04/09/2025 - Sistemas Operativos (Ude@)

0. Votación

1. ¿Ya formo grupo para el laboratorio? (Opción única)

10/10 (100%) han respondido

Si	(6/10) 60%
No	(4/10) 40%

Actualizar lista de parejas

2. ¿Ya tiene la cuenta en github? (Opción única)

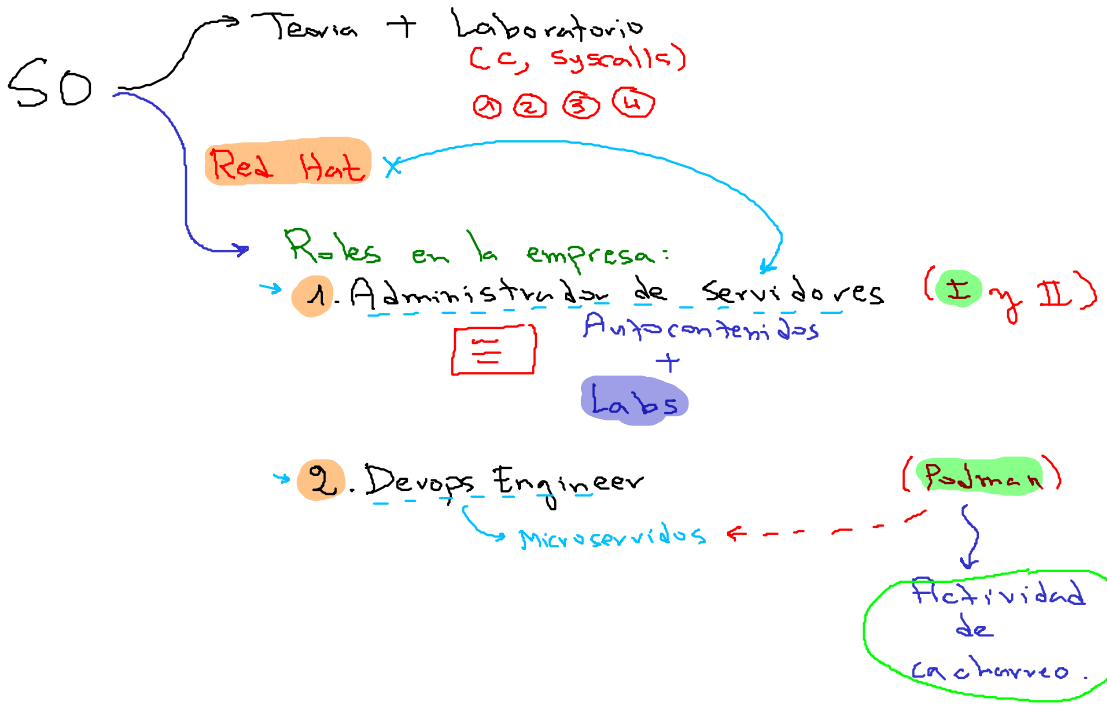
10/10 (100%) han respondido

Si	(10/10) 100%
No	(0/10) 0%

3. ¿Ya se registro en la plataforma de Red Hat? (Opción única)

