|  |
| --- |
| **Question 1.** |
| **Answer.** |
| **Q 2. Design an algorithm for deadlock detection.** |
| **Answer** |
| **Q 3. Compare deterministic modeling and queuing models for evaluation the performance of CPU scheduling algorithms.** |
| **Answer** |
| **Q 4. Enlist the limitations of layered structure operating systems. Give your suggestions to overcome these limitations.** |
| **Answer** |
| **Q 5. Starvation and deadlock are same. Accept or reject with solid reasons** |
| **Answer**  **Q 6. What are the differences between?**   1. **Process scheduling and thread scheduling** 2. **Preemptive and non-preemptive scheduling algorithms?** |
| **Answer** |
| **Q 7. How does Win32 API identify the priority class of a process?** |
| **Answer** |
| **Q 8. During the execution of a process, in how many queues it may be queued and why. Support you answer with a diagram** |
| **Answer** |
| **Q 9. Consider a system running ten I/O-bound task and on CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 millisecond and that all processes are long-running tasks. Describe the CPU utilization for a round-robin scheduler when:**   * **a. The time quantum is 1 millisecond** * **b. The time quantum is 10 milliseconds** |
| **Answer** |
| **Q 10. Most Round-Robin Schedulers Use A Fixed Size Quantum. Give An Argument In Favor Of A Small Quantum. Now Give An Argument In Favor Of A LargeQuantum. Compare And Contrast The Types Of Systems And Jobs To Which The Arguments Apply. Are There Any For Which Both Are Reasonable?** |
| **Answer**  A small quantum reduces the response time for all processes, which is important for interactive processes. However, a long quantum reduces the overhead of process switching, which increases throughput and CPU utilization. A short quantum is useful for a general-purpose computer. A long quantum is useful for a batch system. You might have a system that uses a short quantum when there are jobs that need a quick response time, then lengthens the quantum when heavy computation needs to be done. |
| **Q 11 Consider the following snapshot of a system:**  **Allocation Max Available**  **A B C D** A B C D **A B C D**  P0 0 0 1 2 0 0 1 2 1 5 2 0  P1 1 0 0 0 1 7 5 0  P2 1 3 5 4 2 3 5 6  P3 0 6 3 2 0 6 5 2  P4 0 0 1 4 0 6 5 6  Answer the following questions using the banker’s algorithm:  (a) what is the content of the matrix Need?  (b) Is the system in a safe state?  (c) If a request from process P1 arrives for (0, 4, 2, 0), can the request be granted  immediately? |
| **Answer**  (a) Need = Max - Allocation  Need  A B C D  P0 0 0 0 0  P1 0 7 5 0  P2 1 0 0 2  P3 0 0 2 0  P4 0 6 4 2  (b)   * Need P0 <= Available so execute P0   Available = 1 5 3 2   * Need P2 <= Available so execute P2   Available = 2 8 8 6   * Need P1 <= Available so execute P1   Available = 3 8 8 6   * Need P3 <= Available so execute P3   Available = 3 14 11 8   * Need P4 <= Available so execute P4   Available = 3 14 12 12  The system is in a safe state: P0, P2, P1, P3, P4 is the safe sequence.  (c) After satisfying P1 request (0, 4, 2, 0), the system becomes the following state.  Allocation Max Need Available  A B C D A B C D A B C D A B C D  P0 0 0 1 2 0 0 1 2 0 0 0 0 1 1 0 0  P1 1 4 2 0 1 7 5 0 0 3 3 0  P2 1 3 5 4 2 3 5 6 1 0 0 2  P3 0 6 3 2 0 6 5 2 0 0 2 0  P4 0 0 1 4 0 6 5 6 0 6 4 2  By using the safety algorithm   * Need P0 <= Available so execute P0   Available = 1 1 1 2   * Need P2 <= Available so execute P2   Available = 2 4 6 6   * Need P1 <= Available so execute P1   Available = 3 8 8 6   * Need P3 <= Available so execute P3   Available = 3 14 11 8   * Need P4 <= Available so execute P4   Available = 3 14 12 12  the system is still in a safe state and P0, P2, P1, P3, P4 is a  safe sequence. |
| **Q 12. An algorithm to detect a cycle in a Resource-Allocation graph requires an order of n2 operations (where n is the number of vertices in the graph). Explain and proof it mathematically?** |
| **Answer**  Let’s consider the gaph has **n** processes or nodes.  Maintain a datastructure such as P1 -> P2 -> P3  Deadlock case is P1 -> P2 -> P1  Algorithm of Resource allocation graph is given as  Global queue Q  For each process Pi up to n:  Create Pi entry in Q  For each Pj up to n:  Push Pj to Pi entry  Check if allocating a resource to Pj deadlocks Pi  Clearly there are two for loops in the algorithm so the complexity is O(n2) |
| **Q 13. Write an algorithm for producer consumer problem using monitors** |
