Sistemas Operativos

Curso 2014

Planificación

Agenda

- Introducción.
- Despachador.
- Clases de procesos.
- Esquemas de planificación.
- Criterios de planificación.
- Algoritmos de planificación.
  - FCFS.
  - SJF.
  - Prioridad.
  - Round-Robin.
  - Multilevel-Queue.
  - Multilevel-Feedback-Queue.
- Sistemas multiprocesadores.

Introducción

- La planificación (*scheduling*) es la base para lograr la multiprogramación.
- Un sistema multiprogramado tendrá varios procesos que requerirán el recurso procesador a la vez.
- Esto sucede cuando los procesos están en estado *ready* (pronto).
- Si existe un procesador disponible y existen procesos en estado *ready*, se debe elegir el que será asignado al recurso para ejecutar.
- El componente del sistema operativo que realiza la elección del proceso es llamada planificador (*scheduler*).

Introducción

- **Despachador**: módulo del SO que da el control de la CPU al proceso seleccionado por el planificador de corto plazo
- Esto implica
  - Cambio de contexto: Salvar registros del procesador en PCB del proceso saliente. Cargar los registros con los datos del PCB del proceso entrante.
  - Cambiar el bit de modo a usuario.
  - Saltar a la instrucción adecuada que había quedado el proceso que se asignó a la CPU (registro *program counter*).
- La latencia del despachador debe ser la menor posible
- El planificador es el responsable de seleccionar el próximo proceso a ejecutarse.

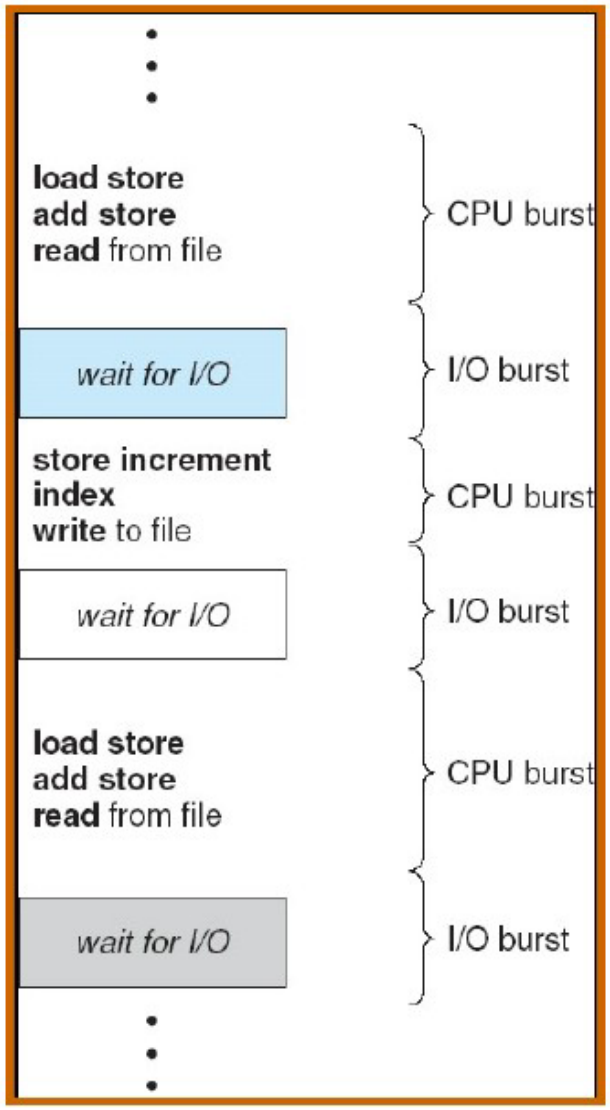
Tipos de planificador

- Largo plazo
  - Determina qué programas son admitidos al sistema para ejecución
  - Controla el grado de multiprogramación
  - Mientras más procesos son admitidos, cada uno tendrá un porcentaje menor de uso del procesador
- Mediano plazo
  - Determina si agregar más programas a los que ya están parcialmente o totalmente en memoria principal
- Corto plazo
  - Determina qué proceso es ejecutado en el procesador
  - Se ejecuta frecuentemente y debe ser eficiente
  - Evento típicos que desencadena al despachador: interrupciones del reloj (quantum), interrupciones de I/O, llamados al sistemas, señales, etc.

