PROCESOS

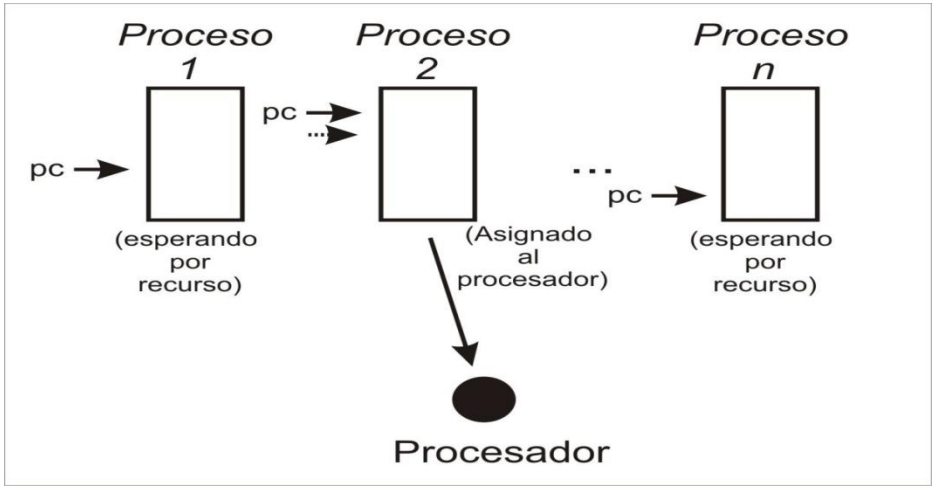
SISTEMAS OPERATIVOS
DEPARTAMENTO DE COMPUTACION-FACYT
PROFESORA MIRELLA HERRERA

PROCESOS

- Instancia de un programa en ejecución
- Desde el punto de vista del Sistema Operativo es la entidad más pequeña, individualmente planificable, formada por código (instrucciones de máquina y llamadas al sistema) y datos; caracterizada por atributos (asignados por el programador del sistema o por el S.O. tales como prioridad, derechos de acceso, entre otros) y un estado dinámico (inactivo, listo, en ejecución, espera o completado)

PROCESOS

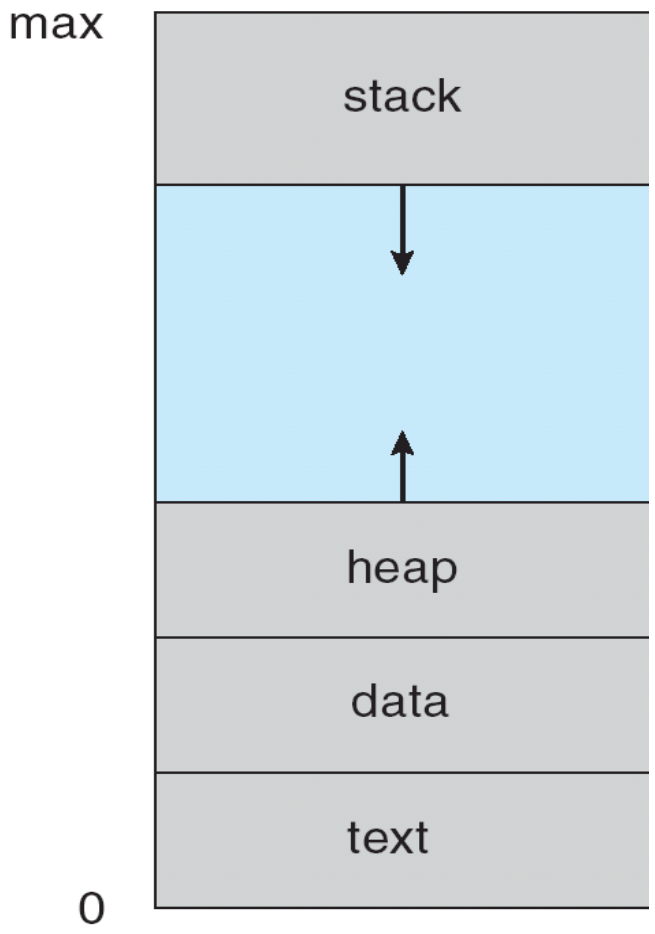
Cada proceso tiene su *program counter*, y avanza cuando el proceso tiene asignado el recurso procesador. A su vez, a cada proceso se le asigna un número que lo identifica entre los demás: identificador de proceso (*process id*).