| **Answer**   * Paradigm for cooperating processes:   + *producer* process produces information that is consumed by a *consumer* process * Two variations:   + **unbounded-buffer** places no practical limit on the size of the buffer:     - Producer never waits     - Consumer waits if there is no buffer to consume   + **bounded-buffer** assumes that there is a fixed buffer size     - Producer must wait if all buffers are full     - Consumer waits if there is no buffer to consume |
| **Q 14. What is primary difference between deadlock and starvation** |
| **Answer**   |  |  |  | | --- | --- | --- | | **S.No** | **Deadlock** | **Starvation** | | 1 | All processes keep waiting for each other to complete and none get executed | High priority processes keep executing and low priority processes are blocked | | 2 | Resources are blocked by the processes | Resources are continuously utilized by high priority processes | | 3 | Necessary conditions Mutual Exclusion, Hold and Wait, No preemption, Circular Wait | Priorities are assigned to the processes | | 4 | Also known as Circular wait | Also know as lived lock | | 5 | It can be prevented by avoiding the necessary conditions for deadlock | It can be prevented by Aging | |
| **Q 15. What are the necessary conditions for a deadlock? List and describe them briefly** |
| **Answer**  Deadlock can arise if four conditions hold simultaneously.   * **Mutual exclusion:** only one process at a time can use a resource * **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes * **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task * **Circular wait:** there exists a set {*P*0, *P*1, …, *P*n} of waiting processes such that *P*0 is waiting for a resource that is held by *P*1, *P*1 is waiting for a resource that is held by *P*2, …, *Pn*–1 is waiting for a resource that is held by *P*n, and *P*n is waiting for a resource that is held by *P*0. |
| **Q 15. You are given the following**   * **A set of Processes P = {P1, P2, P3}** * **A set of Recources R = R1, R2, R3, R4}**   **Resource instances**   * **One instance of resource type R1** * **Two instances of resource type R2** * **One instance of resource type R3** * **Three instances of resource type R4**   **Process states**   * **Process P1 is holding an instance of resource type R2 and is waiting for an instance of resource type R1** * **Process P2 is holding an instance of R1 and an instance of R2 and is waiting for an instance of R3** * **Process P3 is holding an instance of R3**  1. **Draw resource-allocation graph by using data as given above** 2. **From resource-allocation graph produced in a). conclude whether the system is in deadlock or not in deadlock state; describe reason of your conclusion** 3. **Write Banker’s algorithm along with safety and resource-request algorithms** |
| **Answer**  **a)**    **b)** Even the resource-allocation graph contains cycle but it is not in deadlock state because   * P3 will release R3 * P2 will use R3 and release R1 and R2 * P1 will use R1 and R2   Leaving the system in safe state  **c)**  Banker’s Safety Algorithm   |  | | --- | | Let *n* = number of processes, and *m* = number of resources types.  Available: Vector of length m. If available [j] = k, there are k instances of resource type Rj available Max: n x m matrix. If Max [i,j] = k, then process Pi may request at most k instances of resource type Rj Allocation: n x m matrix. If Allocation[i,j] = k then Pi is currently allocated k instances of Rj Need: n x m matrix. If Need[i,j] = k, then Pi may need k more instances of Rj to complete its task Need [i,j] = Max[i,j] – Allocation [i,j]   1. Let ***Work***and ***Finish*** be vectors of length *m* and *n*, respectively. Initialize:   ***Work* = *Available***  ***Finish* [*i*] = *false* for *i* = 0, 1, …, *n-* 1**   1. Find an ***i***such that both:   (a) ***Finish* [*i*] == *false***  (b) ***Needi* ≤ *Work***  If no such ***i*** exists, go to step 4   1. ***Work* = *Work* + *Allocationi*  *Finish*[*i*] = *true*** go to step 2 2. If ***Finish* [*i*] == *true*** for all ***i***, then the system is in a safe state |   Resource-Request Algorithm   |  | | --- | | ***Requesti*** = request vector for process ***Pi***. If ***Requesti*[*j*] = *k*** then process ***Pi*** wants ***k*** instances of resource type ***Rj***   * 1. If ***Requesti* ≤ *Needi*** go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim   2. If ***Requesti* ≤ *Available***, go to step 3. Otherwise ***Pi*** must wait, since resources are not available   3. Pretend to allocate requested resources to ***Pi*** by modifying the state as follows:   ***Available* = *Available* – *Requesti;***  ***Allocationi*= *Allocationi* + *Requesti*;**  ***Needi* = *Needi* – *Requesti;***   * + - If safe ⇒ the resources are allocated to ***Pi***     - If unsafe ⇒ ***Pi*** must wait, and the old resource-allocation state is restored | |
