Chapters 9
Uniprocessor Scheduling

Part 0: The Story so Far

"This, has been the story, so far"

Process vs Threads

Process: A unit of isolation / protection

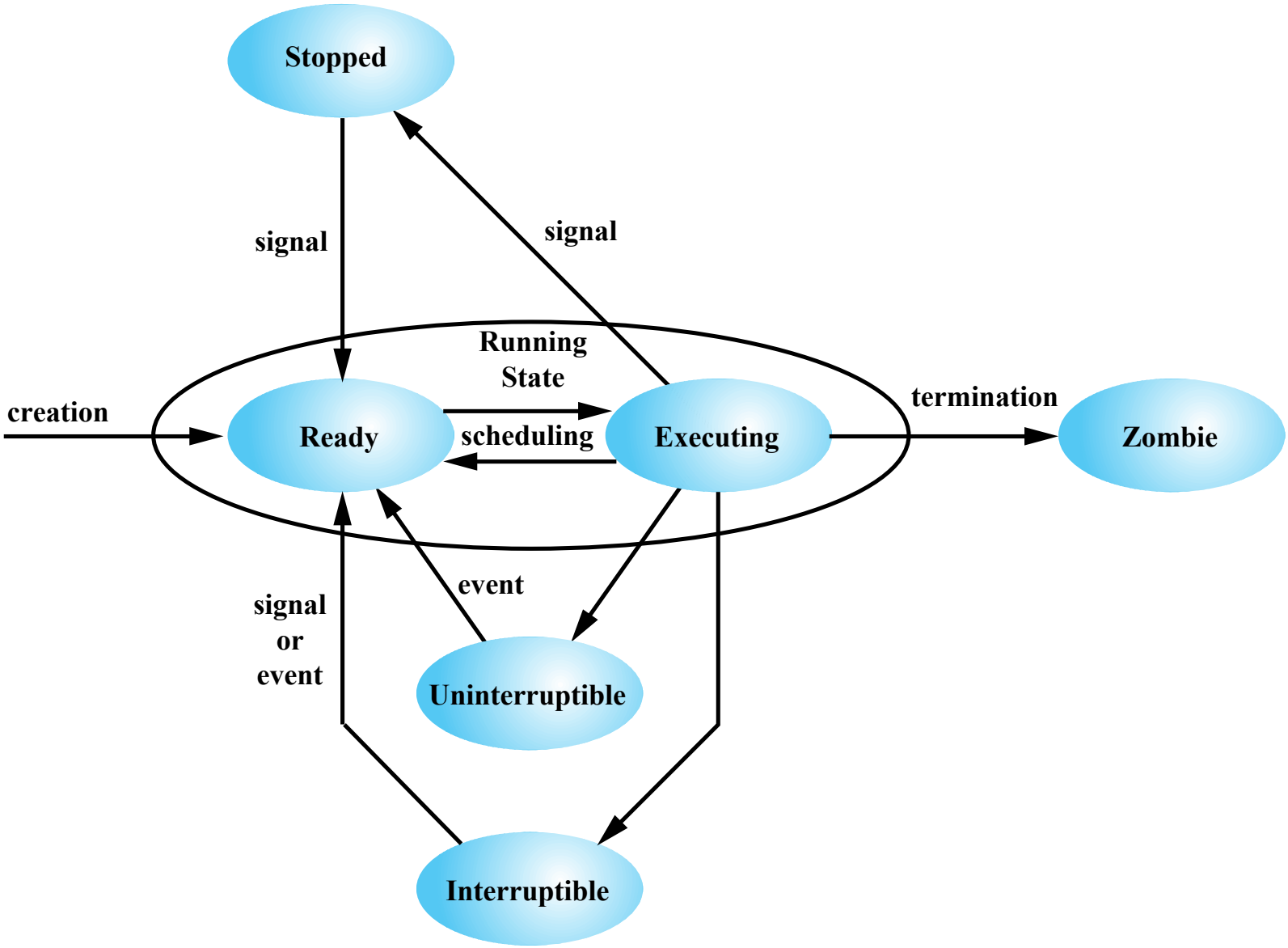
Thread: A point of execution within a program

