**ABES Engineering College, Ghaziabad**

# B. Tech (Second Year) Even Semester Sessional Test-1 Solution

**Session: 2022-2023**

**Course Code: KCS401**

**Course Name: Operating System**

**Maximum Marks: 75**

|  |  |  |  |
| --- | --- | --- | --- |
| **Q. No.** | **Question** | **Marks** | **CO** |
| **Section-A Total Marks : 5\*2=10** | | | |
| **1** | **Attempt ALL Parts** |  | |
| **a)** | Define Operating system and mention its major functions.  Sol:  **Operating System** is an interface between the user and the hardware and enables the interaction of a computer’s **hardware and software.**  **(1 MARKS)**  Major Functions of operating system are:   1. file management, 2. memory management, 3. storage management, 4. process management, 5. handling input and output, 6. and controlling peripheral devices such as disk drives and printers.   **(1 MARKS)** | **1+1** | **CO1** |
| **b)** | Explain the concept of Re-entrant Kernel in brief.  Sol:  A re-entrant kernel is the one which allows multiple processes to be executing in the kernel mode at any given point of time and that too without causing any consistency problems(burden of doing out weight) among the kernel data structures.  •Well, we know that in a single processor system only one process can execute at any given instant but there could be other processes blocked in kernel mode waiting to be executed.  •For example, in a re-entrant kernel a process waiting on a ‘read()’ call may decide to release CPU to a process which is waiting for execution in kernel mode.  **(2 MARKS)** | **2** | **CO1** |
| **c)** | What are the performance criteria in CPU scheduling.  Sol:  The various performance criteria in CPU scheduling are:   1. CPU utilization. 2. Throughput 3. Waiting Time 4. Turn Around Time 5. Response Time 6. Priority 7. Balanced Utilization 8. Fairness   **(2 MARKS)** | **2** | **CO2** |
| **d)** | Describe the typical elements of Process Control block.  Sol:  Typical elements of PCB are:   Process State This specifies the process state i.e. new, ready, running, waiting or terminated. Process Number This shows the number of the particular process. Program Counter This contains the address of the next instruction that needs to be executed in the process. Registers This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc. List of Open Files These are the different files that are associated with the process  **(1 MARKS Diagram + 1 MARKS for explanation)** | **2** | **CO2** |
| **e)** | Explain the key factors which are necessary for principle of concurrency.  Sol:  The key factors which are necessary for principle of concurrency are:   1. Interleaving 2. Synchronization 3. Mutual Exclusion 4. Deadlock Avoidance 5. Process or thread coordination 6. Resource Allocation   **(2 MARKS )** | **2** | **CO3** |
| **Section-B Total Marks : 3\*5 = 15** | | | |
| **2** | **Attempt ANY ONE part from the following** |  | |
| **a)** | Explain various states of a process with the help of a diagram.  Sol: 1.New A program which is going to be picked up by the OS into the main memory is called a new process. 2. Ready Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned. The OS picks the new processes from the secondary memory and put all of them in the main memory.  The processes which are ready for the execution and reside in the main memory are called ready state processes. There can be many processes present in the ready state. 3. Running One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm. Hence, if we have only one CPU in our system, the number of running processes for a particular time will always be one. If we have n processors in the system then we can have n processes running simultaneously. 4. Block or wait From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process.  When a process waits for a certain resource to be assigned or for the input from the user then the OS move this process to the block or wait state and assigns the CPU to the other processes. 5. Completion or termination When a process finishes its execution, it comes in the termination state. All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system. 6. Suspend ready A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state.  If the main memory is full and a higher priority process comes for the execution, then the OS must make the room for the process in the main memory by throwing the lower priority process out into the secondary memory. The suspend ready processes remain in the secondary memory until the main memory gets available. 7. Suspend wait Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory. Since it is already waiting for some resource to get available hence it is better if it waits in the secondary memory and make room for the higher priority process. These processes complete their execution once the main memory gets available and their wait is finished.  **(3 MARKS )**  Q) Explain, how many processes can be in a ready state considering uniprocessor system environment?  Ans) In a Uniprocesssor system considering the size of RAM infinite , if there a N processes which have arrived and are ready to run , 1 process will be sent to Running state so in total there will be N-1 processes in ready state  **(2 MARKS )** | **3+2** | **CO1** |
| **b)** | Explain the fork () system call with the help of suitable example.  Sol:  A new process known as a "child process" is created with the fork system call which runs concurrently with the process called the parent process. fork system call in OS returns an integer value and requires no arguments. After the creation of a new child process, both processes then execute the next command following the fork system call. Therefore, we must separate the parent from the child by checking the returned value of the fork ():   * **Negative:** A child process could not be successfully created if the fork () returns a **negative** value. * **Zero:** A new child process is successfully created if the fork () returns a **zero**. * **Positive:** The positive value is the process ID of a child's process to the parent. The process ID is the type of pidt that is defined in sys/types.h.   However, the parent and child processes are stored in different memory locations. These memory spaces contain the same information, therefore any operations carried out by one process do not affect the other.  **(3 MARKS)**  Example:  Let us consider the following code        **(2 MARKS)** | **5** | **CO1** |
| **3** | **Attempt ANY ONE part from the following** |  | |
| **a)** | Consider the following set of processes, assumed to have arrived at time 0. Consider the CPU scheduling algorithms Shortest Job First (SJF) and Round Robin (RR). For RR, assume that the processes are scheduled in the order P1, P2, P3, P4.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Processes | P1 | P2 | P3 | P4 | | Burst Time | 8 | 7 | 2 | 4 |   If the time quantum for RR is 4 ms, then calculate the absolute value of the difference between the average turnaround times (in ms) of SJF and RR (round off to 2 decimal places).  **(GATE CS-2020)**  **Solution :**  **SJF**      Turn Around Time (TAT) = (21 – 0) + (13 – 0) + (2 – 0) + (6 – 0)  Average TAT = 42/4 = 10.5  **(2 MARKS)**  **Round Robin**    **(2 MARKS)**  Turn Around Time (TAT) = (18 – 0) + (21 – 0) + (10 – 0) + (14 – 0) = 18 + 21 + 10 + 14 Average TAT= 63/4 = 15.75 Hence, ⏐SJF (TAT) – RR(TAT)⏐ = ⏐10.5 – 15.75⏐ = 5.25  **(1 MARKS)** | **5** | **CO2** |
| **b)** | Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds) and priority (0 is highest priority) shown below. None of the processes have I/O burst time.   |  |  |  |  | | --- | --- | --- | --- | | Process | Arrival Time | Burst Time | Priority | | P1 | 0 | 11 | 2 | | P2 | 5 | 28 | 0 | | P3 | 12 | 2 | 3 | | P4 | 2 | 10 | 1 | | P5 | 9 | 16 | 4 |   Calculate the average waiting time (in milliseconds) of all the processes using pre-emptive priority scheduling algorithm.  **(GATE CS-2017)**  **Sol:**    **(5 MARKS)** | **5** | **CO2** |
| **4** | **Attempt ANY ONE part from the following** |  | |
| **a)** | Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared Boolean variables S1 and S2 are randomly assigned  Method Used by P1  Method Used by P2  while (S1 != S2) ; while (S1 == S2) ;  Critica1 Section Critica1 Section  S1 = S2; S2 = not (S1);  Elaborate the following criteria's with respect to the above-mentioned problem   1. Mutual Exclusion   Sol ) Mutual Exclusion ensures that if a process is inside crictical section , no other process could enter into it . So if process P1 is running first , it checks the value of S1 and S2 . If both values are different only in this case it could enter into its critical section . Now let us suppose that values were different and process P1 enters critical section and context switch happened then also P2 process couldnt enter critical section as both S1 and S2 are still different and process P2 could only enter if S1 and S2 are same .vice versa is also true .  **(2.5 MARKS)**   1. No Progress   here Progress Requirement is not satisfied. Suppose when s1=1 and s2=0 and process p1 is not interested to enter into critical section but p2 want to enter critical section. P2 is not able to enter critical section in this as only when p1 finishes execution, then only p2 can enter (then only s1 = s2 condition be satisfied). Progress will not be satisfied when any process which is not interested to enter into the critical section will not allow other interested process to enter into the critical section.  **(2.5 MARKS)**  **(GATE CS-2010)** | **2.5+2.5** | **CO3** |
