**Slide 6 Exercises**

1). Consider the following set of processes with their *arrival time*, *CPU burst time* and *priority* details:

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival Time (ms)** | **CPU Burst (ms)** | **Priority** |
| *P*1 | 0 | 10 | 3 |
| *P*2 | 1 | 6 | 4 |
| *P*3 | 2 | 4 | 1 |
| *P*4 | 3 | 8 | 2 |

Draw the *Gantt chart* and show the *average waiting time* of the execution of these processes using the following scheduling algorithms:

 1.1) First-Come First-Served (FCFS)

 1.2) Shortest-Job-First (SJF)

 1.3) Priority

 1.4) Round-Robin (RR) (time quantum = 4).

2). Consider the following set of processes with their *arrival time*, *CPU burst time* and *priority* details:

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival Time (ms)** | **CPU Burst (ms)** | **Priority** |
| *P*1 | 0 | 8 | 4 |
| *P*2 | 1 | 4 | 1 |
| *P*3 | 2 | 6 | 3 |
| *P*4 | 3 | 12 | 2 |

Draw the *Gantt chart*, and show the *average waiting time* of the processes based on the following scheduling algorithms:

 2.1) First-Come First-Served (FCFS)

 2.2) Shortest-Job-First (SJF)

 2.3) Priority

 2.4) Round-Robin (time quantum = 4).

3). Consider the following set of processes with their *arrival time*, *CPU burst time* and *priority* details:

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival Time (ms)** | **CPU Burst (ms)** | **Priority** |
| *P*1 | 0 | 14 | 3 |
| *P*2 | 1 | 8 | 1 |
| *P*3 | 2 | 6 | 4 |
| *P*4 | 3 | 4 | 2 |

Draw the *Gantt chart*, and show the *average waiting time* of the processes based on the following scheduling algorithms:

 3.1) First-Come First-Served (FCFS)

 3.2) Shortest-Job-First (SJF)

 3.3) Priority

 3.4) Round-Robin (time quantum = 4).

4). A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given n processes to be scheduled on one processor, how many different schedules are possible?

5). Explain the difference between preemptive and nonpreemptive scheduling.

6). Assume that an operating system maps user-level threads to the kernel using the many-to-many model and that the mapping is done through the use of LWPs. Furthermore, the system allows program developers to create real-time threads. Is it necessary to bind a real-time thread to an LWP?

7). Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs?