28 Jan 2025 - Operating Systems - II - Week 04

CPU Scheduling

CPU scheduler ⟶ 4 cases : process

1. switches from running to waiting
2. switches from running to ready
3. switches from waiting to ready
4. terminates

running → running in the CPU

ready state ⇒ in queue

1 & 4 : non preemptive (by itself)

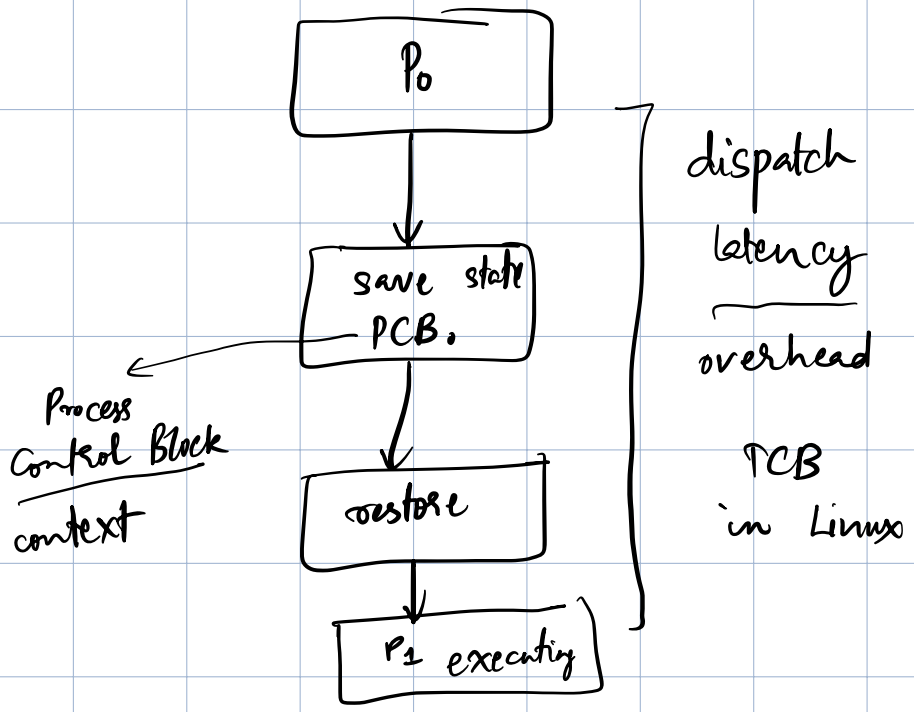
2 & 3 : preemptive

Process state diagram

why ① non preemptive?

→ I/O burst for process

Dispatcher



Scheduling criteria

→ CPU utilization — keep CPU busy → when is time wasted?
→ context switching

→ throughput — # of processes

→ Turnaround

→ Waiting

→ Response

FCFS

bad turnaround time

SJF

→ theoretically best

→ less predictable

estimation itself is an overhead

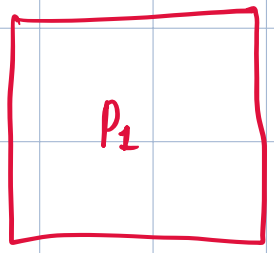
Shortest Remaining Time First

→ starvation: smaller jobs keep arriving

Scheduler where?

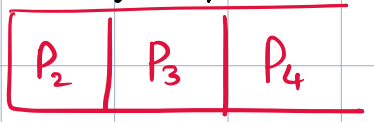
scheduler is also a program

→ runs on the same CPU

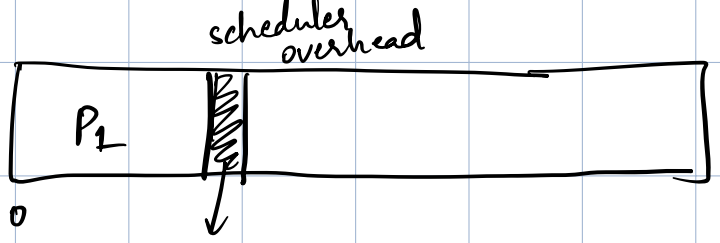


CPU

Ready queue



→ timer creates interrupt at regular intervals (time quantum)



