

**SCHOOL OF COMPUTING AND IT**

**IV Semester; First- Sessional Examination, Feb-2016**

**Branch: B.Tech. (CSE / IT)**

**(Solution)**

**Subject Code: CS1401**

**Subject Name : Operating Systems and Linux**

**Max. Marks: 20 Duration: 1 hour**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Instructions: 1. All questions are compulsory

2. Missing data if any can be suitably assumed.

3. Numbers in [ ] indicate marks.

|  |  |  |
| --- | --- | --- |
| **1.** | In the following program (written in ‘C’), specify the line numbers where system call will be occured.   1. #include <stdio.h> 2. #include<stdlib.h> 3. void main() 4. { 5. int \*p; 6. int n; 7. printf(“How many numbers\t”); 8. scanf(“%d”,&n); 9. p=(int \*) malloc (n\* sizeof (int)); 10. for (i=0; i<n; i++) 11. scanf(“%d”, p+i); 12. for(i=0; i <n; i++) 13. printf(“\n%d”,\*(p+i)); 14. free(p); 15. }   **System Call will occur at line Nos. 7,8,9,11,13,14 || 7,8,11,13**  2. In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state: Now consider the following statements:  Specify whether the statement is True/False . Justify your answer with specific reason.   1. If a process makes a transition D, it would result in another process making transition A immediately. 2. A process P2 in blocked state can make transition E while another process P1 is in running state. 3. The OS uses preemptive scheduling. 4. The OS uses non-preemptive scheduling.   **Solution**   1. False   **Reason**: If a process makes a transition D, it would result in another process making transition B, not A.   1. True.   **Reason:** A process can move to ready state when I/O completes irrespective of other process being in running state or not.   1. True   **Reason :**  Because there is a transition from running to ready state.   1. False   **Reason** :  as the OS uses preemptive scheduling. | **[2]**  **[4x1=4]** |
| **3.**  **4.** | The process arrives in a system at a rate of 20 processes per minutes and the average service time for each job is 3 seconds. What will be system load?  **20\*3= 60 seconds or 1 minute / load =60/60=1 or 100% loaded**  Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:  Process Burst Time Priority Arrival Time  Job1 9 2 0  Job2 4 1 1  Job3 2 3 3  Job4 6 4 5  The processes are assumed to have arrived in the order Job1, Job2, Job3 and Job4.   1. Draw four Gantt charts illustrating the execution of these processes using preemptive-SJF, and RR (quantum = 2) scheduling.  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 0-J1-1 | 1. J2 -5 | 5- J3 -7 | 7- J4 -13 | 13- J1-21 |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0-J1-2 | J2-4 | J1-6 | J3-8 | J2-10 | J4-12 | J1-14 | J4-16 | J1-18 | J4-20 | J1-21 |  1. What is the average turnaround time of processes for preemptive-SJFs and RR scheduling algorithms?   TAT-P-SJF:- 9.25 AND TAT-RR-12.5   1. What is the average waiting time of processes for preemptive-SJFs and RR scheduling algorithms?   P-SJF:- 4 RR-7.25  d. Draw an Gant chart for FCFS | **[2]**  **[2]**  **[2]**  **[2]**  **[1]** |
| **5** | Explain   1. How interrupts support multiprogramming?   The earliest operating systems were used to control single-user computer systems. In those days, the operating system would read in one job, find the data and devices the job needed, let the job run to completion, and then read in another job. In contrast, the computer systems that OS manages are capable of multiprogramming, or executing many programs concurrently. With multiprogramming, when a job cannot use the processor, the system can suspend, or interrupt, the job, freeing the processor to work on another job. OS makes multiprogramming possible by capturing and saving all the relevant information about the interrupted program before allowing another program to execute. When the interrupted program is ready to begin executing again, it can resume execution just where it left off. Multiprogramming allows OS to run thousands of programs simultaneously for users who might be working on different projects at different physical locations around the world   1. How round robin scheduling is equivalent to FCFS and when?   If the time slice length is too large than the process burst time.   1. What is the difference between concurrency and parallelism?   Concurrency is when two or more tasks can start, run, and complete in overlapping time periods. It doesn't necessarily mean they'll ever both be running at the same instant. Eg. multitasking on a single-core machine.  Parallelism is when tasks literally run at the same time, eg. on a multicore processor.   1. Why *one to many* multithreading model is not preferred?   This model is not possible, this model says that one user thread will schedule on multiple kernel threads that is not possible.   1. Why *many to one* multithreading model cannot make use of multiprocessors in the system?   Because only one user thread can be scheduled over kernel thread which execute on processor. | **[5x1=5** |