

Introducción a los Sistemas Operativos

Administración de Procesos

Práctica 4



- Programa en ejecución
- Los conceptos de tarea, Job y proceso hacen referencia a lo mismo
- Según su historial de ejecución, los podemos clasificar:
 - CPU Bound (ligados a la CPU)
 - I/O Bound (ligados a entrada/salida)



Programa

- ✓ Es estático
- ✓ No tiene program counter
- ✓ Existe desde que se edita hasta que se borra



Proceso

- ✓ Es dinámico
- ✓ Tiene program counter
- ✓ Su ciclo de vida comprende desde que se lo “dispara” hasta que termina



Procesos - PCB - Process Control Block

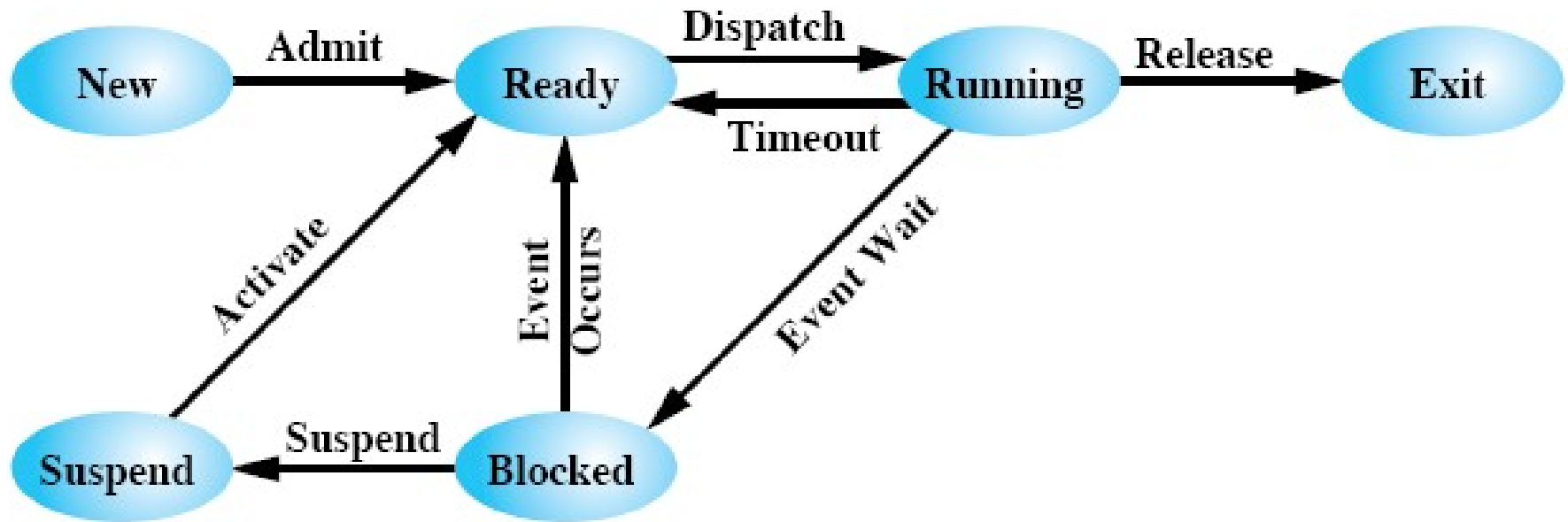
Identifier
State
Priority
Program counter
Memory pointers
Context data
I/O status information
Accounting information
⋮

- ✓ Una por proceso
- ✓ Tiene información de cada proceso
- ✓ Es lo primero que se crea cuando se crea un proceso y lo último que se borra cuando termina



Procesos (cont.) - Estados

En su ciclo de vida, el proceso pasa por diferentes estados



Objetivos del planificador

- ❑ Es la clave de la multiprogramación.
- ❑ Está diseñado de manera apropiada para cumplir las metas de:
 - Menor Tiempo de Respuesta
 - Mayor rendimiento
 - Uso eficiente del procesador

