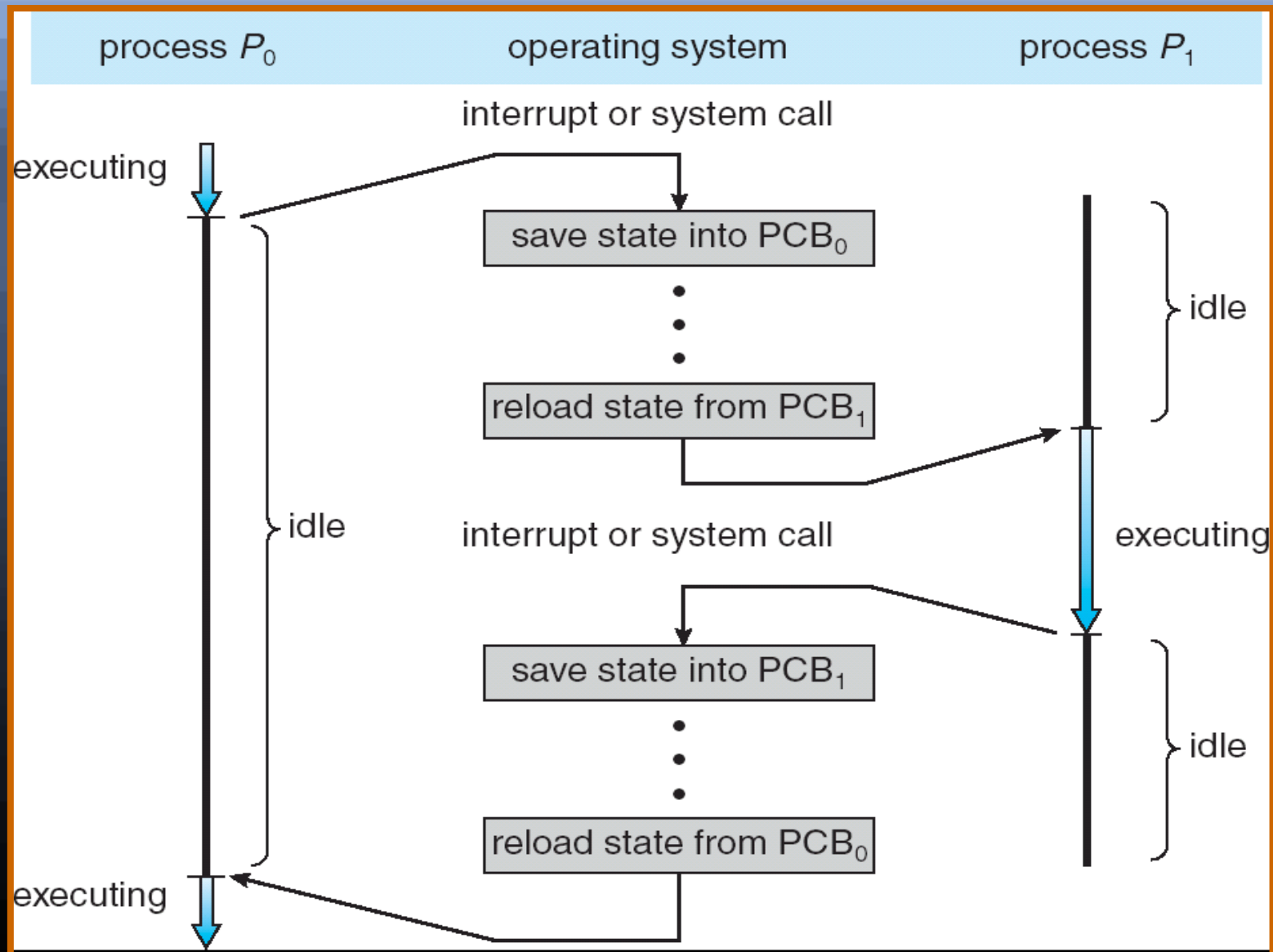
# Bloque de Control Procesos (PCB)

¿Cómo representa el Sistema Operativo los procesos???

- Número de Proceso (PID)
- Estado
- PC ( contador de programa)
- Registros (generales, puntero de pila)
- Espacio de memoria que ocupa
- Descriptores de archivos abiertos
- Usuario / grupo
- etc.



