1. (15%) Some computer systems do not provide a privileged mode of operation in hardware. Is it possible to construct a secure operating system for these computer systems? Give arguments both that it is and that it is not possible.
2. (10%) Suppose you have a lab full of networked machines with broken clocks. The clocks can poll for the current time but cannot generate any interrupts. The machines are otherwise fine. The lack of a hardware clock would appear to make time slicing and preemptive multitasking impossible.
   1. Discuss how you can achieve the next best thing namely, cooperative multitasking, without modifying programs to make explicit calls to relinquish the processor. Your solution should still let a process run for at least a single time quantum.
   2. Now suppose you introduce one special machine to the network with a working clock. How can this single machine fix the problem and make all the machines in the lab preemptively multitasked.
3. (10%) Two computers science student, George and Mary, are having a discussion about i-nodes. George maintains that memories have gotten so large and so cheap that when a file is opened, it is simpler and faster just to fetch a new copy of i-node into i-node table, rather than search the entire table to see if it is already there. Mary disagrees. Who is right and why? (An i-node (index node) is a [data structure](http://en.wikipedia.org/wiki/Data_structure) found in many [Unix](http://en.wikipedia.org/wiki/Unix) [file systems](http://en.wikipedia.org/wiki/File_system). Each i-node stores all the information about a file system object ([file](http://en.wikipedia.org/wiki/Computer_file), [device node](http://en.wikipedia.org/wiki/Device_node), [socket](http://en.wikipedia.org/wiki/Socket), [pipe](http://en.wikipedia.org/wiki/Pipeline_(Unix)), etc.), except data content and file name.)
4. (15%) Peterson’s solution is a class software-based solution to the critical-section problem. This solution is restricted to two processes. The following is the structure of process *Pi* (other process *Pj*) and shared data. (In this case, the processes are numbered *P0* and *P1*.)

|  |  |
| --- | --- |
| Shared data | The structure of process *Pi* |
| boolean flag[2];  int turn;  flag[0]=flag[1]=false;  turn=0 or 1, initially | do{  flag[i] = true;  turn=j;  while flag[j]&&(turn==j);  critical section  flag[i] = false;  remainder section  While(true) |

Please prove that this solution satisfies the following three requirements:

1. Mutual exclusion.
2. Progress.
3. Bounded waiting.
4. (10%) Consider the deadlock situation that could occur in the dining-philosophers problem when the philosophers obtain the chopsticks one at a time. Discuss how the four necessary conditions for deadlock indeed hold in this setting. Discuss how deadlocks could be avoided by eliminating any one of the four conditions.
5. (15%)
   1. Please discuss parameter passing approaches for remote process calls (RPC)
   2. Please show why asynchronous RPC (non-blocking RPC) could be useful.
6. (15%)
   1. Explain why spinlocks are not appropriate for uniprocessor systems yet may be suitable for multiprocessor systems.
   2. Please describe the procedures and steps of a device driver to perform (a) Interrupt I/O (b) DMA
7. (10%) Please explain shortly about why below technologies are employed in the OS memory management?

(a) Paging (b) TLB (c) Segmentation (d) Segmentation with paging.