PROCESOS

Un proceso en memoria se constituye de varias secciones:

- Código(text): Instrucciones del proceso
- Datos(data): Variables globales del proceso
- Memoria dinámica (Heap): Memoria dinámica que genera el proceso.
- Pila (Stack): Utilizado para preservar el estado en la invocación anidada de procedimientos y funciones.



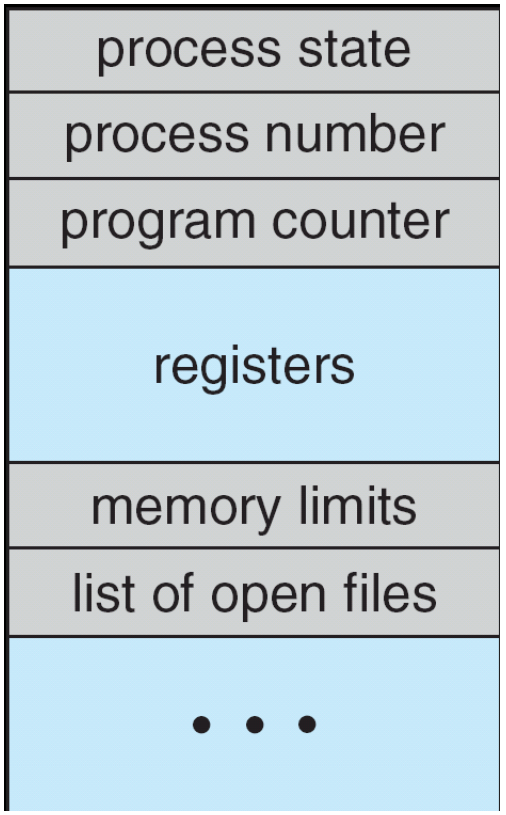
PROCESOS

El estado de un proceso es definido por la actividad corriente en que se encuentra. Los estados de un proceso son:

- Nuevo(new): Cuando el proceso es creado
- Ejecutando (running):El proceso tiene asignado un procesador y está ejecutando sus instrucciones
- Bloqueado (waiting):El proceso está esperando por un evento (que se complete un pedido de E/S o una señal)
- Listo (ready): El proceso está listo para ejecutar, solo necesita del recurso procesador
- Finalizado(terminated): El proceso finalizó su ejecución

BLOQUE DE CONTROL DE PROCESOS PCB

Es una estructura de datos en la que el S.O. agrupa toda la información que necesita conocer respecto a un proceso en particular. Cada vez que se crea un proceso el S.O. crea una PCB y cuando el proceso termina su PCB es liberada



CREACIÓN DE PROCESOS

- Envío de un trabajo por lotes (batch)
- Conexión de usuario (log-in)
- Creados para proporcionar servicios como imprimir
- Procesos que crean otros procesos

TERMINACIÓN DE PROCESOS

- Un trabajo por lotes ejecuta la instrucción *Halt*
- Un usuario se desconecta (log-out)
- Salir de una aplicación
- Errores y condiciones de fallo

TERMINACIÓN DE PROCESOS

Razones para la terminación de un Proceso:

