|  |  |
| --- | --- |
| What is CPU utilization ? |  |
| Keep CPU busy as possible |  |

|  |  |
| --- | --- |
| What is CPU-I/O burst cycle ? |  |
| Process execution consists of a cycle of CPU execution and I/O wait  That means the CPU either work or wait for I/O |  |

|  |  |
| --- | --- |
| What is CPU scheduler (short-term scheduler) ? |  |
| Selects process from the ready queue to be execute |  |

|  |  |
| --- | --- |
| When the CU scheduler will take place (work)? |  |
| 1. Switches from running to waiting state (I/O interrupt) **nonpreemptive**  2. Switches from running to ready state (timer interrupt) **preemptive**  3. Switches from waiting to ready(higher priority process arrive to ready queue) **preemptive**  4. Terminates **nonpreemptive** |  |

|  |  |
| --- | --- |
| What is the preemptive? |  |
| * **nonpreemptive** : once CPU given to the process it cannot be preempted until completes 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) |  |

|  |  |
| --- | --- |
| What is dispatcher ? |  |
| Dispatcher module gives control of the CPU to the process selected by the short-term scheduler |  |

|  |  |
| --- | --- |
| What is the task of dispatcher ? |  |
| * switching context * switching to user mode * jumping to the proper location in the user program to restart that program |  |

|  |  |
| --- | --- |
| What is the dispatcher latency ? |  |
| time it takes for the dispatcher to stop one process and start another running |  |

|  |  |
| --- | --- |
| What is the scheduling criteria ? |  |
| * **CPU utilization** : keep the CPU as busy as possible * **Throughput** : # of processes that complete their execution per time unit * **Turnaround time** : amount of time to execute a particular process * **Waiting time** : amount of time a process has been waiting in the ready queue * **Response time** : amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment) |  |

|  |  |
| --- | --- |
| What we must keep for scheduling criteria ? |  |
| * **Max** CPU utilization * **Max** throughput * **Min** turnaround time * **Min** waiting time * **Min** response time |  |

|  |  |
| --- | --- |
| What is the types of scheduler algorithms ? |  |
| 1. First come, first served (FCFS) 2. Shortest Job First (SJF) 3. Priority 4. Round robin (RR) 5. Multilevel queue |  |

|  |  |
| --- | --- |
| Explain FCFS scheduling ? |  |
| * **Advantages** : very simple in implementation * **disadvantages** : when process with high burst time come first the other processes will wait for long time to start execute. |  |

|  |  |
| --- | --- |
| explain SJR scheduling ? |  |
| Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time  **Advantages :** SJF is optimal . gives minimum average waiting time for a given set of processes  **Disadvantages** : The difficulty is knowing the length of the next CPU request |  |

|  |  |
| --- | --- |
| explain the previous preemptive example ? |  |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **17** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |  | | **0** | 1 | 2 | 3 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 6 | 7 | **P1** | |  |  |  |  |  |  |  |  |  | **0** | 1 | 2 | 2 | 3 | 4 | --- | --- | **P2** | |  |  |  |  |  |  |  |  |  |  |  | **0** | 1 | --- | --- | --- | --- | **P3** | |  |  |  |  |  | **0** | 1 | 2 | 3 | 4 | 4 | 4 | --- | --- | **---** | **---** | **---** | **P4** | |  |

|  |  |
| --- | --- |
| How to Determining Length of Next CPU Burst ? |  |
| 1.PNG   * If we assume that : **t0 = 6** and **T0 = 10**   2.PNG |  |

|  |  |
| --- | --- |
| Explain the priority scheduling ? |  |
| * A priority number (integer) is associated with each process * The CPU is allocated to the process with the highest priority (smallest) * 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 (over head) * **Disadvantage** : not suitable to time sharing |  |

|  |  |
| --- | --- |
| Explain round robin scheduling ? |  |
| * Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue. * If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units. * **Advantage** : not suitable to time sharing * **Disadvantage** : higher average turnaround than SJF put better response * **Performance**   + *If q* large ⇒ FIFO   + *If q* small ⇒ *overhead*   3.PNG |  |

|  |  |
| --- | --- |
| Are there a relation between time quantum and the average of turnaround ? |  |
| NO |  |

|  |  |
| --- | --- |
| Explain multilevel scheduling ? |  |
| * Ready queue is partitioned into separate queues: foreground (interactive) background (batch) * Each queue has its own scheduling algorithm   + foreground – RR   + background – FCFS |  |

|  |  |
| --- | --- |
| What is types of scheduling between queues in multilevel scheduling ? |  |
| * **Fixed priority scheduling**: (i.e., serve all from foreground then from background). Possibility of starvation. * **Time slice** : each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR and 20% to background in FCFS |  |

|  |  |
| --- | --- |
| Explain multilevel feedback queue ? |  |
| A process can move between the various queues |  |

|  |  |
| --- | --- |
| What is the parameters that Multilevel-feedback-queue scheduler defined by them ? |  |
| * number of queues * scheduling algorithms for each queue * method used to determine when to upgrade a process * method used to determine when to demote a process * method used to determine which queue a process will enter when that process needs service |  |

|  |  |
| --- | --- |
| Give example of multilevel feedback queue ? |  |
| * Three queues:   + *Q*0 – RR with time quantum 8 milliseconds   + *Q*1 – RR time quantum 16 milliseconds   + *Q*2 – FCFS * Scheduling   + A new job enters queue *Q0* which is servedRR. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue *Q*1.   + At *Q*1 job is again served RR and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue *Q*2. |  |