CS 202: SCHEDULING

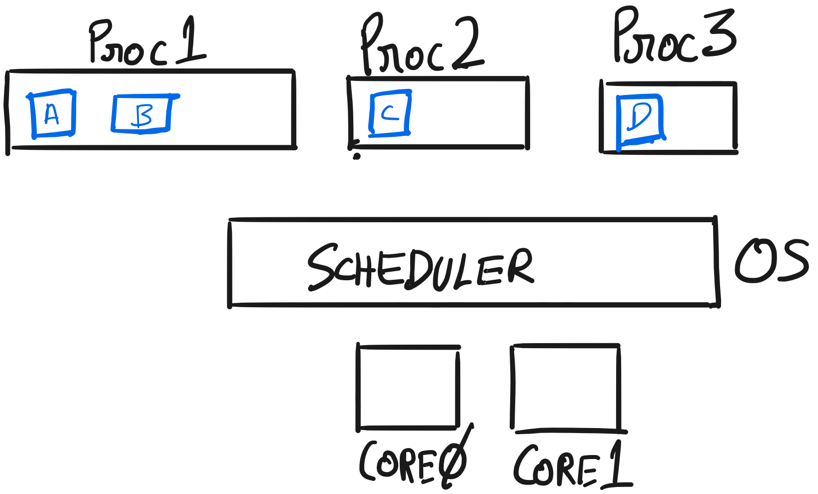
- Last class: Finished concurrency
- Next few classes
    - Today: Scheduling
    - Thursday: Dive into a bug from the past
        ↳ Want to derive lessons for system design. PLEASE READ BEFORE CLASS!
    - Next Tuesday: NO CLASS (MONDAY SCHEDULE)

Some comments on feedback

- Being lost, or unclear
    ↳ Please ask – even fine to say none of this makes sense. Happy to revisit again.
- Labs not covered in lectures
    - stat(2)
- How this all fits together.

Back to scheduled programming

- We talked about threads & processes



CORE IDEA: ONLY SOME THREADS EXECUTING AT A TIME.

TWO QUESTIONS:

- Mechanism: How to switch between threa

CONTEXT SWITCHES Nov.

THINGS TO KNOW

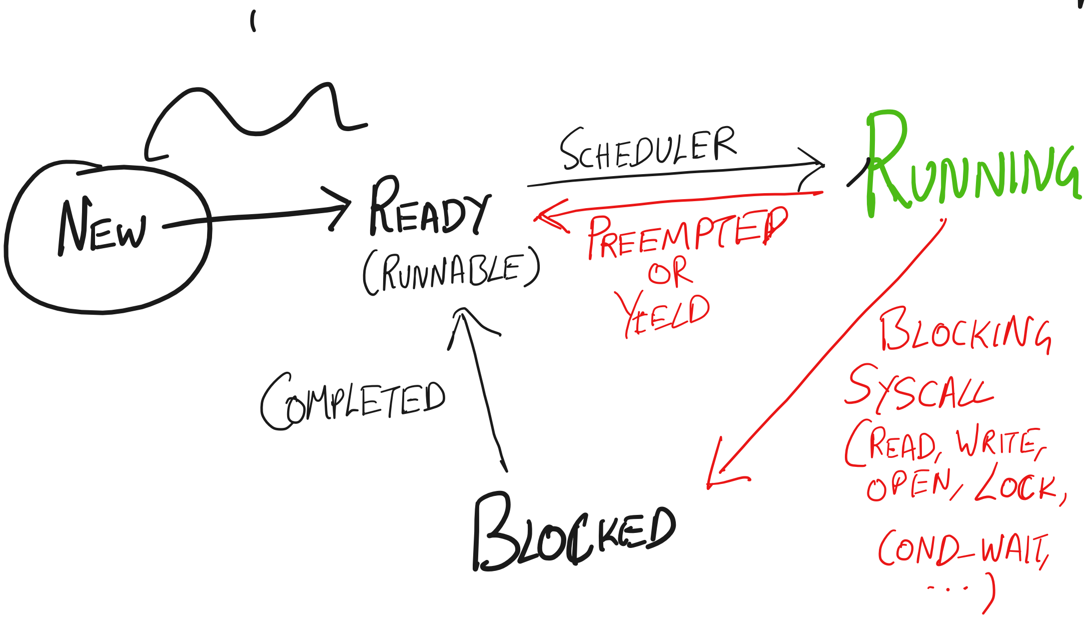
↳ COSTLY: Takes cycles, affects caches.

