Contenido 3: Scheduling

Lesly Reyes
lareyes2@uc.cl

Álvaro Quintana
alvaroq@uc.cl

Departamento de Ciencias de la Computación
Pontificia Universidad Católica Chile

August 19, 2024

Contenido

Scheduling
- Planificación de CPU
- Distintos niveles
- Modelo de ejecución de un proceso
- Tipos de scheduling

Algoritmos de Scheduling (BATCH)
- First-come, First-served (FCFS)
- Shortest Job First (SJF))

Algoritmos de Scheduling (Interactive)
- Round-Robin (RR)
- Priority Scheduling
- Multilevel Feedback Queue (MLFQ)

Algoritmos de Scheduling (Real Time)
- Procesos Real Time

Material Adicional

Scheduling

CPU Scheduling a.k.a. Planificación de CPU

Tenemos:

- Múltiples procesos en memoria (multiprogramación), ordenados en una tabla de PCBs.
- Algunos procesos en estado ready (listos para ejecutar)
- CPU que puede atender solo un proceso a la vez

Y queremos:

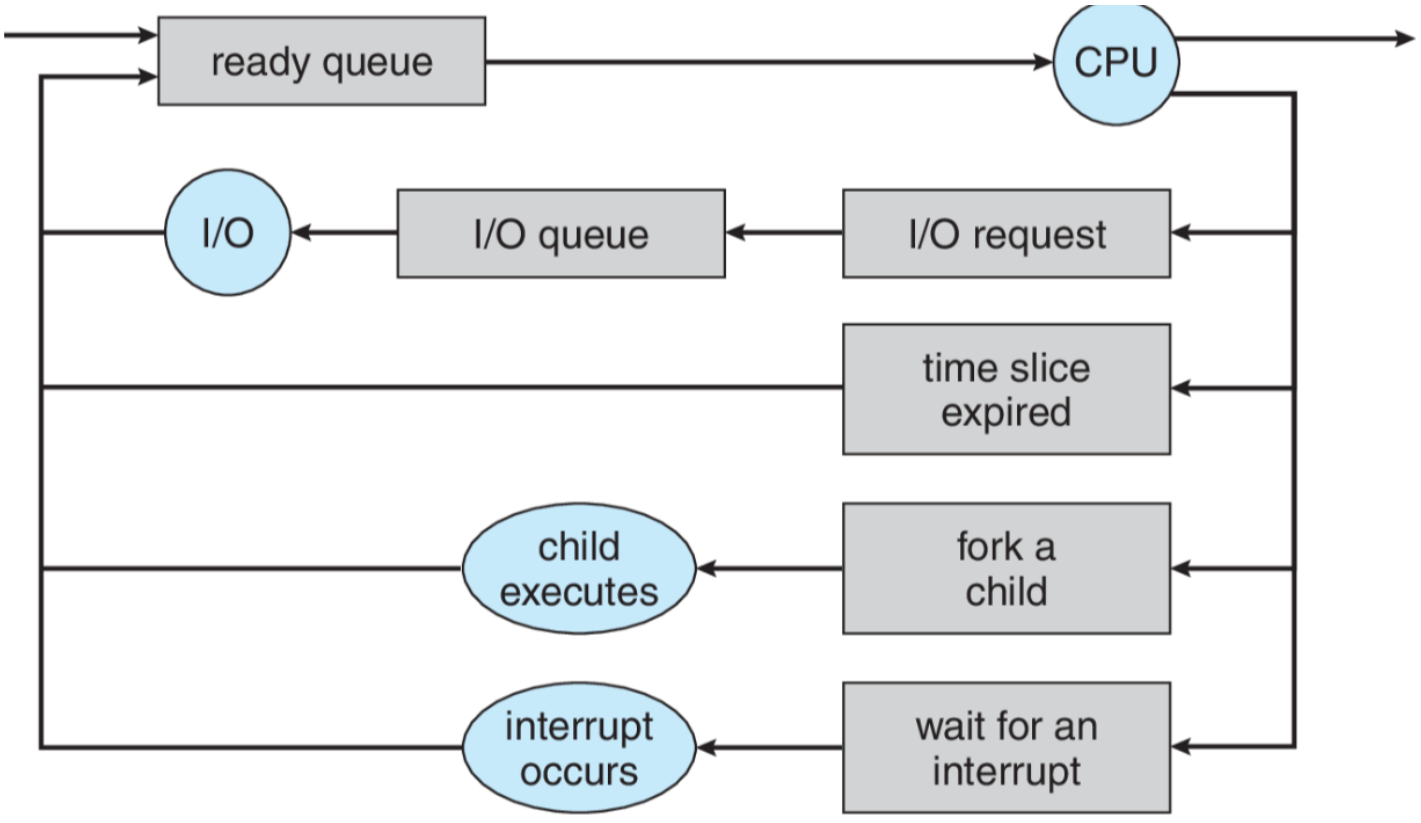
- **Multitasking:** asignar tiempo a múltiples procesos

