



3. Describe the main advantage of multiprogramming. In early computers, all data read/written was handled by the CPU since there was no Direct Memory Access. How does this organization effect multiprogramming?

4. Which of the following should only be allowed in kernel mode? Justify your answer.

(a) Disable all interrupts

(b) Read the time-of-day clock

(c) Set the time-of-day clock

(d) Change the memory map

5. Describe in detail the following system calls: fork, exec, and unlink. (Hint: use the unix `'man'` command to find more information about these system calls. It is not enough to take the two or three sentence description from the book.) For each of the system calls, give a condition that causes it to fail.

6. Can the system call

```
count = write(fd, buffer, nbytes);
```

every return a value in count other than `nbytes`? If so, why?

7. To a programmer, a system call look like any other call to a library procedure. Is it important that a programmer know which library procedures result in system calls? Under which circumstances, and why?

8. Describe some tradeoffs that operating system designers face. How are these tradeoffs resolved? Does it vary from system to system.