

# ***Introducción a los Sistemas Operativos***

## **Procesos - II**

### **Profesores:**

Lía Molinari

Juan Pablo Pérez

Nicolás del Río



- ✓ Versión: Agosto 2019
- ✓ Palabras Claves: Procesos, Estados, Scheduler, Long Term, Medium Term, Short Term

Los temas vistos en estas diapositivas han sido mayormente extraídos del libro de William Stallings (Sistemas Operativos: Aspectos internos y principios de diseño) y del libro de Silberschatz (Operating Systems Concepts)



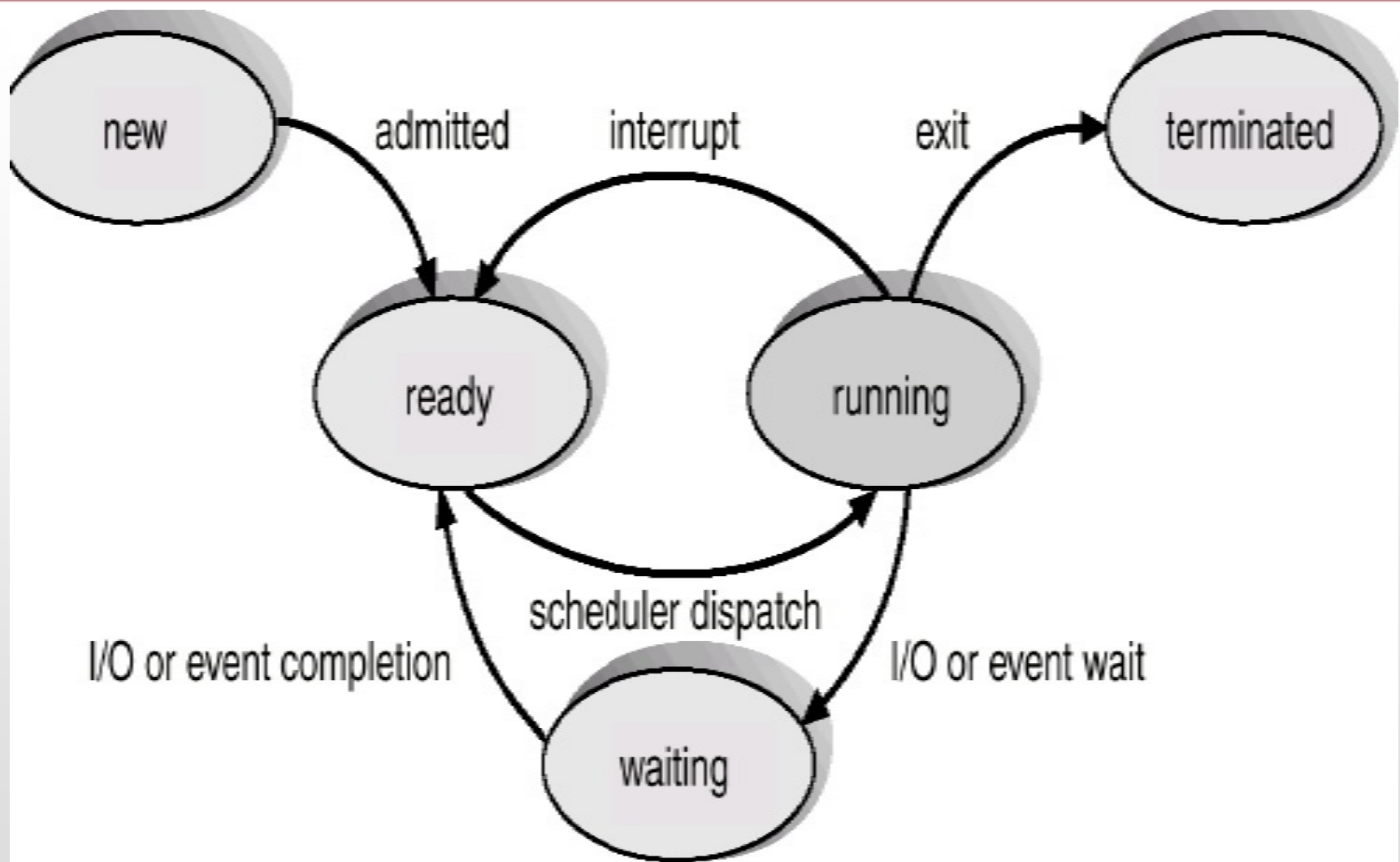
# Estados de un proceso

En su ciclo de vida, un proceso pasa por diferentes estados.

