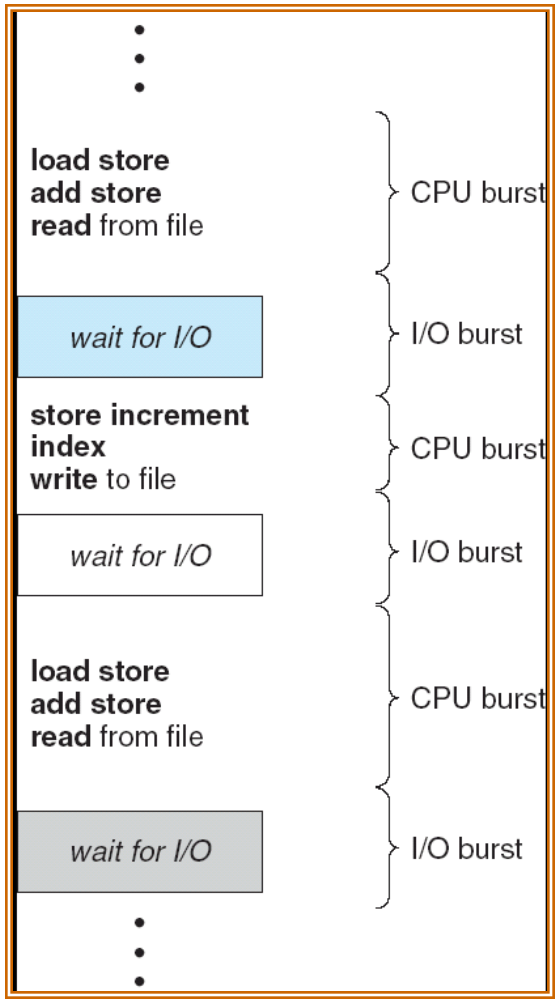
5: CPU Scheduling

Sistemas Operativos 1
Ing. Alejandro León Liu

- Calendarización de CPU
- Algoritmos de calendarización
- Criterios de selección de algoritmos de calendarización

- Sistema con un procesador
  - Un proceso corre a la vez
- Multiprogramación
  - I/O: Cambio de proceso
  - Tener un proceso ejecutándose todo el tiempo
- Tiempo compartido (Multitasking)
  - Cambio frecuente: Interacción

- Ejecución de procesos
  - Ciclo de
    - CPU burst
    - I/O burst
  - Ayuda en
    - Selección algoritmo calendarización
  - CPU scheduler
    - Calendarizador a corto plazo
    - Seleccionar siguiente proceso ready
    - No necesariamente FIFO



- CPU calendariza nuevo proceso
  - 1) Proceso pasa de running a waiting (I/O)
  - 2) Proceso pasa de running a ready (INT)
  - 3) Proceso pasa de waiting a ready (I/O completada)
  - 4) Proceso termina

- Calendarizador cooperativo (No preemptive): 1 y 4
- Calendarizador preemptive: 1, 2, 3, 4
  - Timer
  - Acceso compartido a datos
  - Secciones críticas en kernel
    - No permiten interrupciones

- Dispatcher
  - Da control al nuevo proceso elegido por el CPU scheduler
  - Encargado de:
    - Context switch
    - Cambiar a user mode
    - Saltar a ubicación de user program y reanudar ejecución
  - Dispatch latency: Tiempo en parar un proceso y reanudar otro.