Sistema Operativo debe elegir cuál de todos los procesos en estado ready debe ejecutar a continuación: scheduling

Scheduling

Responsable: scheduler, a.k.a. planificador

Scheduling puede ser visto como un sistema de manejo de colas



Scheduling

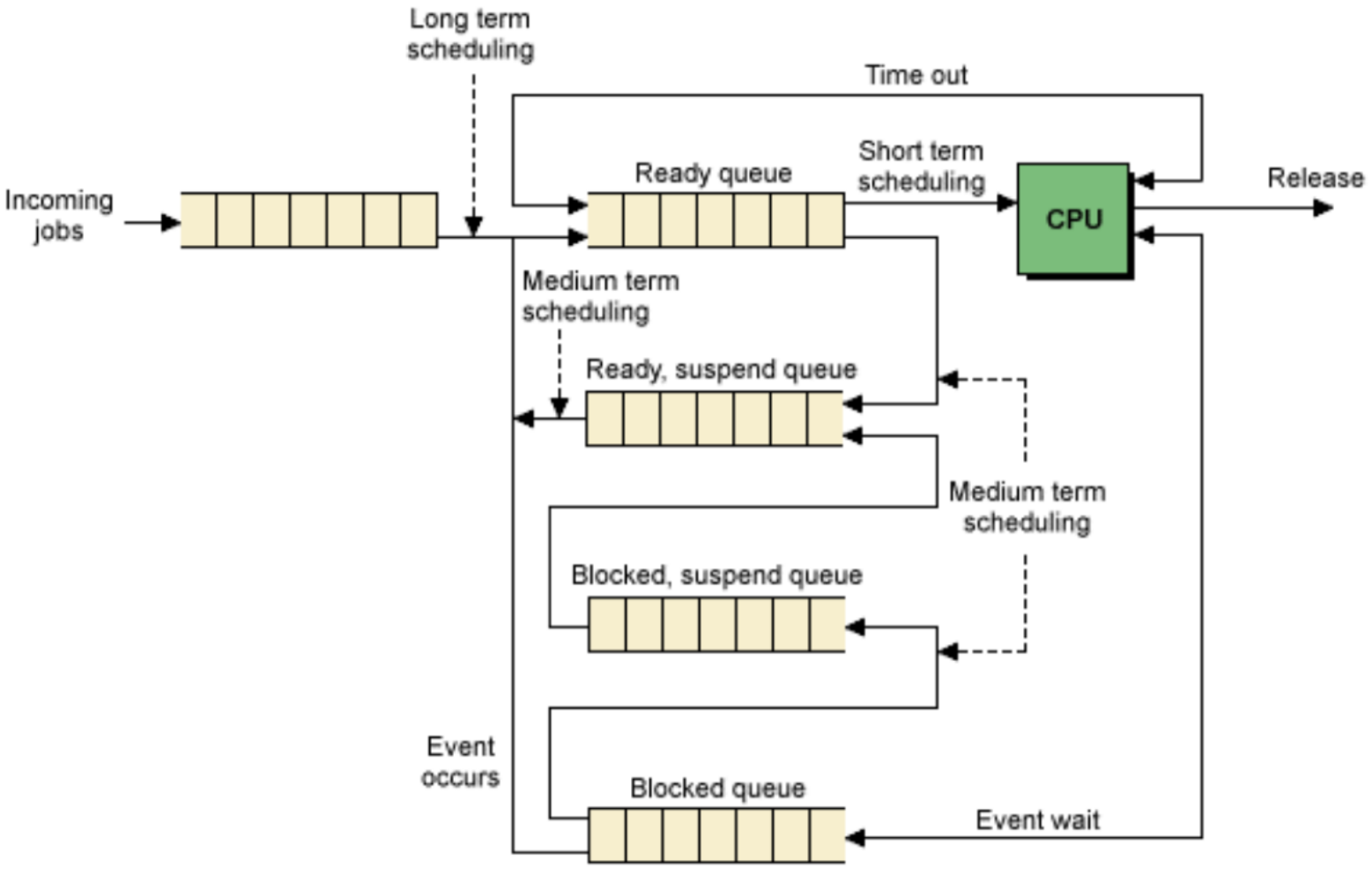
Distintos niveles: Long-term, medium-term y short-term