| **Q 16. Enlist advantages and disadvantages of user level threads over kernel level threads. What types of threads used by Solaris?** |
| **Answer**   * **User threads** - management done by user-level threads library   + Advantages     - are easier and faster to create than kernel-level threads. They can also be more easily managed.     - can be run on any operating system.     - There are no kernel mode privileges required for thread switching in user-level threads   + Disadvantages     - Multithreaded applications in user-level threads cannot use multiprocessing to their advantage     - The entire process is blocked if one user-level thread performs blocking operation.   Solaris Green Threads are used by solaris which are based on many-to-one model |
| **Q 17. Once a system decides to create a new process, what steps the system takes from creation to process termination?** |
| **Answer**  --Creation--  When a process creates a child process   1. It will be allocated resources to accomplish its task. It may gain resources directly from OS or from the subset of the parent’s resources 2. Parent will pass initialization data (input) to the child process 3. There are two possibilites of child execution    1. Parent and child execute concurrently    2. Parent waits until the child completes its task and then terminates it   --Termination—  A parent may terminate a child when   * The child has exceeded allocated resource usage * Task assigned to child is no longer required * Parent is exiting and OS don’t allow a child to continue its execution |
| **Q 18. Which of the following components of program are shared across threads in a multithreaded process?**   * **Register Values** * **Heap Memory** * **Global Variables** * **Stack Memory** |
| **Answer**  The Global values and heap memory are shared across a multithreaded process. Register values and stack memory are private to each thread |
| **Q 19. Briefly explain scheduler activations** |
| **Answer**  A scheme for communication between the user-thread library and the kernel is known as scheduler activation. It works as follows:  The kernel provides an application with a set of virtual processors (LWPs), and the application can schedule user threads onto an available virtual processor.  Furthermore, the kernel must inform an application about certain events. This procedure is known as an upcall. Upcalls are handled by the thread library with an upcall handler, and upcall handlers must run on a virtual processor. |
| **Q 20. Consider the following set of processes which the length of CPU-burst time given in milliseconds.**  **Process Burst Time Priority**  **P1 10 3**  **P2 1 1**  **P3 2 3**  **Processes are assumed to have arrived in the order P1, P2, P3 all at time 0**   1. **Draw gantt charts illustrating the execution of these processes using SJF, and non-preemptive priority scheduling (a smaller priority number implies a highest priority)** 2. **What is the turnaround time of each process for each scheduling algorithm in part a)?** 3. **What is the waiting time of each process for each scheduling algorithm in part a)?** 4. **Which of the scheduler in part a) results in the minimal average waiting time (overall processes)?** |
| **Answer**  **a)**    **b)**  Turnaround time  SJF non-preemptive priority  P1 13 11  P2 1 1  P3 3 13  **c)**  Waiting time  SJF non-preemptive priority  P1 3 1  P2 0 0  P3 1 11  **d)**  Average waiting time  SJF => (13 + 1 + 3)/3 = 5.6 milliseconds  non-preemptive priority => (1 + 0 + 11)/3 = 4 milliseconds |
| **Q 21. Consider a system with one printer that could be shared among multiple processes. Can this system go into deadlock state if more than two processes try to access it simultaneously? Why or why not?** |