CRITERIOS DE SELECCIÓN DE ALGORITMO DE CALENDARIZACIÓN DE CPU

- Utilización de CPU
  - < 40%: sistema poco cargado
  - > 90%: sistema cargado
- Throughput
  - Número de procesos completados / unidad de tiempo
- Turnaround time
  - Tiempo total de ejecución de proceso
- Waiting time
  - Tiempo total en estado waiting
- Response time
  - Tiempo de creación a tiempo de primera respuesta

ALGORITMOS DE CALENDARIZACIÓN

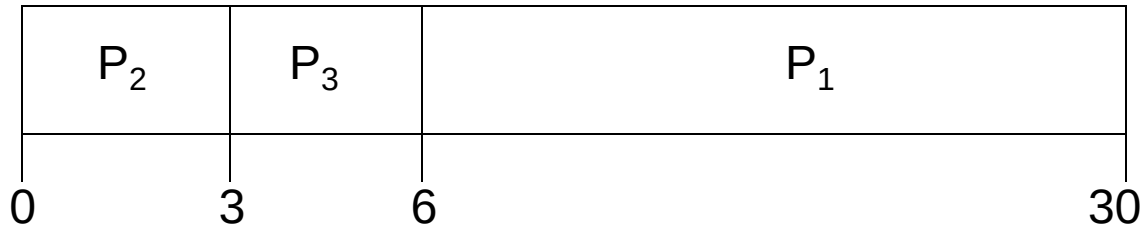
- First come, first serve: implementación simple

Process	Burst Time
P_1	24
P_2	3
P_3	3

  - Tiempo de llegada: P_1 , P_2 , P_3

  - Waiting time P_1 = 0; P_2 = 24; P_3 = 27
  - Average waiting time: (0 + 24 + 27)/3 = 17

- Tiempo de llegada P_2 , P_3 , P_1



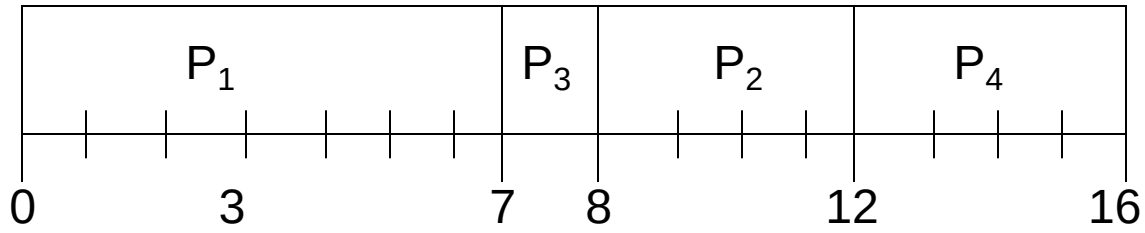
- Waiting time $P_1 = 6; P_2 = 0; P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- *Convoy effect:* procesos cortos esperan por procesos largos

- Shortest job, First scheduling (SJF)
  - Asignar CPU a proceso con menor CPU burst
  - Minimizar waiting time
  - Difícil saber CPU burst
    - Estimar
  - Non-preemptive: Al asignar CPU, no se puede asignar a otro proceso hasta que se termine el CPU burst
  - Preemptive: Shortest-Remaining-Time-First (SRTF)