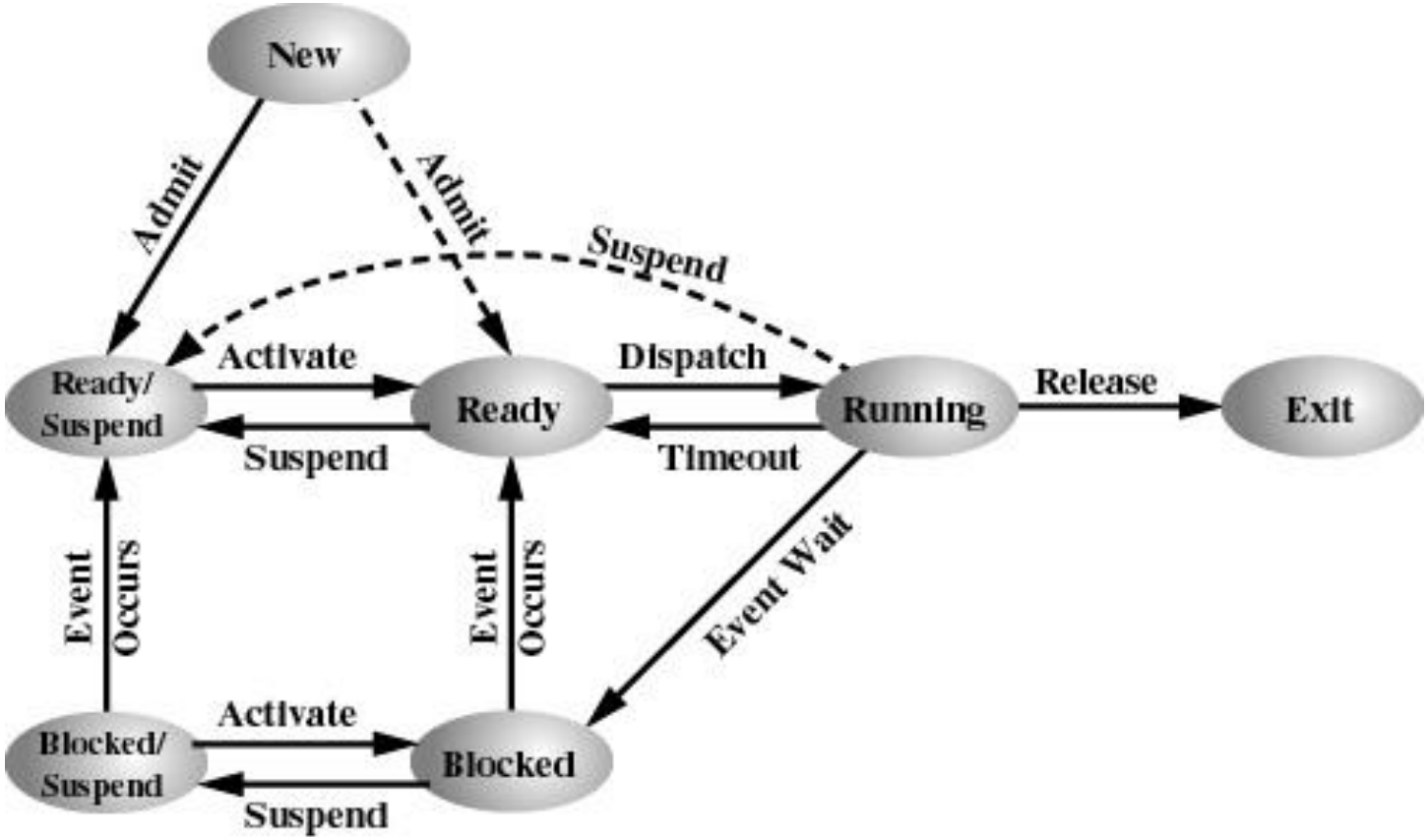
- Finalización normal
- Se excede el tiempo límite
- No hay memoria disponible
- Violación de límites
- Error de protección
  - ejemplo: escribir en un fichero de sólo lectura
- Error aritmético
- Tiempo de espera sobrepasado
  - un proceso espera un evento más tiempo del máximo especificado

TERMINACIÓN DE PROCESOS

Razones para la terminación de un Proceso:

- Fallo de entrada/salida (E/S)
- Instrucción inválida
  - cuando se intenta ejecutar datos
- Instrucciones privilegiadas
- Uso incorrecto de datos
- Intervención del Sistema Operativo
  - cuando se detecta un interbloqueo (*deadlock*)
- El padre termina, así que los hijos mueren
- Por petición del proceso padre

DIAGRAMA DE ESTADO DE PROCESOS



(b) With Two Suspend States

TRANSICIÓN DE ESTADOS

Razones para la Suspensión de un Proceso:

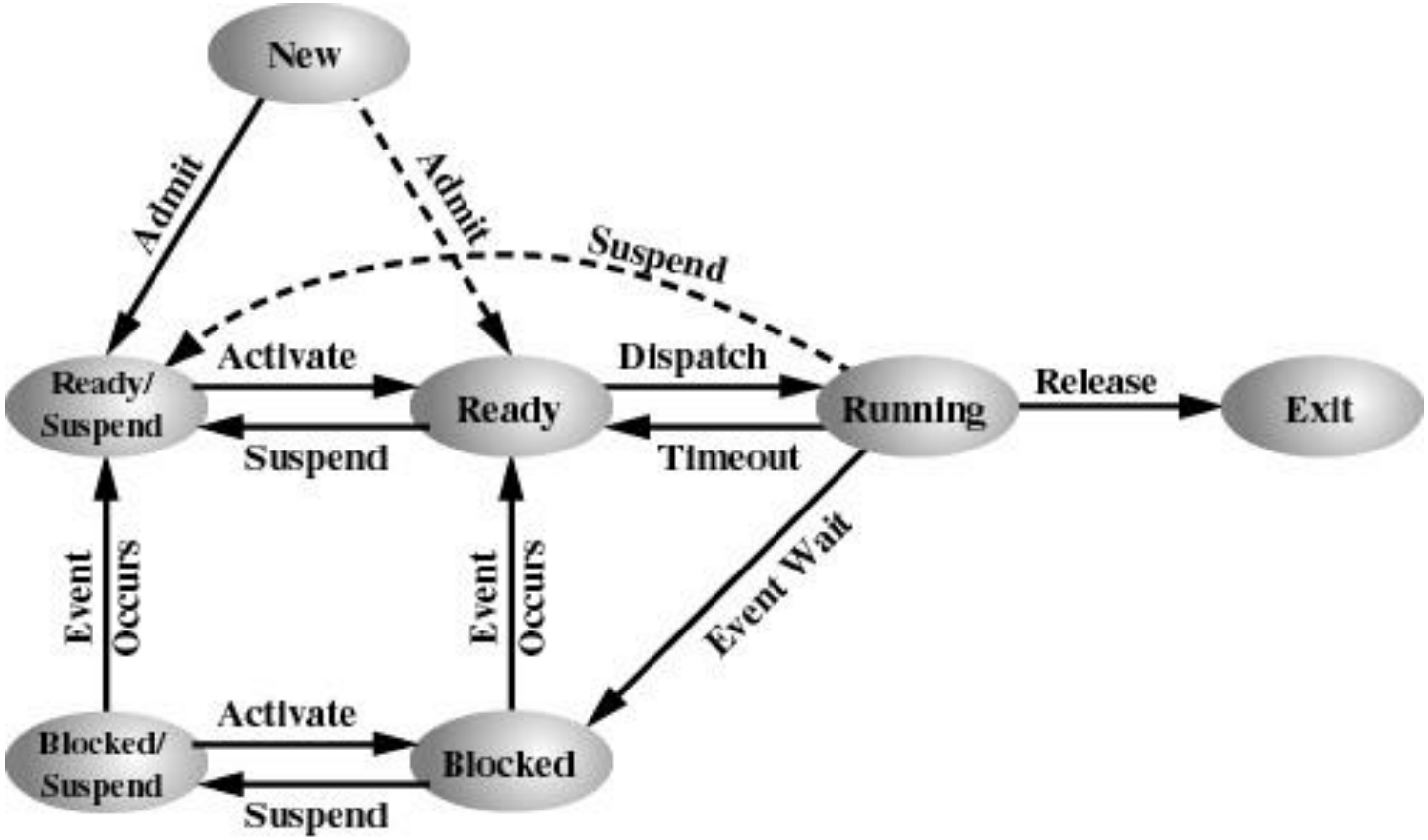
- *Swapping*
  - el S.O. necesita liberar suficiente memoria principal para ubicar un proceso que está listo para ejecutar
- Otra razón del sistema operativo
  - el S.O. puede suspender un proceso del que sospecha que puede estar causando problemas
- Petición interactiva del usuario
  - un usuario puede querer suspender un proceso por motivos de depuración, o relacionado con el uso de un recurso

TRANSICIÓN DE ESTADOS

Razones para la Suspensión de un Proceso:

- Temporización
  - un proceso puede ejecutarse periódicamente (monitorización o contabilidad) y suspenderse hasta el siguiente intervalo de tiempo
- Petición del proceso padre
  - un proceso puede suspender al descendiente para examinarlo, modificarlo o coordinar la actividad de varios

DIAGRAMA DE ESTADO DE PROCESOS



(b) With Two Suspend States

ESTADOS DE LOS PROCESOS

- Listo. Está en memoria principal disponible para ejecución
- Bloqueado. Está en memoria principal esperando un evento
- Listo suspendido. Está en memoria secundaria esperando un evento
- Bloqueado suspendido. Está en memoria secundaria disponible para ejecutarse