Clases de procesos

- Existen distintas políticas de planificación que serán exitosas según la clase de procesos que ejecuten.
- En general, los procesos tienden a ser o más intensivos en el uso de procesador, o más intensos en el uso de operaciones de E/S.
- Los procesos tienen ciclos de ráfagas de ejecución (*CPU-burst*) y ciclos de ráfagas de espera de operaciones de E/S (*I/O burst*)
  - **Procesos *CPU-bound***: Los procesos que contienen un alto uso de procesador son llamados *CPU-bound* o *compute-bound*.
  - **Procesos *I/O-bound***: Los procesos que realizan muchos accesos a operaciones de E/S son llamados *I/O-bound*.
- La prioridad que tenga un proceso frente a los demás para acceder al recurso será inversamente proporcional al uso que haga del recurso.

Clases de procesos



Esquemas de planificación

- Los momentos en que el planificador es invocado son:
  1. Cuando un proceso se bloquea: por ejemplo cuando inicia una operación de E/S o espera a que termine un hijo, etc.
  2. Cuando un proceso cambia del estado *ejecutando* al estado *pronto*. Por ejemplo al ocurrir una interrupción.
  3. Cuando ocurre una interrupción de E/S y un proceso pasa del estado *bloqueado* a *pronto*.
  4. Cuando se crea un proceso
  5. Cuando un proceso finaliza su ejecución.
- Cuando ocurre 1 o 5, el planificador es invocado debido a que el proceso en ejecución libera el procesador.
- Si el planificador es invocado cuando ocurre 2, 3 o 4, se dice que este es **expropiativo** (*preemptive*), ya que puede quitar el procesador al proceso que estaba en ejecución.

Esquemas de planificación

- Sistemas operativos con planificadores **no expropiativos** (*non-preemptive*) son los que asignan el recurso procesador a un proceso y hasta que este no lo libere, ya sea porque finaliza su ejecución o se bloquea, no se vuelve a ejecutar el planificador.
- Sistemas operativos con planificadores **expropiativos** (*preemptive*) son que los que pueden expropiar el recurso procesador a un proceso cuando otro proceso entra en estado pronto (ya sea porque es nuevo o porque se desbloqueó) o porque se le impone un límite de tiempo para ejecutar.

Esquemas de planificación

- Los esquemas de planificación son útiles según el ambiente donde sean aplicados:
  - **Sistemas por lotes**: Como no existe interacción con usuarios, los planificadores no expropiativos son ideales.
  - **Sistemas interactivos**: Debido a que existen procesos de usuarios ejecutando a la vez, los planificadores expropiativos son ideales para mantener un buen tiempo de respuesta para los usuarios.
  - **Sistemas de tiempo real**: No es necesario un planificador expropiativo ya que los procesos puede que no ejecuten por un buen tiempo, pero cuando lo hacen es por un período muy corto.

Criterios de planificación

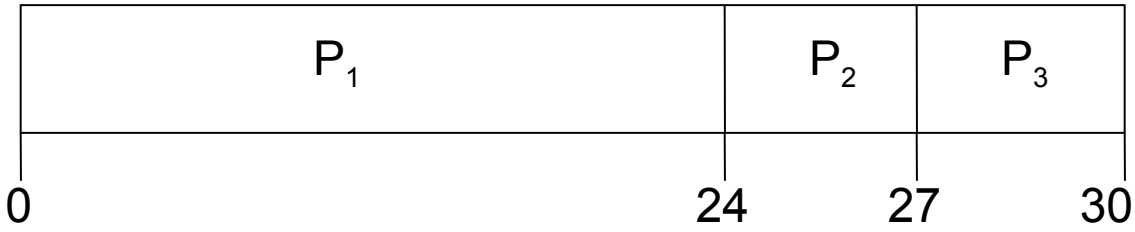
- Los algoritmos de planificación tendrán distintas propiedades y favorecerán cierta clase de procesos.
- Es necesario definir criterios para poder evaluar los algoritmos de planificación:
  - **Utilización de CPU** (*CPU utilization*): Es el porcentaje de uso (en cuanto a ejecución de tareas de usuario o del sistema que son consideradas útiles) que tiene un procesador.
  - **Rendimiento** (*Throughput*): Es el número de procesos que ejecutaron completamente por unidad de tiempo (una hora p.ej.).
  - **Tiempo de retorno** (*Turnaround time*): Es el intervalo de tiempo desde que un proceso es cargado hasta que este finaliza su ejecución.
  - **Tiempo de espera** (*Waiting time*): Es la suma de los intervalos de tiempo que un proceso estuvo en la cola de procesos listos (*ready queue*).
  - **Tiempo de respuesta** (*Response time*): Es el intervalo de tiempo desde que un proceso es cargado hasta que brinda su primer respuesta. Es útil en sistemas interactivos.