# Conmutación entre Procesos (cambio de contexto)

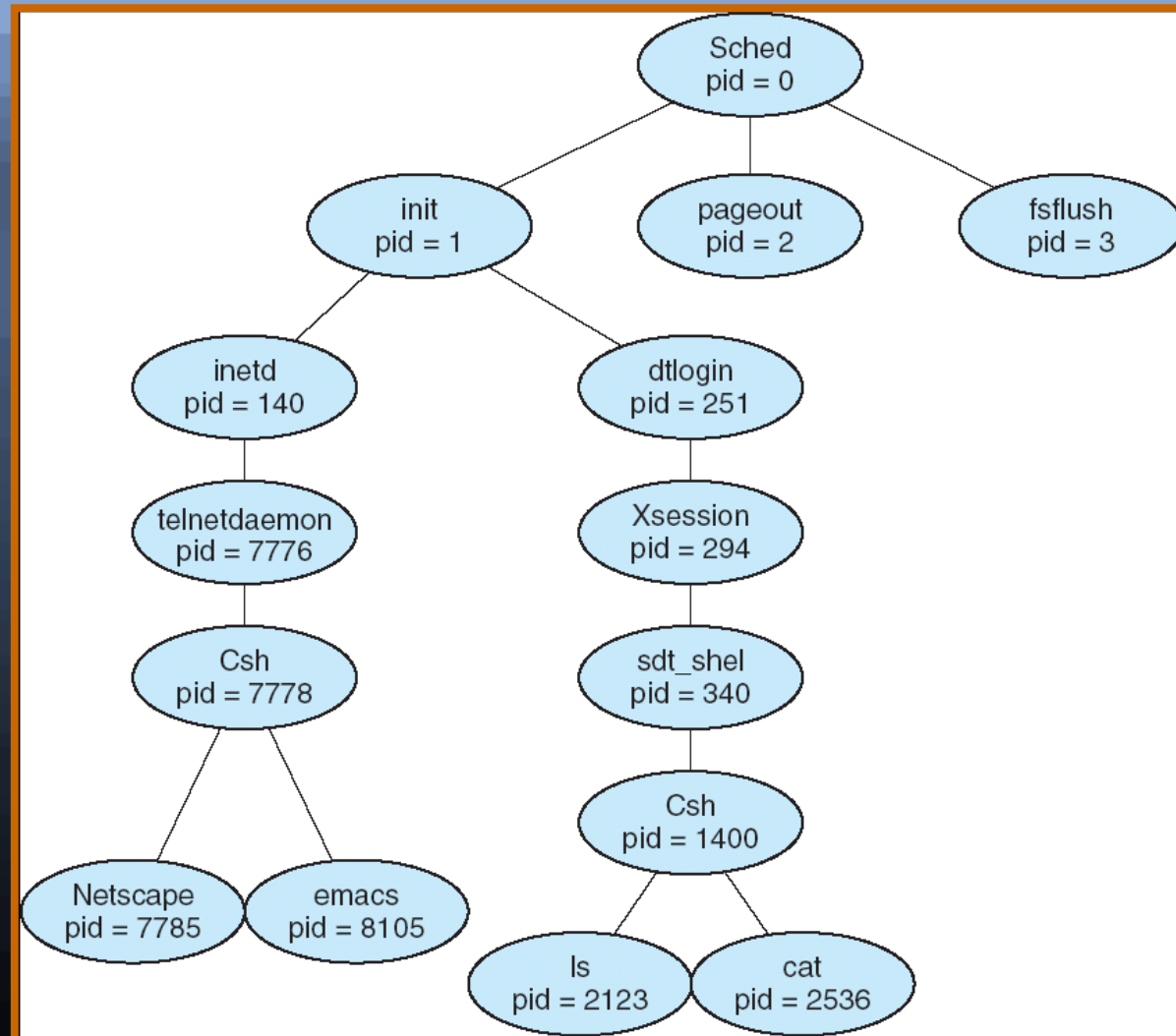




# Jerarquía de Procesos

Relación entre procesos:

- Arbol de procesos
  - Comando pstree
  - Proceso Sched
  - Proceso init
- Llamadas a sistema:
  - “getpid()”
  - “getppid()”



# Creación de Procesos

¿Para qué creo procesos?

