CSCI315 – Operating Systems Design

Department of Computer Science
Bucknell University

CPU Scheduling Algorithms

FCFS, SJF, Priority

Ch 5.3

This set of notes is based on notes from the textbook authors, as well as L. Felipe Perrone, Joshua Stough, and other instructors.
Xiannong Meng, Fall 2021.

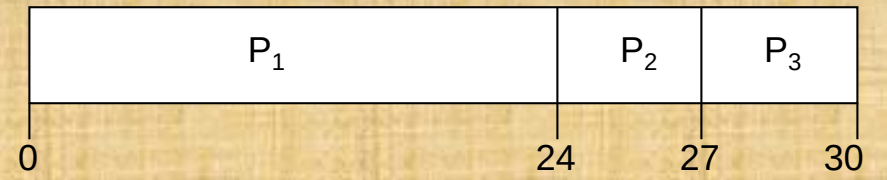
CPU Scheduling Algorithms

- In last segment, we discussed the basic idea of CPU scheduling.
- We'd like to arrange the execution of processes to gain the best performance.
  - maximum throughput, utilization
  - minimum waiting time, turn-around time, response time …
- In this segment, we will look at some algorithms aiming to achieve these goals.
- In these studies, we ignore the other cost such as context switching, just concentrate on CPU time.

First-Come, First-Served (FCFS)

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The **Gantt Chart** for the schedule is:



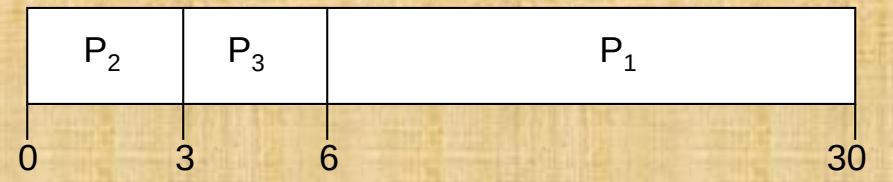
- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Issues with FCFS

Suppose that the processes arrive in the order

$$P_2 , P_3 , P_1$$

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6; P_2 = 0, P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case.
- *Convoy effect:* all process are stuck waiting until a long process terminates.

Shortest-Job-First (SJF)

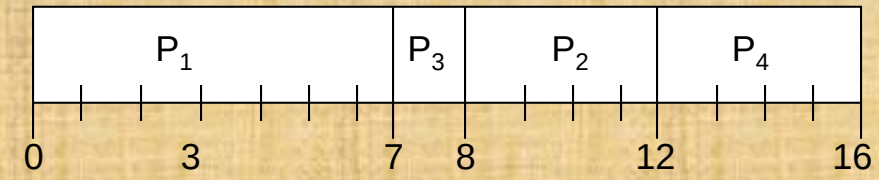
- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
- Two schemes:
  - **Non-preemptive** – once CPU given to a process it cannot be preempted until completing its CPU burst.
  - **Preemptive** – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF).
- SJF is **optimal** – gives minimum average waiting time for a given set of processes.

Question: Is this practical? How can one determine the length of a CPU-burst?

Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (non-preemptive)