First Come First Served (FCFS)

- Los procesos son ejecutados en el orden que llegan a la cola de procesos listos.
- La implementación es fácil a través de una cola FIFO.
- Es adecuado para sistemas por lotes (*batch*).
- Es un algoritmo no expropiativo: una vez que el procesador le es asignado a un proceso este lo mantiene hasta que termina o se bloquea (por ejemplo al generar un pedido de E/S).
- El tiempo de espera promedio por lo general es alto.

First Come First Served

Proceso	Burst Time
P1	24
P2	3
P3	3



- Tiempo de espera: P1 = 0; P2 = 24; P3 = 27
- Tiempo de espera promedio: (0 + 24 + 27)/3 = 17

Shortest Job First (SJF)

- El algoritmo asocia a los procesos el largo de su próximo *CPU-burst*.
- Cuando el procesador queda disponible se le asigna al proceso que tenga el menor *CPU-burst*.
- Si dos procesos tiene el mismo *CPU-burst* se desempata de alguna forma.
- Su funcionamiento depende de conocer los tiempos de ejecución lo cual en la mayoría de los casos no sucede.
- Es adecuado para sistemas por lotes (*batch*).

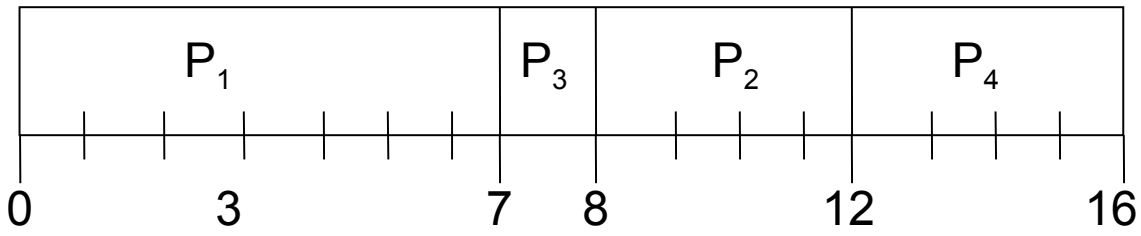
Shortest Job First (SJF)

- Dos esquemas:
  - **No expropiativo**: una vez que se le asigna el procesador a un proceso no se le podrá quitar.
  - **Expropiativo**: Si un nuevo proceso aparece en la lista de procesos listos con menor *CPU-burst*, se le quita la CPU para asignarla al nuevo proceso.
- Este algoritmo es óptimo para el tiempo de espera, pero requiere que todos los procesos participantes estén al comienzo (si no es expropiativo) y además hay que saber el tiempo del próximo *CPU-burst*.
- Es usado para planificación de largo plazo más que para planificación de corto plazo.

Shortest Job First (SJF) – No expropiativo

Proceso	Tiempo de arribo	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

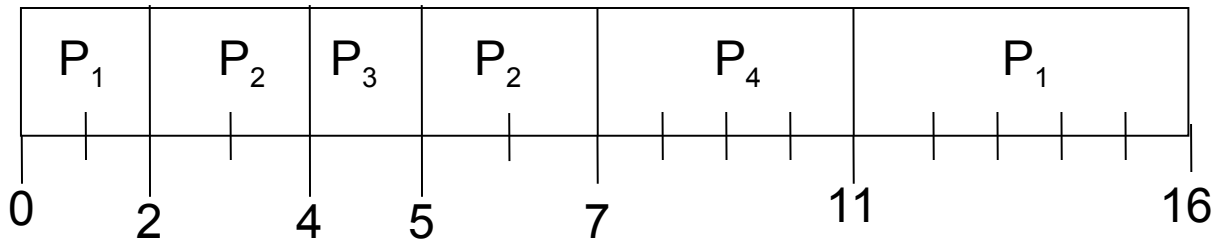
- Tiempo de espera promedio: (0 + 6 + 3 + 7)/4 = 4