- Policy: WHAT TO RUN NEXT

TODAY

PROCESS/THREAD STATE — PCB ~ TCB

man 2 yield



PREEMPTION: Timer interrupt.
Make sure thread/process does not hold onto processor (core) for too long.

Yield: Syscall, invoke scheduler.

PREEMPTIVE SCHEDULING:

Preemption

COOPERATIVE SCHEDULING:

Yield

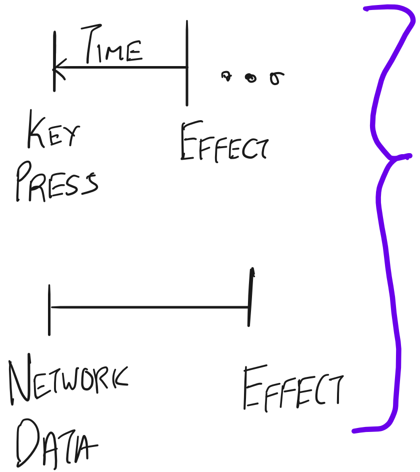
SCHEDULING POLICY (ALGORITHM)

- CAN DO NEARLY ANYTHING TO CHOOSE NEXT THREAD/PROCESS
  (someone, somewhere probably tried)

- GOING TO LOOK AT A FEW TODAY
  ... BUT NEED A WAY TO COMPARE THEM.

METRICS

- TURNAROUND TIME/COMPLETION TIME/RESPONSE TIME
  "How quickly does a process appear to work"



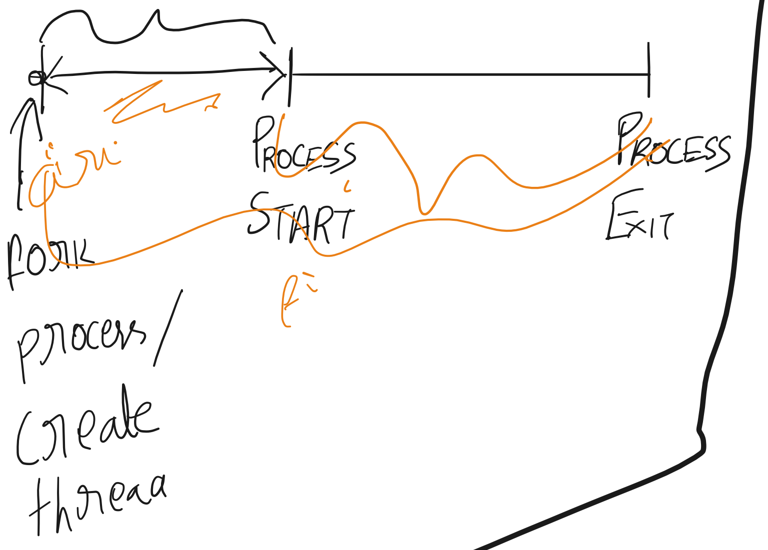
INTERACTIVE TASKS:

- Editors
- Browsers
- Terminals
- ...

TEXTBOOK: OUTPUT TIME*

*Many texts call this response time

TEXTBOOK: RESPONSE TIME



BATCH PROCESSING

- Video/Audio encoding
- Scientific Computation
  - ↳ Simulation
- Model training
- ...