The book *loves* to mix these two terms in this chapter…

Task:

- single threaded process
- single thread of a multi threaded process
- "Job"

Tasks Use the CPU



Reminder: Uniprocessor

Uniprocessor = Single core CPU

- only one task can be running at a time
- Old school
- hard enough as is

Part 1: Introduction

Definitions

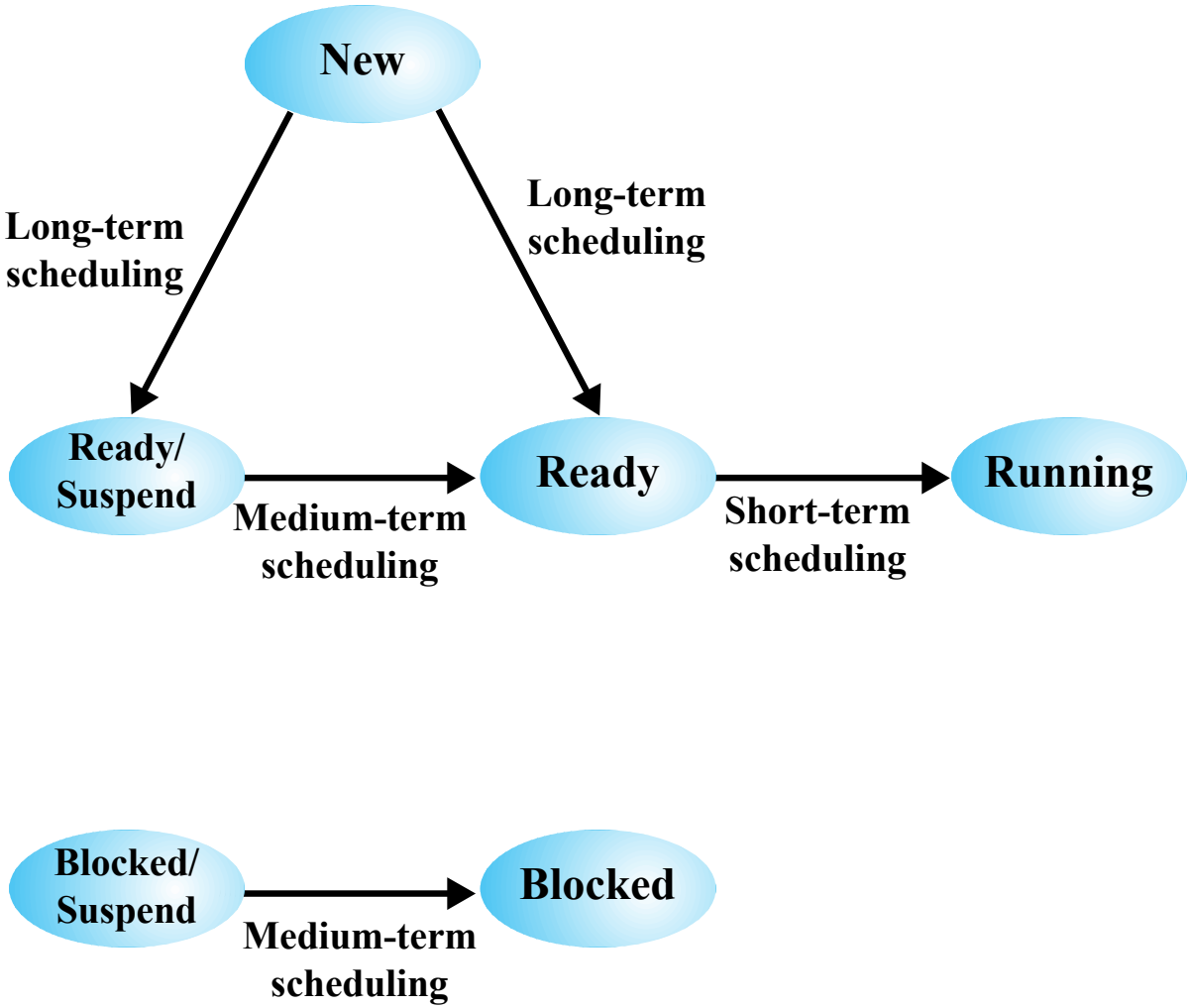
Long-term Scheduling: which tasks should even be in the pool of consideration

Medium-term Scheduling: which tasks should be in RAM

Short-term Scheduling: which availible task should be executed on the CPU

I/O Scheduling: which pending I/O request should be handled

Visual #1: State View



Visual #2: Hierarchical View

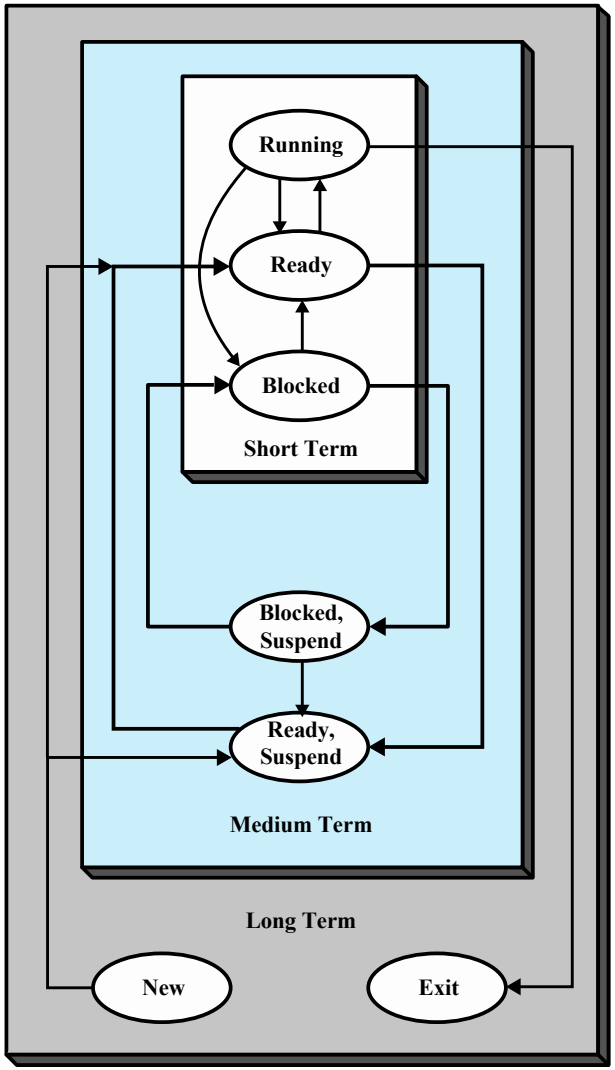


Figure 9.2 Levels of Scheduling

Visual #3: Queue View

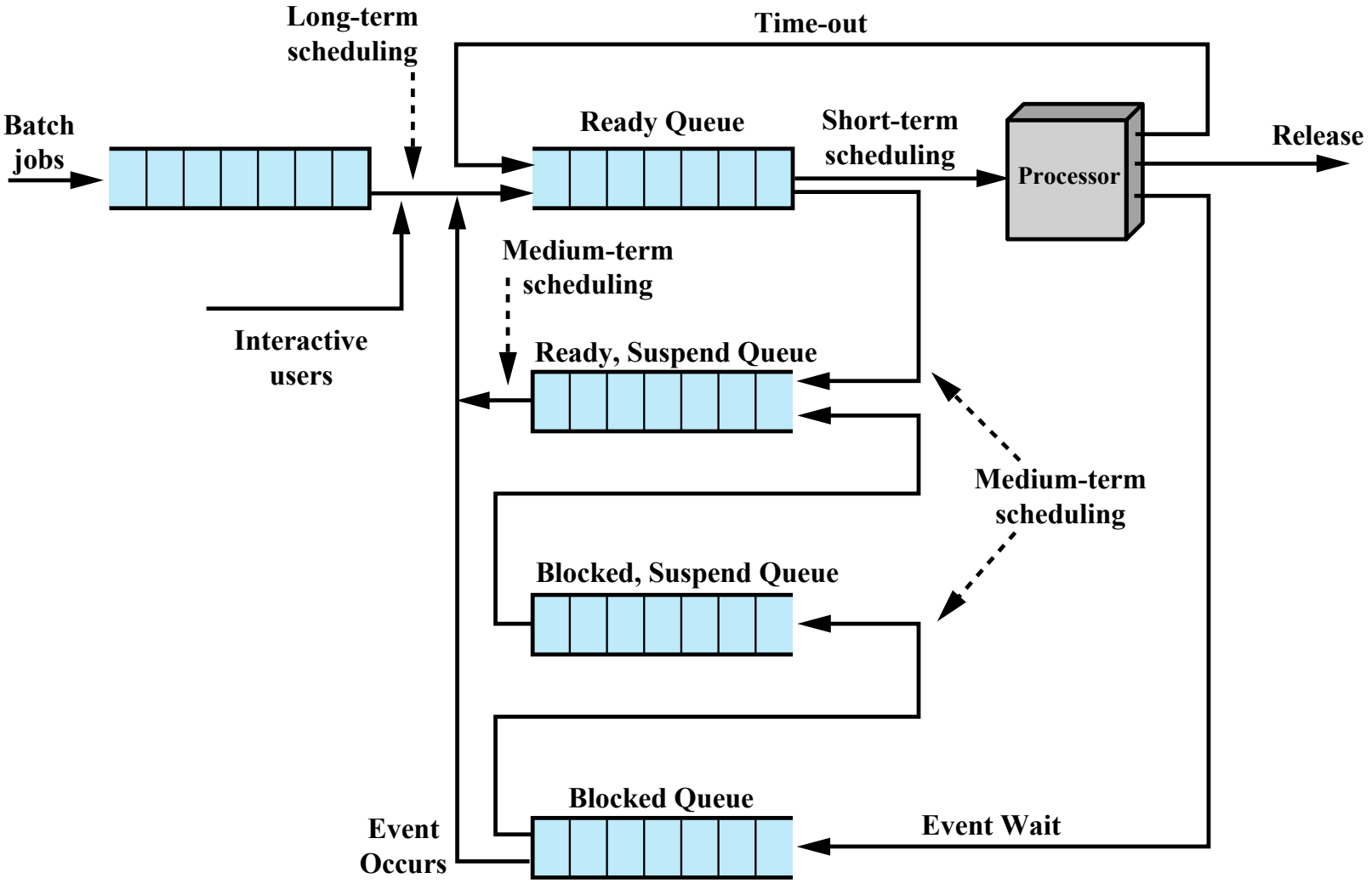


Figure 9.3 Queuing Diagram for Scheduling

Which will We Care About?