- Inicialización del Sistema
  - Init
  - Servicios
- Nuevos servicios
  - Comandos, servicios
  - Background, foreground (bg, fg, &, at)
- División funcional de tarea a realizar: ¿Ejemplos?
- ¿Cómo muestro procesos existentes en linux?
  - Comando ps, top, pstree.
- ¿Cómo creo procesos en linux?
  - Desde Interpretador de comandos (o doble “click”)
  - Llamada a sistema “fork()”

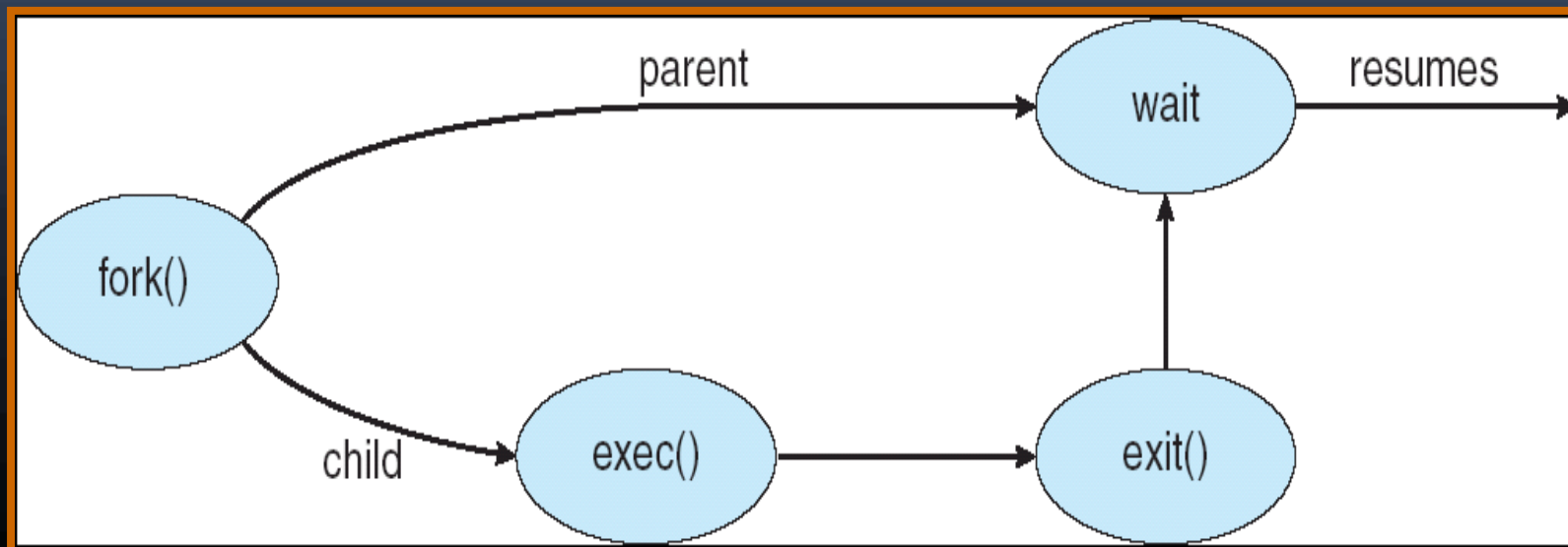
# Creación de Procesos con fork()

## Alternativas Padre:

- Ejecuta concurrentemente con hijo
- Espera que finalice el hijo

- Alternativas Hijo:

- Usa mismo programa y datos que padre
- Carga un nuevo programa



# Terminación de Procesos

## Tipos de terminación

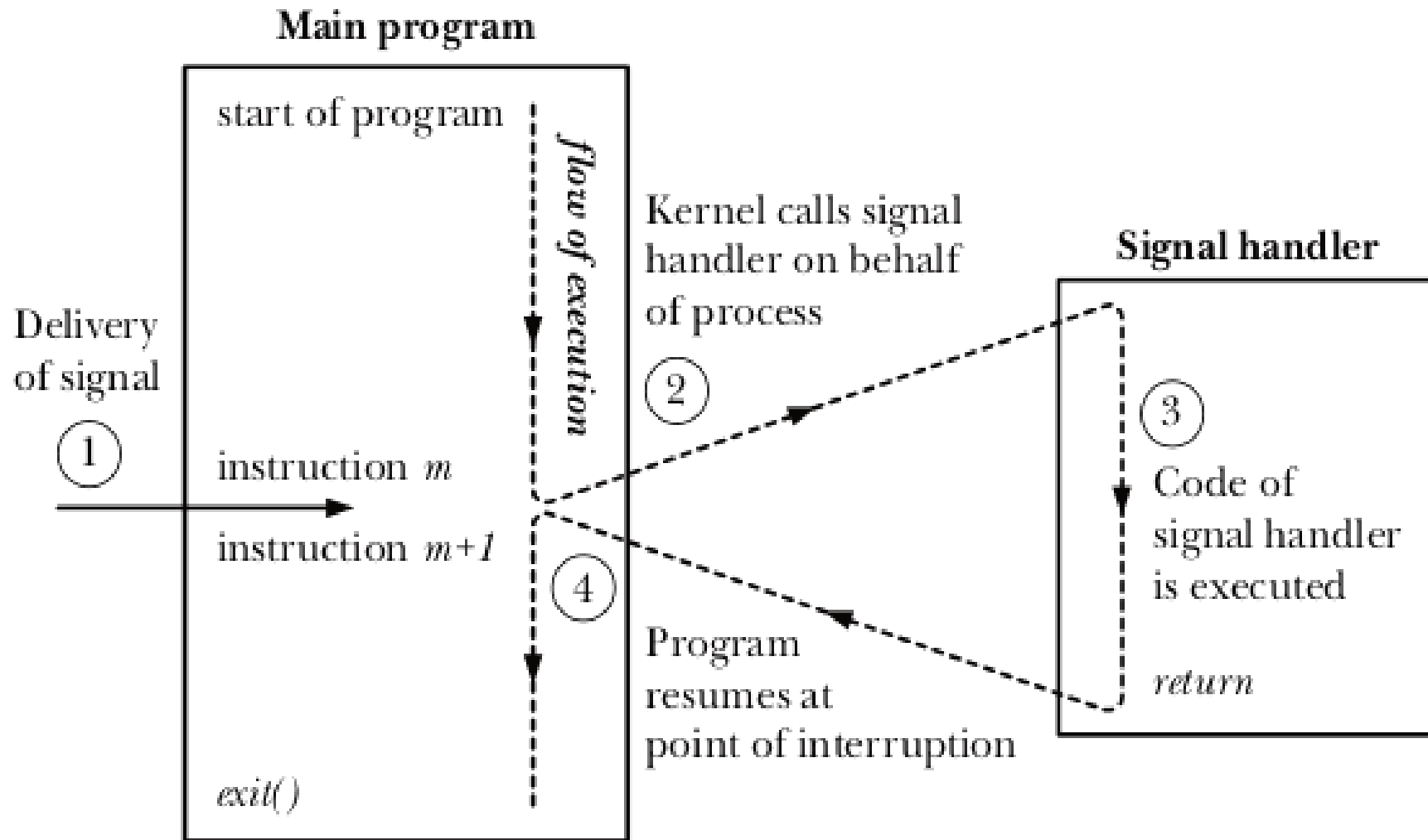
- Terminación Normal
  - Tareas que realiza el SO
  - echo \$?
- Terminación Anormal Voluntaria
  - echo \$?
  - standard error
- Terminación Anormal Involuntaria
  - No hay mas memoria
  - Violación de Segmento
  - División por cero
- Recepción de una señal (Que es una señal?)
  - Comando kill, llamada a sistema “kill()”
  - Handler de señales, llamada a sistema “signal()”

# Señales

¿Qué son ?

- Una señal es una notificación asíncrona entregada a un proceso a partir de la ocurrencia de un evento para .
- Para que se usan ?
  - Aviso del kernel de una excepción de Hardware
    - División por cero, Violación de Segmento, etc.
  - Notificar al proceso de un evento de software
    - Un hijo termina la ejecución, un contador llega a cero, etc.
  - Para controlarlo desde el teclado
    - Suspende el proceso, terminar el proceso.

# Manejador de la Señal



# Manejador de la Señal

- Cuando un proceso recibe una señal este puede:
  - Ignorar la señal.
  - Dejar que se ejecute una acción por defecto relacionada con la señal.
  - Ejecuta un manejador para la señal: programa encargado de “hacer algo” con la señal recibida.