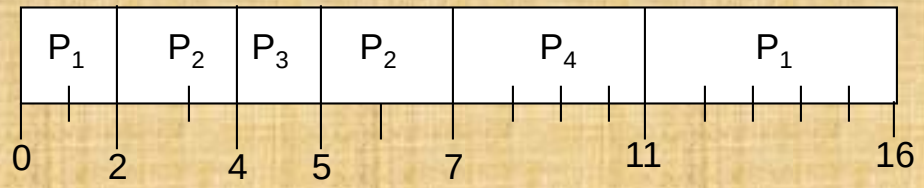


- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

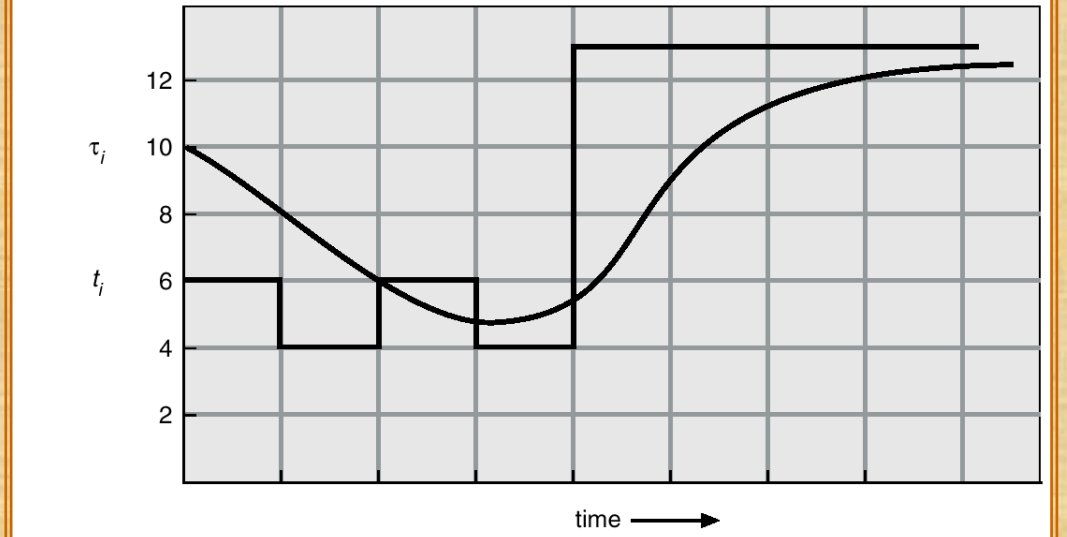
Determining Length of Next CPU-Burst

- We can only *estimate* the length.
- This can be done by using the length of previous CPU bursts, using exponential averaging:

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\tau_n$$

1. t_n = actual lenght of n^{th} CPU burst
2. τ_n = predicted value for the CPU burst at time n
3. $0 \leq \alpha \leq 1$
4. The effect of the value of α?

Prediction of the Length of the Next CPU-Burst



CPU burst (t_i)	6	4	6	4	13	13	13	. . .
"guess" (τ_i)	10	8	6	6	5	9	11	12	. . .

The graph is shown when α is 0.5

Example

- Given the actual (measured) CPU bursts are 6, 4, 6, 4, 13, 13, 13, and the initial estimate of τ is 10 as in previous slide, show the **first three** predictions when α takes the value of
  - 0.2
  - 0.7
- When α is 0.2, estimates are 9.2, 8.16, 7.73
- When α is 0.7, estimates are 7.2, 4.96, 5.69
- See an example computation on next slide

Example Computation

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n$$

t_i = 6, 4, 6, 4, 13, 13, 13 --- these are measured time

τ_0 = 10, α = 0.2, t_0 = 6

τ_1 = 0.2*6 + 0.8*10 = 9.2

τ_2 = 0.2*4 + 0.8*9.2 = 8.16

τ_3 = 0.2*6 + 0.8*8.16 = 7.73

Priority Scheduling

- A priority number (integer) is associated with each process.
- The CPU is allocated to the process with the highest priority (typically, smallest integer ≡ highest priority)
  - Preemptive
  - Non-preemptive
- SJF is a priority scheduling where priority is the predicted next CPU-burst time.
- Problem: **Starvation** – low priority processes may never execute.
- Solution: **Aging** – as time progresses increase the priority of the process.

