Algoritmos de Planificación del Procesador

Victor Tortolero, 24.569.609

Universidad de Carabobo
Facultad de Ciencia y Tecnología
Sistemas Operativos

25 de marzo de 2016

Algoritmo: First Come, First Serve (FCFS)

Se atiende a los procesos por orden de llegada, usando una cola. Es no apropiativo.

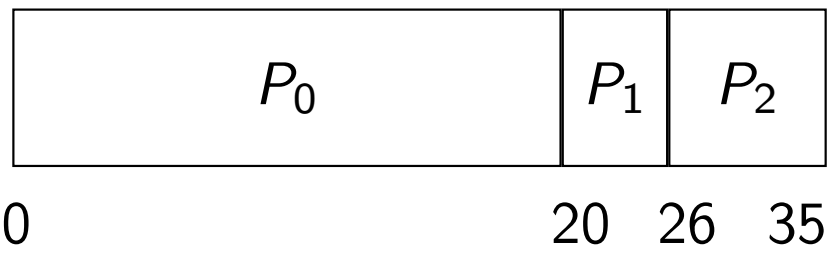
- **Ventajas**
  - Es fácil de entender e implementar.
- **Desventajas**
  - Aunque normalmente es justo en como dedica tiempo de CPU a los procesos, los procesos largos hacen esperar a los cortos.
  - El tiempo de espera es alto por lo que carece de rendimiento.

Algoritmo: First Come, First Serve (FCFS)

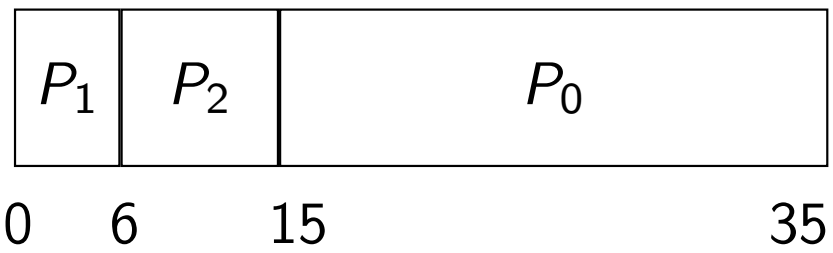
Ejemplo

Proceso	Tiempo de Ráfaga
P_0	20
P_1	6
P_2	9

$P_0 \rightarrow P_1 \rightarrow P_2$
$T_p = \frac{0+20+26}{3} = 15{,}333$



$P_1 \rightarrow P_2 \rightarrow P_0$
$T_p = \frac{0+6+15}{3} = 8$



Algoritmo: Shortest Job First (SJF)

Se atiende al proceso que tenga la próxima ráfaga de CPU mas corta, es no apropiativo.

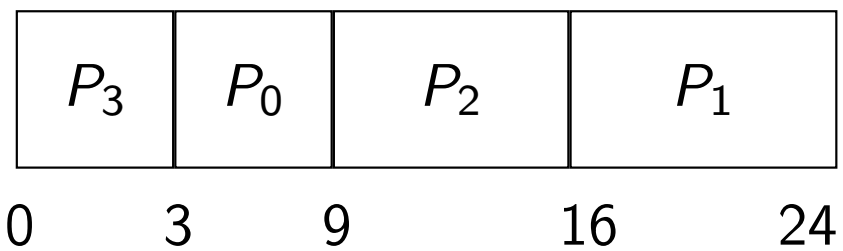
- **Ventajas**
  - Tiempo promedio de espera reducido.
- **Desventajas**
  - Difícil de implementar ya que es complicado conocer la duración de la siguiente ráfaga de CPU para un proceso.

Algoritmo: Shortest Job First (SJF)

Ejemplo

Proceso	Tiempo de Ráfaga(ms)	Tiempo de llegada(ms)
P_0	6	0
P_1	8	0
P_2	7	0
P_3	3	0

Tiempo de espera Promedio: $\frac{0+3+9+16}{4} = 7ms$



Algoritmo: Shortest Remaining Time First (SRTF)

Se atiende al proceso que tenga el menor tiempo de ráfaga total restante. Es apropiativo.

- **Ventajas**
  - Es eficiente.
  - Presenta un buen tiempo promedio de servicio.
  - Logra que la lista de procesos preparados sea lo más corta posible.

