INTRODUCTION

1. What are the five major activities of an operating system with regard to process management?
2. Describe a Process Control Block. What are some key components? How is it used?
3. Provide two programming examples in which multithreading provides better performance than a single-threaded solution.
4. Provide two programming examples in which multithreading does NOT provide better performance than a single-threaded solution.
5. What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?
6. Describe actions taken by the kernel to context switch between processes.
7. How does signal() operation associated with monitors differ from the corresponding operation defined for semaphores
8. Draw the 5-state process model. Label the states and describe what causes a process to move to/from each state.
9. What is the difference between kernel mode and user mode? Explain how having two distinct modes aids in designing an operating system
10. Which of the following instructions should be allowed only in kernel mode?
	1. Disable all interrupts
	2. Read the time of day clock
	3. Set the time of day clock
	4. Change the memory map
11. What is a trap instruction? Explain its use in operating systems?
12. What is an interrupt? Explain its use in an operating system?
13. To a programmer, a system call looks like any other call to ta library procedure. Is it important that a programmer know which library procedures result in system calls? Under what circumstances and why?

PROCESSES AND THREADS

1. What are the three requirements for the critical-section problem?
2. When an interrupt or system call transfers control to the operating system, a kernel stack area separate from the stack of it interrupted process is generally used. Why?
3. On all current computers, at least part of the interrupt handlers are written in assembly language. Why?
4. The register set is listed as a per-thread rather than a per-process item? Why? After all, the machine has only one set of registers.
5. Why would a thread ever voluntarily give up the CPU by calling thread\_yield? After, all, since there is not periodic clock interrupt, it may never get the CPU back.
6. Can a thread ever be preempted by a clock interrupt? If so, under what circumstances/ If not, why not?
7. Draw the 5-state thread model. Label the states and describe what causes a thread to move to/from each state.
8. When a process creates a new process using the *fork()I operation*, which of the following states is shared between the parent process and the childe process
	1. Stack
	2. Heap
	3. Shared Memory segments
9. Consider the following piece of code:

void main() {

 fork();

 fork();

 exit();

}

 How many processes are created upon execution of this process?

1. What resources are used when a thread is created? How do they differ from those used when a process is created?
2. Which of the following are shared across threads in a multithreaded process?
	1. Register values
	2. Heap memory
	3. Global variables
	4. Stack memory

SYNCHRONIZATION

1. What is meant by a code’s ‘critical section’?
2. What are the 3 requirements for a solution to the critical-section problem.
3. Describe the Peterson solution to the critical section problem.
4. Does Peterson’s solution to the mutual-exclusion problem work; when process scheduling is preemptive? How about when it is not preemptive?
5. What is a race-condition?
6. Given some code, place semaphores and mutexes correctly to assure the program satisfies mutual exclusion.
7. What is a spinlock? Why is it not appropriate for single-processor systems yet are often used in multiprocessor systems?
8. Explain why interrupts are not appropriate for implementing synchronization problems in multiprocessor systems.

INTERPROCESS COMMUNICATION / SYNCHRONIZATION

SCHEDULING

1. CPU Scheduling: Round Robin (RR) with quantum=1

Consider the following set of processes, with the length of the CPU burst given in milliseconds:

|  |  |  |  |
| --- | --- | --- | --- |
| **Process ID** | **Arrival Time** | **Burst Time** | **Priority** |
| P1 | 0 | 4 | 3 |
| P2 | 1 | 5 | 2 |
| P3 | 3 | 2 | 4 |
| P4 | 9 | 6 | 1 |
| P5 | 10 | 3 | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |   |

For each process compute the following. Show your work:

 Wait time

 Turnaround time

 Throughput:

For all the processes compute the following and show your work:

 Average Wait time

 Average Turnaround time

 Average Throughput:

1. Repeat for FIFO
2. Repeat for SJF
3. Explain the concepts behind Multilevel-feedback queue scheduling
4. Describe a scheduling technique for multiprocessor systems
5. Explain how time quantum value and context switching time affect each other in round-robin scheduling algorithm
6. Five batch jobs, A through E arrive at the computer center at almost the same time. They have estimated runtimes of 10, 6,2,4 and 8 minutes. Their (externally determined) priorities are 3,5,2,1 and 4, respectively with 5 being the highest priority. For each of the following scheduling algorithms determine the mean process turnaround time. Ignore process switching overhead.
	1. Round Robin (quantum time 2)
	2. Priority scheduling
	3. FCFS (assume order: A, B, C, D, E)
	4. Shortest job first