TRANSICIÓN DE ESTADOS

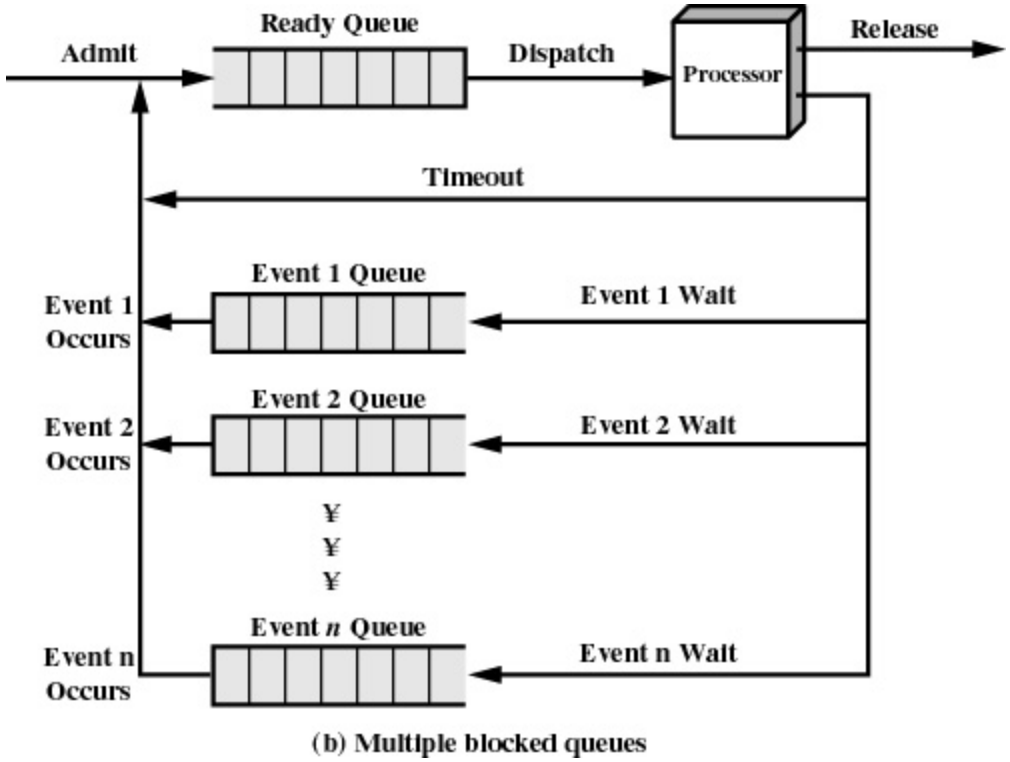
- Bloqueado =>Bloqueado suspendido. Si no hay procesos listos, un proceso bloqueado se pasa a disco para hacer espacio en memoria o para liberar memoria.
- Bloqueado suspendido => Listo suspendido. Sucede lo que estaba esperando.
- Listo suspendido => Listo. Si no hay procesos listos, necesitamos traer uno para ejecutarlo. También puede que un proceso listo suspendido tenga la mayor prioridad.
- Listo => Listo suspendido. Normalmente se suspenden procesos bloqueados, pero se puede preferir suspender un proceso listo de baja prioridad ante uno bloqueado de alta prioridad. Se busca obtener más memoria libre.

TRANSICIÓN DE ESTADOS

- Nuevo => Listo suspendido/Listo. Creamos el proceso cuando hay muchos bloqueados, o inmediatamente.
- Bloqueado suspendido => Bloqueado. Un proceso termina liberando memoria, uno de los bloqueados suspendidos tiene la mayor prioridad y el SO sospecha que el evento que espera sucederá en breve.
- Ejecutando => Listo/Suspendido. Un proceso de mayor prioridad despierta de bloqueado suspendido.
- Cualquier estado => Saliente.

LISTAS Y COLAS DE PROCESOS

- Lista de procesos del sistema (job queue): En esta lista están todos los procesos del sistema. Al crearse un nuevo proceso se agrega el PCB a esta lista. Cuando el proceso termina su ejecución es borrado.
- Cola de procesos listos (ready queue): Esta cola se compondrá de los procesos que estén en estado listo. La estructura de esta cola dependerá de la estrategia de planificación utilizada.
- Cola de espera de dispositivos (device queue): Los procesos que esperan por un dispositivo de E/S particular, son agrupados en una lista específica al dispositivo. Cada dispositivo de E/S tendrá su cola de espera.