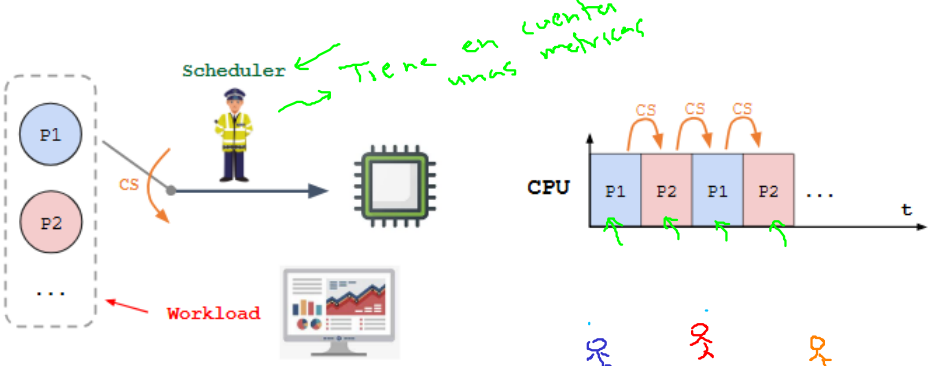
10/10 (100%) han respondido

Si	(8/10) 80%
No	(2/10) 20%



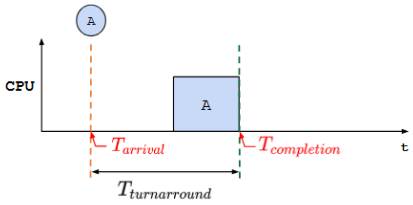
1. Repaso clase anterior

El Scheduler



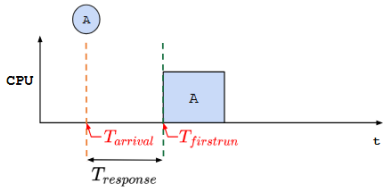
Metricas:
1. Turnaround time
2. Fairness (Equidad)
3. Response time

Métricas - Turnaround time



$$T_{turnarround} = T_{completion} - T_{arrival}$$

Métricas - Response Time



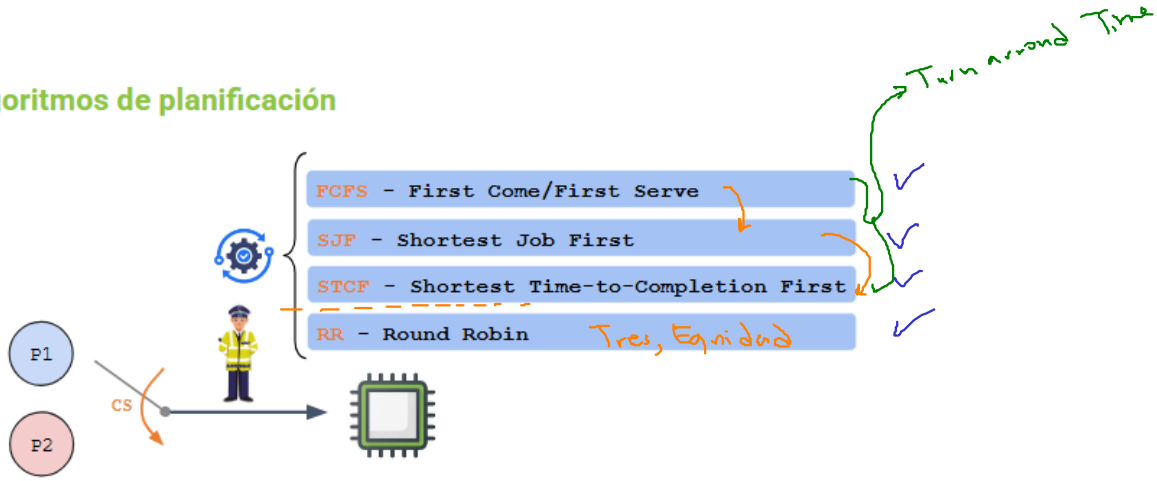
$$T_{response} = T_{firstrun} - T_{arrival}$$

Suposiciones ideales

1. Cada trabajo se ejecuta por la misma cantidad de tiempo
2. Todos los trabajos son iniciados al mismo tiempo (arrival time)
3. Una vez iniciado, cada trabajo se ejecuta hasta su finalización
4. Todos los trabajos solo usan la CPU (no I/O)
5. El tiempo de ejecución de cada trabajo es conocido (runtime).

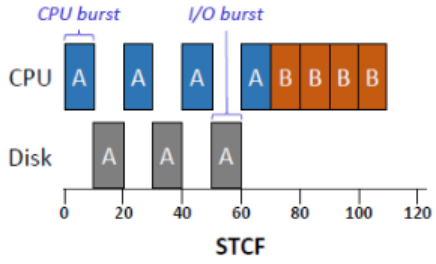
A medida que avanzamos, las suposiciones se iban relajando para acercarnos más a la realidad

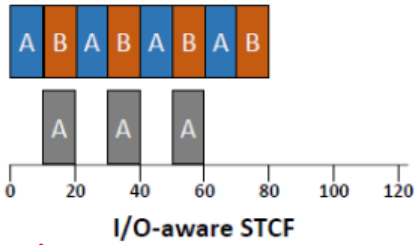
Algoritmos de planificación



Incorporando I/O

Not I/O aware	I/O aware (Overlap)
Pobre uso de los recursos.	Overlap: Permite un mejor uso de los recursos.