- Non preemptive

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

  - SJF

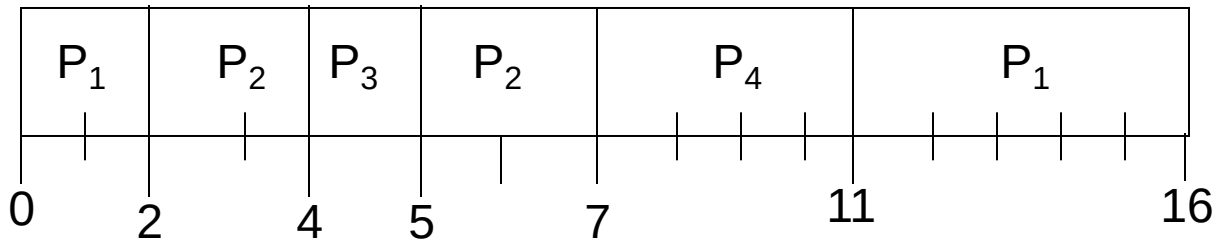


  - Average waiting time = (0 + 6 + 3 + 7)/4 = 4

- Preemptive

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

  - SJF



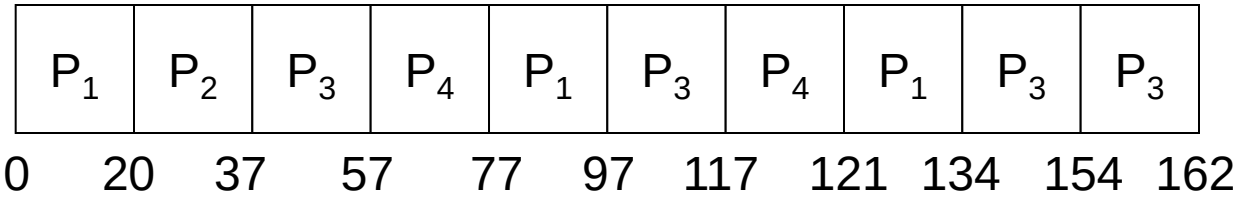
  - Average waiting time = (9 + 1 + 0 +2)/4 = 3

- Priority Scheduling
  - Asociar prioridad a cada proceso
    - Externa: importancia del proceso
    - Interna: memoria, archivos abiertos, burst time average
  - Starvation – Procesos con poca prioridad nunca se ejecutan
    - Al apagar la IBM 7094 en 1973, un proceso creado en 1967 todavía no había sido ejecutado
    - Aging: al pasar el tiempo, aumentar prioridad
  - Preemptive: Es posible asignar CPU a nuevo proceso de alta prioridad
  - Non preemptive: nuevo proceso de alta prioridad será el siguiente en ser ejecutado

- Round Robin
  - Multitasking (tiempo compartido)
  - Cola circular
  - Time quantum (10 – 100 ms): ningún proceso tendrá el CPU más que un time quantum. (Excepto si no hay otros procesos) q pequeño:
    - q grande: FIFO
    - q pequeño:
      - Parece que cada proceso tiene un CPU
      - Mayor a tiempo de context switch

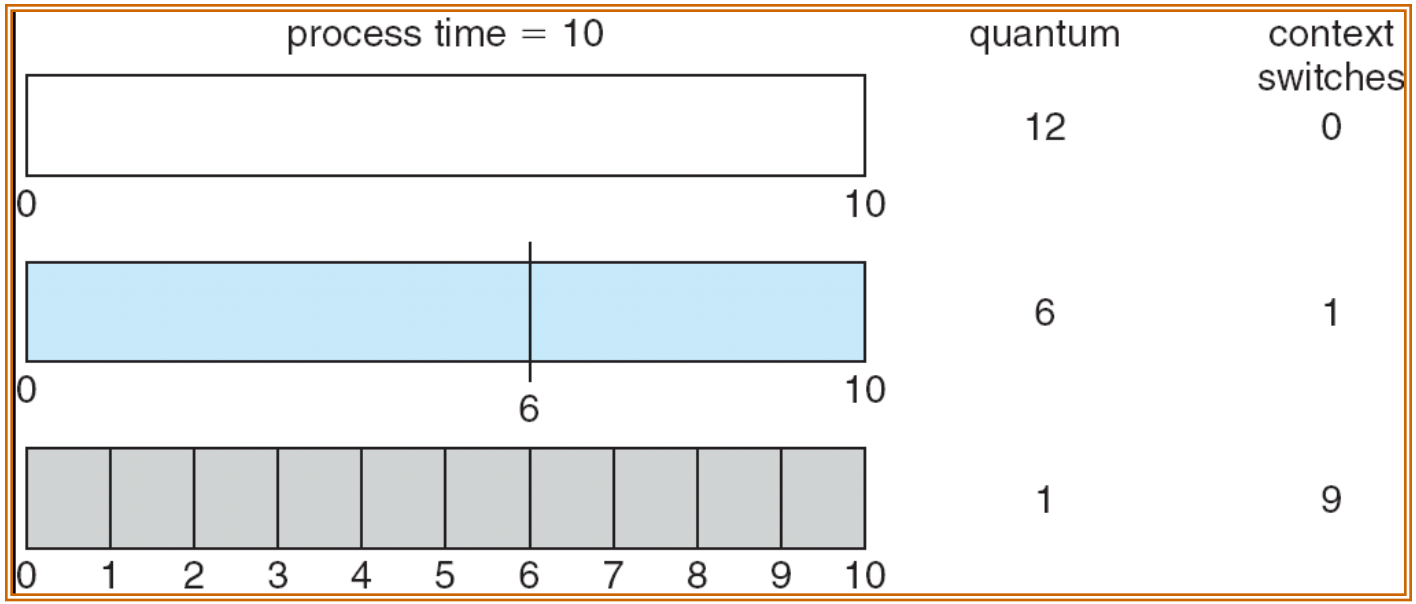
Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- q = 20