Shortest Job First (SJF) – Expropiativo

Proceso	Tiempo de arribo	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

- Tiempo de espera promedio: (9 + 1 + 0 + 2)/4 = 3



Basados en Prioridad

- A cada proceso se le asigna un número entero que representa su prioridad.
- El planificador asigna el procesador al proceso con la más alta prioridad.
- Se utiliza en general un esquema expropiativo ya que si un proceso con mayor prioridad que el que esta ejecutando arriba a la lista de procesos listos (*ready queue*), será asignado al procesador.
- SJF se puede ver como un algoritmo de prioridad donde la prioridad esta dada por el próximo *CPU-burst*.
- Es adecuado para sistemas interactivos.

Basados en Prioridad

- Sufre de posposición indefinida ya que un proceso de baja prioridad quizás no pueda ejecutar nunca.
- La solución es utilizar prioridades dinámicas de envejecimiento: incrementar la prioridad según pasa el tiempo sin ejecutar.
- La prioridad de un proceso para el uso del recurso procesador deberá ser inversamente proporcional al uso que el proceso haga del mismo.
- Por lo tanto un proceso tipo *I/O-bound* deberá tener, en general, mayor prioridad que uno tipo *CPU-bound*.

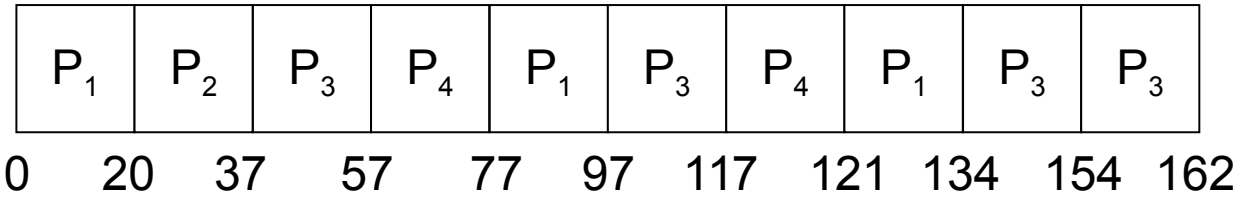
Round Robin (RR)

- A cada proceso se le brinda un intervalo de tiempo para el uso del procesador (*time quantum*).
- Al finalizar el tiempo, el procesador le es expropiado y vuelve al estado pronto (*ready*) al final de la cola.
- Es fácil de implementar ya que solamente es necesario una cola de procesos listos. Cuando un proceso consume su *quantum* es puesto al final de la cola.
- El *quantum* debe ser bastante mayor a lo que lleva realizar un cambio de contexto, sino se tendrá mucho *overhead*. A su vez, el tiempo de *quantum* incide en los tiempos de retorno.
- Es ideal para sistemas de tiempo compartido.
- No hay posposición indefinida

Round Robin (RR)