Tiempo de ejecución se conoce

2. MLFQ (Multi-level Feedback Queue)

i. Contextualización

Observación: Afirmar que el tiempo de ejecución de cada proceso es conocido **es difícil**.

Proceso	Arrival time	Run-time
A	0	100
B	20	10
C	20	10

Suposiciones ideales

~~1. Cada trabajo se ejecuta por la misma cantidad de tiempo~~
~~2. Todos los trabajos son iniciados al mismo tiempo (arrival time)~~
~~3. Una vez iniciado, cada trabajo se ejecuta hasta su finalización~~
~~4. Todos los trabajos solo usan la CPU (no I/O)~~
~~5. El tiempo de ejecución de cada trabajo es conocido (runtime).~~

Proceso	Características
A	• **CPU**: 40 ◦ **CPU burst**: 10 • **I/O**: 30 ◦ **I/O burst**: 10
B	• **CPU**: 40

Punto de partida

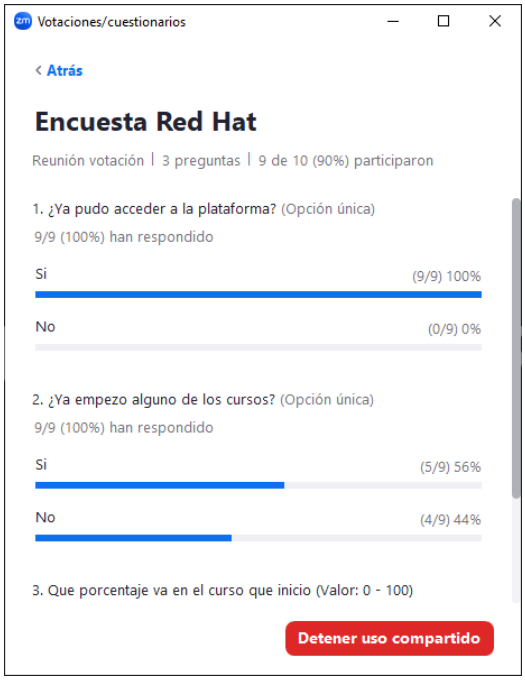
- Al relajar todas las suposiciones ideales llegamos a la conclusión de que el Scheduler no conoce nada para tomar la decisión sobre cuál es el próximo proceso que ejecutará la CPU.

- ¿Cómo llevar a cabo el proceso de planificación sin tener un conocimiento perfecto de la situación?
- ¿Cómo diseñar un **planificador de propósito general** que funcione bien para todo tipo de procesos (interactivos y batch)?

ii. Tipos de procesos (Trabajos)

09/09/2025 - Sistemas Operativos (UdeC)

1. Repaso clase anterior



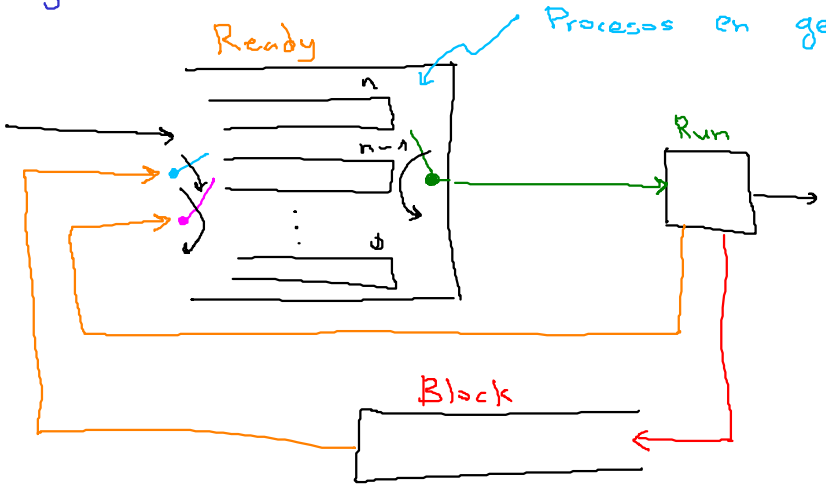
Planificador general → Tipos de Procesos

- Batch (CPU Bound) — CPU/IO
  - CPU
  - Tturnaround
- Interactivos (I/O Bound)
  - Tresponse
  - CPU / I/O

Políticas vistas:

1. FCFS
2. SJF
3. * STCF ✓
4. * RR ✓ (Fairness)

Mejora:



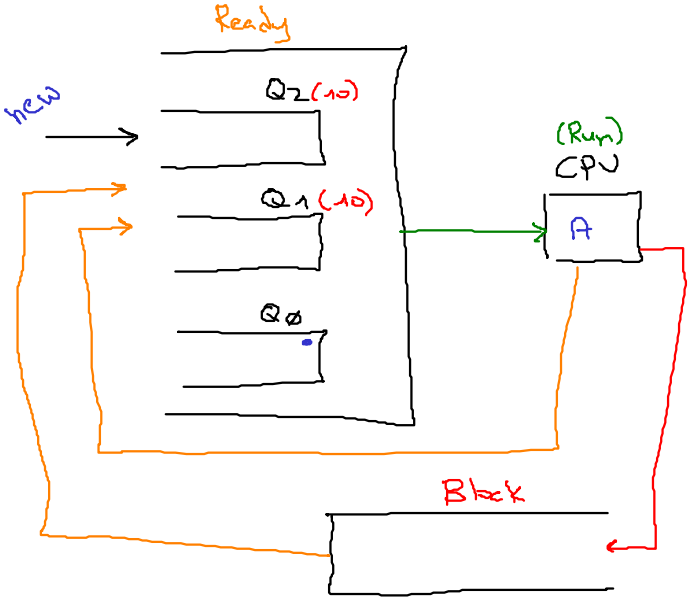
MLFQ: Reglas básicas

- Regla 1: Si **prioridad(A) > prioridad(B)**; **A** se ejecuta.
- Regla 2: Si **prioridad(A) = prioridad(B)**; **RR** para **A** y **B**.
- Regla 3: Cuando un trabajo llega al sistema es ubicado en la cola con la prioridad más alta.
- Regla 4a: Si el trabajo usa completamente el quantum de tiempo, se reduce su prioridad.
- Regla 4b: Si el trabajo entrega la CPU antes de finalizar su quantum de tiempo, mantiene el mismo nivel de prioridad.
- Regla 5: Después de un tiempo **S**, mueva todos los trabajos al mayor nivel de prioridad

A medida que avancemos en nuestro estudio iremos analizando la implicación de cada una de estas reglas.

Ejemplos

Caso 1.



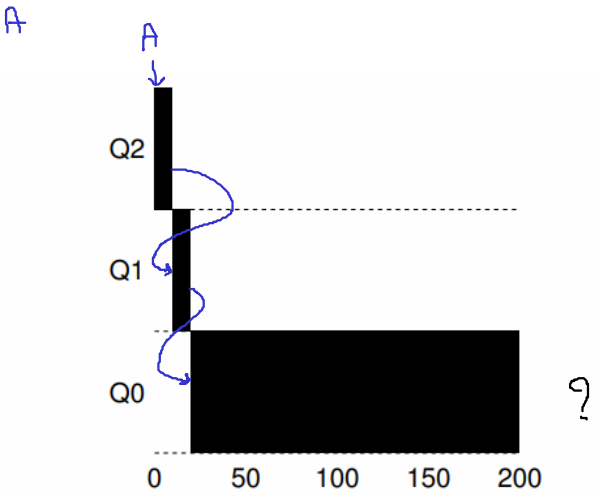
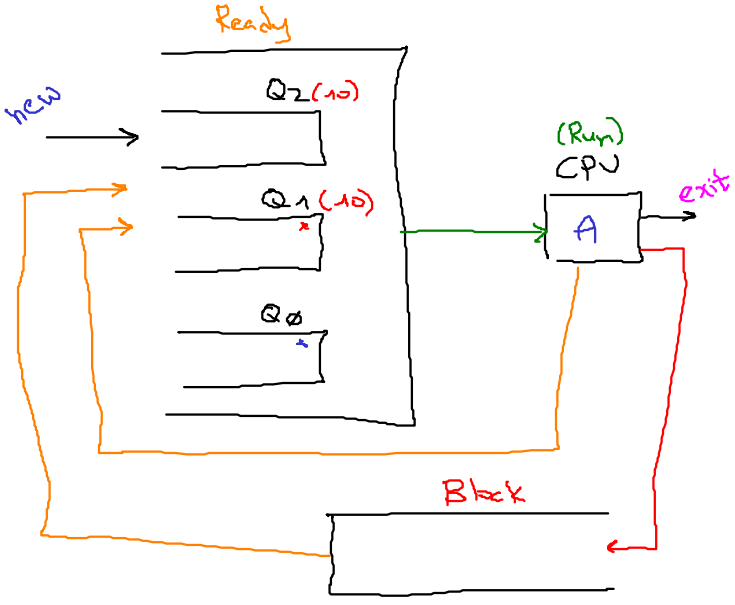
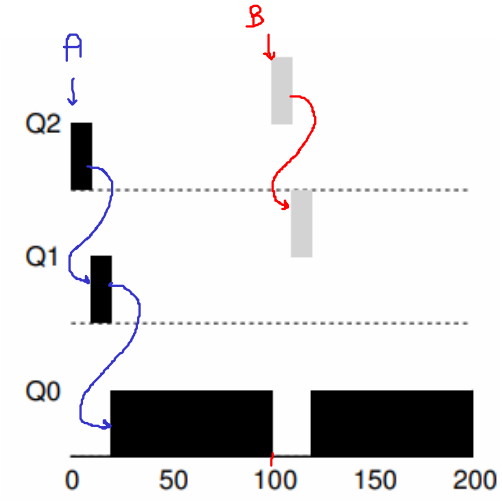