| **Answer**  There are certainly other factors included other than the information provided   * Mutual Exclusion: if resources are shareable then no deadlock otherwise possibility of deadlock * Hold and Wait: A process can only request printer and other resources when it holds none. If this is true then possibility of deadlock otherwise no deadlock * No preemption: If a process using printer have to wait then it should be preempted otherwise possibility of deadlock * Circular wait: Enumeration of resource types. Resources can be allocated in increasing order. Otherwise possibility of deadlock |
| **Q 22. Is the following system of four processes (P1, P2, P3 and P4) with two resource types (A, B) deadlocked?**  **Process Allocation Need**  **A B A B**  **P1 1 3 1 2**  **P2 4 1 4 3**  **P3 1 2 1 7**  **P4 2 0 5 1**   1. **If the contents of Availability vector are (1, 4) for resource types (A, B) respectively.** 2. **If the contents of Availability vector are (2, 3) for resource types (A, B) respectively.** 3. **If the contents of Availability vector are (2, 4) for resource types (A, B) respectively.** |
| **Answer**  **a)**   1. Need P1 <= Available so execute P1   Available = 2 7   1. Need P3 <= Available so execute P3   Available = 3 9  No process can be executed further so the system is deadlocked  **b)**   1. Need P1 <= Available so execute P1   Available = 2 6  No process can be executed further so the system is deadlocked  **c)**   1. Need P1 <= Available so execute P1   Available = 3 7   1. Need P3 <= Available so execute P3   Available = 4 9   1. Need P2 <= Available so execute P2   Available = 8 10   1. Need P4 <= Available so execute P4   Available = 10 10  System is in safe state. Safe sequence is P1, P3, P2, P4 |
| **Q 23 Consider the following set of processes with the length of the CPU-burst time given in milliseconds.**  **Process Burst Time Priority**  **P1 10 3**  **P2 1 1**  **P3 2 3**  **P4 1 4**  **P5 5 2**  **The processes are assumed to have arrived in the order P1, P2, P3, P4, P5 all at time 0**   1. **Draw four gantt charts illustrating the execution of these processes using FCFS, SJF and non-preemptive priority (a smaller priority number implies a higher priority) and RR (quantum = 5) scheduling** 2. **Which of the scheduler in part a) results in minimal average waiting time (over all processes)?** |
| **a)**    **b)**    Waiting time for  FCFS = (0 + 10 + 11 + 13 + 14)/5 = 9.6 milliseconds  RR = (0 + 1 + 5 + 3 + 9)/5 = 5.4 milliseconds  SJF = (9 + 0 + 2 + 1 + 4)/5 = 3.2 milliseconds  Priority = (6 + 0 + 16 + 18 + 1)/5 = 8.2 milliseconds |
| **Q 24. Write down two benefits and two drawbacks of each threading model** |
| **Answer**  --One to one model—   1. Advantages 2. Scalable paralellism. Multithreaded programs written under the one-to-one model can achieve significant speedups when migrated from uniprocessors to multiprocessors 3. When one user thread and its kernel thread block, the other user threads can continue to execute 4. Disadvantages 5. Expensive creation. Under the one-to-one model, every thread creation requires explicit kernel involvement and consumes kernel resources 6. Resource inefficiency. Every thread created by the user requires kernel memory for a stack, as well as some sort of kernel data structure to keep track of it   --Many to many model—   1. Advantages 2. Large numbers of threads can be supported relatively cheaply. The creation of a user thread does not necessarily require the (relatively expensive) creation of a kernel thread 3. Synchronization can also be inexpensive: the implementation of synchronization primitives involves primarily user-level code 4. Disadvantages 5. Kernel mode becomes a resource whose unavailability can cause one group of threads to block another group of threads 6. more complicated to implement because you would have to manage the kernel mode threads as resources.   --Many to one model—   1. Advantages 2. Cheap thread creation. To create a new thread, the threads library need only create a context (i.e. a stack and registers) for the new thread 3. Portability. Because user-level threads packages are implemented entirely with standard UNIX and POSIX library calls 4. Disadvantages 5. No parallelism. Multithreaded programs under the many-to-one model will run no faster on multiprocessors than they run on uniprocessors 6. Since there is only one kernel thread, if a user thread executes a blocking system call, the entire process blocks, since no other user thread can execute until the kernel thread (which is blocked in the system call) becomes available |
