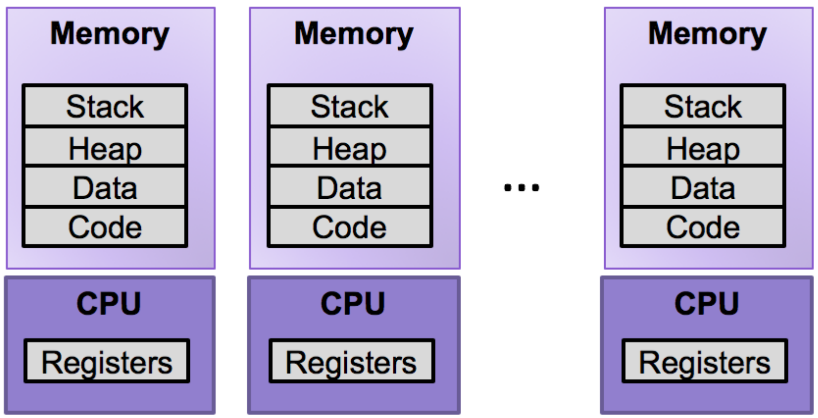
Lecture 18: CPU Scheduling

CS 105

Fall 2024

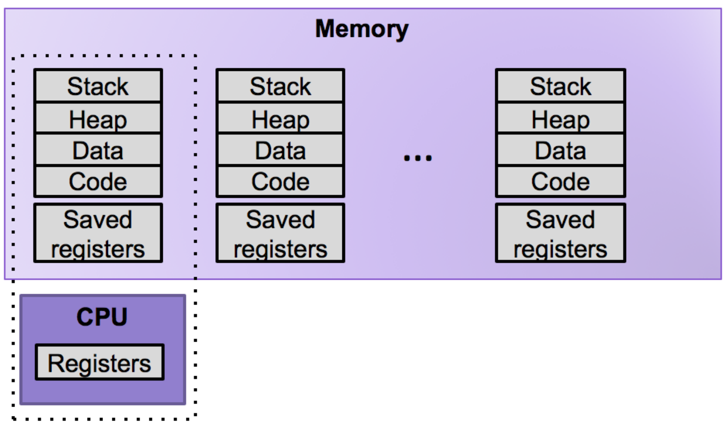
Review: Multiprocessing

The Illusion



- Abstraction: logical control flow within a process

The Reality



- Context switching b/n processes
- User cannot predict how instructions will interleave

Real-world Examples

- Restaurants handling orders
- DMV handling customers
- Students handling assignments
- Hospitals handling patients

Possible Metrics

- **Latency:** how much time between when a job is requested and when a job is completed
- **Response time:** how much time between when a job is requested and when you start processing the job
- **Throughput:** the rate at which jobs are completed

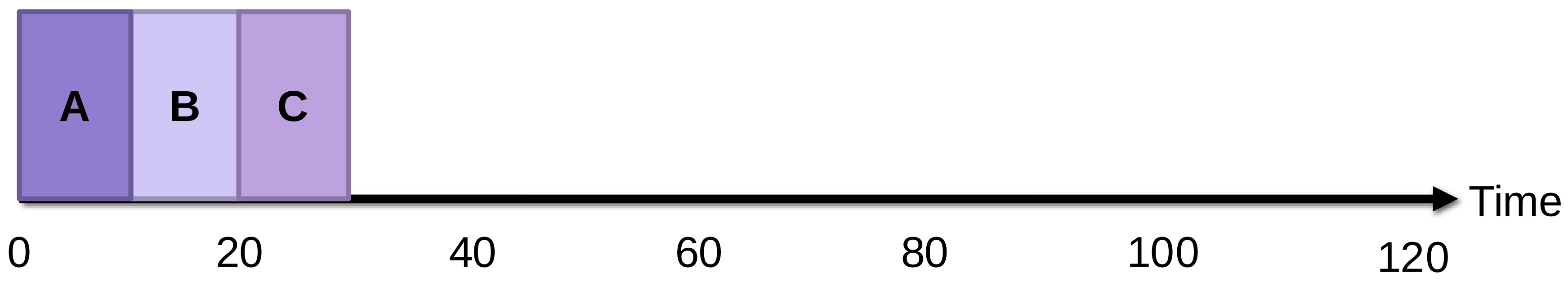
Simplifying Assumptions (for now)