ignored usually for simple analysis

Scheduler switch

Context switch time

depends on

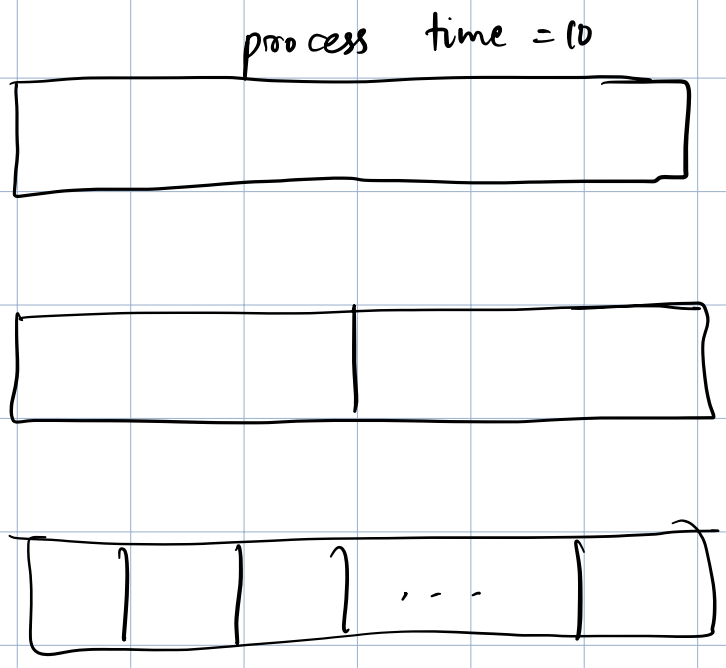
- → data: larger PCB ⟹ more time
- → hardware
- → OS

Scheduling

depends on

- → scheduling algorithm

Linux : BST



quantum	context switches
12	0
5	1
1	9

80% CPU burst > q (→ time quantum)

fork and pthread_create → create new process

- system call to invoke scheduler → parent has to wait due to overhead
- called by a process running in the CPU

In Windows: pthread_create: lighter than fork

Priority scheduling

→ starvation: low priority processes

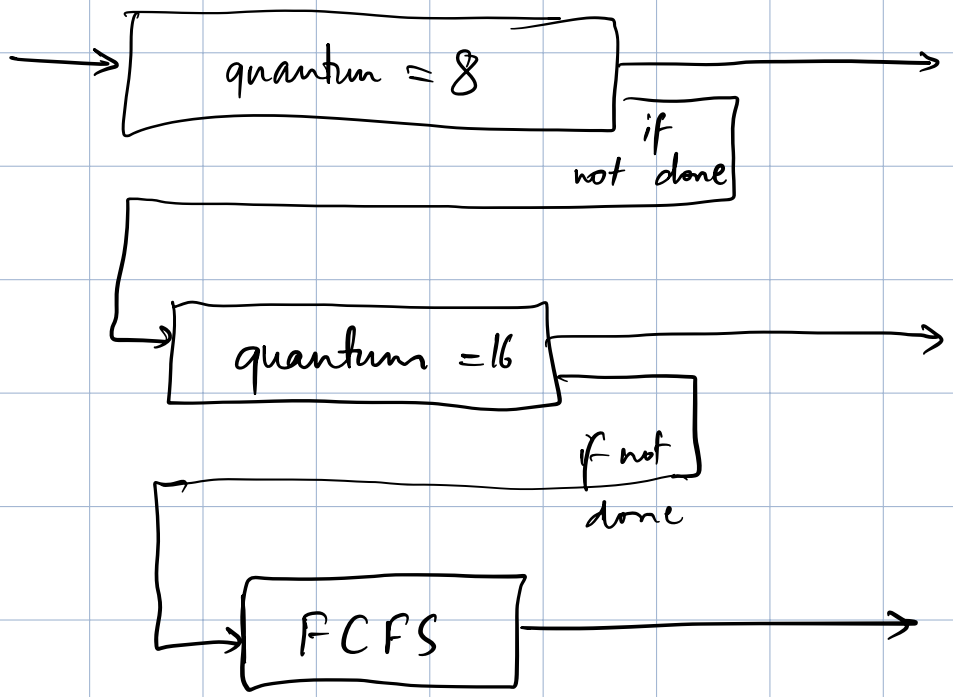
→ solution: aging

→ RR: if same priority

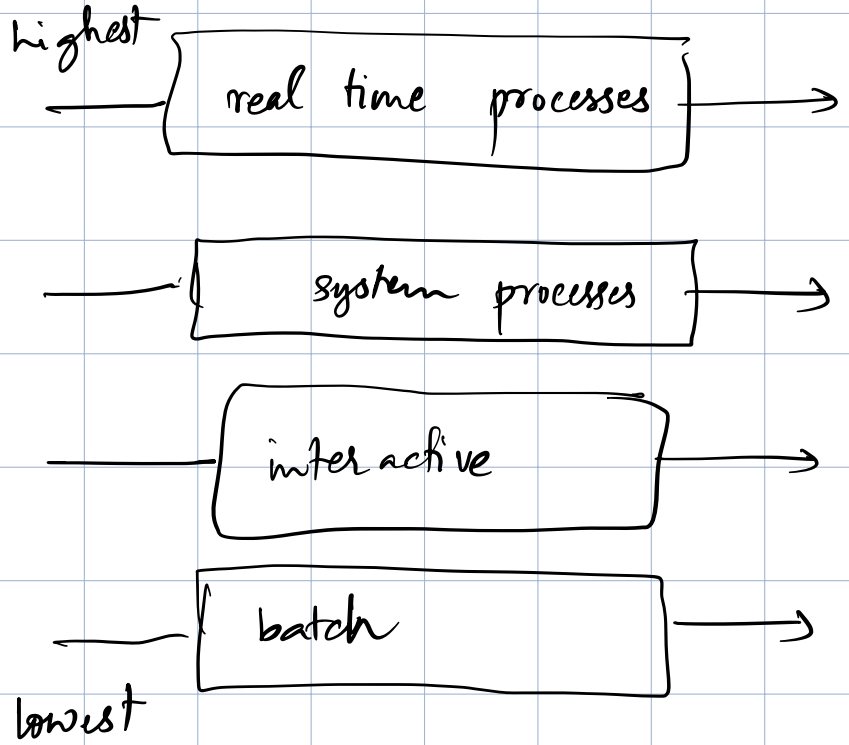
Multilevel Queue

→ prioritize based on process type.

→ each queue can have different scheduling algorithm



deadline → railway line
Hard → gates
→ ABS



Multilevel queue is defined by

- no. of queues
- scheduling algorithm for each queue

Single process scheduling

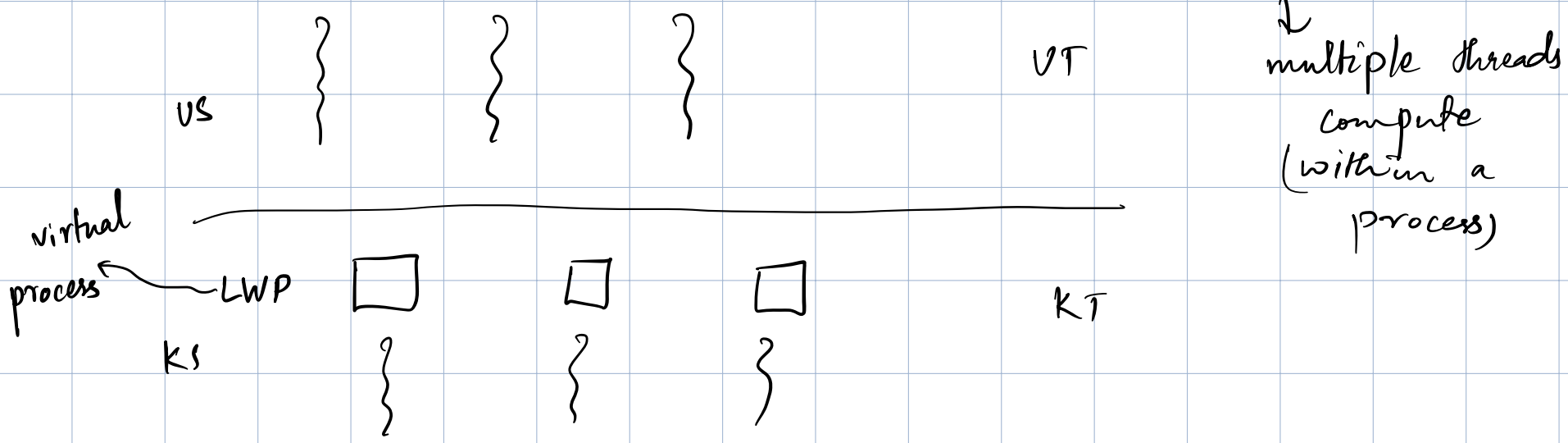
Thread scheduling ⇝ distinction b/w user level and kernel level threads

→ API

→ M-to-one & M-to-M models, thread library schedules user-level threads to run on LWP

→ process contention scope

↳ multiple threads compete (within a process)



→ Pthread Scheduling APIs

Next class: multiple cores scheduling