| **Q 25. Imposing a total ordering of all resource types helps to prevent the occurance of deadlock in an operating system. How can you prove it?** |
| **Answer**  To illustrate, we let R = {R1, R2, ..., Rm} be the set of resource types. We assign to each resource type a unique integer number, which allows us to compare two resources and to determine whether one precedes another in our ordering. Formally, we define a one-to-one function F: R→N, where N is the set of natural numbers. For example, if the set of resource types R includes tape drives, disk drives, and printers, then the function F might be defined as follows:  F(tape drive) = 1  F(disk drive) = 5  F(printer) = 12   1. A process can initially request any number of instances of a resource type say, Ri . After that, the process can request instances of resource type Rj if and only if F(Rj ) > F(Ri ). a process requesting an instance of resource type Rj must have released any resources Ri such that  F(Ri ) ≥ F(Rj ) 2. If several instances of the same resource type are needed, a single request for all of them must be issued   If these two protocols are used, then the circular-wait condition cannot hold. We can demonstrate this fact by assuming that a circular wait exists (proof by contradiction). Let the set of processes involved in the circular wait be {P0, P1, ..., Pn}, where Pi is waiting for a resource Ri , which is held by process Pi+1. (Modulo arithmetic is used on the indexes, so that Pn is waiting for a resource Rn held by P0.) Then, since process Pi+1 is holding resource Ri while requesting resource Ri+1, we must have F(Ri ) < F(Ri+1) for all i. But this condition means that F(R0) < F(R1) < ... < F(Rn) < F(R0). By transitivity, F(R0) < F(R0), which is impossible. Therefore, there can be no circular wait. |
| **Q 26. How data can be shared between threads?** |
| **Answer**  Threads share data by sharing   * Code Segment * Heap Memory * Global Variables   But they don’t share   * Register Values * Stack Memory |
| **Q 27. How do you say that four conditions for deadlock are not completely independent?** |
| **Answer**   * The circular-wait condition implies the hold-and-wait condition, so the four conditions are not completely independent. * If Mutual exclusion is not applicable and all resources are indefinitely shareable then none of the remaining three conditions could arise |
| **Q28. Enlist the services offered by the Linux Operating System.** |
| **Answer**  One set of operating-system services provides functions that are helpful to the user   * **User interface**   + Command Line Interpreter     - Bourne Shell Command Interpreter   + Graphics User Interface     - CDE, KDE, GNOME * **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error) * **I/O operations** - A running program may require I/O, which may involve a file or an I/O device * **File-system manipulation** - Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management. * **Communications** – Processes may exchange information, on the same computer or between computers over a network * **Error detection** – OS needs to be constantly aware of possible errors   Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing   * **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them * **Logging -** To keep track of which users use how much and what kinds of computer resources * **Protection and security** |
| **Q 29. Compare methods to pass parameters to the operating system** |
| **Answer**   * Often, more information is required than simply identity of desired system call   + Exact type and amount of information vary according to OS and call * Three general methods used to pass parameters to the OS   + Simplest: pass the parameters in registers     - In some cases, may be more parameters than registers   + Parameters stored in a block*,* or table, in memory, and address of block passed as a parameter in a register     - This approach taken by Linux and Solaris   + Parameters placed, or **pushed***,* onto the **stack**by the program and **popped**off the stack by the operating system   + Block and stack methods do not limit the number or length of parameters being passed |
| **Q 30. Distinguish betweem the client-server and peer-to-peer models of distributed systems** |