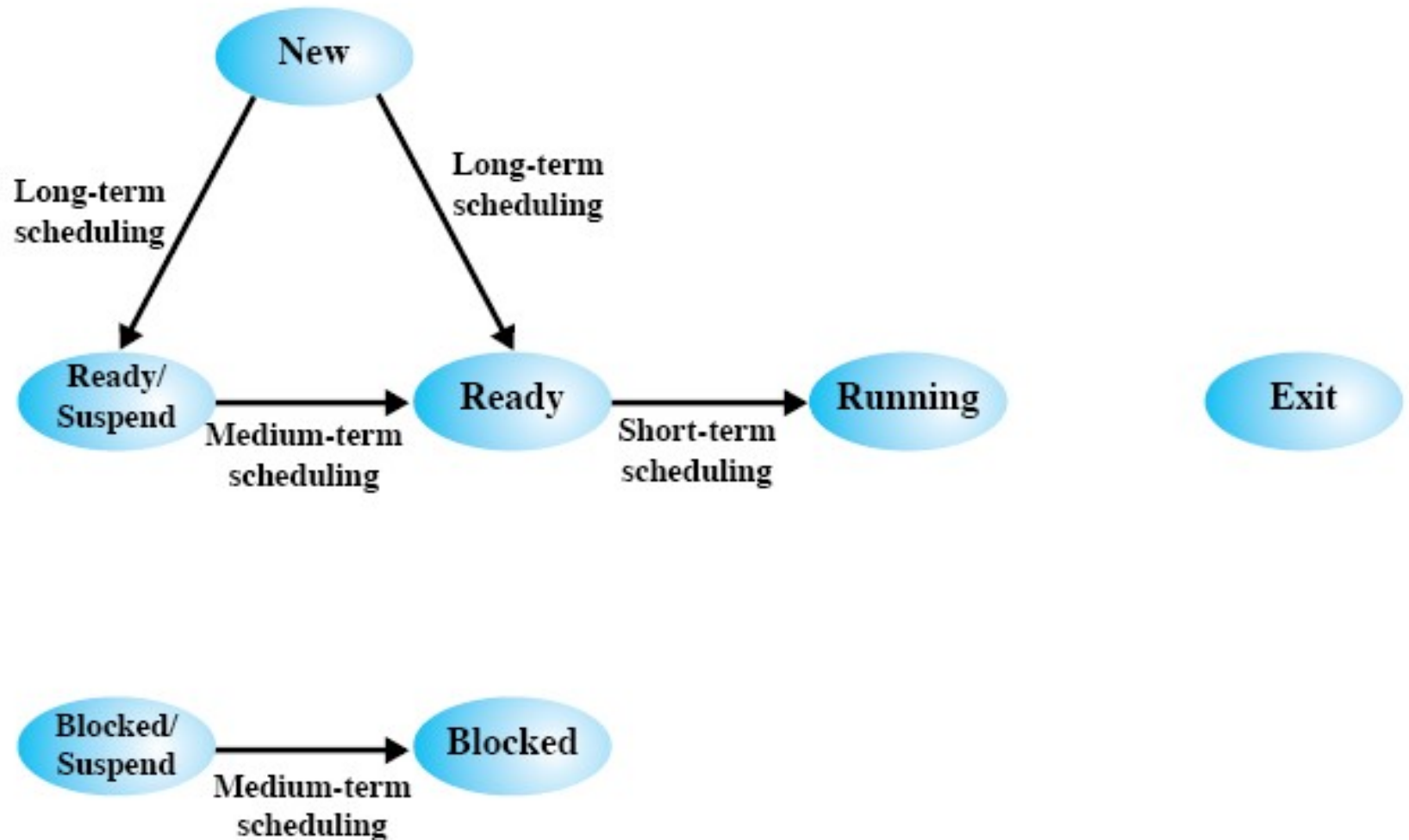


Planificadores

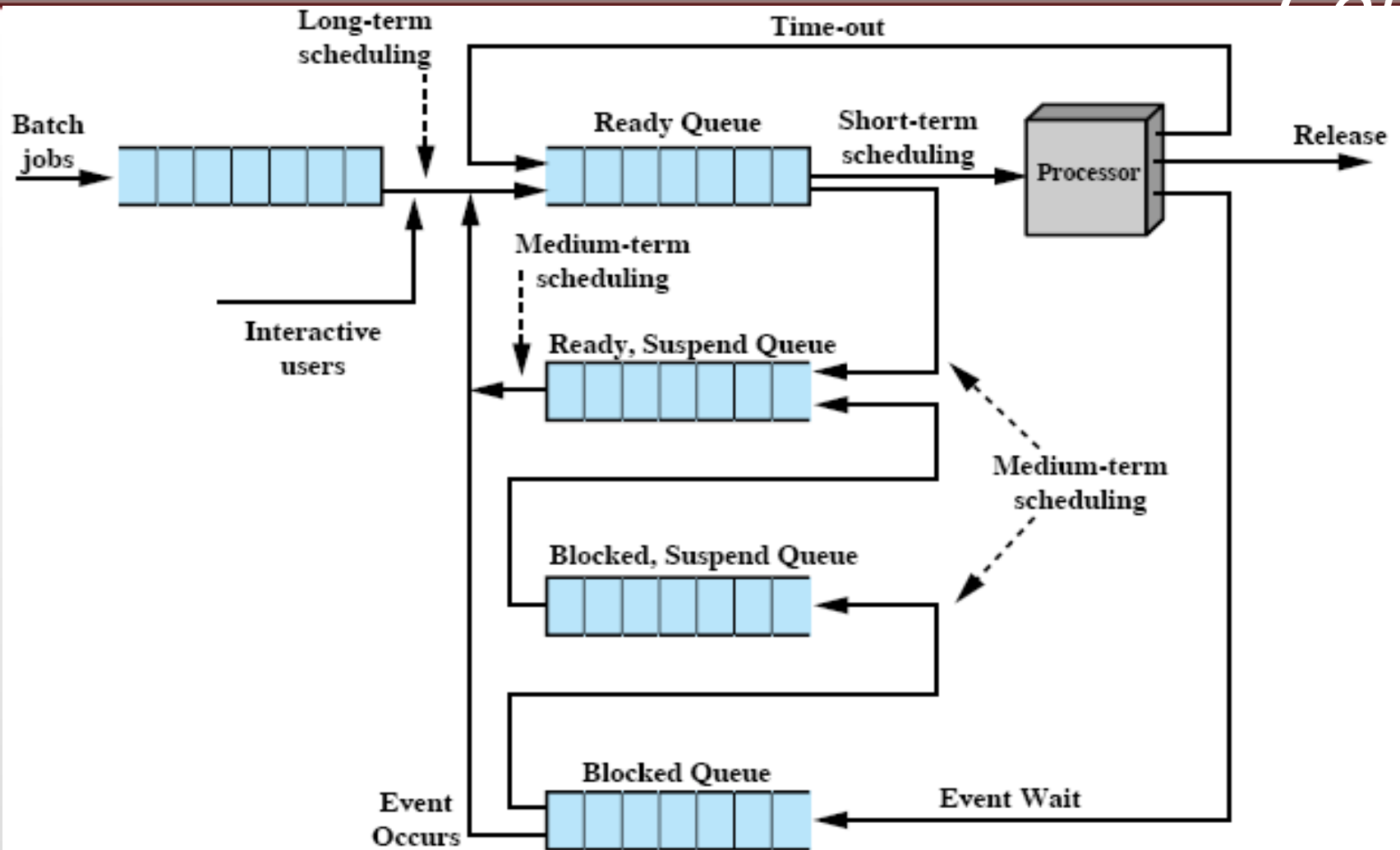
- ❑ Long term scheduler: Admite nuevos procesos a memoria (controla el grado de multiprogramación)
- ❑ Medium term scheduler: Swapping (intercambio) entre disco y memoria cuando el SO lo determina (puede disminuir el grado de multiprogramación)
- ❑ Short term scheduler: Que proceso listo se ejecuta



Relación entre planificadores y Estados



Relación entre planificadores y Colas



Tiempos de los procesos

- Retorno
 - Tiempo que transcurre entre que el proceso llega al sistema hasta que completa su ejecución
- Espera
 - Tiempo que el proceso se encuentra en el sistema esperando (sin ejecutarse) ($TR - T_{cpu}$)
- Promedios
 - Promedios de los anteriores



Apropiación vs. No apropiación

- Nonpreemptive
 - Una vez que un proceso esta en estado de ejecución, continua hasta que termina o se bloquea por algún evento (por ej. I/O).
- Preemptive
 - El proceso en ejecución puede ser interrumpido y llevado a la cola de listos por el SO.
 - Mayor overhead pero mejor servicio
 - Un proceso no monopoliza el procesador.