- ✓ Nuevo (new)
- ✓ Listo para ejecutar (ready)
- ✓ Ejecutándose (running)
- ✓ En espera (waiting)
- ✓ Terminado (terminated)



# Estados de un proceso (cont.)

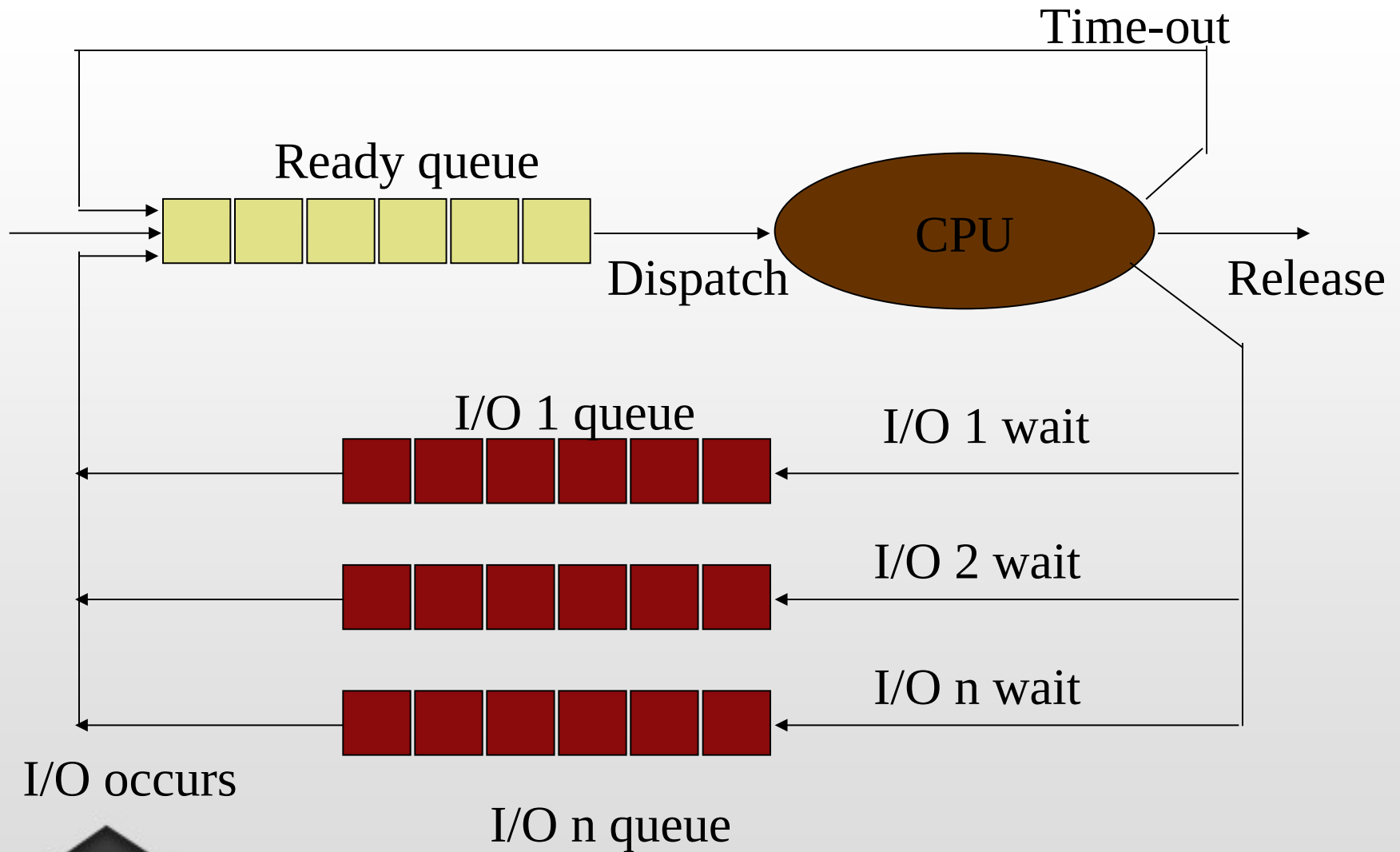


# Colas en la planificación de procesos

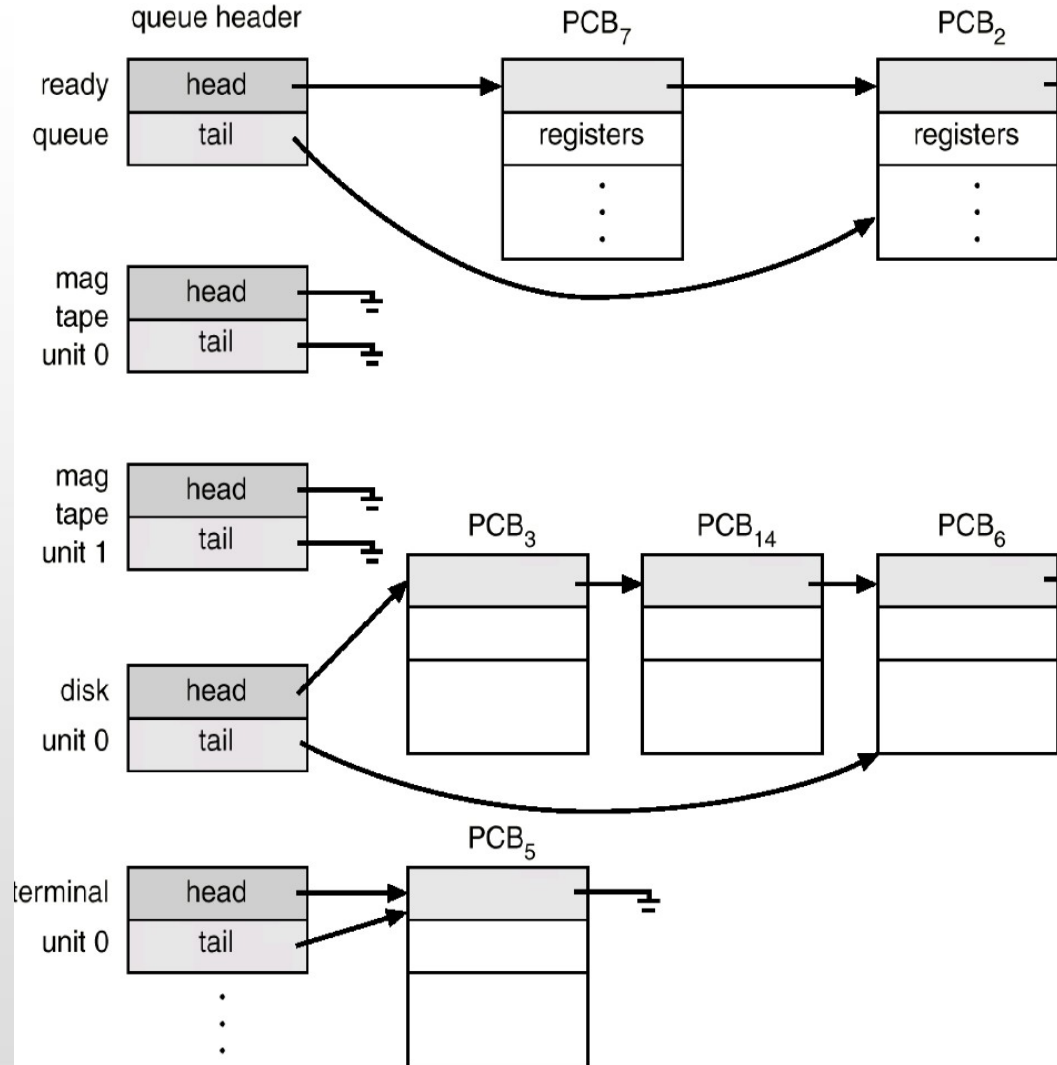
- ✓ Para realizar la planificación, el SO utiliza la PCB de cada proceso como una abstracción del mismo
- ✓ Las PCB se enlazan en Colas siguiendo un orden determinado
- ✓ Ejemplos
  - ✓ Cola de trabajos o procesos
    - ✓ Contiene todas las PCB de procesos en el sistema
  - ✓ Cola de procesos listos
    - ✓ PCB de procesos residentes en memoria principal esperando para ejecutarse
  - ✓ Cola de dispositivos