Process Priority in Linux

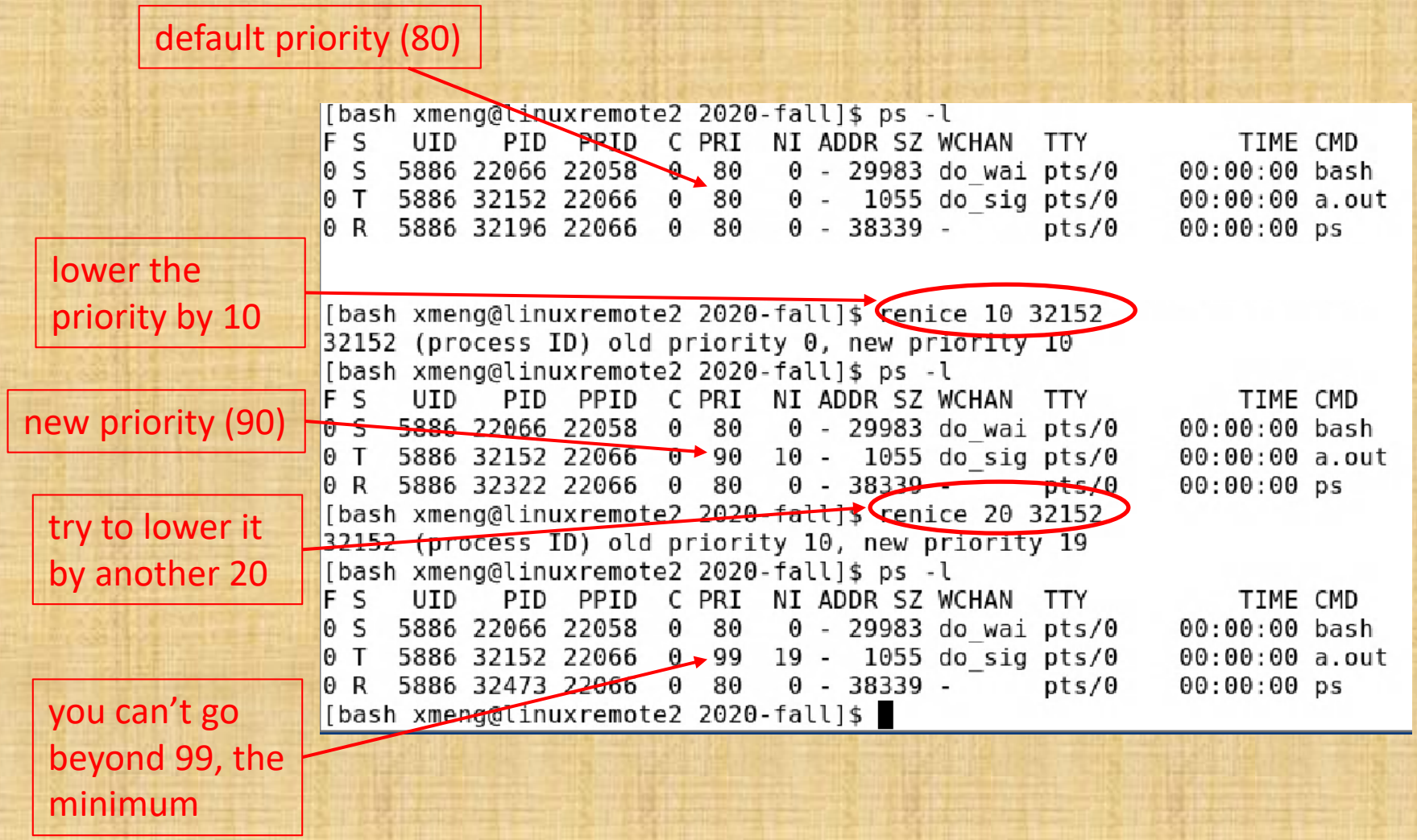
- Priority scheduling is commonly used in production OSes such as Linux
- In Linux, the priority values range from 1 (most favorite) to 99 (least favorite)
- Try `ps -l` command on a Linux terminal
- Default priority of a user process is 80.
- We can run a CPU intensive job and use the `nice` command to set its priority, or `renice` command to change its priority. (Range of `renice` is 0 to 20.)

https://study.com/academy/lesson/process-priorities-in-linux-definition-modification.html

To check priority levels

Use Linux command **chrt** to check levels of priority.

```
[bash xmeng@linuxremote2 ~]$ chrt -m
SCHED_OTHER min/max priority    : 0/0
SCHED_FIFO min/max priority     : 1/99
SCHED_RR min/max priority       : 1/99
SCHED_BATCH min/max priority    : 0/0
SCHED_IDLE min/max priority     : 0/0
SCHED_DEADLINE min/max priority : 0/0
[bash xmeng@linuxremote2 ~]$
```


default priority (80)
lower the priority by 10
new priority (90)
try to lower it by another 20
you can't go beyond 99, the minimum
[bash xmeng@linuxremote2 2020-fall]$ ps -l
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
0 S 5886 22066 22058 0 80 0 - 29983 do_wai pts/0 00:00:00 bash
0 T 5886 32152 22066 0 80 0 - 1055 do_sig pts/0 00:00:00 a.out
0 R 5886 32196 22066 0 80 0 - 38339 - pts/0 00:00:00 ps
[bash xmeng@linuxremote2 2020-fall]$ renice 10 32152
32152 (process ID) old priority 0, new priority 10
[bash xmeng@linuxremote2 2020-fall]$ ps -l
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
0 S 5886 22066 22058 0 80 0 - 29983 do_wai pts/0 00:00:00 bash
0 T 5886 32152 22066 0 90 10 - 1055 do_sig pts/0 00:00:00 a.out
0 R 5886 32322 22066 0 80 0 - 38339 - pts/0 00:00:00 ps
[bash xmeng@linuxremote2 2020-fall]$ renice 20 32152
32152 (process ID) old priority 10, new priority 19
[bash xmeng@linuxremote2 2020-fall]$ ps -l
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
0 S 5886 22066 22058 0 80 0 - 29983 do_wai pts/0 00:00:00 bash
0 T 5886 32152 22066 0 99 19 - 1055 do_sig pts/0 00:00:00 a.out
0 R 5886 32473 22066 0 80 0 - 38339 - pts/0 00:00:00 ps
[bash xmeng@linuxremote2 2020-fall]$