| **Answer**  --Client-Server Computing--   * Dumb terminals supplanted by smart PCs * Many systems now **servers**, responding to requests generated by **clients**   + **Compute-server system** provides an interface to client to request services (i.e., database)   + **File-server system** provides interface for clients to store and retrieve files   1_18.pdf  --Peer-to-Peer—   * P2P does not distinguish clients and servers   + Instead all nodes are considered peers   + May each act as client, server or both   + Node must join P2P network     - Registers its service with central lookup service on network, or     - Broadcast request for service and respond to requests for service via ***discovery protocol***   + Examples includeNapsterandGnutella*,* **Voice over IP** (**VoIP**)such as Skype   1_19.pdf |
| **Q 31. What is the purpose of interrupts? What are differences between a trap and an interrupt? Can traps be generated intentionally by a user program? If so, for what purpose?** |
| **Answer**   * Interrupt transfers control to the interrupt service routine generally, through the **interrupt****vector**, which contains the addresses of all the service routines. * A **trap** is a software-generated interrupt caused either by an error or a user request and other type of interrupt is called hardware interrupt. Hardware interrupt is generated by one of the hardware devices * A user program can generate interrupt due to the following reasons   + I/O request   + Request for system service – system call   + Software error e.g., division by zero, infinite loop, process trying to modify each other or the operating system |
| **Q 32. Define the essential properties of the following types of operating systems**   1. **Time sharing** 2. **Real time** 3. **Clustered** |
| **Answer**  **a) Multitasking (Timesharing)**   * A logical extension of Batch systems– the CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing   + **Response time** should be < 1 second   + Each user has at least one program executing in memory 🢡**process**   + If several jobs ready to run at the same time 🢡 **CPU scheduling**   + If processes don’t fit in memory, **swapping** moves them in and out to run   + **Virtual memory** allows execution of processes not completely in memory   **b)** **Real time Embedded Systems**   * Real-time embedded systems most prevalent form of computers   + Vary considerable, special purpose, limited purpose OS, **real-time OS**   + Use expanding * Many other special computing environments as well   + Some have OSes, some perform tasks without an OS * Real-time OS has well-defined fixed time constraints   + Processing ***must*** be done within constraint   + Correct operation only if constraints met   **c) Clustered Systems**   * Like multiprocessor systems, but multiple systems working together   + Usually sharing storage via a storage-area network (SAN)   + Provides a high-availability service which survives failures     - Asymmetric clustering has one machine in hot-standby mode     - Symmetric clustering has multiple nodes running applications, monitoring each other   + Some clusters are for high-performance computing (HPC)     - Applications must be written to use parallelization   + Some have distributed lock manager (DLM) to avoid conflicting operations |
| **Q 33. Enlist four problems with layered approach. Elaborate only with figure different layers in UNIX operating system structure.** |
| **Answer** |
| **Q 34. Once the process is allocated the CPU and is executing, elaborate only with queuing diagram what several events could occur?** |
| **Answer** |
| **Q 35. A process P1 is executing on the CPU. A process P2 of higher priority required to be allocated the CPU. What information of P1 would be saved by the operating system?** |
| **Answer**   * When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch** * **Context** of a process represented in the PCB * Context-switch time is pure overhead; the system does no useful work while switching   + The more complex the OS and the PCB 🡺 the longer the context switch * Time dependent on hardware support   + Some hardware provides multiple sets of registers per CPU 🡺 multiple contexts loaded at once |
| **Q 36. Write down the reasons when a parent process terminates the execution of its child processes.** |
| **Answer**   * Parent may terminate the execution of children processes using the **abort()** system call. Some reasons for doing so:   + Child has exceeded allocated resources   + Task assigned to child is no longer required   + The parent is exiting, and the operating systems does not allow a child to continue if its parent terminates |