    - ✓ PCB de procesos esperando por un dispositivo de I/O



# Colas en la planificación de procesos (cont.)



# Colas en la planificación de procesos (cont.)



# Módulos de la planificación

- ✓ Son módulos (SW) del Kernel que realizan distintas tareas asociadas a la planificación.
- ✓ Se ejecutan ante determinados eventos que así lo requieren:
  - ✓ Creación/Terminación de procesos
  - ✓ Eventos de Sincronización o de E/S
  - ✓ Finalización de lapso de tiempo
  - ✓ Etc





# *Módulos de la planificación (cont.)*

- ✓ Scheduler de long term
- ✓ Scheduler de short term
- ✓ Scheduler de medium term

Su nombre proviene de la frecuencia de ejecución.



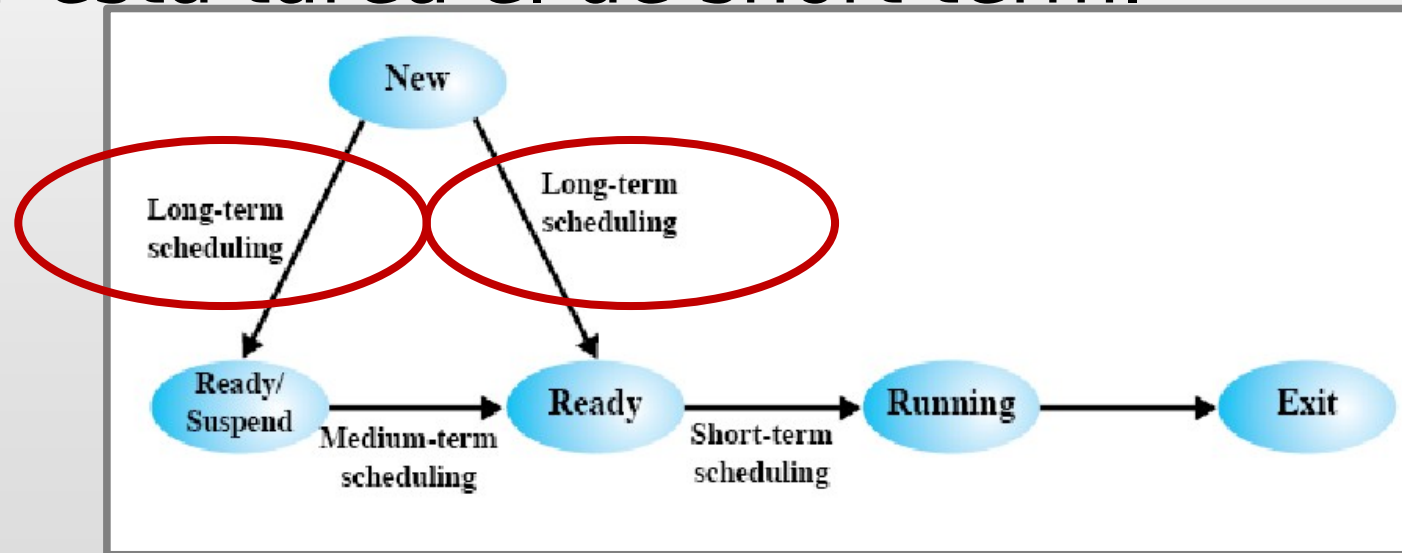
# Módulos de la planificación (cont.)