Distintos niveles de scheduling en el sistema operativo:

- **Long-term** Scheduler
  - Admite procesos en la cola ready.
  - Determina el grado de multiprogramación (cantidad de procesos en memoria).
- **Medium-term** Scheduler
  - Modifica temporalmente el grado de multiprogramación.
  - Ejecuta swapping copiando memoria RAM a disco, y de disco a RAM.
- **Short-term** Scheduler (a.k.a. dispatcher)
  - Selecciona un proceso de la cola ready para ejecutar.
  - Ejecuta el cambio de contexto

Scheduling

Distintos niveles: Long-term, medium-term y short-term



Scheduling

¿Y si estamos siempre haciendo Scheduling en vez de ejecutar programas?

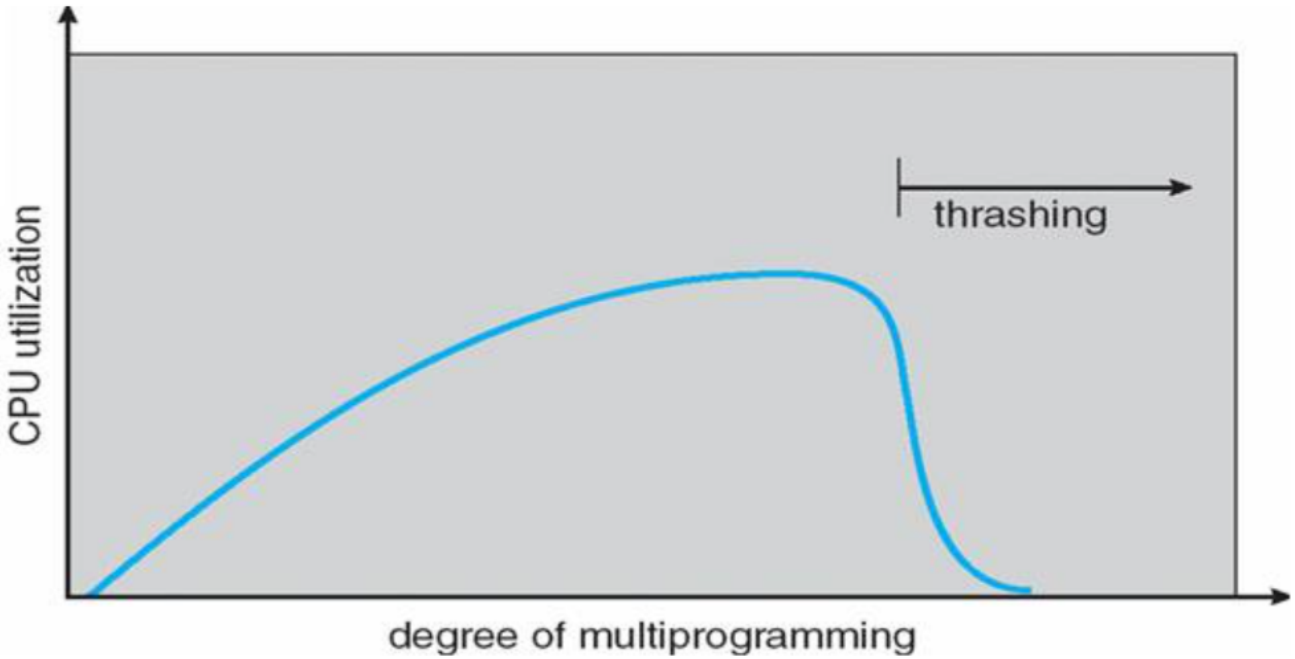
Scheduling es importante para **multitasking** . . . pero scheduling y context switch son sólo overhead