Proceso	Burst Time
P1	53
P2	17
P3	68
P4	24

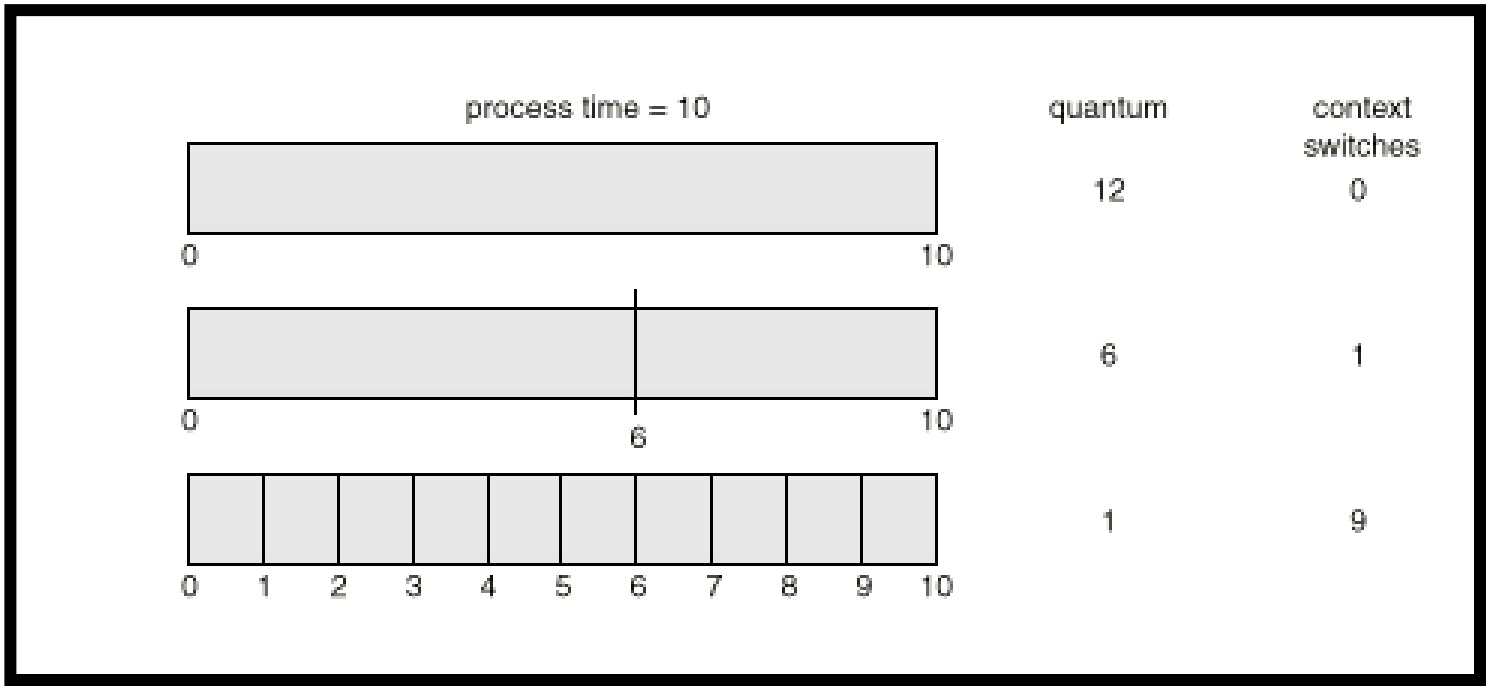
quantum = 20



- Por lo general, tiene un mayor tiempo de retorno que el *SJF*, pero mejora el tiempo de respuesta.