- ✓ Otros módulos: **Dispatcher** y **Loader**.
- ✓ Pueden no existir como módulos separados de los schedulers vistos, pero la función debe cumplirse.
- ✓ **Dispatcher**: hace cambio de contexto, cambio de modo de ejecución..."despacha" el proceso elegido por el *Short Term* (es decir, "salta" a la instrucción a ejecutar).
- ✓ **Loader**: carga en memoria el proceso elegido por el *long term*.



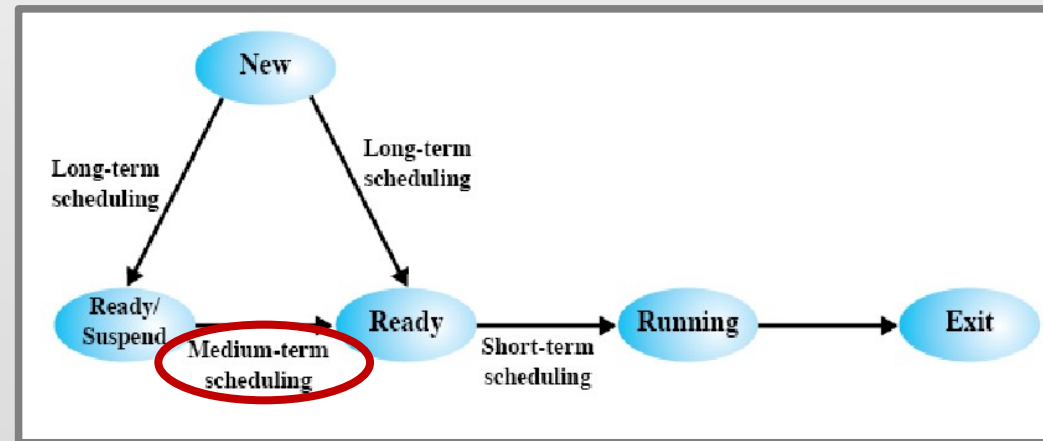
# Long term Scheduler

- ✓ Controla el *grado de multiprogramación*, es decir, la cantidad de procesos en memoria.
- ✓ Puede no existir este scheduler y absorber esta tarea el de short term.



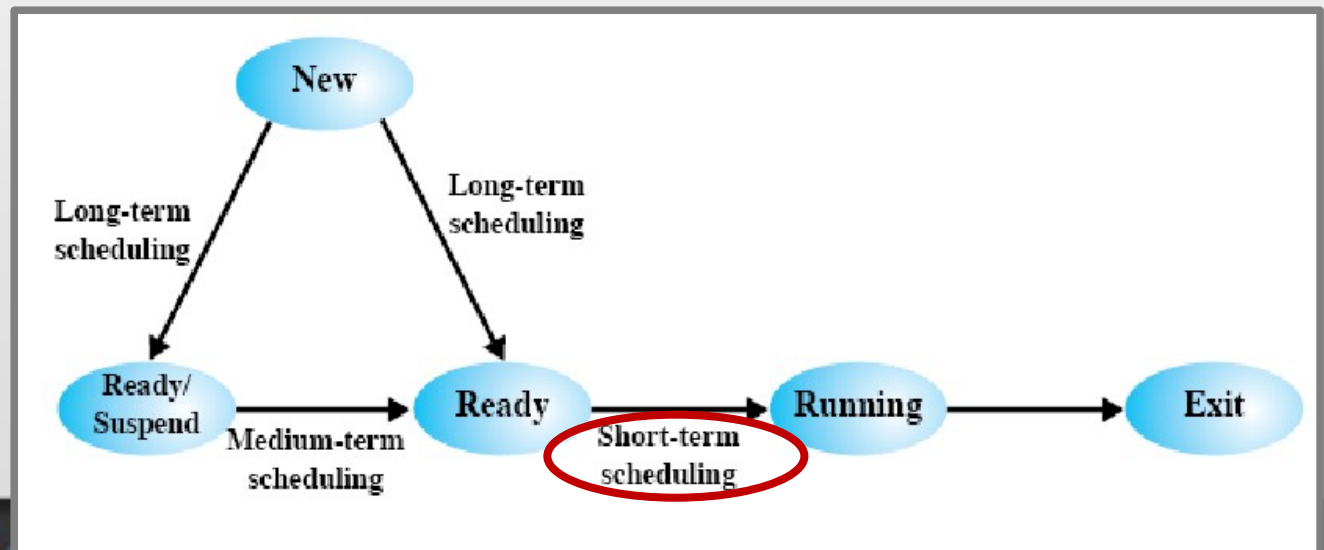
# Medium Term Scheduler (swapping)

- ✓ Si es necesario, reduce el grado de multiprogramación
- ✓ Saca temporalmente de memoria los procesos que sea necesario para mantener el equilibrio del sistema.
- ✓ Términos asociados: *swap out* (sacar de memoria), *swap in* (volver a memoria).

