3SH3.Questions.txt

Question: What is a microkernel?
Answer: A microkernel aims to move as much code from kernel space into user space. The goal is to have the minimum amount of code running in kernel space. An advantage of this is increased security, because less code is running in kernel space. A disadvantage is the performance overhead of communication between user and kernel space.

Question: What is the difference between Linux and Unix?
Answer: Both are a family of operating systems. Unix was developed at Bell Labs by Ken Thompson, Dennis Ritchie, and others. Linux was was developed later by Linus Torvalds. Unix is (mostly) closed source. On the other hand, Linux is open source. Linux is based off of Unix; it is a Unix-like operating system - almost a clone.

Question: Why are microkernels more secure? Is it because less code is running with high-access; thus, the attack surface is less.
Answer: Microkernels move code from the kernel to the user space. So, if there is some vulnerability in the code, and an attacker manages to exploit it, then the attacker does not have full control of the kernel, but only user space. Compromising the kernel gives attackers full control to do whatever they want. We want the kernel to be as secure as possible, and minimizing the attack vector is a valid strategy.

Question: Is the third column of 'lsmod' the ‘pid’?
Answer: No, the third column in ‘lsmod’ is not the PID. The third column indicates how many instances of the module is being used. A value of 0 means that the module is not being used.

Question: In `simple.ko` does the `ko` stand for Kernel Object? Similar to how `o` stands for object in the `simple.o` file?
Answer: Yes. [(∗.ko)] stands for kernel object, and [(∗.o)] stands for object.

Question: Is the kernel part of the operating system or is the OS part of the kernel?
Answer: The kernel is part of the operating system. It is one of the first processes that runs when the operating system starts up. The relationship between the OS and the kernel is like a car and its engine.

Question: What does the OS do that the kernel doesn’t do?
Answer: The kernel runs all the time, while the operating system runs only when it is told to do something. The kernel is the middle man between hardware and software. If any program/process/application wants CPU time, memory, etc., then it must ask the kernel.

Question: The software or algorithm that is used for scheduling, where

is it stored when it’s in use?
Answer: The scheduling software is part of the operating system and is stored in secondary memory. When the operating system is booted, essential system processes like the scheduling algorithm are written to main memory, and this region is protected.

Question: Why is the PID of the child process 0? Isn’t a PID of 0 reserved for the Kernel?
Answer: The PID, relative to the child process, is always 0. But getting the PID of the child process from the parent is greater than zero.

Question: What is the difference between a thread and a process?
Answer: Both are independent sequences of execution. A process has its own memory, data, address space, etc. A thread shares this data with other threads, except for the stack. Processes are heavy on the system, and threads are light. Every process is a thread, but not every thread is a process.