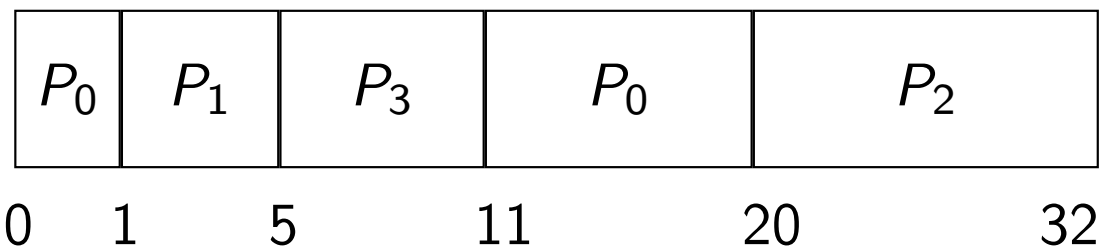
- **Desventajas**
  - Los procesos largos no se ejecutaran mientras existan procesos cortos en la cola.

Algoritmo: Shortest Remaining Time First (SRTF)

Ejemplo

Proceso	Tiempo de Ráfaga(ms)	Tiempo de llegada(ms)
P_0	10	0
P_1	4	1
P_2	12	2
P_3	6	3

Tiempo de espera Promedio: $\frac{0+1+2+8+15+35}{6} = 10{,}166ms$



Algoritmo: Round Robin (RR)

Se ejecuta cada proceso en la cola por un tiempo Q determinado.

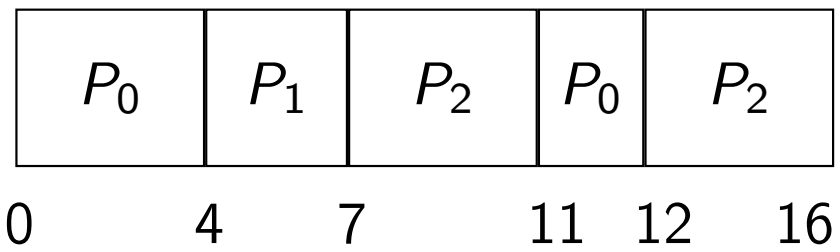
- **Ventajas**
  - Es justo con todos los procesos.
- **Desventajas**
  - Si el valor del quantum es mayor que el tiempo requerido por el proceso mas largo, se convierte en FCFS.
  - Si el valor del quantum es muy pequeño se producen muchos cambios de contexto lo que es ineficiente.

Algoritmo: Round Robin (RR)

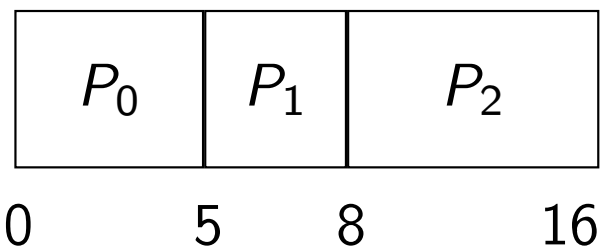
Ejemplo

Proceso	Tiempo de Ráfaga(ms)	Tiempo de llegada(ms)
P_0	5	0
P_1	3	1
P_2	8	2

Q = 4



Q = 8



Algoritmo: Prioridades

Se asocia a cada proceso un numero de prioridad (mientras menor sea el número, mas alta la prioridad), y se ejecuta el proceso con la prioridad mas alta. Si hay varios procesos con la misma prioridad, se resuelve con FCFS.

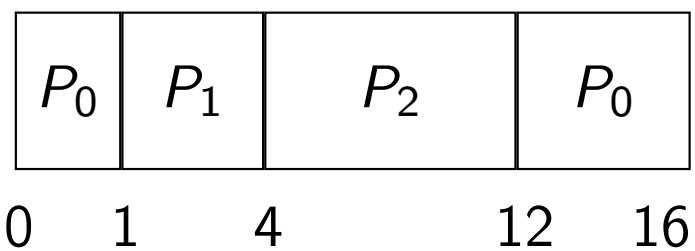
- **Ventajas**
  - Puede ser apropiativo o no apropiativo.
- **Desventajas**
  - Si no se usa envejecimiento, un proceso con muy baja prioridad puede llegar a no ejecutarse nunca.

