CS 202 (-002): Lecture 9: Scheduling

Notes / Announcement

- Next class/quiz
  - Important to do the Reading
    - ↳ Report on a real concurrency bug with real implications.
- No class on Tuesday, Oct. 7
- Observations from concurrency quiz
  - + Need to review coding standard
    - Several suggestions to move cv_signal outside critical section.
  - + "Must wait before signaling"
    - Wait & signal from different threads. Else DEADLOCK

Pred ≡ count < len
iter == current
...

```
mutex_lock(l);
while (!pred) {
    cv_wait(l, c);
}
mutex_unlock(l);
```

pred cannot hold until wait. Why?

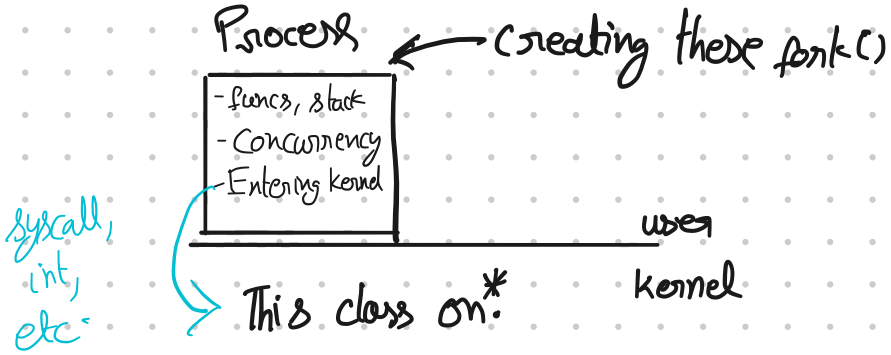
```
mutex_lock(l);
if (!pred) {
  //Change world
  // to fix pred
  cv_signal (l, c)
}
mutex_unlock(l);
```

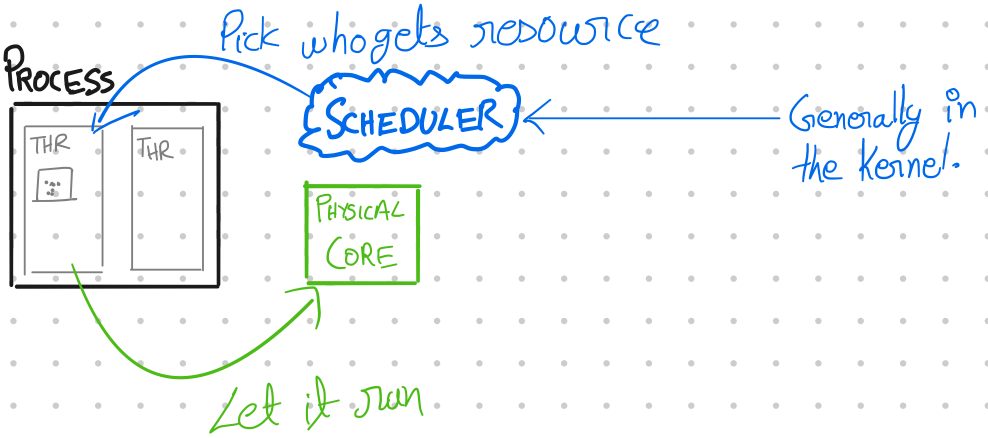
+ General advice: MORE PRACTICE

LAB 3

SCHEDULING

Where we are





Mechanics

1. Enter the scheduler
2. Choose what to run
   - (a) Track what can run
   - (b) Select next
3. Start running —— LATER (~ lec 16) CONTEXT SWITCH.

ENTRY POINT

1. Blocking calls – Pause execution until some condition is fulfilled
   - + mutex_lock cv_wait ...
   - + read write open
   - ...
2. Yield – give someone else a chance to run.