Algoritmos de planificación - *First-Come-First-Served* (FCFS)

- ❑ Cada proceso se coloca en la cola de listos
- ❑ Cuando hay que elegir un proceso para ejecutar, se selecciona el mas viejo en la cola de listos (FIFO).
- ❑ No favorece a ningún tipo de procesos, porque se van a ir ejecutando en orden de llegada, pero en principio podríamos decir que los CPU Bound terminan en su primer ráfaga, mientras que los I/O bound necesitan mas ráfagas (por su naturaleza)



Scheduling – Ejemplo

Job	Inst. Llegada	CPU	Prioridad
1	0	9	3
2	1	5	2
3	2	3	1
4	3	7	2

Recordemos: En FCFS el criterio de selección es el orden de llegada!

#Ejemplo 1

**TAREA “1” PRIORIDAD=3 INICIO=0
[CPU, 9]**

**TAREA “2” PRIORIDAD=2 INICIO=1
[CPU, 5]**

**TAREA “3” PRIORIDAD=1 INICIO=2
[CPU, 3]**

**TAREA “4” PRIORIDAD=2 INICIO=3
[CPU, 7]**

¿Tiempos de Retorno y Espera?



- ❑ Política nonpreemptive que selecciona el proceso mas corto primero.
- ❑ Procesos cortos se colocan delante de procesos largos.
- ❑ Los procesos largos pueden sufrir starvation (Inanición).
- ❑ Veamos el ejemplo 1 nuevamente



- Política basada en un reloj
- Quantum: Medida que determina cuanto tiempo podrá usar el procesador cada proceso.
 - Pequeño: Overhead de Context Switch
 - Grande: ¿Pensar?
- Cuando un proceso es expulsado de la CPU es colocado al final de la Ready Queue y se selecciona otro (FIFO Circular)



- ❑ Recordar: Cada proceso se ejecuta durante una fracción de tiempo \square QUANTUM (Q)
- ❑ Existe “contador” que indica las unidades de CPU en las que se ejecuto. Cuando el mismo llega a 0 (cero) el proceso es expulsado.
- ❑ Existen 2 variantes con respecto al valor inicial del “contador” cuando un proceso es asignado a la CPU
 - TIMER VARIABLE
 - TIMER FIJO



- El “contador” se inicializa en Q
 contador := Q
 cada vez que un proceso es
 asignado a la CPU.
- Este Esquema:
 - Mas utilizado en los algoritmos RR
 - Utilizado por el simulador

Veamos el ejemplo 1 nuevamente

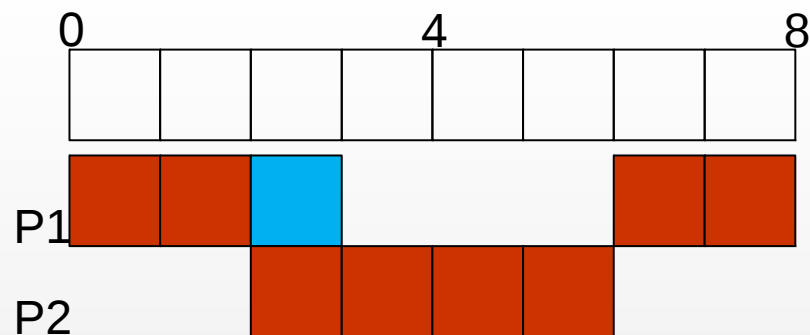


- El “contador” se inicializa a Q **solo** cuando su valor es 0 (cero)
 - if (contador == 0) contador = Q;
- Es como si el “contador” se compartiera entre los procesos
- Ejemplo (Quantum = 4)
 - P1 toma la CPU y se ejecuta por 2 unidades
 - P2 (al dejar P1 la CPU) comienza con el contador = 2