For the rest of this chapter…

later

- Long-term Scheduling
- Medium-term Scheduling

control the 'degree of multi programming'

how many tasks we can consider

CMP

- Short-term Scheduling
  → ran very often
- ~~I/O Scheduling~~

Part 2: Non-Preemptive Policies

FCFS, SPN

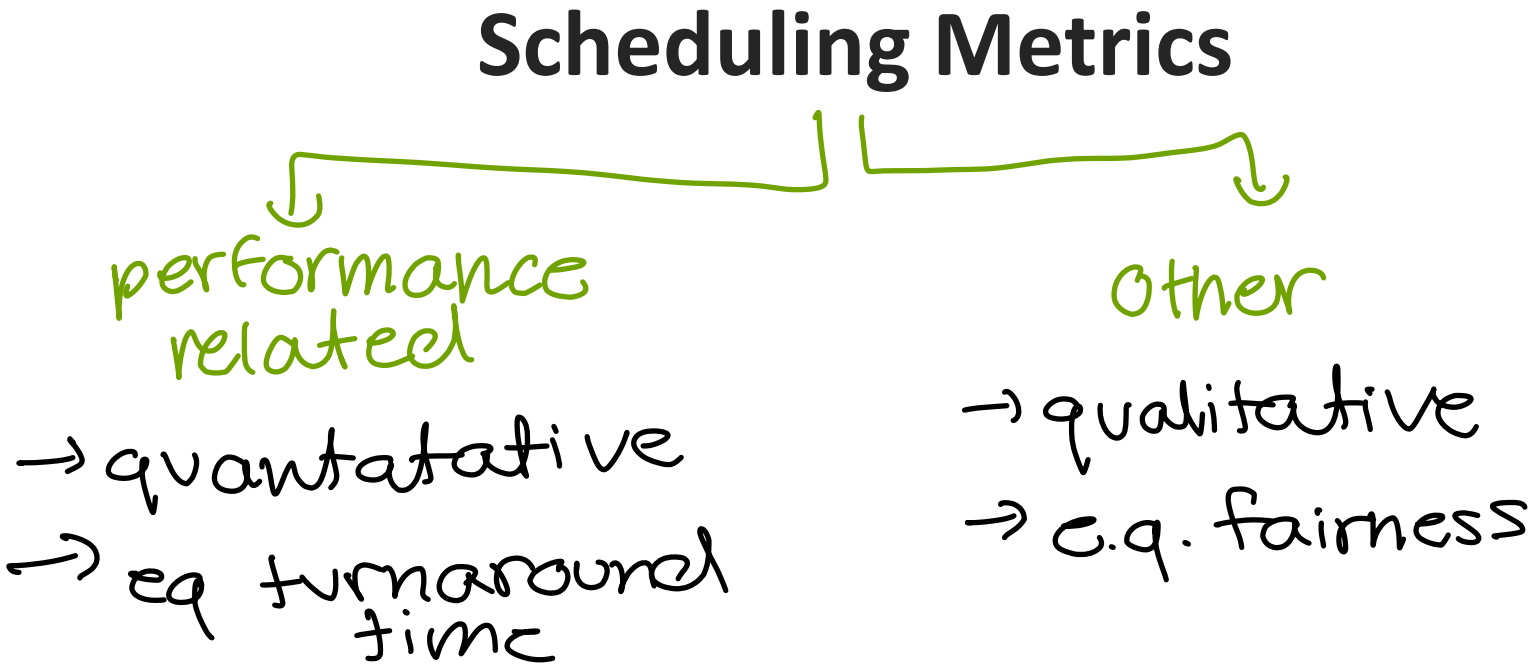
Scheduling Assumptions (for now)

To start, we'll assume that *all tasks*:

1. Run for the same amount of time
2. Arrive at the same time
3. Once started, run to completion
   (no preemption)
4. Only use the CPU
5. Have a known run-time

Scheduling Metrics

In addition, we need some way to *compare* schedules



Scheduling Metric #1

Turnaround Time:

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

$T_{completion}$: time when a task finishes

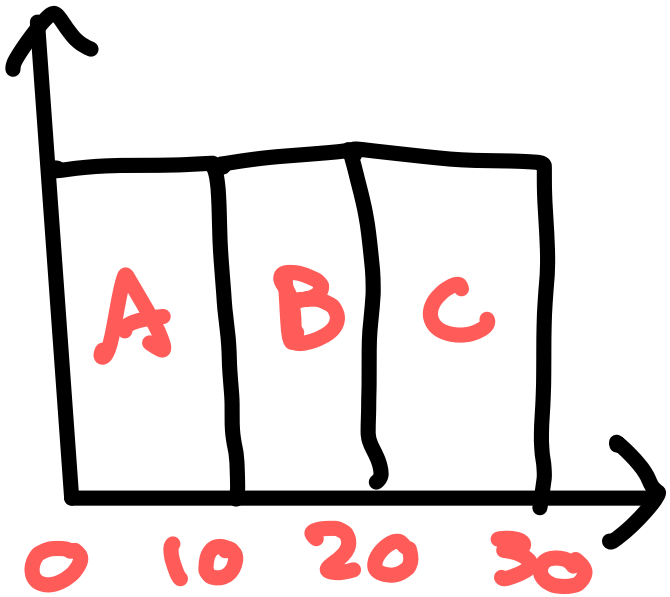
$T_{arrival}$: time when the task initially goes to ready

Policy #1: FCFS / FIFO

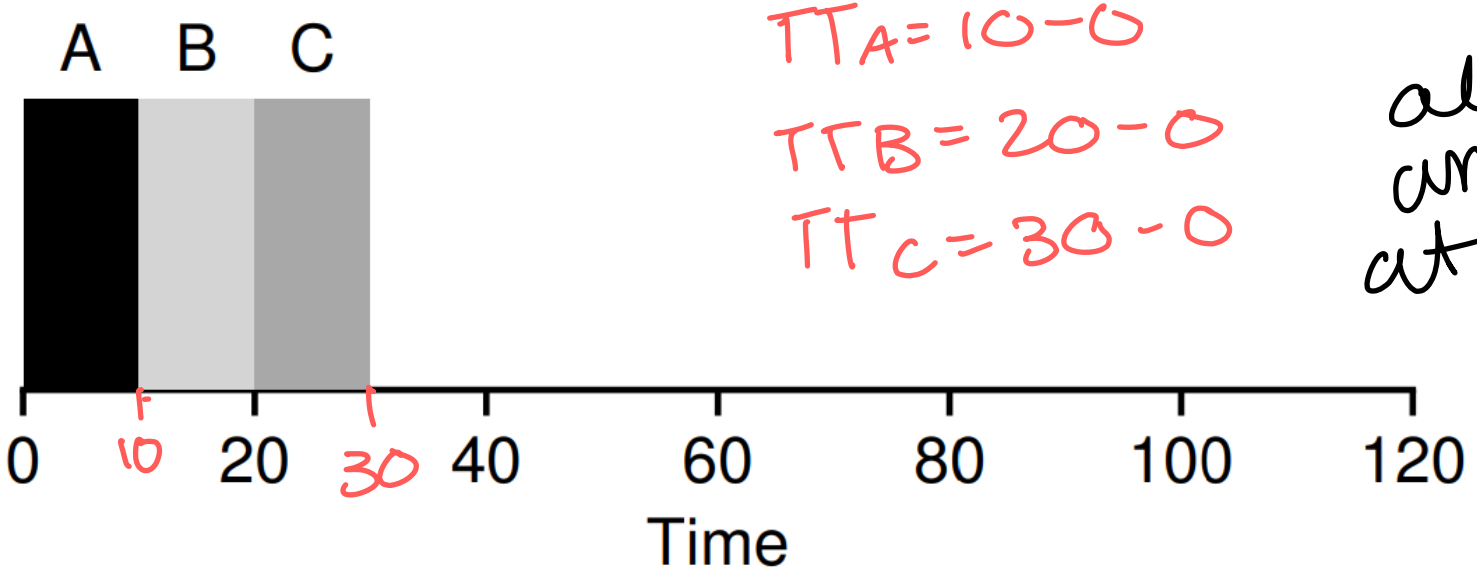
First come first served

Example #1: 3 tasks: A, B, & C.