1) Once you start a job, you complete that job before beginning the next job
2) The run-time of each job is known in advance
3) All jobs only use the CPU

First In, First Out (FIFO)

- Jobs are scheduled in the order they arrive

- Example:
  - Job A arrives at time 0, takes time 10 to complete
  - Job B arrives at time 5, takes time 10 to complete
  - Job C arrives at time 10, takes time 10 to complete



- Average Latency = $\frac{10+15+20}{3} = 15$
- Average Response = $\frac{0+5+10}{3} = 5$
- Throughput = $\frac{3}{30} = .1$

Exercise 1: First In, First Out (FIFO)

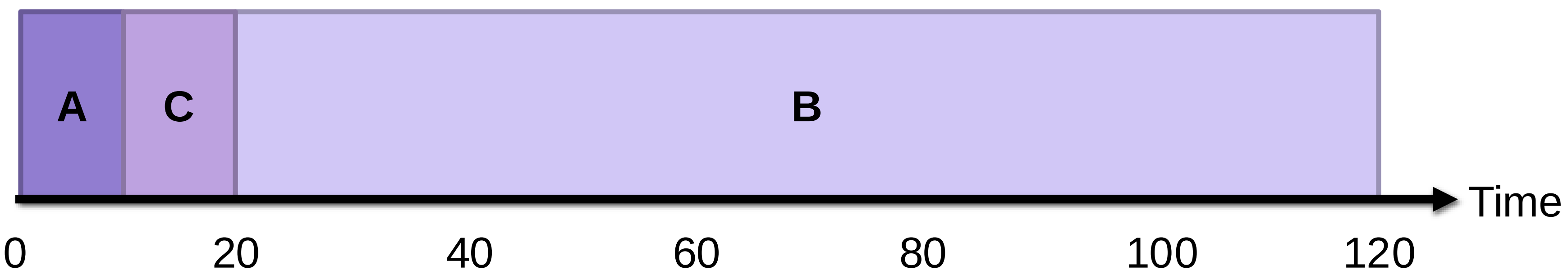
- Jobs are scheduled in the order they arrive

- Example:
  - Job A arrives at time 0, takes time 100 to complete
  - Job B arrives at time 5, takes time 10 to complete
  - Job C arrives at time 10, takes time 10 to complete

Shortest Job First (SJF)

- Jobs are scheduled in order of length (shortest first)

- Example:
  - Job A arrives at time 0, takes time 10 to complete
  - Job B arrives at time 5, takes time 100 to complete
  - Job C arrives at time 10, takes time 10 to complete



- Average Latency = $\frac{10+115+10}{3} = 45$
- Average Response = $\frac{0+15+0}{3} = 5$
- Throughput = $\frac{3}{120} = .025$

Exercise 2: Shortest Job First (SJF)

- Jobs are scheduled in order of length (shortest first)