Algoritmo: Prioridades

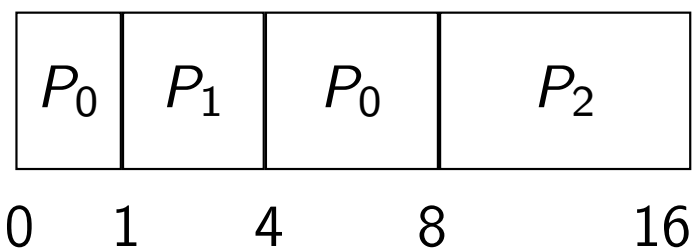
Ejemplo

Proceso	T. de Ráfaga(ms)	T. de llegada(ms)	Prioridad
P_0	5	0	3
P_1	3	1	1
P_2	8	2	2

Sin envejecimiento



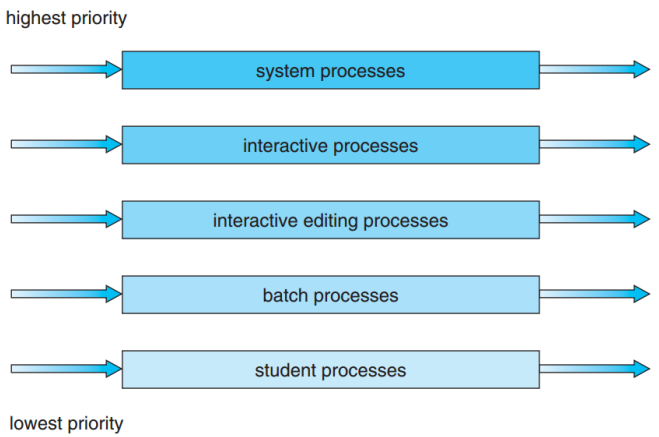
Envejecimiento T=2



Algoritmo: Colas Multinivel (MLQ)

Los procesos se agrupan por clasificación y se asignan a diferentes colas, cada cola puede tener su propio algoritmo de planificación. Se elige la cola a usar se usa un algoritmo de prioridades sin envejecimiento o se asigna un porcentaje de tiempo a cada cola.

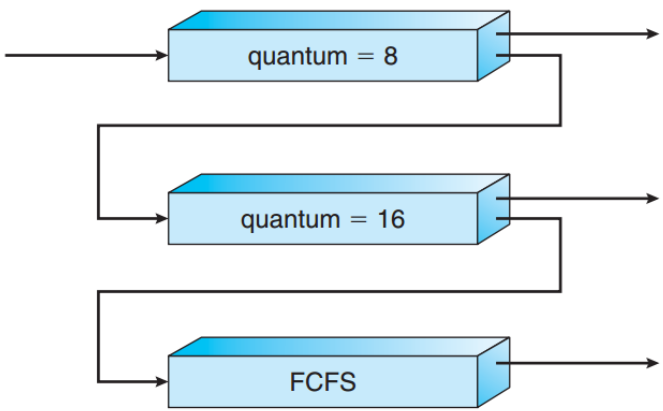
- **Ventajas**
  - Es muy adaptable a las necesidades del sistema, ya que cada cola puede ser gestionada de forma diferente.
- **Desventajas**
  - Es el algoritmo de planificacion mas complejo, requiere de algun mecanismo para definir el valor de todos los parametros.



Algoritmo: Colas Multinivel con Retroalimentacion (MLFQ)

Parecido al MLQ, pero aqui los procesos no estan restringidos a quedarse en la cola en la que entraron, pueden cambiar de una cola a otra. Agrupa los procesos según las características de sus ráfagas de CPU.

- **Ventajas**
  - Es general y adaptable a cualquier sistema.
- **Desventajas**
  - Es bastante complejo y costoso.