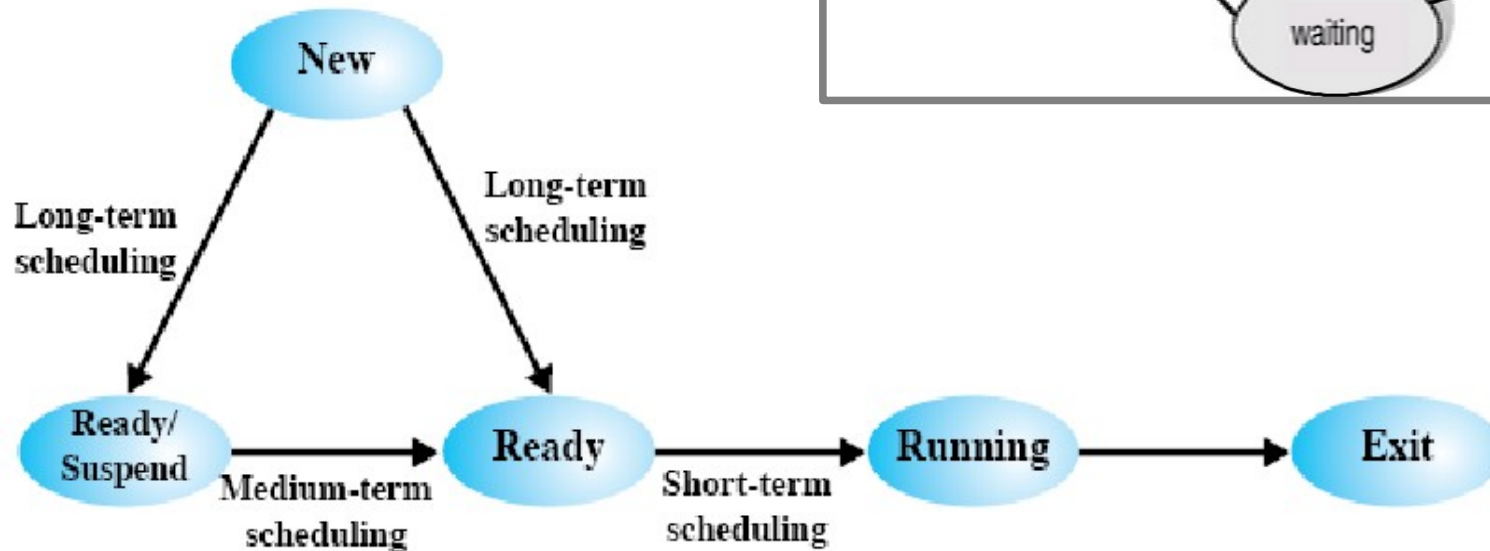
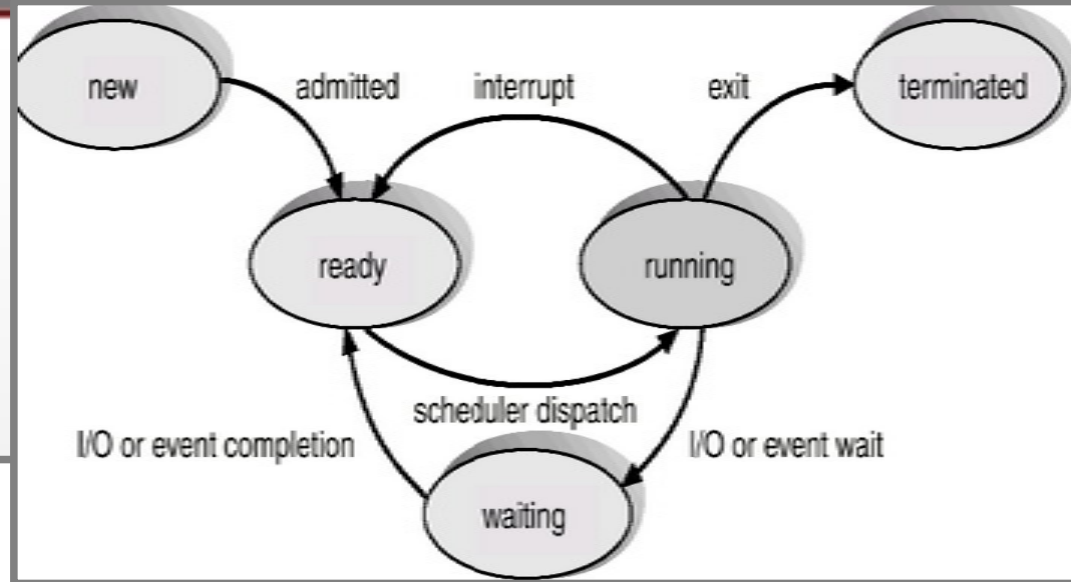