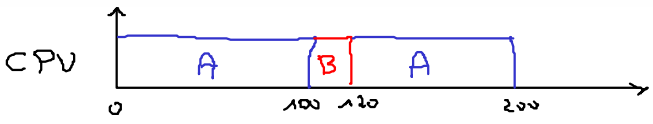
Figure 8.2: **Long-running Job Over Time**

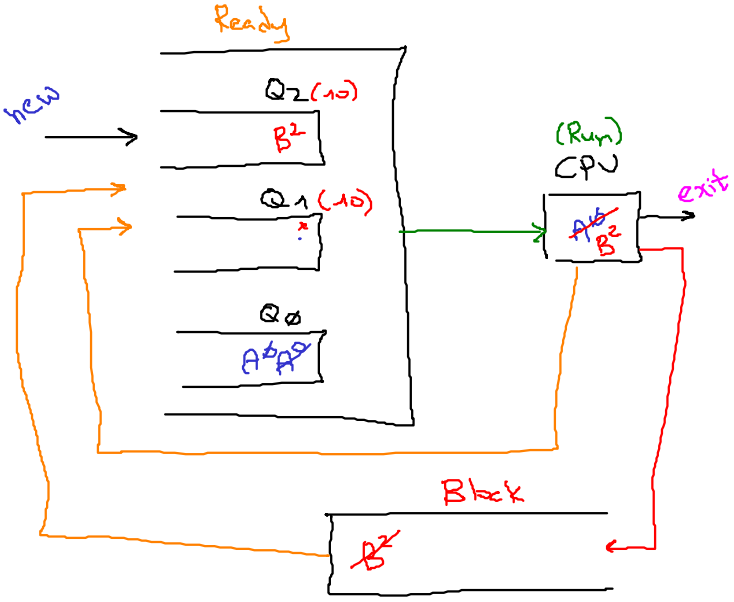
Caso 2

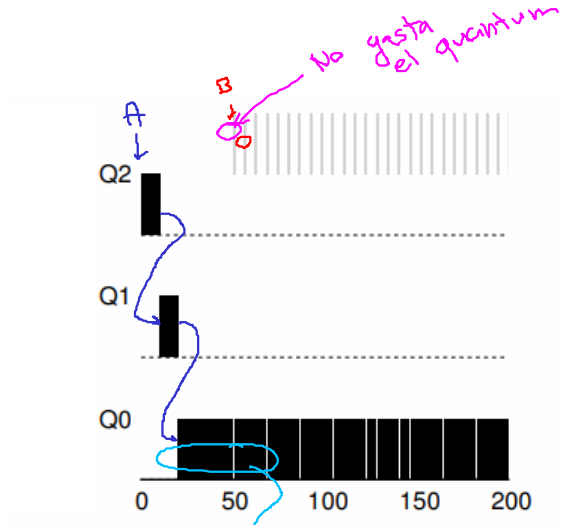




Along Came An Interactive Job: Two Examples







Along Came An Interactive Job: Two Examples