| **b)** | What is race condition?  Sol) Each of the processes has some sharable resources and some non-shareable resources. The sharable resources can be shared among the cooperating processes. The non-cooperating processes don’t need to share the resources. When we [synchronize the processes](https://t4tutorials.com/process-synchronization-solved-mcqs-questions-answers/) and the synchronization is not proper then the race condition occurs. we can define the race condition as follows;  A race condition is a condition when there are many [processes and every process shares the data](https://t4tutorials.com/monitors-examples-process-synchronization-role-procedures-shared-data-operating-systems-os/) with each other and accessing the data concurrently, and the output of execution depends on a particular sequence in which they share the data and access.  **(1 MARKS)**  Q )Consider the following two processes P1 and P2. Explain in detail whether the following code raise to race condition if shared variable is initialized as 5.   |  |  | | --- | --- | | P1 | P2 | | x=shared;  x++;  Sleep(1)  Shared=x; | y=shared;  y--;  Sleep(1)  Shared=y; |     In the above code two parallel processes are given and a global variable named as shared is given whose initial value is 5. P1 is incrementing this value by 1 and P2 is decrementing its value by 1 which means that finally shared shall contain 5. But actually,,both processes are using sleep() function which is causing context switching.  Consider the following case :  P2 function starts first . It initializes y = 5 and decrements it by 1 . So y is now containing 4 and then it context switches to P1 due to sleep(). In Process P1 , x is initialized to 5 and incremented by 1 , ie, x now contains 6 and again there is a context switch , which sends control to P2 process,and now shared has been assigned value of y ie , 4 and P2 process terminates . Control goes back to process P1 where shared has been assigned the value of x ie, 6 . Which has lead to inconsistent results.  In the similar manner if process P1 starts first it will further leads to inconsistent result .    **(4 MARKS)**  **(GATE CS-2009)** | **1+4** | **CO3** |
| **Section-C Total Marks : 5\*10 = 50** | | | |
| **5** | **Attempt ANY ONE part from the following** |  | |
| **a)** | Explain in detail about the multi-threaded operating system and also discuss its types. Sol) Multithreading Model: Multithreading allows the application to divide its task into individual threads. In multi-threads, the same process or task can be done by the number of threads, or we can say that there is more than one thread to perform the task in multithreading. With the use of multithreading, multitasking can be achieved.    The main drawback of single threading systems is that only one task can be performed at a time, so to overcome the drawback of this single threading, there is multithreading that allows multiple tasks to be performed.    **(4 MARKS)**  **There exists three established multithreading models classifying these relationships are:**   * Many to one multithreading model * One to one multithreading model * Many to Many multithreading models  Many to one multithreading model: The many to one model maps many user levels threads to one kernel thread. This type of relationship facilitates an effective context-switching environment, easily implemented even on the simple kernel with no thread support.  The disadvantage of this model is that since there is only one kernel-level thread schedule at any given time, this model cannot take advantage of the hardware acceleration offered by multithreaded processes or multi-processor systems. In this, all the thread management is done in the userspace. If blocking comes, this model blocks the whole system.  Multithreading Models in Operating system  In the above figure, the many to one model associates all user-level threads to single kernel-level threads.  **(2 MARKS)** One to one multithreading model The one-to-one model maps a single user-level thread to a single kernel-level thread. This type of relationship facilitates the running of multiple threads in parallel. However, this benefit comes with its drawback. The generation of every new user thread must include creating a corresponding kernel thread causing an overhead, which can hinder the performance of the parent process. Windows series and Linux operating systems try to tackle this problem by limiting the growth of the thread count.  Multithreading Models in Operating system  In the above figure, one model associates that one user-level thread to a single kernel-level thread.  **(2 MARKS)** Many to Many Model multithreading model In this type of model, there are several user-level threads and several kernel-level threads. The number of kernel threads created depends upon a particular application. The developer can create as many threads at both levels but may not be the same. The many to many model is a compromise between the other two models. In this model, if any thread makes a blocking system call, the kernel can schedule another thread for execution. Also, with the introduction of multiple threads, complexity is not present as in the previous models. Though this model allows the creation of multiple kernel threads, true concurrency cannot be achieved by this model. This is because the kernel can schedule only one process at a time.  Multithreading Models in Operating system  Many to many versions of the multithreading model associate several user-level threads to the same or much less variety of kernel-level threads  **(2 MARKS)** | **5 +5** | **CO1** |