# Short Term Scheduler

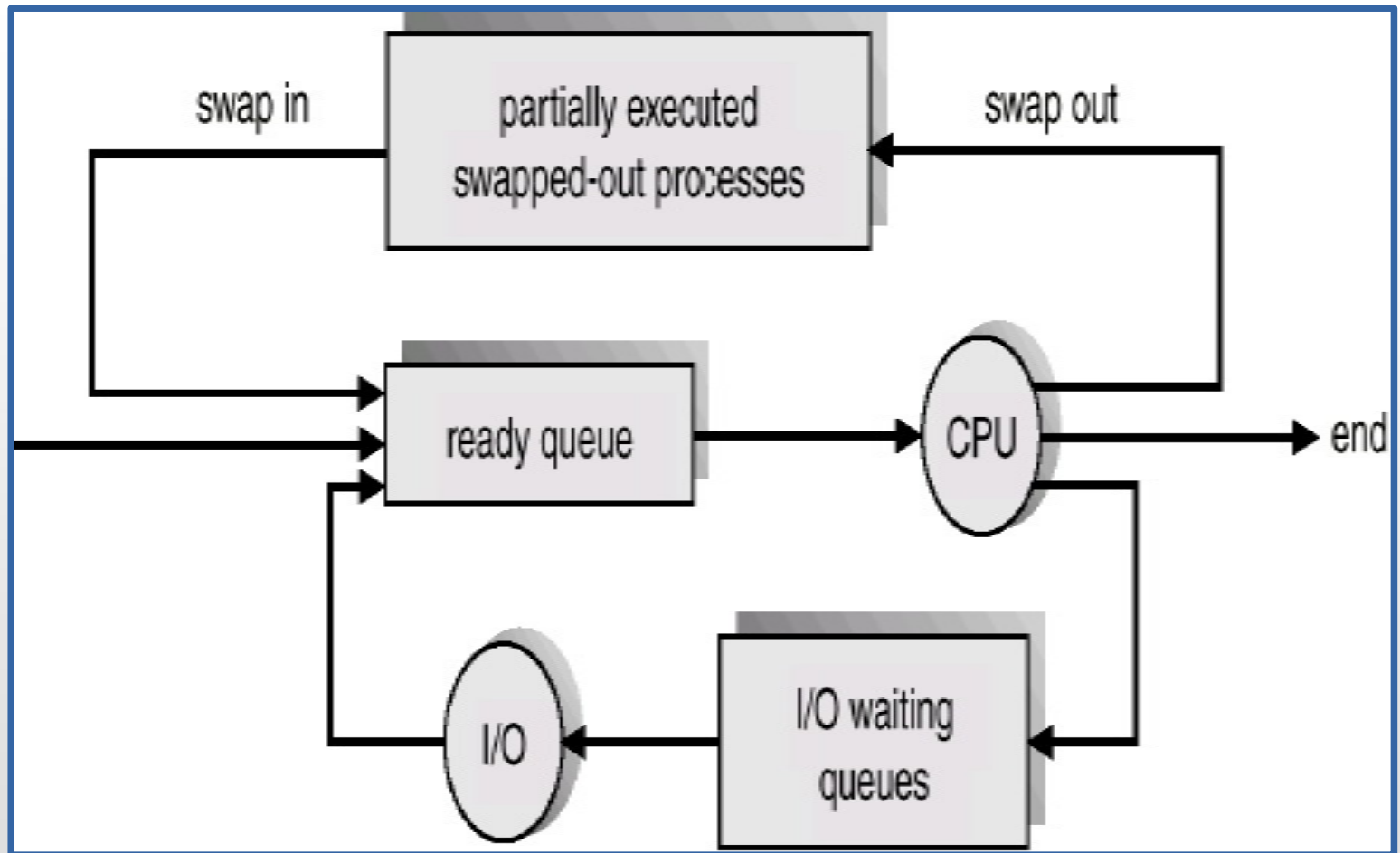
- ✓ Decide a cuál de los procesos en la cola de listos se elige para que use la CPU.
- ✓ Términos asociados: apropiativo, no apropiativo, algoritmo de scheduling



