System Calls, Traps, Exceptions

1. What is a system call? How is a system call different from a normal function call?

2. What is the difference between a system call, hardware interrupt, a software interrupt (trap), and an exception? Give examples of each.

3. How are trap handlers, exception handlers, and interrupt handlers different from system calls?

4. What steps take place when a system call is invoked by a process?

5. What is a system call table? Why is it needed? OR What role does it play in OS security?

6. Explain the CPU-privilege transitions during a system call.

7. (a) Why do some operating systems, such as Linux, map themselves (i.e. the kernel code and data) into the address space of each process? (b) What is the alternative?

8. Assume a mainstream monolithic OS, such as Linux. When a process makes a system call, how can the system call mechanism avoid any context switching overhead between the calling process and the OS? (as opposed to the overhead seen when switching between two processes).

9. How does the OS ensure that a user-level process does not jump to, and execute, arbitrary code in the OS memory?

10. Does a system call invocation by a process trigger a context switch? Why or why not?