Operating Systems Assignments

Faculty Name : Himadi Nath Saha

Sl	Assignment Question	Week
1	The Sun Ultra SPARC processor has multiple register sets. Specify the actions of a context switch if the new context is already loaded into one of the register sets. What else must happen if the new context is in memory rather than in a register set and all the register sets are in use?	1
2	When a process creates a new process using the fork() operation, which of the following state is shared between the parent process and the child process? a. Stack b. Heap c. Shared memory segments	1
3	Again considering the RPC mechanism, consider the "exactly once" semantic. Does the algorithm for implementing this semantic execute correctly even if the "ACK" message back to the client is lost due to a network problem? Specify the sequence of messages and whether "exactly once" is still preserved.	1
4	Suppose that the following processes arrive for execution at the times indicated. Each process will run the listed amount of time. In answering the questions, use non-preemptive scheduling and base all decisions on the information you have at the time the decision must be made. ---------- ---------- ---------- Process Arrival Time Burst Time ---------- ---------- ---------- P_1 0.0 8 P_2 0.4 4 P_3 1.0 1 a. What is the average turnaround time for these processes with the FCFS scheduling algorithm? b. What is the average turnaround time for these processes with the SJF scheduling algorithm? c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P_1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1unit and then SJF scheduling is used. Remember that processes P_1 and P_2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.	2
5	Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on. These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum).What (if any) relation holds between the following pairs of sets of algorithms? a. Priority and SJF b. Multilevel feedback queues and FCFS c. Priority and FCFS d. RR and SJF	3

6	Suppose that a scheduling algorithm (at the level of short-term CPU scheduling) favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?	4
7	Assume an operating system maps user-level threads to the kernel using the many-to-many model where the mapping is done through the use of LWPs. Furthermore, the system allows program developers to create real-time threads. Is it necessary to bind a real-time thread to an LWP?	4
8	*The Cigarette-Smokers Problem*. Consider a system with three *smoker* processes and one *agent* process. Each smoker continuously rolls a cigarette and then smokes it. But to roll and smoke a cigarette, the smoker needs three ingredients: tobacco, paper, and matches. One of the smoker processes has paper, another has tobacco, and the third has matches. The agent has an infinite supply of all three materials. The agent places two of the ingredients on the table. The smoker who has the remaining ingredient then makes and smokes a cigarette, signaling the agent on completion. The agent then puts out another two of the three ingredients, and the cycle repeats. Write a program to synchronize the agent and the smokers using Java synchronization.	5
9	The wait () statement in all Java program examples was part of a "while" loop. Explain why you would always need to use a "while" statement when using wait () and why you would never use an "if" statement.	5
10	Give the reasons why Solaris, Windows XP, and Linux implement multiple locking mechanisms. Describe the circumstances under which they use spinlocks, mutexes, semaphores, adaptive mutexes, and condition variables. In each case, explain why the mechanism is needed.	6
11	Consider a computer system that runs 5,000 jobs per month with no deadlock-prevention or deadlock-avoidance scheme. Deadlocks occur about twice per month, and the operator must terminate and rerun about 10 jobs per deadlock. Each job is worth about $2 (in CPU time), and the jobs terminated tend to be about half-done when they are aborted. A systems programmer has estimated that a deadlock-avoidance algorithm (like the banker's algorithm) could be installed in the system with an increase in the average execution time per job of about 10 percent. Since the machine currently has 30-percent idle time, all 5,000 jobs per month could still be run, although turnaround time would increase by about 20 percent on average. a. What are the arguments for installing the deadlock-avoidance algorithm? b. What are the arguments against installing the deadlock-avoidance algorithm? c. Can a system detect that some of its processes are starving? If you answer "yes," explain how it can. If you answer "no," explain how the system can deal with the starvation problem.	7
12	Consider the following resource-allocation policy. Requests and releases for resources are allowed at any time. If a request for resources cannot be satisfied because the resources are not available, then we check any processes that are blocked, waiting for resources. If they have the desired resources, then these resources are taken away from them and are given to the requesting process. The vector of resources for which the process is waiting is increased to include the resources that were taken away. For example, consider a system with three resource types and the vector *Available* initialized to (4,2,2). If process P_0 asks for (2,2,1), it gets them. If P_1 asks for (1,0,1), it gets them. Then, if P_0 asks for (0,0,1), it is blocked (resource not available). If P_2 now asks for (2,0,0), it gets the available one (1,0,0) and one that was allocated to P_0 (since P_0 is blocked). P_0's *Allocation* vector goes down to (1,2,1) and its *Need* vector goes up to (1,0,1). a. Can deadlock occur? If you answer "yes", give an example. If you answer "no," specify which necessary condition cannot occur. b. Can indefinite blocking occur? Explain your answer.	8

13	Suppose that you have coded the deadlock-avoidance safety algorithm and now have been asked to implement the deadlock-detection algorithm. Can you do so by simply using the safety algorithm code and redefining $Max_i = Waiting_i + Allocation_i$, where $Waiting_i$ is a vector specifying the resources process i is waiting for, and $Allocation_i$ is? Explain your answer.	9
14	Is it possible to have a deadlock involving only one single process? Explain your answer.	9
15	Assume an operating system maps user-level threads to the kernel using the many-to-many model and the mapping is done through LWPs. Furthermore, the system allows developers to create real-time threads. Is it necessary to bind a real-time thread to an LWP? Explain.	10
16	Write a multithreaded pthread program to implement multiplication. Use C, C++ or Java	10