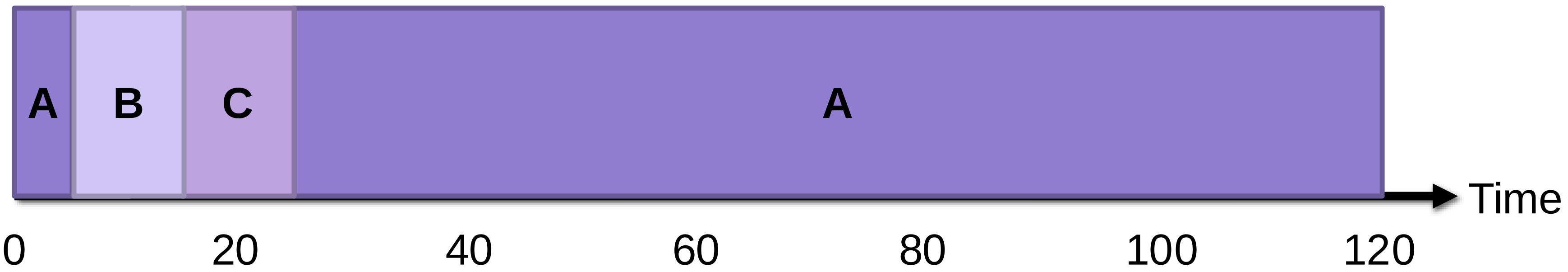
- Example:
  - Job A arrives at time 0, takes time 100 to complete
  - Job B arrives at time 5, takes time 10 to complete
  - Job C arrives at time 10, takes time 10 to complete

Simplifying Assumptions (for now)

1) ~~Once you start a job, you complete that job before beginning the next job~~
2) The run-time of each job is known in advance
3) All jobs only use the CPU

Shortest Time-to-Completion First (STCF)

- The job with the shortest time-to-completion is scheduled next
- If a job arrives with a shorter time-to-completion then the current job, it preempts the current job
- Example:
  - Job A arrives at time 0, takes time 100 to complete
  - Job B arrives at time 5, takes time 10 to complete
  - Job C arrives at time 10, takes time 10 to complete



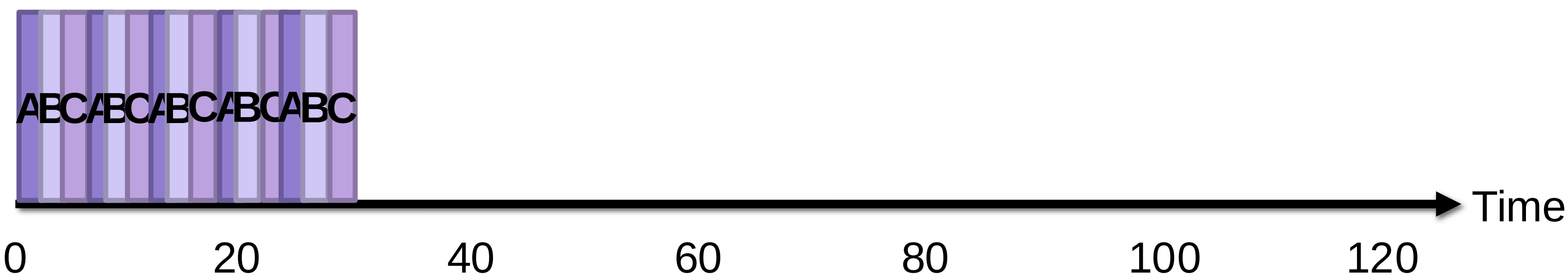
- Average Latency = $\frac{120+10+15}{3}$ = 48.3
- Average Response = $\frac{0+0+5}{3}$ = 1.6
- Throughput = $\frac{3}{120}$ = .025

Simplifying Assumptions (for now)

1) ~~Once you start a job, you complete that job before beginning the next job~~
2) ~~The run time of each job is known in advance~~
3) All jobs only use the CPU

Round Robin (RR)

- Run jobs for a fixed time slice (e.g., 2), cycle through all job that are not yet completed
- Example:
  - Job A arrives at time 0, takes time 10 to complete
  - Job B arrives at time 0, takes time 10 to complete
  - Job C arrives at time 0, takes time 10 to complete



- Average Latency = $\frac{26+28+30}{3} = 28$
- Average Response = $\frac{0+2+4}{3} = 2$
- Throughput = $\frac{3}{30} = .1$

Exercise 3: Round Robin (RR)

- Run jobs for a fixed time slice (e.g., 2), cycle through all job that are not yet completed
- Example:
  - Job A arrives at time 0, takes time 100 to complete
  - Job B arrives at time 10, takes time 10 to complete
  - Job C arrives at time 10, takes time 10 to complete