# Estados y schedulers

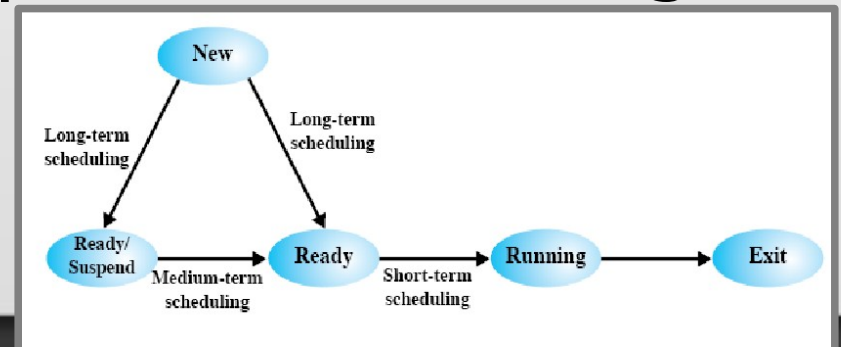


# Procesos en espera y swapeados



# Sobre el estado nuevo (new)

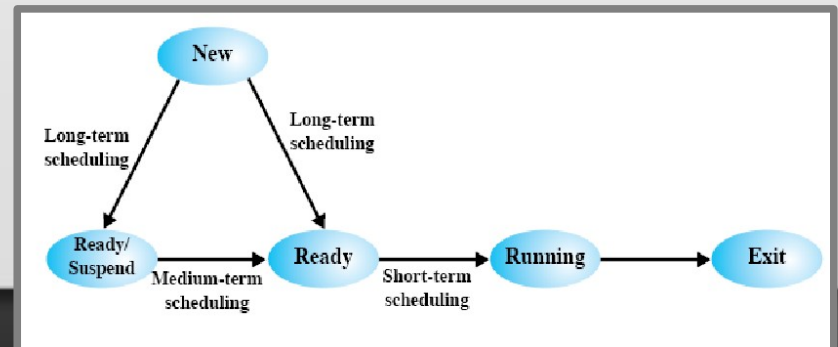
- ✓ Un usuario “dispara” el proceso. Un proceso es creado por otro proceso: su proceso padre.
- ✓ En este estado se crean las estructuras asociadas, y el proceso queda en la *cola de procesos*, normalmente en espera de ser cargado en memoria





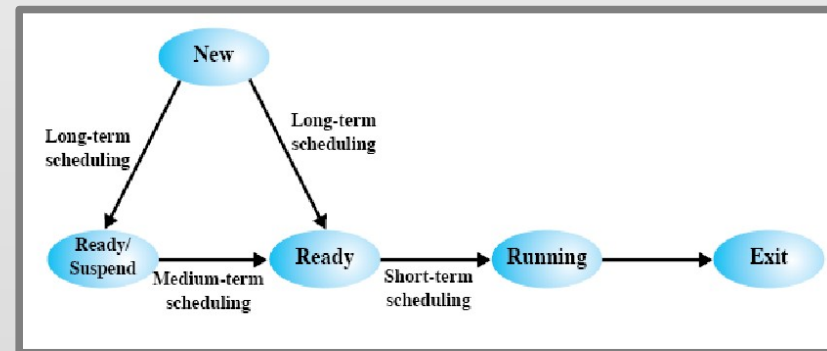
# Sobre el estado listo (ready)

- ✓ Luego que el scheduler de largo plazo eligió al proceso para cargarlo en memoria, el proceso queda en estado listo
- ✓ El proceso sólo necesita que se le asigne CPU
- ✓ Está en la cola de procesos listos (ready queue)



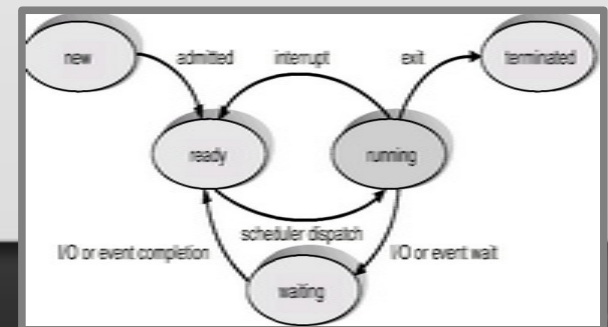
# Sobre el estado en ejecución (running)

- ✓ El scheduler de corto plazo lo eligió para asignarle CPU
- ✓ Tendrá la CPU hasta que se termine el período de tiempo asignado (quantum o time slice), termine o hasta que necesite realizar alguna operación de E/S



# *Sobre el estado de espera (waiting)*

- ✓ El proceso necesita que se cumpla el evento esperado para continuar.
- ✓ El evento puede ser la terminación de una E/S solicitada, o la llegada de una señal por parte de otro proceso.
- ✓ Sigue en memoria, pero no tiene la CPU.
- ✓ Al cumplirse el evento, pasará al estado de listo.



# Transiciones

- ✓ **New-Ready:** Por elección del scheduler de largo plazo (carga en memoria)
- ✓ **Ready-Running:** Por elección del scheduler de corto plazo (asignación de CPU)
- ✓ **Running-Waiting:** el proceso “se pone a dormir”, esperando por un evento.
- ✓ **Waiting-Ready:** Terminó la espera y compite nuevamente por la CPU.

