|  |  |
| --- | --- |
| 1. Operating system CANNOT merge I/O devices and files into a combined file because of the similarity of system calls for each. **2)** **RMI is similar to RPC** 2. The single **benefit of a thread pool** is to control the number of threads is **NOT** true. 3. **All contemporary operating systems** support kernel threads. 4. **Race conditions** are prevented by locks 5. Monitors ARE practiced in the modern programming languages. 6. If a resource-allocation graph has a cycle, it does NOT necessarily mean the system must be in a deadlocked state. 7. Ordering resources and requiring the resources to be acquired in order prevents the circular wait from occurring and therefore prevents deadlock from occurring: false 8. The banker's algorithm IS useful in a system with multiple instances of a resource type 9. A **non-preemptive** **scheduling** a process cannot be interrupted once it starts its execution, it will terminate only when it ends while in **preemptive scheduling** a process can be interrupted by another process in mid of its execution also. 10. **LWP (Light Weight Process)**: is an intermediate data structure between user and kernel threads, which appears to be a virtual processor on which process can schedule user thread to run on a kernel thread 11. **Gant Chart:** Is a way to draw scheduled processes such as **priority scheduling.** 12. **Spinlock**: The Process “spins” while waiting for a lock. Spinlocks have the advantage that no context switch is required when a process must wait on a lock, and a context switch may take considerable time. 13. **Semaphore:** Is a software based synchronization tool used against the critical section problem. 14. **Deadlock:** Is a situation where two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes. 15. **Critical Section:** Is a section of code that a process needs to execute, in which the process may be changing common variables, updating a table, or writing a file. 16. **Throughput:** Is measure of work for the number of processes that are completed per cpu time unit. 17. **Long-term scheduler** (or job scheduler) –selects which processes should be brought into the ready queue 18. **Short-term scheduler** (or CPU scheduler) – selects which process should be executed next and allocates CPU 19. **Midterm Scheduler** Sometimes it can be advantage to remove process from memory and thus decrease the degree of multi-programming. This scheme is called swapping 20. **Process P0 and P1 set to 1:** The implementation of a semaphore with a waiting queue may result in a situation where two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes. The event in question is the execution of a signal() operation. When such a state is reached, these processes are said to be deadlocked. **Resource-Allocation Graph 🡪** 21. Given the definition of a **resource-allocation graph**, it can be shown that, if the graph contains no cycles, then no process in the system is deadlocked. If the graph does contain a cycle, then a deadlock may exist. If each resource type has exactly one instance, then a cycle implies that a deadlock has occurred. If the cycle involves only a set of resource types, each of which has only a single instance, then a deadlock has occurred. Each process involved in the cycle is deadlocked. In this case, a cycle in the graph is both a necessary and a sufficient condition for the existence of deadlock. 22. **Page Number:** If the size of the logical address space is 2^m, and a page size is 2^n bytes, then the high-order m−n bits of a logical address designate the page number, and the n low-order bits designate the page offset. 23. **Paging partitions:** M = Size of memory, N = Size of page. 24. **First fit**: Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or at the location where the previous first-fit search ended. We can stop searching as soon as we find a free hole that is large enough. 25. **Best fit**: Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole. 26. **Worst fit**: Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach. 27. **Point-to-Point and Broadcast packet here:** 28. : It maps many user-level threads to one kernel thread. The entire system may block makes a block system call. 29. **One to one**: Each user-level thread has one corresponding kernel thread. The only drawback is that creating a user thread requires creating the corresponding kernel thread 30. **Many to many**: Multiplexes many user-level threads to a smaller or equal number of kernel threads. User can create as many threads as they want 31. The major difficulty in designing a layered operating system approach is **appropriately defining the various layers** |  |