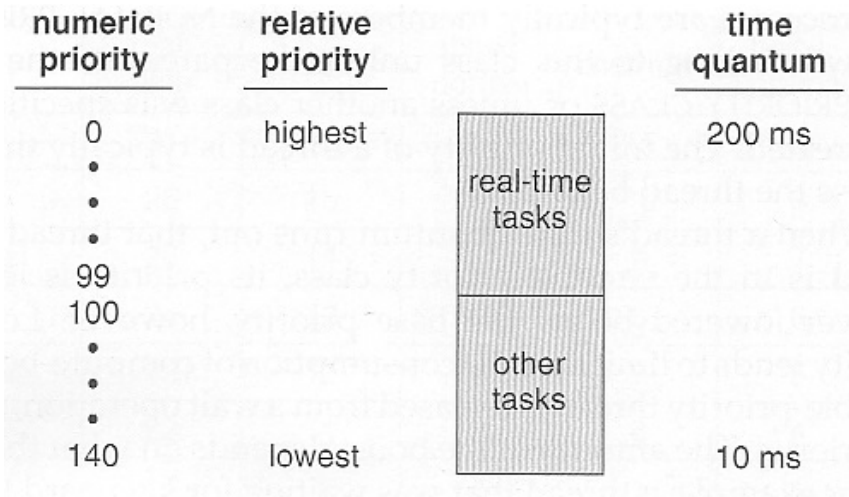
Sistemas de Tiempo Real

Los procesos deben terminar su ejecución en un periodo de tiempo
Estan los Soft Real-Time Systems y los Hard Real-Time Systems.

- **Planificación de tablas estáticas**: Se conocen los procesos críticos a priori.
- **Planificación apropiativa con prioridad estática**: Cada proceso tiene una prioridad segun el tiempo en el que deben ser completados.
- **Planificación dinámica**: Hace un plan de ejecución con la información de los procesos.
- **Dynamic best-effort**: El sistema hace lo que puede por cumplir los tiempos propuestos. Los procesos podrian ser abortados.

Planificación para Linux

- Usa un algoritmo basado en prioridades y apropiativo.
- Algoritmo de planificación que corre en tiempo $O(1)$ sin importar el numero de procesos.
- El kernel mantiene una lista de todas las tareas ejecutables en una estructura de datos denominada cola de ejecución, y cada cola tiene 2 matrices de prioridades (Activa y Caducada).



Planificación para Windows

- Usa un algoritmo apropiativo basado en prioridades.
- Existe un hilo especial "Idle" que se ejecuta cuando ningún otro hilo esta listo.
- Existe una relación entre las prioridades del kernel de Windows y el API de Windows.
- A cada proceso se le da una prioridad base dentro de su clase de prioridad.
- A los procesos en primer plano se les multiplica su quantum de tiempo planificado por 3, para que tengan mejor respuesta.

Planificación en sistemas Multiprocesadores

La planificación se complica porque ahora hay mas de un CPU al que se debe tener ocupado y en uso efectivo de manera constante.

- Compartir la carga ayuda ya que balancea el trabajo entre múltiples procesadores.
- Si los CPU son distintos cada uno tiene su propia cola y algoritmo de planificación.
- Si los CPU son iguales, uno toma el papel de master y corre todo el codigo del kernel y controla todas las actividades, mientras que los demas corren codigo de usuario.

Planificación en sistemas Multicore

- Mejoras en el consuma de energía.
- Varios procesadores en un solo chip físico.
- El sistema operativo siguen siendo multiples procesadores.

Planificación de hilos

- El sistema operativos planifica los hilos a nivel de kernel, los hilos a nivel de usuario son gestionados por una librería de hilos y el kernel no es consciente de ellos.
- Los hilos de usuarios deben ser asignados a un hilo de nivel de kernel asociado.
- La biblioteca de hilos planifica los hilos de usuario para que se ejecuten sobre proceso LWP disponible.
- El kernel usa el ámbito de contienda del sistema (SCS, system contention scope), para decidir que hilo del kernel planificar en un CPU.

Criterios para evaluar los algoritmos

- **Modelado determinista**:
  - Evaluación analítica.
  - Define el rendimiento de cada algoritmo para una carga de trabajo predeterminada.
  - Es simple y rápido.

- **Modelado de colas**:
  - Evaluación analítica.
  - Los resultados suelen ser cuestionables.
  - Los análisis pueden plantearse matemáticamente, pero no se apegan a la realidad.

Criterios para evaluar los algoritmos

- **Simulación**:
  - Gran precisión.
  - Requieren recopilar o generar información para ejecutar la simulación.
  - En la mayoría de los casos requieren mucho tiempo.
  - Mientras mas larga, mas precisa pero requiere mas tiempo de computo.

- **Implementación**:
  - Es el criterio con mayor precisión.
  - Alto coste para su implantación.