- ¿Qué pasa si el scheduler o el context switch toman más tiempo de lo que toma el proceso?
- ¿Qué pasa si se le asigna poco tiempo a cada proceso?
- ¿Qué pasa si hay muchos procesos ready? (long-term scheduler)

Contención de procesos se refleja en thrashing

Scheduling

¿Y si estamos siempre haciendo Scheduling en vez de ejecutar programas?



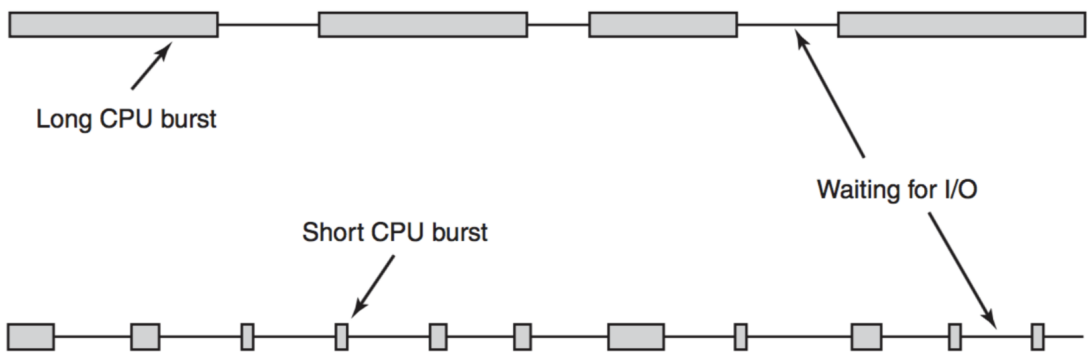
Scheduling

Modelo de ejecución de un proceso

No todos los procesos se comportan igual. Alternan entre dos fases:

- Uso de CPU (CPU-burst)
- Espera por I/O (I/O-burst)

Procesos suelen estar dominados por una u otra etapa

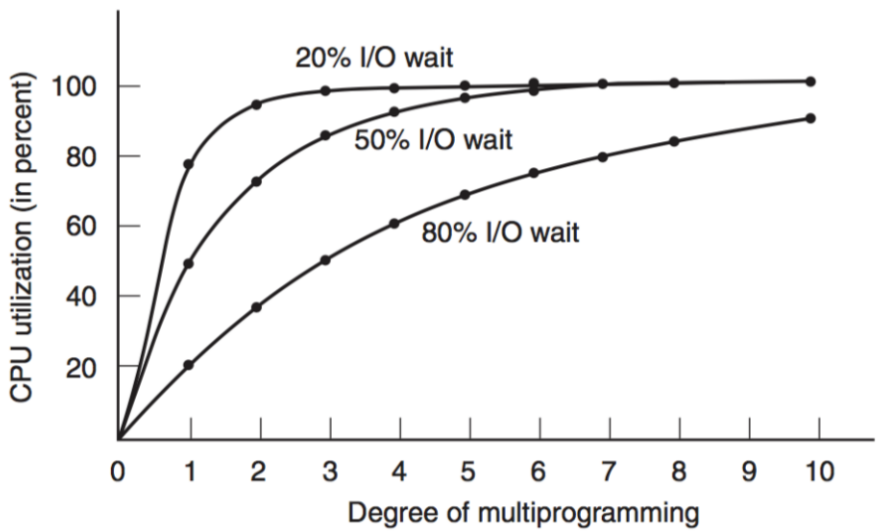


Scheduling

Modelo de ejecución de un proceso

El tiempo que gasta un proceso en espera influye en la utilización de CPU.
Una CPU bajo el 100% **está subutilizada.**