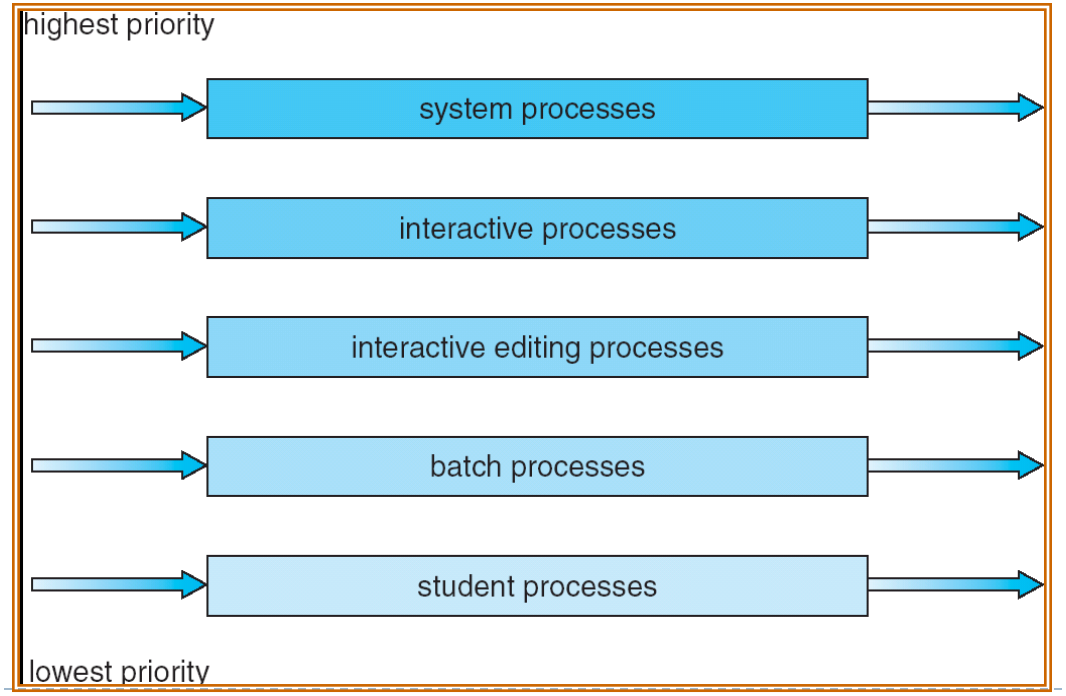


- Mayor turnaround time que SJF, pero mejor respuesta

- N procesos, q quantum
  - Cada proceso tendrá 1/n el CPU
  - En intervalos de tiempo no mayores a q
  - Cada proceso tendrá que esperar como máximo (n-1)*q



- Multilevel Queue Scheduling
  - Particionar cola en varias colas
    - Foreground: procesos interactivos
    - Background: procesos batch

