1. Explain the key differences between early (single user) batch systems, multiprogrammed batch systems, and interactive time-sharing systems.

2. I/O is usually split up into two phases: a command phase (where the physical device controller receives a command from a kernel driver thread) and a completion phase (where the physical device controller indicates completion via a hardware interrupt). Why is this done?

3. In OS/2 and the first Windows NT operating system the video drivers ran in protection ring 1. Later, with Windows NT 4.0 Microsoft switched to a monolithic design so that video drivers ran in ring 0 with the kernel. What are both the advantages and disadvantages of this decision?

4. Is it true that if a process on a multiuser system doesn’t make any system calls and its code doesn’t generate any CPU or memory exceptions then it can be guaranteed to get its full time quantum $\Delta t_{\text{quantum}}$ on the CPU? Why or why not?

5. In using the `signal()` (or `sigaction()`) system call we can set a signal handler (which we write) or we can specify the keywords `SIG_DFL` or `SIG_IGN`. What does each of these last two do?

6. In the System V IPC suite we can set up a message queue and use `msgsnd()` and `msgrcv()` to send and receive messages. Each message has a `type`. Why? How is the type used?

7. In the System V IPC suite we can also let processes communicate via shared memory. What are both the advantages and disadvantages of using shared memory rather than using message queues for interprocess communication?

8. The kernel never (re)starts a user or root process without first setting a timer. Why?


10. The kernel runs in CPU ring 0 whereas root processes run in CPU ring 3 (along with user processes). How could a malicious root process do damage to the integrity of the system?

11. The system call `fork()` creates an offspring process with the same stack, data, and code segments. How does the parent process know it is the parent and not the offspring?

12. Consider the following functions below. Which of these must be system calls, i.e. which cannot be carried out entirely in user mode with only user privileges? You must give reasons for your answers.

   1a. `exp()` (mathematical exponential function)
   1b. `fopen()` (open a disk file)
   1c. `atof()` (convert a string to an integer)
   1d. `strcmp()` (compare two strings for equality)
   1e. `fork()` (create a duplicate process)

13. Criticize or defend the statement “Graphical interfaces do not use or need a command line.”

14. In a monolithic design we identified four layers of functionality: user level, upper kernel level, level-2 deferred procedure calls, and level-1 hardware interrupt handlers. Briefly describe each layer.

15. On most systems, when the user hits CTRL-C on the keyboard, control would transfer to the kernel. The kernel would then send a signal (usually `SIGINT`) to the process. Why not just have the kernel always kill the program? Why is the exception handled in this manner?