(b) Multiple blocked queues

CONTROL DE PROCESOS

Modos de Ejecución de Procesos

- Modo usuario
  - modo con menos privilegios
  - los programas de usuario normalmente se ejecutan en este modo
- Modo del sistema, modo de control o modo *kernel*
  - modo con más privilegios
  - *kernel* del sistema operativo

El proceso está compuesto por la parte usuario (lo que el usuario implementa) y la parte de núcleo (las rutinas del núcleo que utiliza)

CONTROL DE PROCESOS

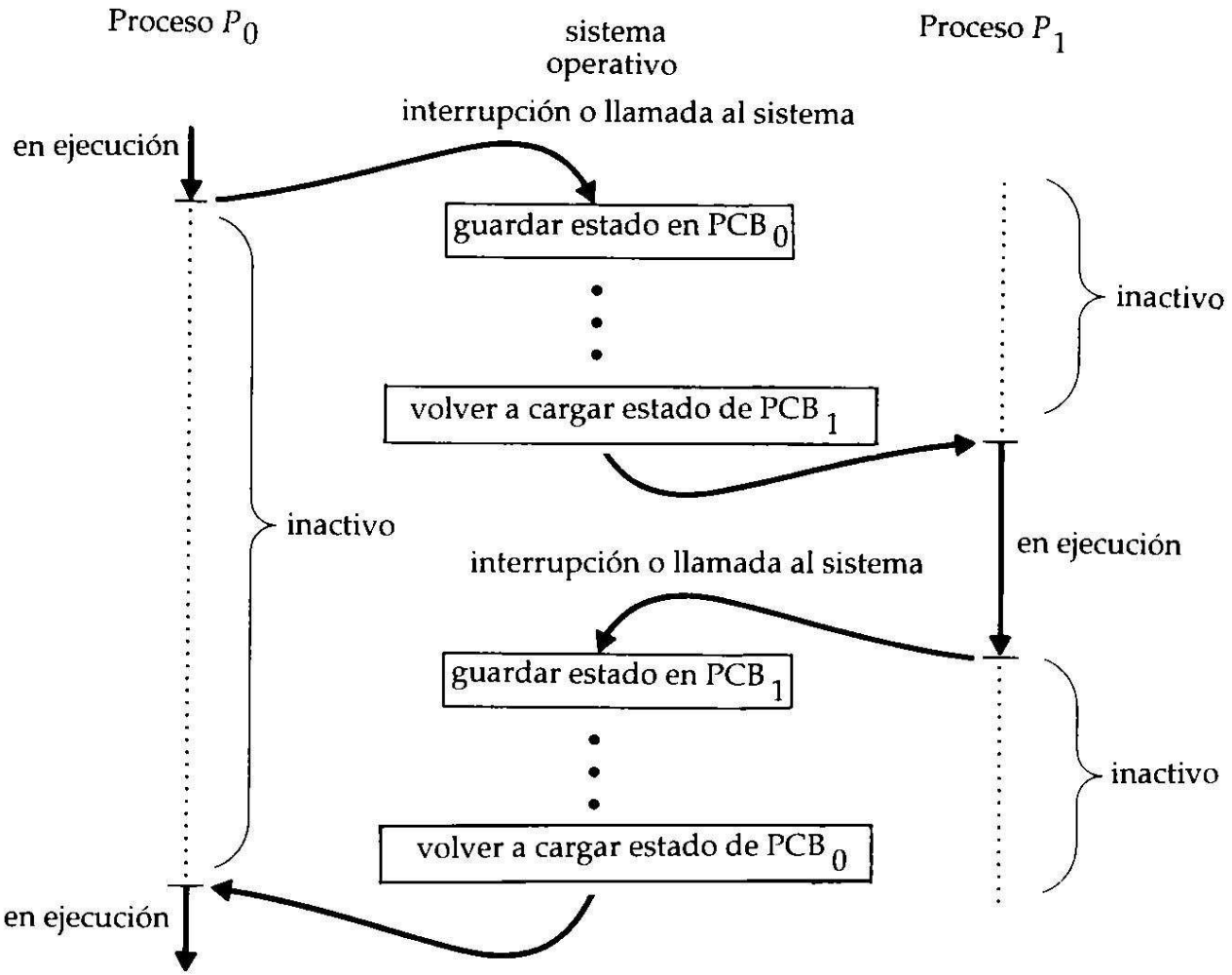
Mecanismo de cambio de modo

1. El PC pasa a apuntar a la rutina de tratamiento de la excepción
2. Cambia de modo usuario a modo núcleo
3. El S.O. guarda el contexto del proceso interrumpido en su PCB