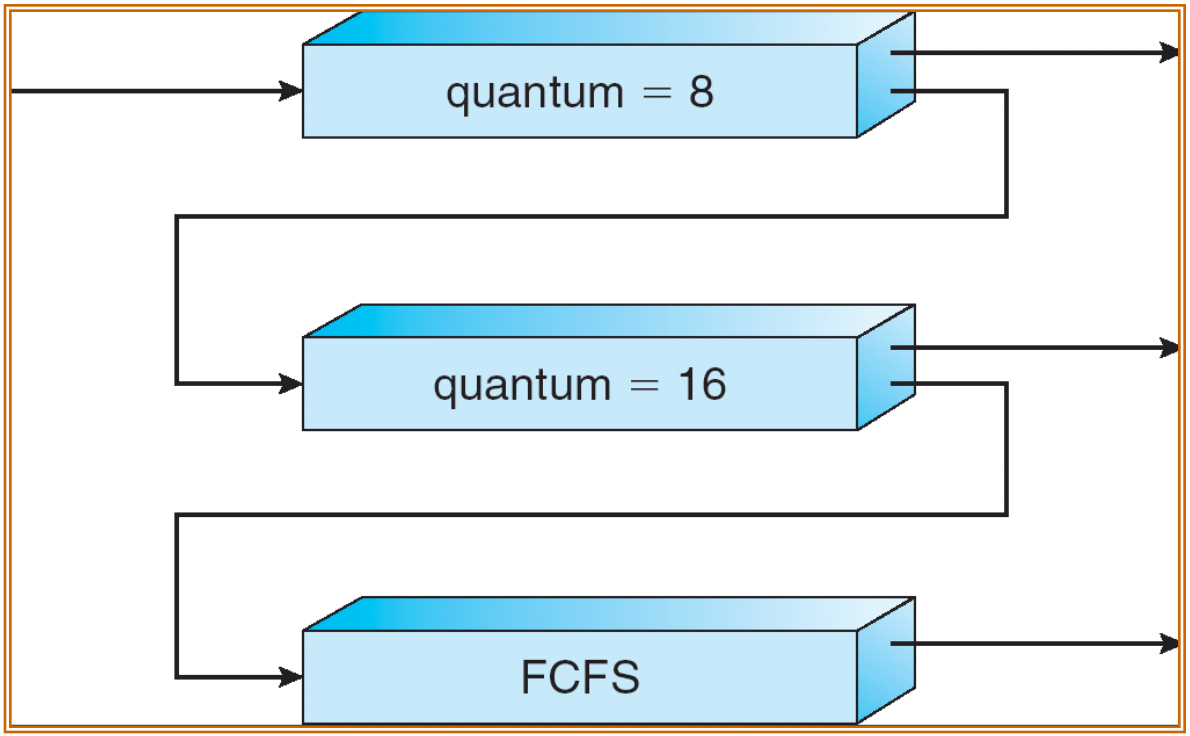


- Cada cola tiene su propio algoritmo de calendarización
- Fixed priority scheduling: Ejecutar todos los de foreground y luego background
  - starvation.
- Time slice – Repartir el tiempo de CPU en cada cola
  - 80%: foreground usando RR
  - 20%: background usando FCFS

- Multilevel Feedback Queue
  - Procesos se pueden mover dentro de las colas
  - Algoritmo más genérico, definido por:
    - Número de colas
    - Algoritmo de calendarización de cada cola
    - Método para determinar cómo procesos suben y bajan de una cola a otra
    - Método para determinar en qué cola ingresa nuevo proceso
  - Algoritmo más complejo