- p es el porcentaje de tiempo en espera por I/O.
- p^n probabilidad que n procesos estén esperando por I/O.
- Utilización de la CPU es $CPU_n = 1 - p^n$.



Scheduling

Tipos de scheduling: Por Interrupción

Por tipo de interrupción
Preemptive(expropiativo):

- Utiliza interrupciones (ej: de reloj (timer)) para decidir cuando sacar a un proceso de ejecución.

Non-Preemptive (colaborativo o no-expropiativo) : Permite que un proceso ejecute hasta que:

- El proceso deja voluntariamente la CPU, ó
- El proceso se bloquea en I/O, ó
- El proceso termina

Por Objetivo

Batch: Trabajo por lotes. Sin interacción.

- Mantener la CPU lo más ocupada posible
- Minimizar turnaround time: tiempo desde envío hasta término.
- Maximizar throughput: número de trabajos por hora

Interactive : Permite que un proceso ejecute hasta que:

- Minimizar tiempo de respuesta
- Satisfacer usuarios

Real Time:

- Tiempo de respuesta debe ser predecible
- Alcanzar deadlines

Scheduling

Conceptos importantes

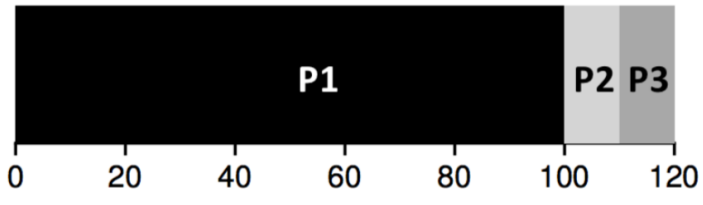
- **BATCH** = un proceso batch es sin ninguna interacción
- **Turnaround** = tiempo que el proceso permanece en el sistema
- **Throughput** = cantidad de procesos terminados por unidad de tiempo

Scheduling

First-come, First-served(FCFS)

Orden de llegada. Cola FIFO.

	Llegada	CPU-burst	Término	Turnaround
P1	0	100	100	100
P2	1	10	110	109
P3	2	10	120	118



Turnaround time promedio: **109** Si P2 hubiese llegado en t=0, y P1 hubiese llegado en t=1, entonces turnaround time promedio $\rightarrow$ 79

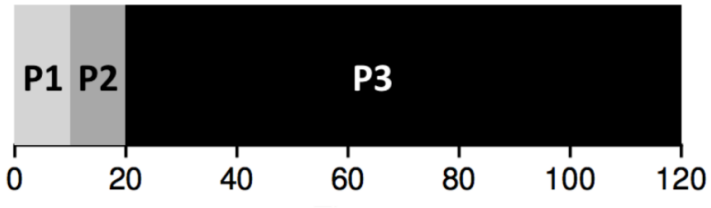
- Non-Preemptive
- Simple
- Poco predecible. Convoy effect

Scheduling

Shortest Job First (SJF)

El más corto primero

	Llegada	CPU-burst	Término	Turnaround
P1	0	10	10	10
P2	2	10	20	18
P3	1	100	120	119