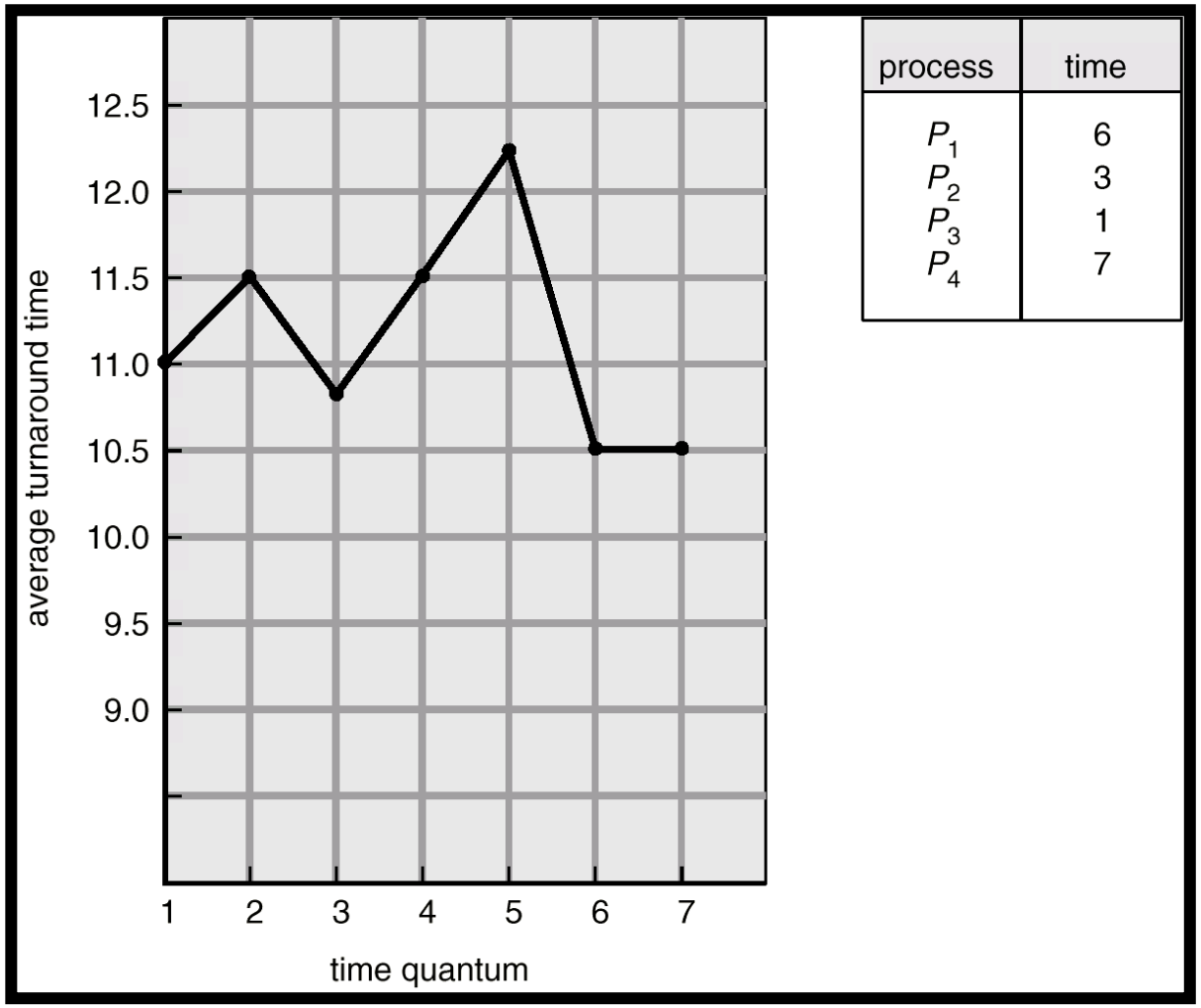
Round Robin (RR)

- Es necesario asignar un tiempo de *quantum* ajustado:
  - Si es muy chico generará muchos cambios de contexto.
  - Si es muy grande, el sistema tenderá a un FCFS.



Round Robin (RR)

- El promedio del tiempo de retorno medio varía según el *quantum*.



Round Robin (RR)