- Arrive at *roughly* the same time ($T_{arrival} = 0$)
- Each job runs for 10 seconds
- Average turnaround time = ?



Example #1



$TT_A = 10 - 0$

$TT_B = 20 - 0$

$TT_C = 30 - 0$

all arrived at t=0

Figure 7.1: **FIFO Simple Example**

$$TT_{average} = \frac{10 + 20 + 30}{3} = 20s$$

Scheduling Assumptions (Rev 1)

All tasks:

1. ~~Run for the same amount of time~~
2. Arrive at the same time (roughly)
3. Once started, run to completion
4. Only use the CPU
5. Have a known run-time.

Example #2

Assume 3 jobs (A, B, & C)

- A runs for 100s, B&C run for 10s each.

Example #2

Assume 3 jobs (A, B, & C)

$TT_A = 100 - 0 = 100$

$TT_B = 110 - 0 = 110$

$TT_C = 120$

- A runs for 100s, B&C run for 10s each.

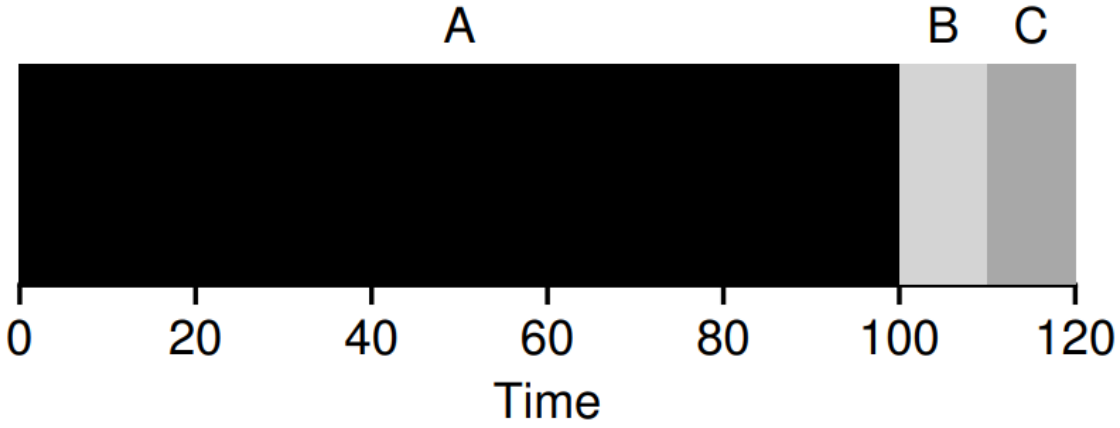


Figure 7.2: **Why FIFO Is Not That Great**

$$TT_{average} = \frac{100+110+120}{3} = 110$$

The Convoy Effect

A heavyweight task makes all others wait

→ Hurting the overall system