Comparing Scheduling Algorithms

- FIFO
  - works well if jobs are short
  - otherwise bad latency and bad response time

- STCF
  - good latency
  - very uneven response time (bad fairness)
  - assumes run-time of each job is known in advance

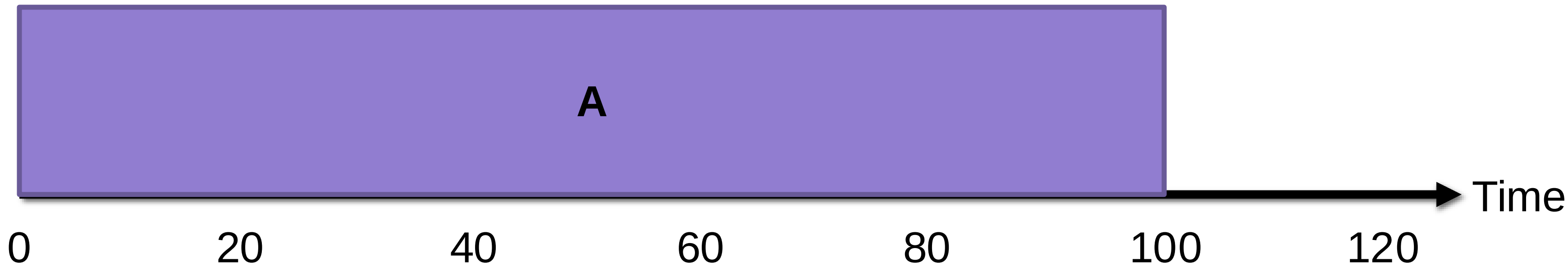
- RR
  - good response time
  - bad latency + overhead of context switching

Simplifying Assumptions (for now)

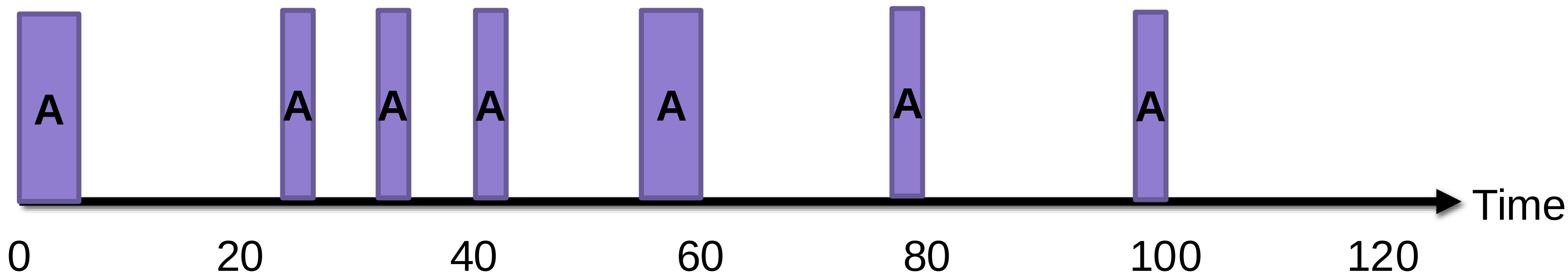
1) ~~Once you start a job, you complete that job before beginning the next job~~
2) ~~The run time of each job is known in advance~~
3) ~~All jobs only use the CPU~~

Processes are not all the same

- CPU-bound processes use a lot of CPU
  - e.g., compiling, scientific computing applications, mp3 encoding



- I/O-bound processes use CPU in short bursts
  - e.g., browsing small webpages, indexing a file system



- Balanced processes are somewhere in between
  - e.g., playing videos, moving windows around