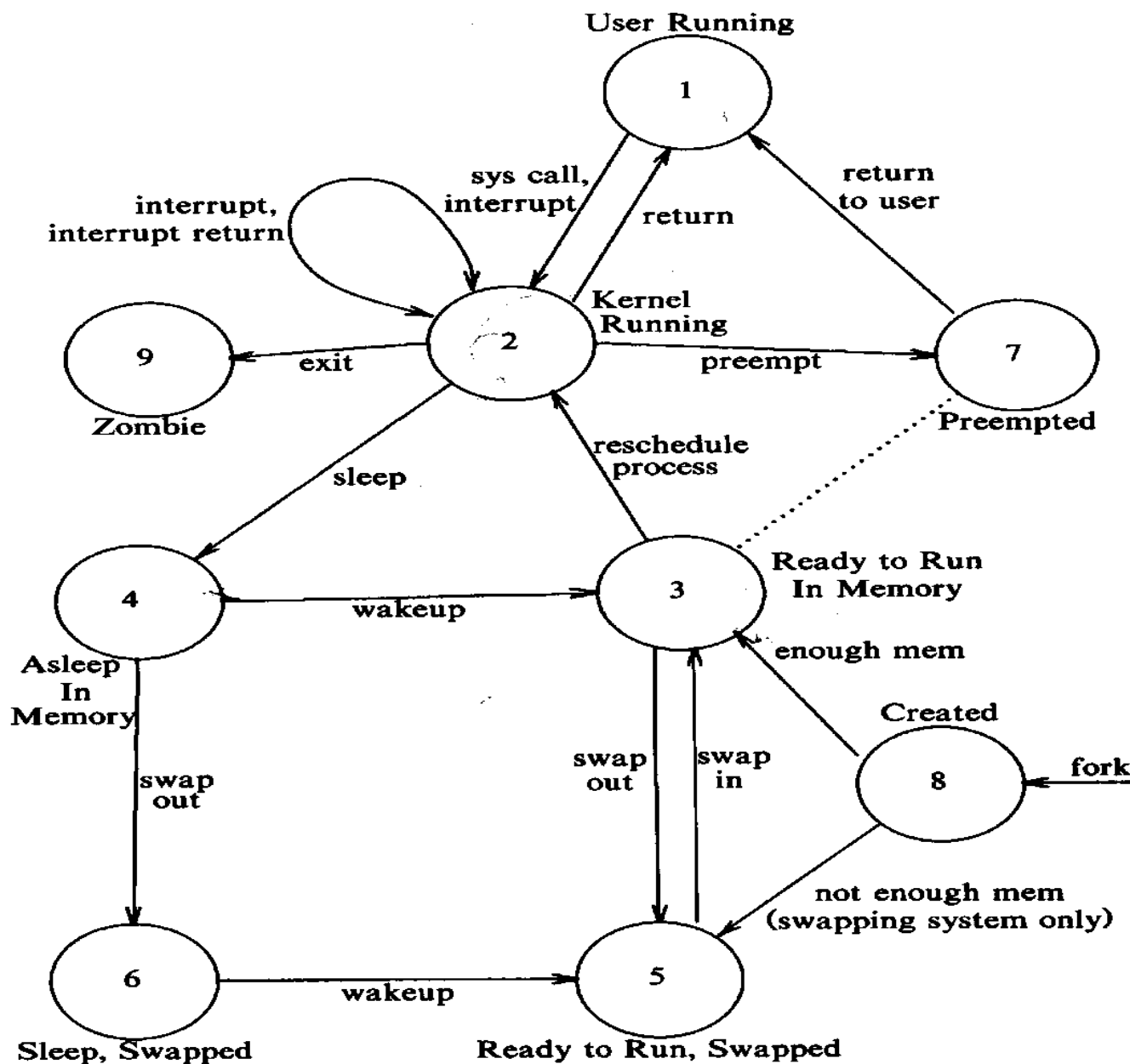


# Caso especial: running-ready

- ✓ Cuando el proceso termina su quantum (franja de tiempo) sin haber necesitado ser interrumpirlo por un evento, pasa al estado de ready, para competir por CPU, *pues no está esperando por ningún evento...*
- ✓ Se trata de un caso distinto a los anteriores, porque el procesos es expulso de la CPU contra su voluntad
- ✓ Esta situación se da en algoritmos apropiativos



## Diagrama incluyendo swapping



**Figure 6.1. Process State Transition Diagram**

# *Explicación por estado*

- ✓ 1. Ejecución en modo usuario
- ✓ 2. Ejecución en modo kernel
- ✓ 3. El proceso está listo para ser ejecutado cuando sea elegido.
- ✓ 4. Proceso en espera en memoria principal.
- ✓ 5. Proceso listo, pero el swapper debe llevar al proceso a memoria ppal antes que el kernel lo pueda elegir para ejecutar.



# Explicación por estado (cont.)

- ✓ 6. Proceso en espera en memoria secundaria.
- ✓ 7. Proceso retornando desde el modo kernel al user. Pero el kernel se apropia, hace un context switch para darle la CPU a otro proceso.
- ✓ 8. Proceso recientemente creado y en transición: existe, pero aun no está listo para ejecutar, ni está dormido.
- ✓ 9. El proceso ejecutó la system call *exit* y *está en estado zombie*. Ya no existe más, pero se registran datos sobre su uso, código resultante del exit. Es el estado final.





# Diagrama de transiciones UNIX

