CS 1217

Lecture 10 – Scheduling Policies

Logistics

- Mid semester exam: week **after** mid semester break

Welcome to Meme Tuesday

Manu Awasthi @mnwsth · 5m

Students in the Operating Systems course, when their code compiles in the first attempt

Welcome to Meme Tuesday

Manu Awasthi @mnwsth · 4m

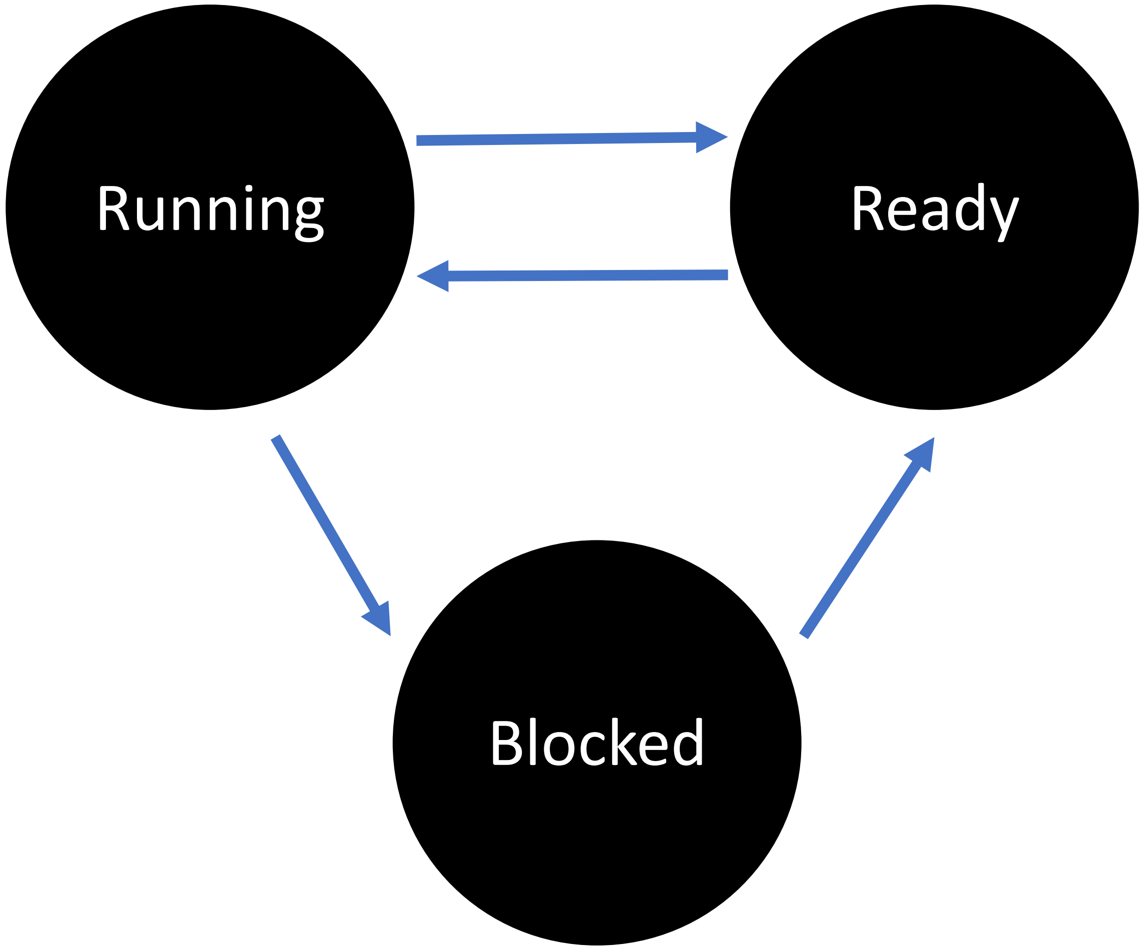
Someone: Why did you enrol in an Operating Systems course?

Student:

Welcome to Meme Tuesday

Why do CPU Scheduling?

Recap: Process States



When does scheduling happen?

- When a process **voluntarily** gives up the CPU by calling `yield()`
- When a process makes a blocking **system call** and must sleep until the call completes.
- When a thread **exits**
- When the **kernel decides** that a thread has run for long enough

- Which one is co-operative? Which one is pre-emptive?

The ”how” of Scheduling

- Mechanism vs Policy

- We need to answer two questions:
  - Which process do we run next?
  - How do we switch between processes?

- How do we **switch between processes**?
  - Perform a **context switch** and move threads between the **ready**, **running**, and **waiting** queues

Lets Play Policy vs. Mechanism

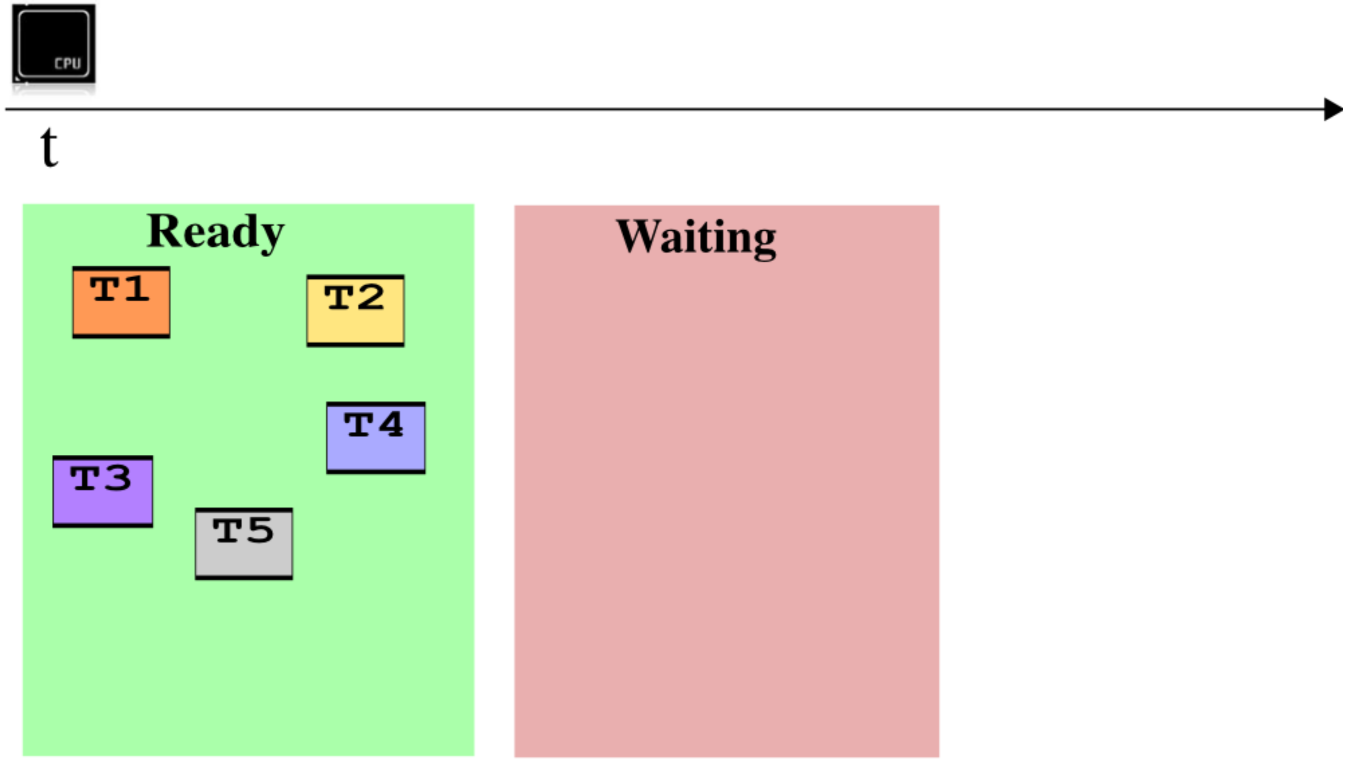
- deciding what process to run
  - Policy
- context switch
  - Mechanism
- maintaining the running, ready and waiting queues
  - Mechanism
- giving preference to interactive tasks
  - Policy
- using timer interrupts to stop running process
  - Mechanism
- choosing a process to run at random
  - Policy

Simplest Scheduling Algorithms

- FIFO or FCFS

- Random

Schedule



Random Scheduling

- Choose a **scheduling quantum**.

- Then:
  - Choose a process at random from the **_____**.
  - Run the process until it it blocks or the scheduling quantum expires.

- What happens when a process leaves the waiting state?

Scheduling Goals

- Need some mechanism to evaluate schedulers
- How well does it meet deadlines—unpredictable or predictable?
- How completely does it allocate system resources?
  - No point having idle CPU, memory, or disk bandwidth when something useful could be happening

- On human-facing systems, deadlines (or **interactivity**) usually wins. Why?

- **Scheduler Performance**: speed vs. schedule optimality

Information for Scheduling

- **What will happen next?** Oracular schedulers *cannot be implemented*.
  - How are they useful then?
- **What has happened?** Use *the past to predict the future*.
  - Where have we seen this before?
- **What does the user want?** How to incorporate user input for scheduling decisions
  - **nice** anyone?

Round Robin Scheduling

- Choose a **scheduling quantum**.
- Establish an ordered ready queue. For example, when a process is created add it to the tail of the ready queue.
- Then:
  - Choose the process at the head of the ready queue.
  - Run the process until it it blocks or the scheduling quantum expires.
  - If its scheduling quantum expires, place it at the tail of the ready queue.
- What happens when a process leaves the waiting state?
- Could put it at the head of the ready queue, or at the tail.

Discussion

- The **random** and **round robin** scheduling algorithms:
- require no information about a process' past, present, or future, and
- accept no user input.

- Both **penalize**—or at least do not reward—processes that give up the CPU before their quantums expire.
- As one exception, **round robin** scheduling is sometimes used once other scheduling decisions have been made and a set of processes are considered equivalent.
- As an example, you might rotate round-robin though a set of threads with the same priority.