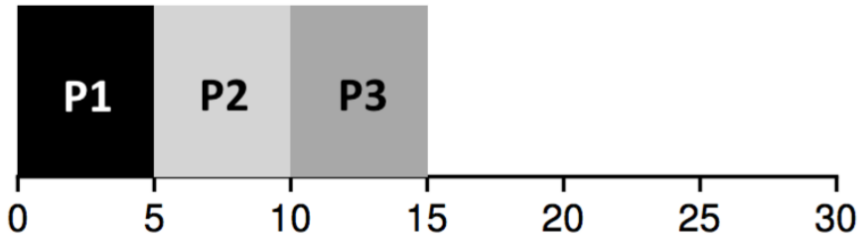
Turnaround time promedio: **49**

- Versión preemptive: Shortest Remaining Time Next. Elige al que le queda menos tiempo.
- No sabemos cuánto demora cada CPU-burst
- Óptimo
- Potencial inanición (starvation) de procesos largos.

Algoritmos de Scheduling (Interactive)

Métrica: tiempo de respuesta (response time).
Tiempo desde llegada (a la cola) hasta primera ejecución.

	Llegada	CPU-burst	Término	Turnaround	Inicio ejec.	T.Respuesta
P1	0	5	5	5	0	0
P2	1	5	10	9	5	4
P3	2	5	15	13	10	8



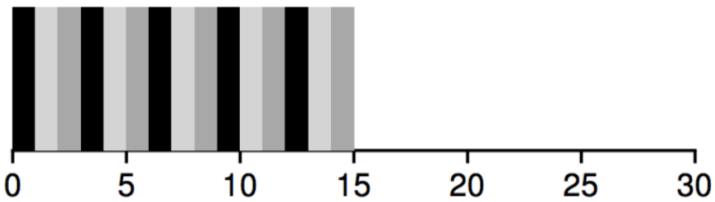
- Turnaround time promedio: 9
- Response time promedio: 4

Algoritmos de Scheduling (Interactive)

Round-Robin (RR)

Un turno (quantum, time slice) para cada uno. **Ejemplo con q=1**

	Llegada	CPU-burst	Término	Turnaround	Inicio ejec.	T.Respuesta
P1	0	5	13	13	0	0
P2	0.1	5	14	13.9	1	0.9
P3	0.2	5	15	14.8	2	1.8



- Turnaround time promedio: 13.9
- Response time promedio: 0.9

  - Con n procesos, cada uno recibe $1/n$ de CPU. Fair!

  - Ningún proceso espera más de $(n-1)$xq para ejecutar

  - Altamente dependiente de la elección de q(¿qué pasa si q es muy pequeño o muy grande?)

Algoritmos de Scheduling (Interactive)

Priority Scheduling

Cada proceso tiene asociada una **prioridad**

- Se atienden por orden de prioridad
- Prioridades iguales: FCFS ó RR
- SJF es un caso particular de este algoritmo (¿por qué?)
- Prioridades pueden ser estáticas o dinámicas
- **Starvation** de procesos con baja prioridad
- **Aging**: incrementar prioridad de procesos que llevan más tiempo

Permite reducir la prioridad de un proceso (incrementa su niceness)

Algoritmos de Scheduling (Interactive)

Multilevel Feedback Queue (MLFQ)

Scheduler que optimize turnaround time (como SJF), pero minimizando response time (como RR)

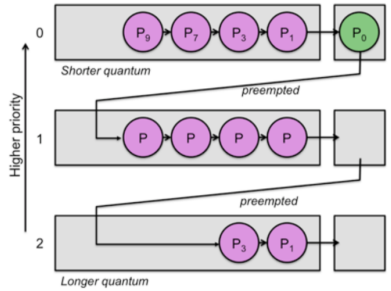
Usado en CTSS (1962) y Multics (1965)

Múltiples colas con distinta prioridad

- R1. Si priority(A) > priority(B)), ejecutar A
- R2. Si priority(A) = priority(B)), ejecutar A y B con RR
- R3. Procesos entran en la cola con **mayor** prioridad
- R4. Si un proceso usa su *q* (acumulado en todos sus turnos), su prioridad se reduce
- R5. Después de un tiempo *S*, todos los procesos se mueven a la cola con mayor prioridad

Algoritmos de Scheduling (Interactive)

Multilevel Feedback Queue (MLFQ)



Base para schedulers de Windows y MacOSX

- R1. Si priority(A) > priority(B)), ejecutar A
- R2. Si priority(A) = priority(B)), ejecutar A y B con RR
- R3. Procesos entran en la cola con **mayor** prioridad
- R4. Si un proceso usa su *q* (acumulado en todos sus turnos), su prioridad se reduce
- R5. Después de un tiempo *S*, todos los procesos se mueven a la cola con mayor prioridad

Los más cortos salen mas rápido y los más largos descienden a una cola de menor prioridad.

