Cairo University
Faculty of Computers & Information
Computer Science Department

Lab 5: CPU Scheduling

Theoretical part:

Definition:

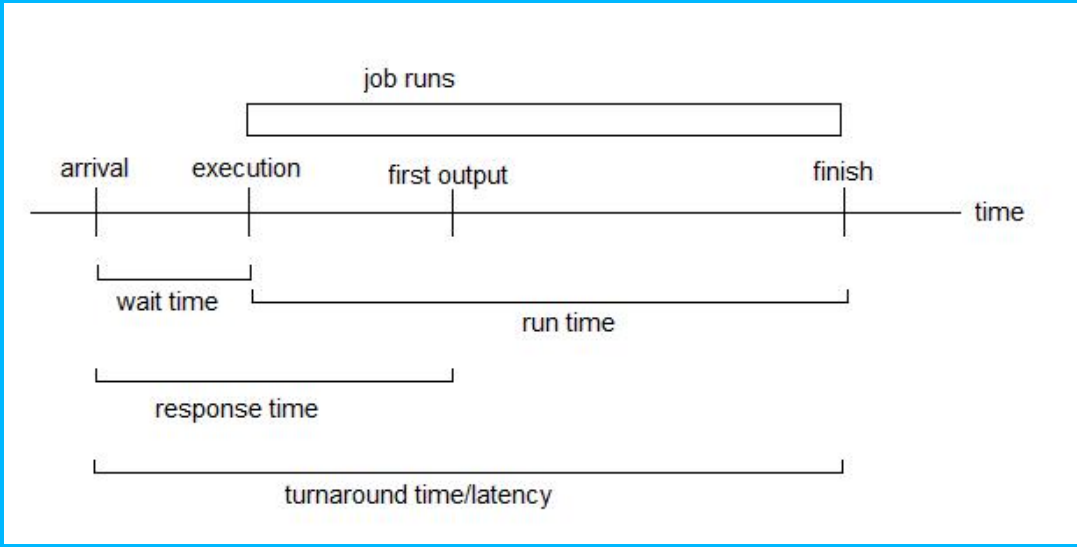
The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process (Determining which processes run when) on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and loaded process shares the CPU using time multiplexing.

Scheduling Criteria

1. **CPU utilization:** it is the percentage of time the CPU is used for a specific time period ones want to keep the CPU as busy as possible (Max.)
2. **Throughput:** # of processes that complete their execution per time unit (Max.)
3. **Waiting time:** How much time processes spend in the ready queue waiting their turn to get on the CPU = start time – arrival time (Min.)
4. **Turnaround time (latency):** total time taken to execute a particular process = waiting time + processing time (Min.)
5. **Response time:** amount of time it takes from when a request was submitted until the first response is produced (Min.)

⇨ In general one wants to optimize the average value of a criteria (Maximize CPU utilization and throughput, and minimize all the others.) However sometimes one wants to do something different, such as to minimize the maximum response time.



Difference between preemptive and non-preemptive scheduling

- Preemptive scheduling: The preemptive scheduling is prioritized. The highest priority process should always be the process that is currently running.
- Non-Preemptive scheduling: When a process enters the state of running, it cannot be preempted (interrupted) until completes finishes its service time.

Give a short description for the following techniques:

1. First Come First-Served (FCFS) Scheduling:
    - Jobs are executed on first come, first serve basis.
    - Could be poor in performance as average wait time is high.
2. Shortest- Job First (SJF) Scheduling:
    - Associate with each process the length of its next CPU burst, Use these lengths to schedule the process with the shortest time
    - Best approach to minimize waiting time.
    - The difficulty is knowing the length of the next CPU request
    - It could be pre-emptive where running process should be the one with shortest running time and it is called Shortest-Remaining-Time-First
3. Priority Scheduling :
    - A priority number (integer) is associated with each process
    - CPU is allocated to the process with the highest priority (smallest integer≡ highest priority)

- Priority can be decided based on memory requirements, time requirements or any other resource requirement.
- Problem Starvation – low priority processes may never execute ➔ Solution Aging – as time progresses increase the priority of the process

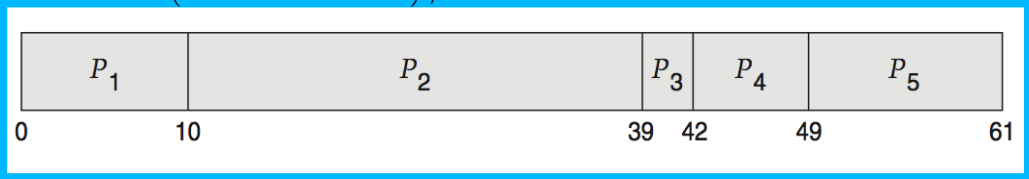
4. Round Robin (RR):
   - Each process is provided a fix time to execute called quantum.
   - Once a process is executed for given time period. Process is pre-empted and added to the end of the ready queue and other process executes for given time period.
   - Context switching is used to save states of pre-empted processes.

