Lab 6: Processes scheduling

Shokhista Ergasheva, Muwaffaq Imam, Artem Kruglov,
Nikita Lozhnikov, Giancarlo Succi, Xavier Vasquez
Herman Tarasau, Firas Jolha

Innopolis University
Course of Operating Systems

October 2022

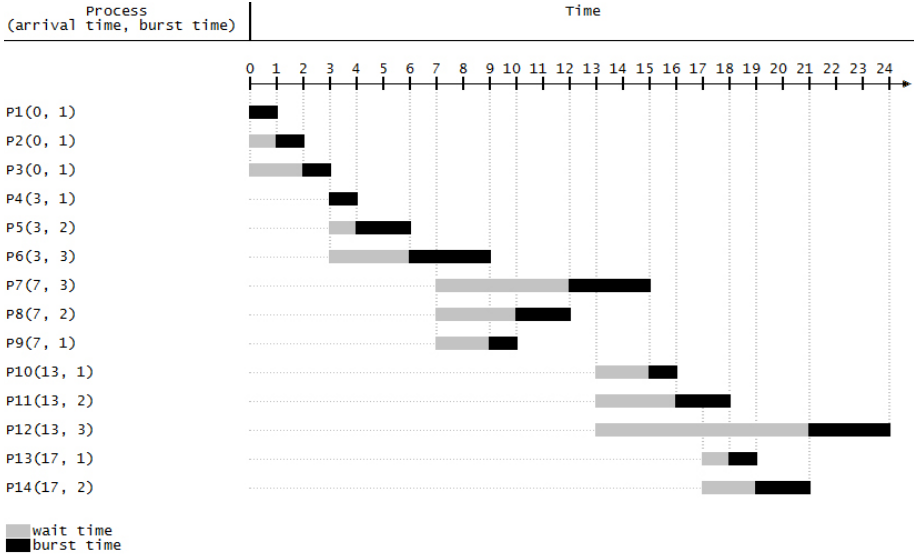
First come, first served

- First in, first out (FIFO), also known as first come, first served (FCFS), is the simplest scheduling algorithm. FIFO simply queues processes in the order that they arrive in the ready queue. This is commonly used for a **task queue**

Shortest job next(1/2)

- Shortest job next (SJN), also known as shortest job first (SJF) or shortest process next (SPN), is a scheduling policy that selects for execution the waiting process with the smallest execution time. SJN is a non-preemptive algorithm. Shortest remaining time is a preemptive variant of SJN.

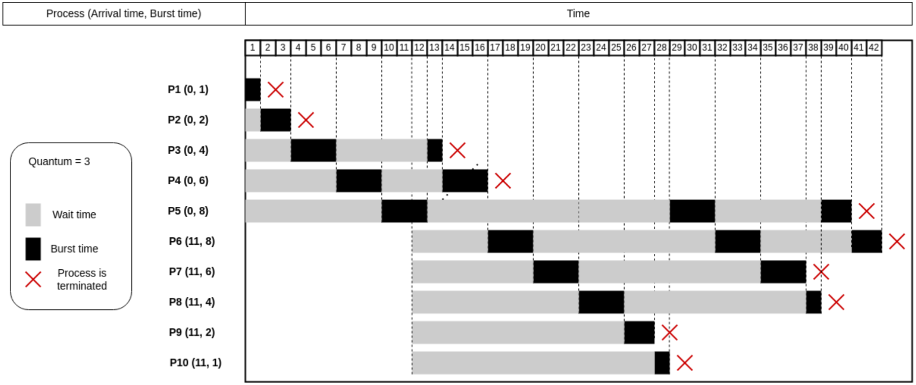
Shortest job next(2/2)



Round robin(1/2)

- A round-robin scheduler generally employs time-sharing, giving each job a time slot or quantum (its allowance of CPU time), and interrupting the job if it is not completed by then. The job is resumed next time a time slot is assigned to that process.
- Round-robin algorithm is a pre-emptive algorithm as the scheduler forces the process out of the CPU once the time quota expires.

Round robin(2/2)



Exercise 1

- Write a program **ex1.c** to simulate the first come, first served (FCFS) algorithm.
- The implementation should show the following metrics
  - Completion time(CT)
  - Turn around time(TAT)
  - Waiting time(WT)
  - Average Turnaround time
  - Average waiting time
- The program should accept from stdin the number of processes and arrival time and burst time for each process.
- Submit **ex1.c** with a supplement script **ex1.sh** to run the program.
- **Note:** Go to the last slides to see the formula of the required metrics.

Exercise 2

- Create a new program **ex2.c**, in which you should implement shortest job first algorithm.
- The implementation should follow the same requirements as the previous exercise.
- Compare the output with the outputs of the previous exercise and save the explanation in **ex2.txt**
- Submit **ex2.txt**, **ex2.c** with a supplement script **ex2.sh** to run the program.

Exercise 3

- Write a program **ex3.c** to simulate the round robin algorithm, which should show the same metrics as the two previous programs.
- Compare the output with the outputs of the previous two exercises and save the explanation in **ex3.txt**
- Submit **ex3.txt**, **ex3.c** with a supplement script **ex3.sh** to run the program.

- Note: for this algorithm, the quantum should be specified by the user from stdin.

Exercise 4 (Optional)

- Write your own implementation of any additional scheduling algorithm you like.
- It should accept number of processes and CPU burst of every process.
- It should represent results as a timeline
- Example:

$$P1 ==== P2 == P3 ==== Pk === Pn$$

where '=' represents a time quantum

Useful terms(1/2)

- **Burst time(BT)**, Burst time is the total time taken by the process for its execution on the CPU
- **Arrival time(AT)**, is the time when a process enters into the ready state and is ready for its execution.
- **Exit time(ET)**, is the time when a process completes its execution and exit from the system.
- **Response time(RT)**, is the time spent when the process is in the ready state and gets the CPU for the first time.
    - RT = Time at which the process gets the CPU for the first time - AT

Useful terms(2/2)

- **Waiting time(WT)**, is the total time spent by the process in the ready state waiting for CPU

$$WT = TAT - BT$$

- **Turnaround time(TAT)**, is the total amount of time spent by the process from coming in the ready state for the first time to its completion

$$TAT = BT + WT$$

$$TAT = ET - AT$$

- **Throughput**, is a way to find the efficiency of a CPU. It can be defined as the number of processes executed by the CPU in a given amount of time

End of lab 6