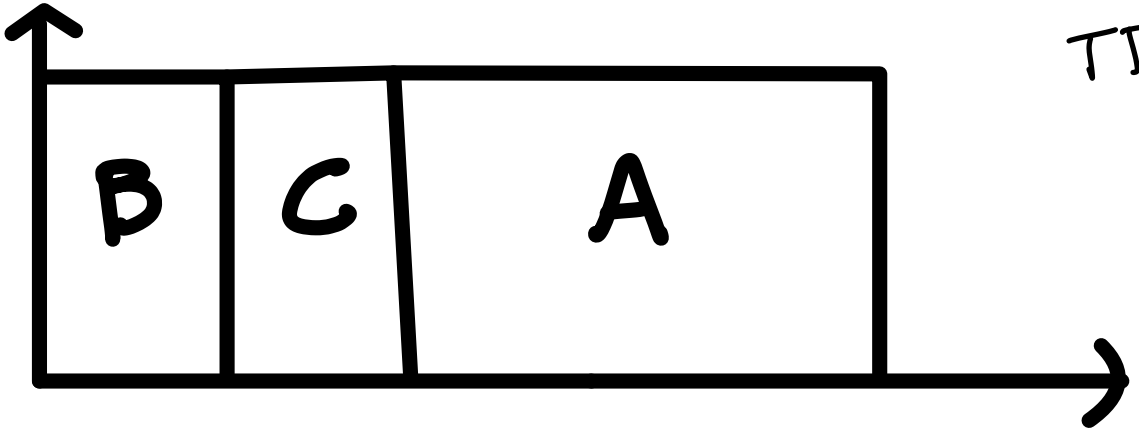
Shortest Process (Task) Next SPN

1) run the shortest task first
2) repeat

long tasks can starve

Re example 2: 3 tasks arrive at same time (A, B, & C)

- A runs for 100s, B&C run for 10s each.



TTavg = 50s

Scheduling Assumptions (Rev 2)

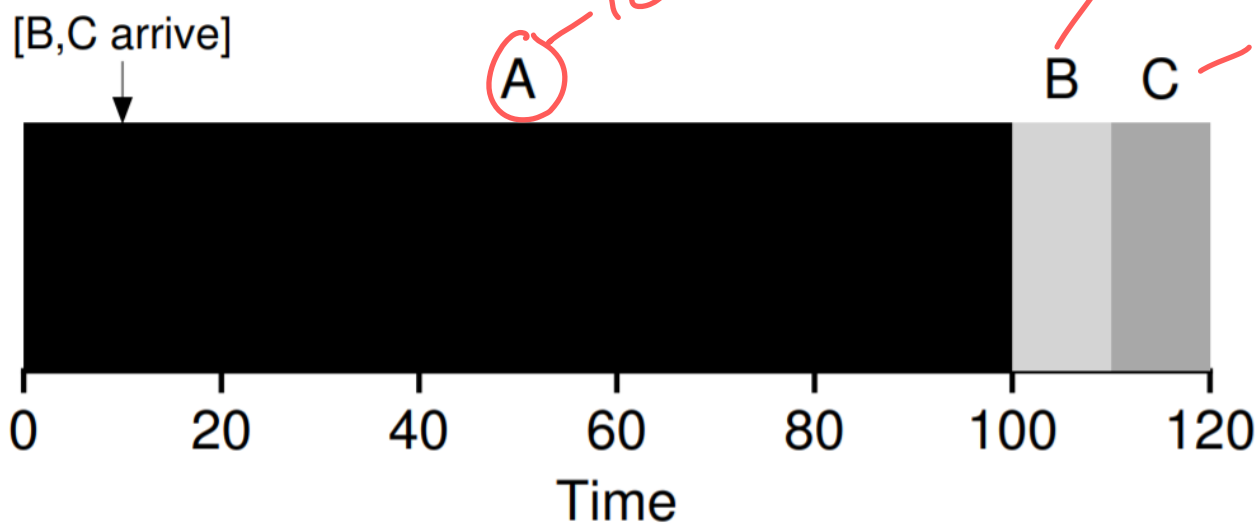
All tasks:

1. ~~Run for the same amount of time~~
2. ~~Arrive at the same time (roughly)~~
3. Once started, run to completion
4. Only use the CPU
5. Have a known run-time.

Example #3

Assume 3 jobs (A, B, & C)

- A runs for 100s, B&C run for 10s each.
- A arrives at 0, B&C at 10s.



100

110 - 10 = 100

120 - 10 = 110

$$TT_{average} = \frac{100 + 100 + 110}{3} = 103.3$$

Scheduling Assumptions (Rev 3)

All tasks:

1. ~~Run for the same amount of time~~
2. ~~Arrive at the same time (roughly)~~
3. ~~Once started, run to completion~~
4. Only use the CPU
5. Have a known run-time.

Part 3: Preemptive Policies

RR, SRT