Algoritmos de planificación - Round Robin Timer Fijo vs. Timer Variable

Timer Variable

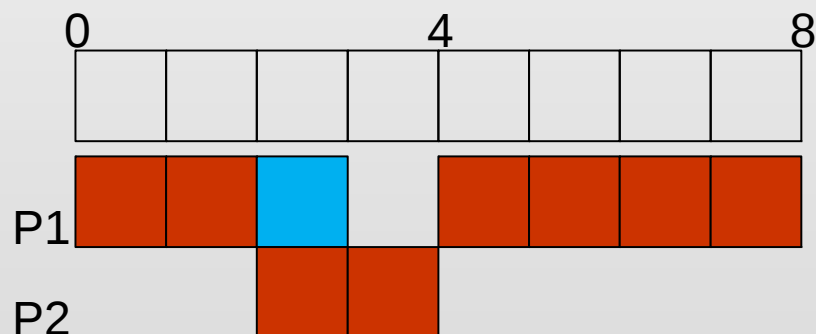


Round Robin, $Q=4$

 = E/S

 = Uso de CPU

Timer Fijo

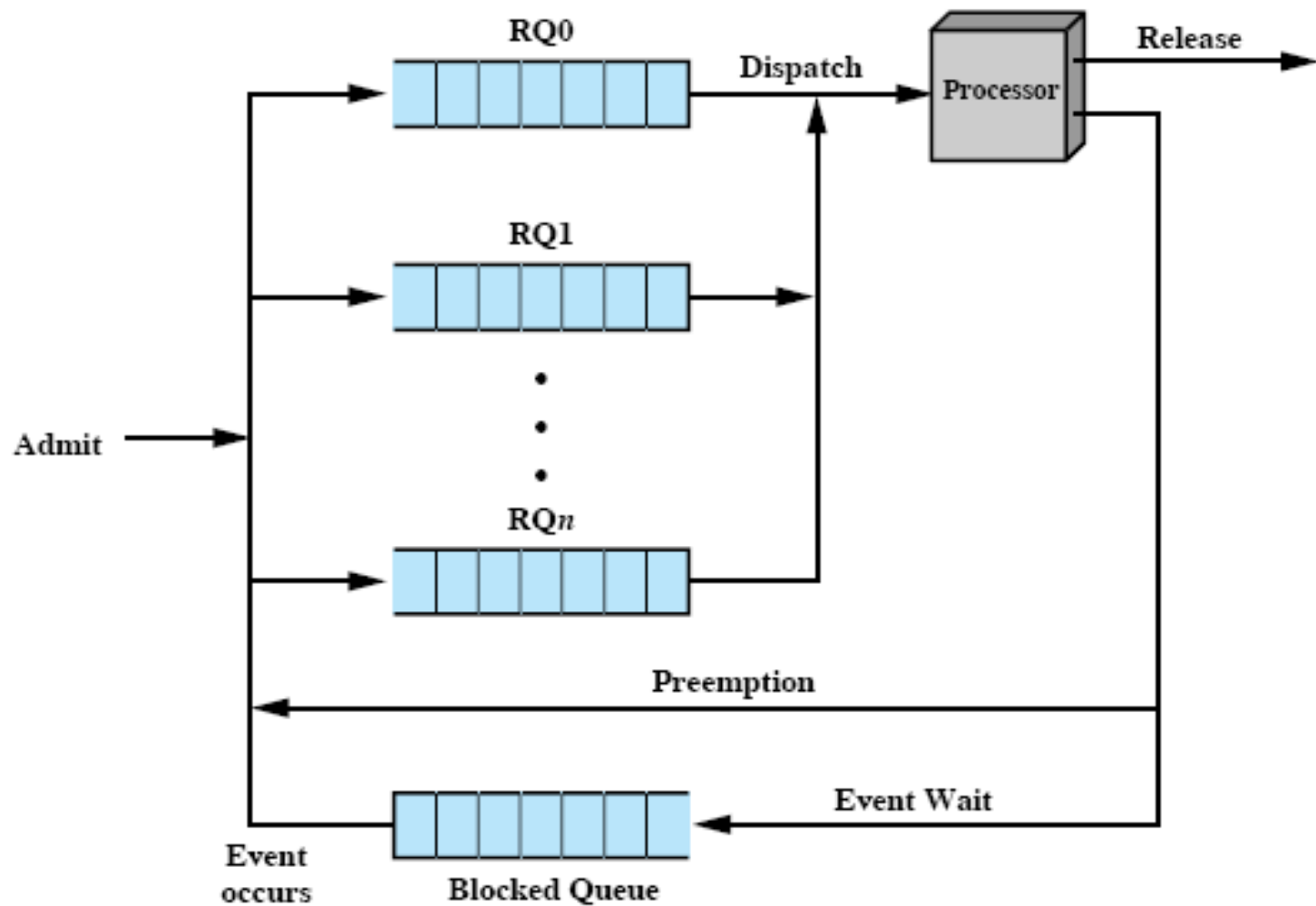


- ❑ Cada proceso tiene un valor que representa su prioridad
- ❑ Scheduler selecciona el proceso de mayor prioridad de los que se encuentran en la Ready Queue.
- ❑ Para simplificar □ Una Ready Queue por cada nivel de prioridad
- ❑ Procesos de Baja Prioridad pueden sufrir starvation (Inanición)
 - Solución: Permitir a un proceso cambiar su prioridad durante su ciclo de vida.
- ❑ Puede ser preemptive

Veamos el ejemplo 1 nuevamente



Algoritmos de planificación - Uso de Prioridades - Varias colas



- ❑ Versión Preemptive de SJF
- ❑ Selecciona el proceso al cual le resta menos tiempo de ejecución.
- ❑ ¿A que tipo de procesos favorece?

❑ I/O Bound

Veamos el ejemplo 1 nuevamente



- Ciclo de Vida de un proceso
 - Uso de CPU + Operaciones de I/O
- Cada dispositivo tiene su cola de procesos en espera
- I/O Scheduler (FCFS, SJF, etc.)
- Vamos a considerar I/O independiente de la CPU
- Uso de CPU + Operaciones de I/O en simultaneo



Algoritmos de planificación – Cada proceso, un recurso

#Ejemplo 2

RECURSO "R1"

RECURSO "R2"

RECURSO "R3"

TAREA "1" INICIO=0

[CPU, 3] [1, 2] [CPU, 2]

TAREA "2" INICIO=1

[CPU, 2] [2, 2] [CPU, 2]

TAREA "3" INICIO=2

[CPU, 2] [3, 3] [CPU, 1]

Job	Inst. Llegada	CPU	E/S (Rec, Inst, dur)
1	0	5	(R1, 3, 2)
2	1	4	(R2, 2, 2)
3	2	3	(R3, 2, 3)



#Ejemplo 3

RECURSO "R1"

RECURSO "R2"