+ sched_yield (2)

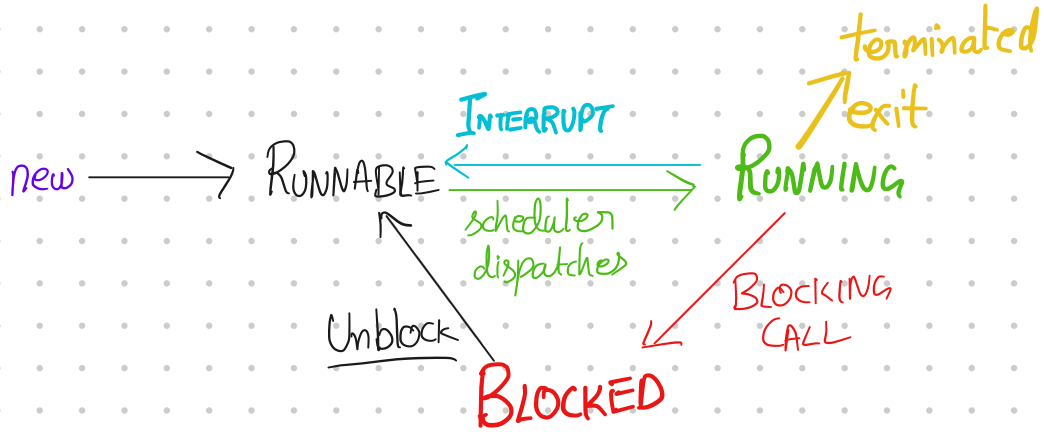
③ Preemption — Timer interrupt to get control back

Tracking What Can Be Run

Tasks

Process/Thread State

(stored in the TCB)



Choose among RUNNABLE tasks.

One thing to keep in mind

- Context switches are expensive
  - ↳ Write all registers → Memory
  - → Caches less likely to be valid
  - → ...

Select Next Task

- Usually, want to select to improve some metric
- Possible Metrics

- Turnaround Time: Time for task to complete
  - Wait time
    Response time
    Output time
    } How long does the user wait
- System throughput
  # of task completed per unit time
- Fairness
  Roughly - everyone gets equal access to processor
  - Usually weighted
- CPU utilization
- Energy
- ...

Which metric — Depends on use case.

Some scheduling algorithms

@ No I/O or other blocking calls
↳ Unrealistic, but simpler

(I) First come first serve

	Arr	Len
P1	0	1
P2	0	2
P3	0	30

Throughput $3/30 = \frac{1}{10}$

P1	0	24s
P2	0.1	3s
P3	0.2	3s

24

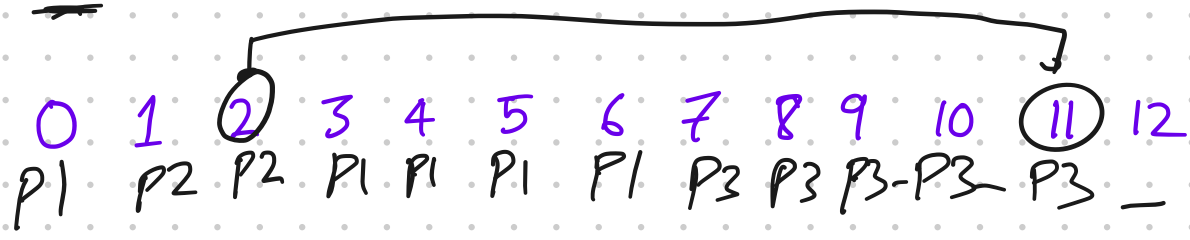
Average turnaround time

$$\frac{24 + (24+3) + (24+3+3)}{3}$$

II Shortest Job First (SJF)
Shortest Time to Completion First (STCF)

	Arr	Len
P1	0	~~5~~ 4
P2	1	~~2~~ 0
P3	2	5

$$\frac{2+7+9}{3}$$



Average Turnaround time

P1 0 4

P2 0 4

SJF: No preemption

0 1

P1

III Round Robin

↳ Preempt (using timer) every 'quanta' time units.