Mecanismo de cambio de proceso

1. Salvar el estado del procesador (PC, registros...)
2. Actualizar campos PCB (estado, contabilidad, auditoría...)
3. Mover el PCB a la cola apropiada (Listo, bloqueado..)
4. Seleccionar el nuevo proceso a ejecutar
5. Actualizar el PCB del proceso elegido (estado Ejecutando...)
6. Actualizar estructuras de datos de gestión de memoria
7. Restaurar el estado del procesador cuando se interrumpió el nuevo proceso

CONTROL DE PROCESOS

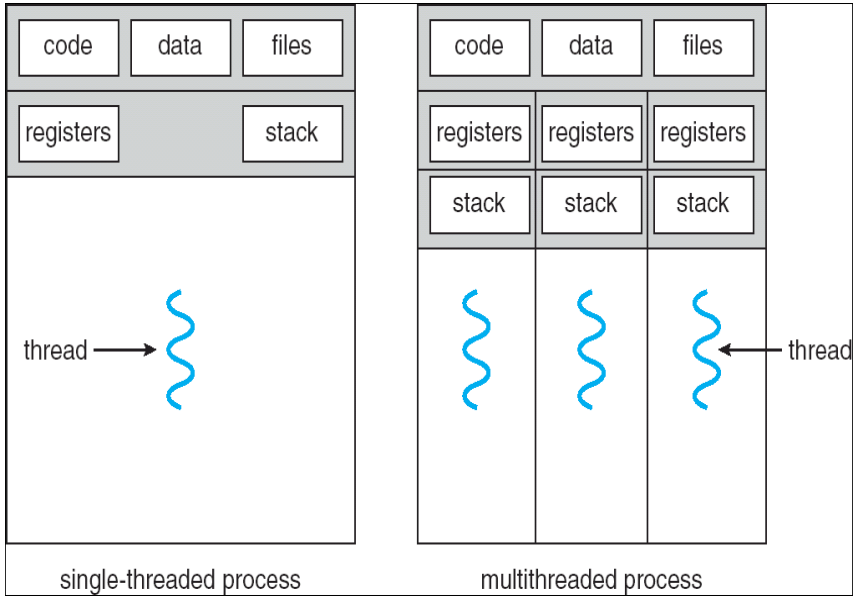


CUÁNDO CONMUTAR PROCESOS

- Interrupción del reloj (timer)
  - el proceso se ha ejecutado durante el máximo periodo de tiempo permitido (time slice)
- Interrupción E/S
- Fallo de memoria
  - la dirección de memoria está en la memoria virtual y hay que traerla a la memoria principal
- Trap (excepción)
  - ocurre un error puede causar que un proceso se mueva al estado de Exit
- Llamada al Supervisor (al sistema)
  - tal como abrir un fichero

HILOS O THREADS

Todos los recursos, sección de código y datos son compartidos por los distintos *threads* de un mismo proceso.



Elementos por hilo	**Elementos por proceso**
Contador de programa	Espacio de direcciones
Pila	Variables globales
Estado + contexto	Ficheros abiertos
Memoria privada	Procesos hijos
	Cronómetros
	Señales
	Semáforos
	Información contable

VENTAJAS DEL USO DE THREADS

Repuesta: Desarrollar una aplicación con varios hilos de control (*threads)* permite tener un mejor tiempo de respuesta.
Compartir recursos: Los *threads* de un proceso comparten la memoria y los recursos que utilizan. A diferencia de *IPC,* no es necesario acceder al *kernel* para comunicar o sincronizar los hilos de ejecución.
Economía: Es más fácil un cambio de contexto entre *threads* ya que no es necesario cambiar el espacio de direccionamiento. A su vez, es más "liviano" para el sistema operativo crear un *thread* que crear un proceso nuevo.
Utilización de arquitecturas con multiprocesadores: Disponer de una arquitectura con más de un procesador permite que los *threads* de un mismo proceso ejecuten en forma paralela