| **b)** | Explain the following concepts:   1. Time sharing operating system   In a time-sharing operating system, each task uses the CPU in such a way that the response time of the CPU is minimized. Each task takes the same amount of time to execute.  The time-sharing operating system is different from a multiprogramming operating system. In a multiprogramming operating system, the main objective is to maximize the use of the CPU. But in Time-sharing OS, the main aim is to minimize the response time of the CPU. Working of Time-Sharing Operating System The Time-Sharing Operating System uses CPU scheduling and multiprogramming.  **CPU scheduling and multiprogramming are used in Timesharing Operating System.**   * When the user performs more than one task, each process's CPU time is divided. * There is a time quantum fixed for each process to execute at a time. This time quantum is minimal and in 10-100 milliseconds. Time quantum is also known as time slot or time slice. * For example, if there are three processes, P1, P2, and P3, running on the system. Suppose the time Quantum is fixed to 4 nanoseconds (ns). Let's see further how these processes will be executed. * Process P1 will execute first for 4ns and as soon as it gets over, process P2 starts executing for 4ns, and when P2 is executed for 4ns then process P3 executes for 4ns. This process continues till all the processes get completed. * In this way, if the process runs for only the fixed time quantum, the switching between the process is very fast. So, the user thinks that all the processes are running simultaneously. In this way response time of the CPU is minimized.   **(3 MARKS)**   1. Multi Programming operating system   A multiprogramming operating system may run many programs on a single processor computer. If one program must wait for an input/output transfer in a multiprogramming operating system, the other programs are ready to use the CPU. As a result, various jobs may share CPU time. However, the execution of their jobs is not defined to be at the same time period.    When a program is being performed, it is known as a "Task", "Process", and "Job". Concurrent program executions improve system resource consumption and throughput as compared to serial and batch processing systems.    The primary goal of multiprogramming is to manage the entire system's resources. The key components of a multiprogramming system are the file system, command processor, transient area, and I/O control system. As a result, multiprogramming operating systems are designed to store different programs based on sub-segmenting parts of the transient area. The resource management routines are linked with the operating system core functions.    Types of the Multiprogramming Operating System  There are mainly two types of multiprogramming operating systems. These are as follows:    Multitasking Operating System  Multiuser Operating System  **(3 MARKS)**   1. System call   A system call is a method for a computer program to request a service from the kernel of the [operating system](https://www.javatpoint.com/os-tutorial) on which it is running. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel.  The **Application Program Interface (API)** connects the operating system's functions to user programs. It acts as a link between the operating system and a process, allowing user-level programs to request operating system services. The kernel system can only be accessed using system calls. System calls are required for any programs that use resources.  Types of System call   1. **Process Control** 2. **File Management** 3. **Device Management** 4. **Information Maintenance** 5. **Communication**   **(4 MARKS)** | **3+3+4** | **CO1** |
| **6** | **Attempt ANY ONE part from the following** |  | |
| **a)** | Explain the difference between   1. Hard Real time and Soft Real time System     **(3 MARKS)**   1. Process and Threads     **(4 MARKS)**   1. Distributed and Clustered operating system   **Distributed operating system:**  In this operating system the machines are present in a loosely coupled environment, which means that the machines are present in different geographical locations.    These machines connect with other machines via a network.    You can understand it in this way, Blockchain technology uses the same concept, and in that the servers are connected to each other via blockchain protocol or you can say that they follow blockchain protocol.    And all these machines act as a single operating system.    So, to manage this environment the operating system used is known as the Distributed operating system.    **Clustered operating system:**  Clustered operating systems are just the opposite of distributed operating systems, here all the different machines are connected to a local network.    As different machines having their own hardware connected to a local network it acts as a server or you can say as a supercomputer.    The computation power of different machines will be combined, resulting in a total increase in the power of computation for a single system.    So, to manage this environment the operating system used is known as the Clustered operating system.  **(3 MARKS)** | **3+4+3** | **CO1** |