→ Go to next task

→ Quanta = 2

	Arr	Len
P1	0	5
P2	1	3
P3	2	5

0	1	2	3	4	5	6	7	8	9	10	11	12
P1	P1	P2	P2	P3	P3	P1	P1					

Average Turnaround time

Estimating Completion Time

- SJF / STCF depend on knowing completion time
  - ↳ Not common
- Solution: Predict using previous observation
  - Exp. weighted moving average [EWMA]

τ_n : Predicted time

t_n : observed time

$\tau_0 = t_0$

$\tau_n = t_n + \alpha \tau_{n-1} \quad 0 \le \alpha \le 1$

$\tau_1 = t_1 + \alpha t_0$

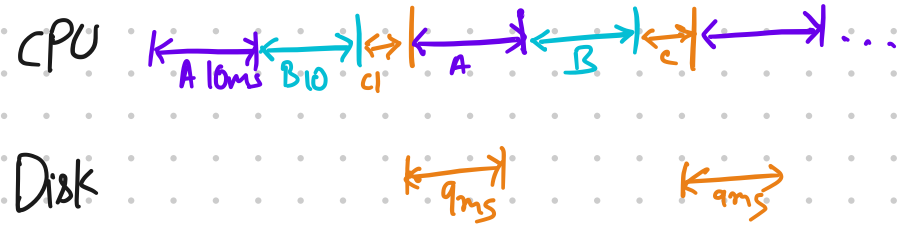
$\tau_3 = t_3 + \alpha t_2 + \alpha^2 t_1 + \alpha^3 t_0$

(b) Adding back I/O

Tasks

A:	CPU Bound	Run for 5 minutes	
B:	CPU Bound	Run for 5 minutes	
C:	I/O Bound	Run for 1ms	} Total 5 min
		Read from disk for 9ms	

- Round robin with 10ms quanta



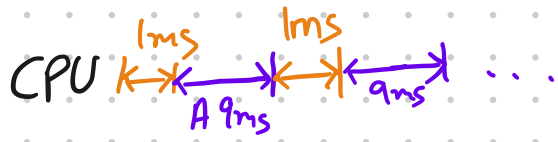
Observe: Disk used 9ms every 21ms

Used 3/7 ~ 42%

Quantum 100ms

9ms every 201ms ~ 4.4%.

- STCF



Disk utilization?

☆ Different trade-offs depending on what a process uses & scheduling policy

IV Priority Scheduling

- Strict Priority

Proc	Posi
A	1
B	2
C	3

[Lower Posiority => More imp.]

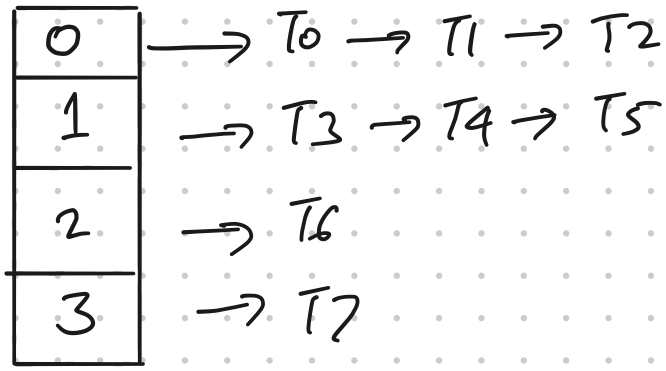
Problems?

- Multilevel feedback queue.

Goal: Improve RESPONSIVENESS & TURNAROUND TIME

RESPONSIVENESS: User interaction => I/O => Blocking

TURNAROUND TIME: Short tasks



- Round robin at each level
- If task uses all of its quanta ↓dec priority
- Periodically set all tasks to have high prio.

- Lottery / Stride Scheduling

Goals: Allow users to specify proportion of CPU time used by each process

gcc - 1 share

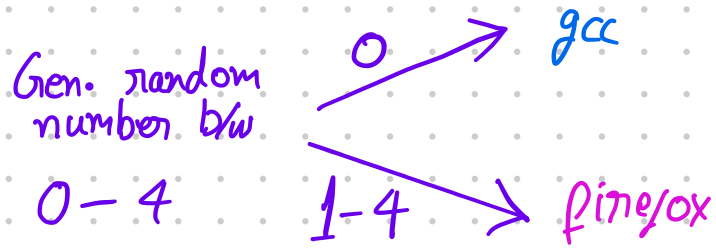
firefox/youtube - 4 shares

How:

- Lottery: Allocate at randomly, in expectation

meeting proportional share

For example



- Stride:

Assign each process a stride that is $\propto \frac{1}{\text{proportion}}$

For example gcc: 8

firefox: 2

Track CPU time in units of stride

↳ Run process that has spent min. tracked time on CPU.

gcc	FF
0	0
8	0
8	2
8	4
8	6
8	8
16	8

CFS

EEVDF