Example on FCFS, SJF and RR with q = 10 for each algorithm we want to calculate average waiting time:

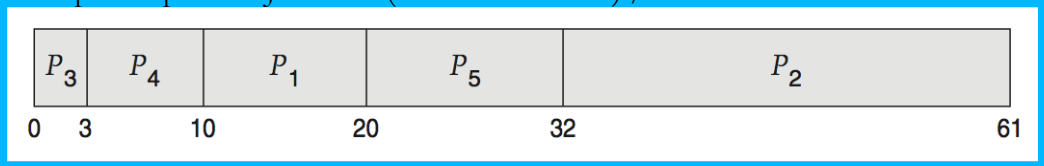
- Consider 5 processes arriving at time 0 with burst time in ms:

Process	Burst Time
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

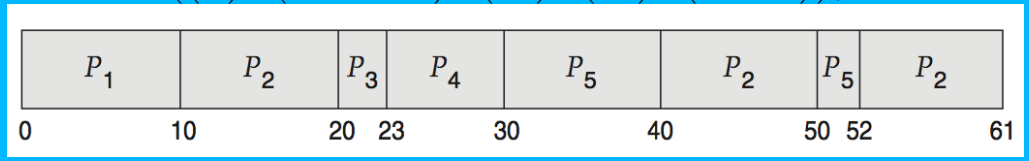
- FCS is 28ms (10 + 39 + 42 + 49) / 5:



- Non-pre-emptive SFJ is 13ms (3 + 10 + 20 + 32) / 5:



- RR is 23ms: ((0) + (10+ 20 + 2) + (20) + (23) + (30 + 10)) /5



Programming part (45 min):

Program that schedule set of processes using the non-pre-emptive SJF technique and calculate waiting time

For each process there will be:

- Name
- Arrival time
- Burst time
- Waiting time

For simplicity we will give the processes the same arrival time equals to 0

`Note: write this program in a file named "Scheduler.java"`

```
import java.util.ArrayList;

//The class of the process
class Process {
	private int arrivalTime;
	private int burstTime;
	private int waitingTime;
	private String name;

	public Process(int arriveTime, int burstTime, String name) {
		this.arrivalTime = arriveTime;
		this.burstTime = burstTime;
		this.name = name;
		waitingTime = 0;
	}

	// Get the arrive time value
	public int getArrivalTime() {
		return arrivalTime;
	}

	// Set the arrive time value
	public void setArrivalTime(int arrivalTime) {
		this.arrivalTime = arrivalTime;
	}

	// Get the burst time value
	public int getBurstTime() {
		return burstTime;
	}

	public int getWaitingTime() {
		return waitingTime;
	}

	// Set the burst time value
	public void setBurstTime(int burstTime) {
		this.burstTime = burstTime;
	}

	// Get the process name
	public String getName() {
		return name;
```

```
	}

	// Set the process name
	public void setName(String name) {
		this.name = name;
	}

	public void setWaitingTime(int waitingTime) {
		this.waitingTime = waitingTime;
	}
}

public class Scheduler {
	int numberOfProcesses;
	int waitingTime = 0;
	// ArrayList to hold the processes
	private ArrayList<Process> processes = new ArrayList<Process>();

	void addProcess(Process p) {
		processes.add(p);
		numberOfProcesses++;
	}

	void schedule_SJF_Non_Prem() {
		// continueScheduling indicates if the scheduling is finished
or not
		boolean continueScheduling = true;
		int waitingTime = 0;
		while (continueScheduling) {
			// set the min burst time to the fist process burst time
			int min = (processes.get(0)).getBurstTime();
			// The location of the min process in the Array list
			int minIndex = 0;
			// Loop to get the location of the process of the min
burst time
			for (int i = 0; i < numberOfProcesses; i++) {
				Process tempProcess = processes.get(i);
				if (tempProcess.getBurstTime() < min) {
					min = tempProcess.getBurstTime();
					minIndex = i;
					waitingTime += min;
				}
			}
			// Now i have the min process
			Process minProcess = processes.get(minIndex);
			System.out.println("The processeor schedule process : "
					+ minProcess.getName() + " that has the
burst time : "
					+ minProcess.getBurstTime() + " waiting
Time: "
					+ minProcess.getWaitingTime());
			waitingTime += minProcess.getBurstTime();
			// Remove the process from the array list as the
processor finished
			// it
			processes.remove(minIndex);
			// decrease the number of processes
			numberOfProcesses--;
			setProcessesWaitingTime(waitingTime);
			// If there are no processes in the array list* that's
mean the
			// //processor completes all the processes to be
scheduled
			if (processes.size() <= 0) {
				continueScheduling = false;
			}
		}
```

```
        }

        private void setProcessesWaitingTime(int waitingTime) {
                for (int i = 0; i < numberOfProcesses; i++) {
                        Process tempProcess = (Process) processes.get(i);
                        tempProcess.setWaitingTime(waitingTime);
                }
        }

        public static void main(String[] args) {
                // Create processes
                Process p1 = new Process(0, 2, "P1");
                Process p2 = new Process(0, 12, "P2");
                Process p3 = new Process(0, 1, "P3");
                Process p4 = new Process(0, 6, "P4");
                Process p5 = new Process(0, 3, "P5");
                Scheduler scheduler = new Scheduler();
                // Add processes to the scheduler
                scheduler.addProcess(p1);
                scheduler.addProcess(p2);
                scheduler.addProcess(p3);
                scheduler.addProcess(p4);
                scheduler.addProcess(p5);
                scheduler.schedule_SJF_Non_Prem();
        }
}
```

Extension:

- .Consider cases where processes the arrival time is difference for example:
    - Adding a variable representing current time and only consider processes with arrival time <= current time for processing
- Calculate average turnaround time and waiting time
- Discuss pre-emptive implementation Shortest-Remaining-Time-First