TAREA "1" INICIO=0

[CPU, 3] [1, 3] [CPU, 2]

TAREA "2" INICIO=1

[CPU, 1] [1, 2] [CPU, 3]

TAREA "3" INICIO=2

[CPU, 2] [2, 3] [CPU, 1]

Job	Inst. Llegada	CPU	E/S (Rec, Inst, dur)
1	0	5	(R1, 3, 3)
2	1	4	(R1, 1, 2)
3	2	3	(R2, 2, 3)



Algoritmos de planificación - Colas Multinivel

- ❑ Scheduler actuales □ Combinación de algoritmos vistos
- ❑ La Ready Queue es dividida en varias colas (Similar a prioridades).
- ❑ Cada cola definida posee su propio algoritmo de scheduling.
- ❑ Los procesos se colocan en las colas según una clasificación que realice el SO
- ❑ A su vez se existe un algoritmo que planifica las colas
- ❑ Realimentación □ Un proceso puede cambiar de una cola a la otra



Algoritmos de planificación - Colas Multinivel -

Ejemplo

- El sistema dispone de 5 colas (1 por cada prioridad), a menor número mayor prioridad
 - El SO selecciona los procesos analizando las colas comenzando desde la de mayor prioridad
 - Las colas 1 a 4 son administradas con RR, $Q=4$. La cola 5 se administra con FIFO
 - Si un proceso en ejecución abandona la CPU antes de completar su quantum por E/S, es movido a la siguiente cola de mayor prioridad
 - Si un proceso en ejecución completa su quantum es movido a la siguiente cola de menor prioridad
 - Los procesos, al ser creados, se les asigna una prioridad de 5
 - Cuando el proceso cambia su prioridad, la mantiene hasta finalizar su ejecución
- ✓ ¿A que procesos beneficia el algoritmo? □ **E/S Bound**
- ✓ ¿Puede ocurrir inanición? □ Si, con los procesos ligados a CPU si siempre llegan procesos ligados a E/S