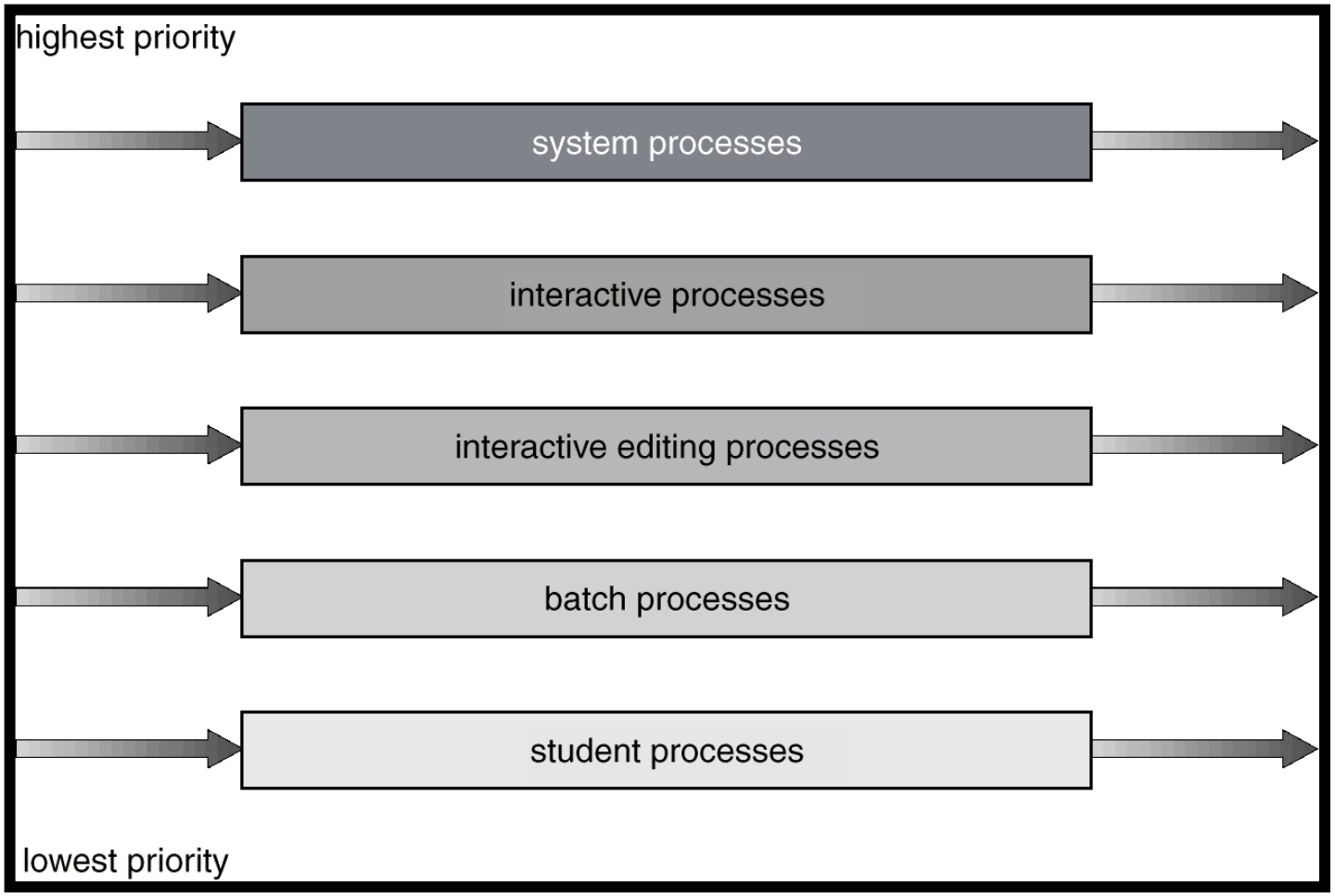
- Comparación con FCFS
  - 10 procesos que necesitan 100 unidades de tiempo son ejecutados a la vez
  - Se muestra el tiempo en que termina cada uno (despreciando el tiempo perdido en cambios de proceso)
  - El *quantum* es de 1 unidad de tiempo

Proceso	FCFS	RR
1	100	991
2	200	992
...	...	...
9	900	999
10	1000	1000

Multilevel Queue

- Si los procesos se pueden clasificar según sus cualidades, es posible dividir la lista de procesos listos (*ready queue*) en varias colas (una para cada clasificación).
- Los procesos son asignados permanentemente a una de las colas.
- Cada cola tendrá su propio algoritmo de planificación propio.
- Además, se debe tener una estrategia de planificación entre las diferentes colas. Por ejemplo, una cola tendrá prioridad sobre otra.