- Colas
  - Q_0 – RR, q = 8ms
  - Q_1 – RR, q = 16 ms
  - Q_2 – FCFS

- Calendarización
  - Proceso nuevo entra a Q_0
    - Recibe 8 ms
    - Si no termina, baja a Q_1
      - Recibe 16 ms
      - Si no termina, baja a Q_2
        - En Q_2 FCFS, si no hay procesos en Q_0 y Q_1
- Alta prioridad a procesos con CPU burst corto
- Procesos largos bajan a cola con menos prioridad


quantum = 8
quantum = 16
FCFS

Resumen de algoritmos de calendarización de CPU

Scheduling algorithm	CPU Utilization	Throughput	Turnaround time	Response time	Deadline handling	Starvation free
First In First Out	Low	Low	High	Low	No	Yes
Shortest remaining time	Medium	High	Medium	Medium	No	No
Fixed priority pre-emptive scheduling	Medium	Low	High	High	Yes	No
Round-robin scheduling	High	Medium	Medium	Low	No	Yes

CALENDARIZACIÓN CON MÚLTIPLES PROCESADORES

- Dos acercamientos
  - Multiprocesamiento asimétrico
    - Un procesador calendariza procesos y realiza I/O
    - Los demás, ejecutan user code
    - Más simple.
    - No es necesaria sincronización de estructuras del sistema
  - Multiprocesamiento simétrico
    - Cada procesador se calendariza
    - Cola común o cola por procesador
    - Programación cuidadosa
      - No queremos que dos procesadores ejecuten el mismo proceso
    - Windows, Solaris, Linux, Mac OS X

- Afinidad de procesador
  - Aprovechar datos de proceso en cache de procesador
  - Evitar migrar procesos de procesador
- Balanceo de carga
  - Sistemas donde se tenga cola por procesador
  - Push
    - Tarea revisa carga en procesadores y distribuye procesos
  - Pull
    - Cuando un procesador está libre jala un proceso de otra cola
  - Rompe afinidad de procesos a procesadores

- Symmetric Multithreading
  - Hyperthreading en procesadores Intel
  - Proveer múltiples procesadores lógicos
    - Contrario a múltiples procesadores físicos
  - Cada procesador lógico tiene
    - Set de registros
    - Manejo de interrupciones
  - Implementado en Hardware
  - Calendarización de procesador físico a cada procesador lógico.

CALENDARIZACIÓN DE THREADS

- Process contention scope (PCS)
  - Muchos a uno, muchos a mucho (kernel threads)
  - Competición de CPU entre user threads del mismo proceso que pelean por algunos kernel threads
- System contention scope (SCS)
  - Una user thread por una kernel thread
  - Windows, Solaris, Linux

- Pthread
  - PTHREAD_SCOPE_PROCESS
  - PTHREAD_SCOPE_SYSTEM

EJEMPLOS

- Solaris
  - Calendarización basada en prioridades.
  - Multilevel feedback queue. Cada cola tiene su algoritmo
    - Real time. (Prioridad: 100-159)
    - System (kernel) (Prioridad: 60-99)
    - Time sharing & Interactive (Prioridad: 0-59)
      - Minimizar response time a procesos interactivos
      - Throughput a CPU-bound processes

- Time sharing & interactive processes
  - Menor prioridad, mayor time quantum

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

EVALUACIÓN DE ALGORITMOS

- ¿Cómo seleccionar algoritmo de calendarización?
- Primer paso definir criterio de evaluación
  - Maximizar utilización de CPU. Tiempo de respuesta menor a 1 segundo.
  - Maximiar throughput de forma que turnaround time es proporcional a execution time

- Método de evaluación
  - Modelo determinístico
    - Define el performance de cada algoritmo con la carga actual.
    - Simple y rápido
    - Aplica únicamente a la carga actual.
    - Requiere el input exacto
    - Útil si se corre la misma carga una y otra vez.
  - Análisis de Colas
    - Medir y aproximar distribución de CPU e I/O bursts.
    - Aproximar tiempos de creación de procesos
    - Para cada algoritmo
      - Es posible calcular throughput, utilización de CPU, waiting time

- Simulaciones
  - Input aleatorio
  - Consume tiempo
  - Necesario un desarrollo complejo

- Todos los modelos son poco precisos
- El ambiente de ejecución puede cambiar

- En algunos casos es posible monitorear desempeño
  - Realizar modificaciones