Algoritmos de Scheduling (Interactive)

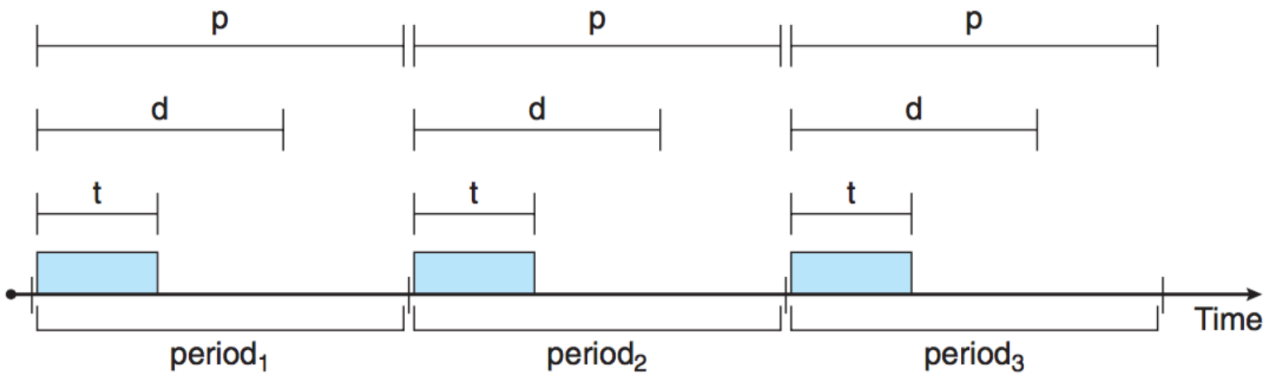
Multilevel Feedback Queue (MLFQ)

- Lottery (1994). Procesos reciben tickets
- Fair Share (1988). Tiempo compartido entre grupos de procesos (usuarios)
- O(1). Dos colas de procesos. Activos y expirados. Linux 2.6, < 2.6.23.
- Brain Fuck Scheduler (BFS) (2009).

Algoritmos de Scheduling (Real Time)

Procesos Real Time

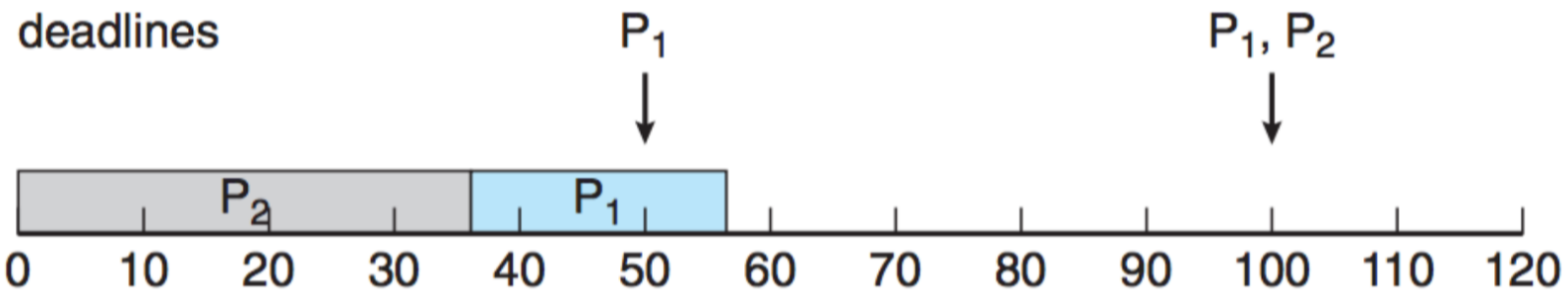
- Poseen deadlines y periodos de ejecución
- Sistema debe determinar si, dado deadline (d), periodo (p), y tiempo de ejecución (t),es capaz de incorporar el procesos a la ejecución.