| **b)** | Discuss monolithic kernel, layered and micro kernel structures of operating system.  Sol)  **Monolithic Kernel**  Functionality of the OS is invoked with simple function calls within the kernel, which is one large program.  • Device drivers are loaded into the running kernel and become part of the kernel.  Figure A monolithic kernel, such as Linux and other Unix systems  **(3 MARKS)**  **Layered Approach** This approach breaks up the operating system into different layers.  • This allows implementers to change the inner workings, and increases modularity.  • As long as the external interface of the routines don’t change, developers have more freedom to change the inner workings of the routines.  • With the layered approach, the bottom layer is the hardware, while the highest layer is the user interface.  Figure Layered  **(3 MARKS)**  **Microkernels** This structures the operating system by removing all nonessential portions of the kernel and implementing them as system and user level programs. Generally, they provide minimal process and memory management, and a communications facility. Communication between components of the OS is provided by message passing.  Figure A Microkernel architecture  The Microsoft Windows NT Operating System. The lowest level is a monolithic kernel, but many OS components are at a higher level, but still part of the OS.  **(4 MARKS)** | **3+3+4** | **CO1** |
| **7** | **Attempt ANY ONE part from the following** |  | |
| **a)** | Consider the resource allocation graph in the figure.  Inserting image...  i) Find if a system is in a deadlock state  ii) If not find a safe sequence.  **Sol:**    No, the system is not in a deadlock state as all the processes terminated successfully.  For part i) -------- **(5 MARKS)**  For part ii) -------- **(5 MARKS)** | **5+5** | **CO2** |
| **b)** | A system has 3 processes, each requiring 2 units of resources R.   1. Find the minimum number of units of ‘R’ such that no deadlock will occur. 2. Find the maximum number of units of ‘R’ such that deadlock will occur.   Explain both the cases with detailed explanation.  **Sol:**      For part i) -------- **(3.5 MARKS)**  For part ii) -------- **(3.5 MARKS)**  For Explanation -------- **(3 MARKS)** | **3.5+3.5+3** | **CO2** |
| **8** | **Attempt ANY ONE part from the following** |  | |
| **a)** | Consider a system that contains 5 processes P1, P2, P3, P4, P5 and 3 resource types. A has 10 instances, B has 5 and C has 7 instances.   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Process | Allocated | | | Max Need | | | Current Available | | | |  | A | B | C | A | B | C | A | B | C | | P1 | 0 | 1 | 0 | 7 | 5 | 3 | 3 | 3 | 2 | | P2 | 2 | 0 | 0 | 3 | 2 | 2 |  |  |  | | P3 | 3 | 0 | 2 | 9 | 0 | 2 |  |  |  | | P4 | 2 | 1 | 1 | 4 | 2 | 2 |  |  |  | | P5 | 0 | 0 | 2 | 5 | 3 | 3 |  |  |  |   Answer the following questions.   1. What is the reference of the need matrix 2. Determine if the system is safe or not. If yes, then write its safe sequence. 3. What will happen if the resource request (1,0,2) for process P1, can the system accept this request immediately.   **Sol:**      For part i) -------- **(4 MARKS)**  For part ii) -------- **(3 MARKS)**  For part iii) -------- **(3 MARKS)** | **4+3+3** | **CO2** |
| **b)** | Consider the following snapshot of a system.   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Process | Allocated | | | | Max | | | | |  | A | B | C | D | A | B | C | D | | P0 | 3 | 0 | 1 | 4 | 5 | 1 | 1 | 7 | | P1 | 2 | 2 | 1 | 0 | 3 | 2 | 1 | 1 | | P2 | 3 | 1 | 2 | 1 | 3 | 3 | 2 | 1 | | P3 | 0 | 5 | 1 | 0 | 4 | 6 | 1 | 2 | | P4 | 4 | 2 | 1 | 2 | 6 | 3 | 2 | 5 |   Using the banker’s algorithm, determine whether or not each of the following states is unsafe. If it is safe, illustrate the order in which the processes may be completed. Otherwise, illustrate why the state is unsafe.   1. Available = (0,3,0,1) 2. Available = (1,0,0,2)   Sol:    For part i) -------- **(5 MARKS)**  For part ii) -------- **(5 MARKS)** | **5+5** | **CO2** |
| **9** | **Attempt ANY ONE part from the following** |  | |
| **a)** | What is a critical section problem. Give the conditions that a solution to the critical section problem must satisfy.  Sol:                  For critical section problem explanation --------**(4 MARKS)**  For Conditions (**1. Mutual exclusion -------- (2 MARKS)**  **2. Progress --------------------(2 MARKS)**  **3. Bounded wait-------------(1 MARKS)**  **4. No assumption of speed----(1 MARKS)**  **Total – 10 marks** | **4+6** | **CO3** |
| **b)** | What is producer- consumer problem. How it can illustrate the classical problem of synchronization. Explain  Sol:            For Producer consumer problem explanation --------**(4 MARKS)**  For (CASE 1 (BEST CASE) **-------- (3 MARKS)**  For (CASE 2 (WORST CASE) **-------- (3 MARKS)** | **4+6** | **CO3** |

CO Course Outcomes mapped with respective question

KL Bloom's knowledge Level (K1, K2, K3, K4, K5, K6)

K1- Remember, K2- Understand, K3-Apply, K4- Analyze, K5: Evaluate, K6- Create