Recitation 4: CPU Scheduling
Scheduling Basic

- **CPU Scheduling** is the act of giving the CPU to a process while other processes are waiting.
  - Scheduling states
    - Process states: New, ready, running, waiting, and terminated
    - CPU states: Idle, and busy
    - Scheduling Queues: Waiting, ready, and running

- **Context switch**: Switching the CPU from one process to another process

- **Mode switch**: Transitioning between user mode and kernel mode

- **Preemption**: The process of forcibly interrupting a running process, performing a context switch and giving the CPU to another process
Types of Scheduling algorithms

- **Non-preemptive scheduling**: If the CPU has been given to a process, the CPU cannot be taken away from that process.
  - Fair because all processes must wait
  - Predictable response time because there is not any high priority process
  - A new process will be scheduled if:
    - A process switches from *running* queue to *waiting* queue
    - A process *terminated*

- **Preemptive scheduling**: If the CPU has been given to a process, the CPU can taken away.
  - The logic is that the processes that are logically runnable will be suspended for a while to run other processes.
  - Fair because processes treated the same
  - Predictable response time because a high priority process cannot displace a low priority one
  - A new process will be scheduled if:
    - A process switches from *running* queue to *ready* queue
    - A process switches from *waiting* queue to *ready* queue
CPU Bound VS I/O Bound Processes

- **CPU bound process**: A process is CPU bound if it spends most of its time using the CPU. A higher CPU clock rate will make a CPU bound process faster.

- **I/O bound process**: A process is I/O bound if it spends most of its time working on a huge file or disk. A faster I/O hardware will make an I/O bound process faster.
Scheduling Criteria

• **CPU Utilization (0 – 100%)**: keep CPU as busy as possible
• **Throughput (#P/T)**: Number of processes that are completed per time unit
• **Turnaround Time**: The interval from the time of submission of the process to the time of completion
• **Waiting Time**: Sum of the periods spend waiting in the ready queue (not in IO queue)
• **Response Time**: The time from the submission of a request until the first response is produced
Scheduling Algorithms

• **CPU scheduling** deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. There are many different CPU-scheduling algorithms:
  • First Come, First Served (FCFS) Scheduling:
  • Shortest Job First (SJF) Scheduling:
  • Shortest Remaining Time First Scheduling:
  • Priority Scheduling:
  • Round Robin (RR) Scheduling:
First Come, First Served (FCFS) Scheduling

- **FCFS** simulates a **FIFO queue** and it is based on queuing
- **FCFS** is a non-preemptive scheduling strategy
- **FCFS** Suffers from **Convey effect** as all the other processes wait for the one big process to get off the CPU.
- **FCFS** has low CPU utilization (↓)
- **FCFS** has high average waiting time (↑)
- **FCFS** consider no priority, therefore there is no starvation if all processes complete eventually
First Come, First Served (FCFS) Scheduling

• Example: Consider a set of processes that arrive at time 0
  • (Process, Burst time): (P1, 24), (P2, 3), (P3, 3)
  • Average waiting time?
  • Throughput?
  • Average turnaround time?

  • Average waiting time? \( (0 + 24 + 27) / 3 = 17 \)
  • Throughput? \( 3 / 30 = 0.1 \)
  • Average turnaround time? \( (24 + 27 + 30) / 3 = 27 \)
Earliest Deadline First (EDF) Scheduling

- **EDF** is a dynamic scheduling algorithm used in real-time operating systems to place processes in a priority queue.
- **EDF** looks for the process with the closest deadline and pick that one to run.
- **EDF** is similar to the algorithm you may use while working on your homework.
Shortest Job First (SJF) Scheduling

• In **SJF**, the process that has the smallest next CPU burst will be scheduled.

• **SJF** Requires advanced knowledge or estimation of the run time of the processes.

• **SJF** is optimal and it cannot be implemented

• **SJF** has low average waiting time (↓)

• **SJF** has the maximum throughput in most scenarios

• Generally, **SJF** is a non-preemptive scheduling algorithm

• **Preemptive SJF** is called **shortest remaining time first**
Shortest Job First (SJF) Scheduling

- Example: Consider a set of processes that arrive at time 0
  - (Process, Burst time): (P1, 6), (P2, 8), (P3, 7), (P4, 3)
  - Average waiting time?
  - Throughput?
  - Average turnaround time?

- Average waiting time? \( \frac{3 + 16 + 9 + 0}{4} = 7 \)
- Throughput? \( \frac{4}{24} = 0.16 \)
- Average turnaround time? \( \frac{9 + 24 + 16 + 3}{4} = 13 \)
Shortest Remaining Time First (SRTF) Scheduling

- **SRTF** is the preemptive version of SJF
- SRTF picks the process with the smallest amount of time remaining until completion is selected to execute.
- In **SRTF**, short processes are handled very quickly
- **SRTF** has the potential of process starvation.
- Waiting time = Start time - Arrive time
Shortest Remaining Time First (SRTF) Scheduling

• Example: Consider a set of processes
  • (Process, Arrival time, Burst time): (P1, 0, 8), (P2, 1, 4), (P3, 2, 9), (P4, 3, 5)
  • Average waiting time?
  • Throughput?
  • Average turnaround time?

• Average waiting time? \( ((10 - 1) + (1 - 1) + (17 - 2) + (5 - 3)) / 4 = 6.5 \)
• Throughput? \( 4 / 26 = 0.15 \)
• Average turnaround time? \( ((17 - 0) + (5 - 1) + (26 - 2) + (10 - 3)) / 4 = 13 \)
Priority Scheduling

- A priority is associated with each process and processes are scheduled based on their priority.
- The CPU is allocated to the highest priority (lowest the better)
- **Starvation** (Waiting for the CPU) of low priority processes
- Waiting time and response time are depended to the priority of the processes.
- Priority scheduling can be implemented in two modes:
  - In Preemptive priority scheduling low priority processes get interrupted by incoming high priority processes
  - Non-preemptive priority scheduling does not give deadlines to running low priority processes.
 Priority Scheduling

• Example: Consider a set of processes that arrive at time 0
  • (Process, Burst time, Priority): (P1, 10, 3), (P2, 1, 1), (P3, 2, 4), (P4, 1, 5), (P5, 5, 2)
  • Average waiting time?
  • Throughput?
  • Average turnaround time?

  • Average waiting time? \( \frac{6 + 0 + 16 + 18 + 1}{5} = 8.2 \)
  • Throughput? \( \frac{4}{19} = 0.21 \)
  • Average turnaround time? \( \frac{16 + 1 + 18 + 19}{4} = 13.5 \)
Round Robin (RR) Scheduling

• **RR** Assigning a fixed time unit (quantum) per process and cycle through processes. (Time quantum + Circular ready queue)

• Basically **RR** is FCFS + Preemption

• **RR** has extensive overhead especially for small quantum

• **RR** has high average waiting time (↑) because waiting time depends on the number of processes

• **RR** has good average response time because response time depends on the length of processes

• Increasing the time quantum, **RR** converges to **FCFS**

• No **starvation** can happen in **RR**
Round Robin (RR) Scheduling

• Example: Consider a set of processes that arrive at time 0, quantum = 4
  • (Process, Burst time): (P1, 24), (P2, 3), (P3, 3)
  • Average waiting time?
  • Throughput?
  • Average turnaround time?
    • Average waiting time? \((6 + 4 + 7) / 3 = 5.66\)
    • Throughput? \(3 / 30 = 0.1\)
    • Average turnaround time? \((30 + 7 + 10) / 3 = 15.6\)