Algoritmos de Scheduling (Real Time)

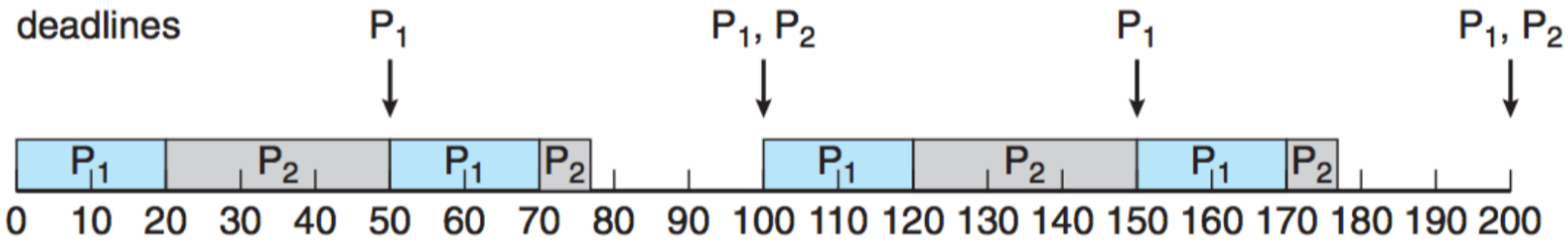
Procesos Real Time

RATE MONOTONIC SCHEDULING



Cada proceso recibe prioridad: $\frac{t_i}{p_i}$

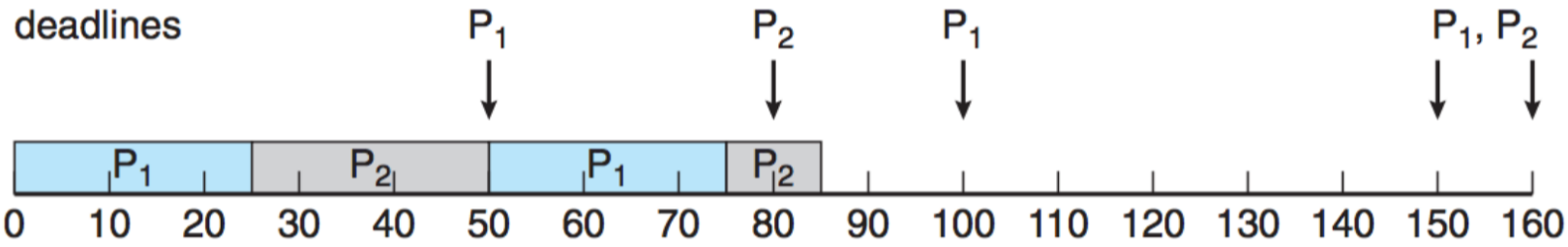
Con $P_1 : \{p_1 = 50, t_1 = 20\}$, y $P_2 : \{p_2 = 100, t_2 = 35\}$



EARLIEST DEADLINE FIRST SCHEDULING (EDF)

Rate Monotonic Scheduling es **estático**. Podría perder *deadlines*

Con $P_1 : \{p_1 = 50, t_1 = 25\}$, y $P_2 : \{p_2 = 80, t_2 = 35\}$



Earliest Deadline First Scheduling es dinámico. Elige siempre el que tiene deadline más cercano.

Material Adicional

- Video explicativo Tipos de schedulersLink Video
- Video explicativo Schedulers batch Link Video
- Video explicativo Schedulers Interactivos: Link Video
- Video explicativo Scheduling tiempo real Link

Próximo contenido a revisar serán Threads