Comparing Scheduling Algorithms

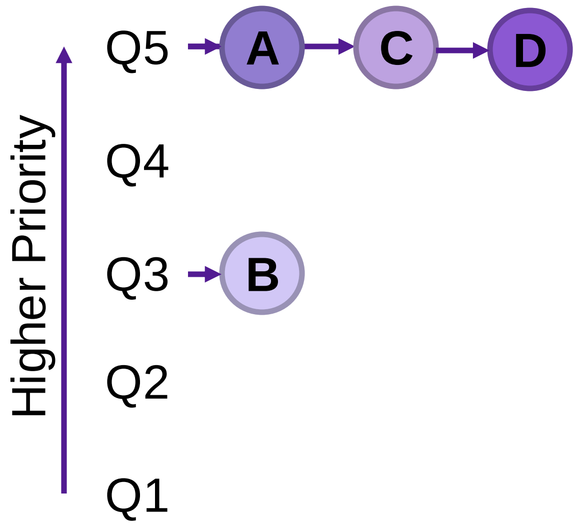
- FIFO
  - works well if jobs are short
  - otherwise bad latency and bad response time
- STCF
  - good latency
  - very uneven response time (bad fairness)
  - assumes run-time of each job is known in advance
- RR
  - good response time
  - bad latency + overhead of context switching
  - poor fairness for mixes of CPU-bound and I/O-bound

Multi-level Feedback Queues

- **Goal:** optimize latency while minimizing response time for interactive jobs without knowing run-time of jobs in advance
- General idea: maintain multiple queues, each with a different priority level

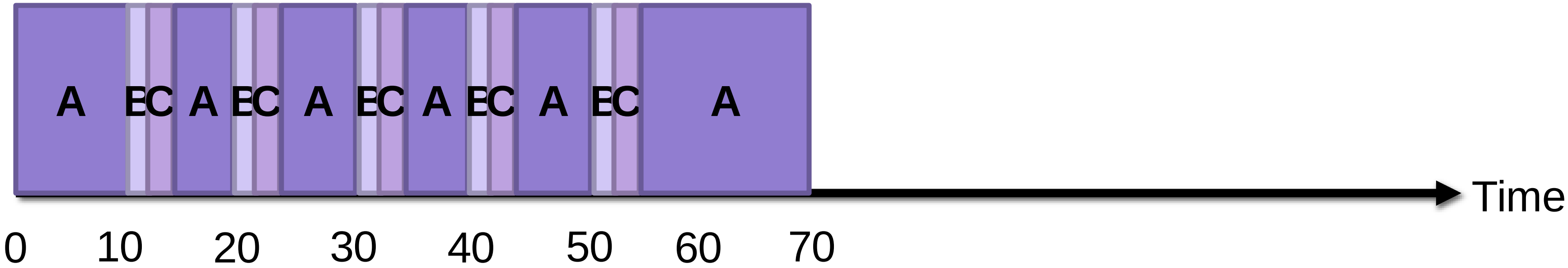
Scheduling rules:

1) If Priority(A) > Priority(B), run A
2) If Priority(A) = Priority(C), run A and C Round Robin
3) When a job enters the system, it is place in the highest priority queue
4) Once a job uses up its time allotment at current priority level, it moves down one queue
5) After some time period, move all jobs in the system to the highest priority queue



Example: Multi-level Feedback Queue

- Multilevel feedback queue with four levels with a time slice of 10 in the highest priority queue, 20 in the next, 40 in the next, and 80 in the lowest priority queue. Priorities reset every 200ms.
- Example:
  - Job A arrives first at time 0 and uses the CPU for 50ms before finishing.
  - Job B arrives at time 1. Job B loops five times; for each iteration of the loop, B uses the CPU for 2ms and then does I/O for 8ms.
  - Job C arrives at time 2. Job C is identical to Job B except for arrival time.



Schedulers in Operating Systems

- **CPU Scheduler** selects next process to run from the runnable pool
- **Page Replacement Scheduler** selects page to evict
- **Disk Scheduler** selects next read/write operation to perform
- **Network Scheduler** selects next packet to send/process