TEXTBOOK: TURNAROUND TIME

- SYSTEM THROUGHPUT

NUMBER OF PROCESSES THAT CAN FINISH/ PRODUCE OUTPUT EVERY SECOND

- FAIRNESS
  - ALL PROCESSES/USERS GET THE SAME CHANCE TO USE PROCESSOR*
  - NO PROCESS STARVES
  - HIGH PRIORITY (IMPORTANT) THREADS/PROCESS GET MOST OF THE TIME

Building Intuition: No I/O

(A) FCFS (First come first served) Non-Preemptive

Batch

① P1 (24 seconds)
② P2 (3 seconds)
③ P3 (2 seconds)

24 + 24+3 +
27+ 2 = 80

Run until Process Finishes

Average Turnaround Time? $\frac{80}{3} \sim 27$

① P1 (24 seconds)
② P2 (4 seconds)
③ P3 (2 seconds)

Average throughput?

1

$3/30 =$

$1/10$

What happens to TURNAROUND time & tput if we change arrival order?

2 + (2+4) + (6+24)

① P3 (2 seconds)
② P2 (4 seconds)
③ P1 (24 seconds)

$\frac{38}{3}$

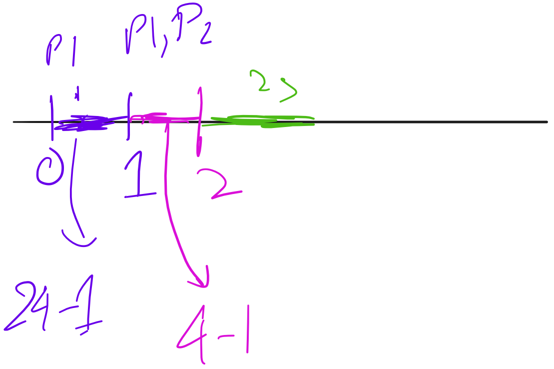
Ⓑ SHORTEST JOB FIRST (SJF) OR SHORTEST TIME TO COMPLETION FIRST (STCF)

Shortest Job First (Non-Preemptive)

① P3 (2 seconds)
② P2 (4 seconds)
③ P1 (24 seconds)

Shortest Time To Completion First (PREEMPTIVE)

PROCESS	ARRIVAL	TIME TO COMPLETION
P1	0	~~24~~
P2	1	~~4~~
P3	2	2



Ⓒ ROUND ROBIN

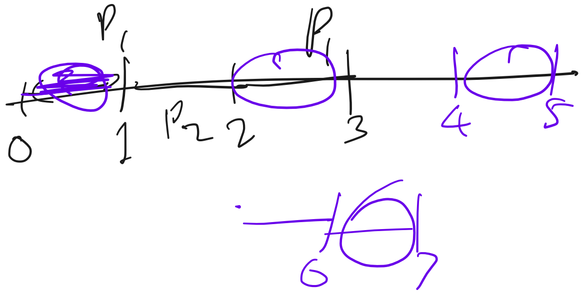
- PREEMPTIVE. <u>QUANTA / TIME SLICE</u> :

HOW LONG A PROCESS GETS TO RUN

- GO TO NEXT PROCESS

(MAKES TPUT & TURNAROUND TIME A BIT HARDER?)

PROCESS	ARRIVAL	TIME TO COMPLETION
P1	0	24
P2	0	6



QUANTA = 1

TURNAROUND TIME 12+12+18

$\frac{42}{\cancel{2}}$

TPUT

ADDING BACK I/O

A: Only CPU, 1 week

B: Only CPU, 1 week

C: Run forever

↳ 1ms CPU $\xrightarrow{\text{Block}}$ 10ms disk

↑ Unblock

- FIFO

- ROUND ROBIN: 100ms QUANTA

- ROUND ROBIN: 1ms QUANTA

STCF (Using CPU Time As Completion Time)

A	1 Week
B	1 Week
C	1 MS

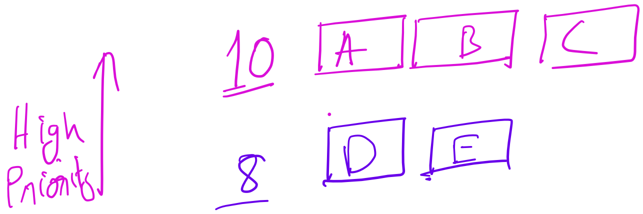
CPU

Disk

Observation: STCF Is Good, But

- Impractical: Don't Know Compute Time A-Priori

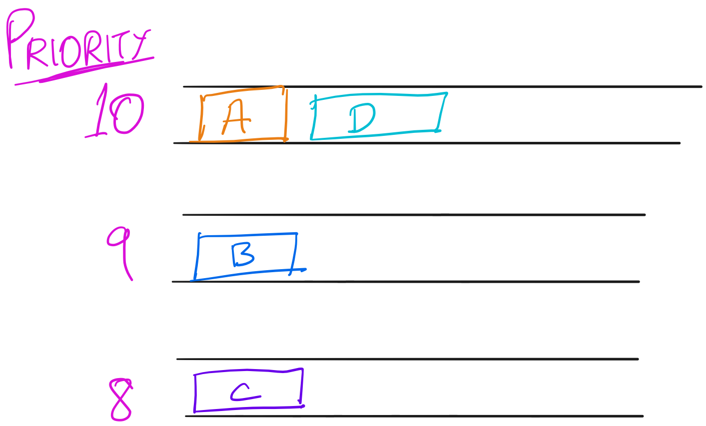
PRIORITY



STRICT PRIORITY

PROBLEM WITH STRICT PRIORITY

MULTI-LEVEL FEEDBACK QUEUE



FEEDBACK: CHANGE PRIORITY BASED ON HOW MUCH CPU

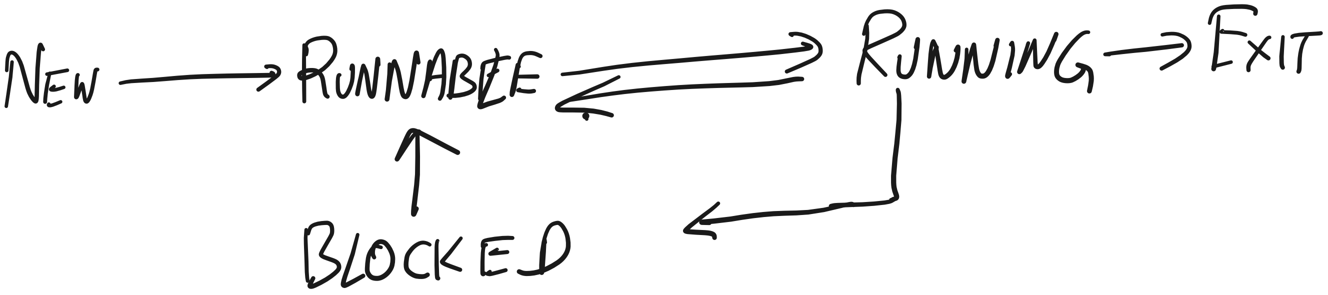
LINUX

- FAIR SCHEDULING
- LOTTERY SCHEDULING

A	10 tickets
B	5 tickets
C	15 tickets

THINGS TO TAKE AWAY

- PROCESS STATES



- SCHEDULING IS A COMMON PROBLEM
  - ↳ MUST CONSIDER REQUIREMENTS WHEN DECIDING ALGORITHM
- SOME OF THE SIMPLE SCHEDULING POLICIES WE TALKED ABOUT ARE VERY COMMON IN PRACTICE:
  - ROUND-ROBIN
  - MLFQ
- OFTEN, VALUABLE TO COMPARE TO GOOD BUT IMPRACTICAL ALGORITHMS (E.G. SCTF) IS OFTEN USEFUL.

([illegible], [illegible] [illegible] [illegible]