Multilevel Queue



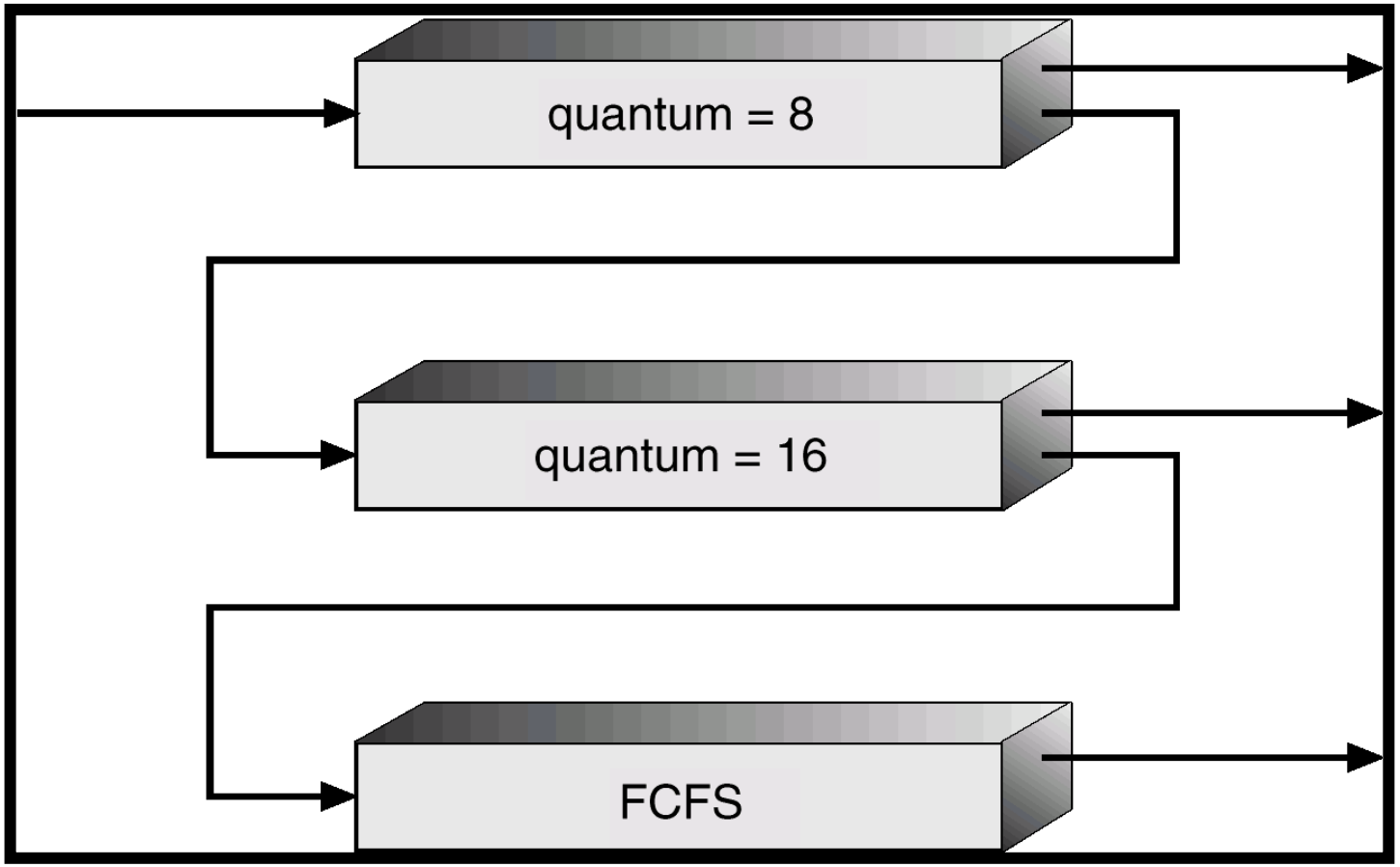
Multilevel Feedback Queue

- Se diferencia con el anterior en que procesos pueden cambiar de cola (nivel).
- Se basa en categorizar los procesos según el uso de CPU (*CPU-burst*) que tengan.
- La cola de mayor prioridad será la de los procesos *I/O-bound* y la de menor la de procesos con alto *CPU-bound*.
- De esta forma, se garantiza que los procesos con poco uso de procesador tengan mayor prioridad, y los que consumen mucho procesador tendrán baja prioridad.
- Los procesos, según el consumo de CPU que hagan, serán promovidos a una cola de mayor prioridad o rebajados a una de menor prioridad.

Multilevel Feedback Queue

- Un planificador *Multilevel Feedback Queue* es definido por:
  - El número de colas.
  - El algoritmo de planificación para cada cola.
  - El método utilizado para promover a un proceso a una cola de mayor prioridad.
  - El método utilizado para bajar a un proceso a una cola de menor prioridad.
  - El método utilizado para determinar a que cola será asignado un proceso cuando esté pronto.

Multilevel Feedback Queue



Sistemas multiprocesadores

- En un sistema simétrico cualquier procesador podrá ejecutar procesos de usuario.
- Una posibilidad es asignar una cola de procesos listos para cada procesador y de esa forma mantener los procesos asignados a un procesador (**afinidad de procesador**).
- Esto es conveniente para aprovechar los datos que están frescos en la memoria cache del procesador, ya que al ejecutar un proceso en un procesador se nutre su cache con datos del proceso.
- De esta forma, se logra mantener un mayor índice de *cache hit* y, por lo tanto, un mayor rendimiento en el sistema.
- Un problema que puede surgir es un desbalance en la cantidad de trabajo por procesador. En estos casos se migrarán procesos de cola para lograr balancear nuevamente la carga.