CHAPTER 1

Introduction

Practice Exercises

1.1 What are the three main purposes of an operating system?
Answer: The three main puropses are:

- To provide an environment for a computer user to execute programs on computer hardware in a convenient and efficient manner.
- To allocate the separate resources of the computer as needed to solve the problem given. The allocation process should be as fair and efficient as possible.
- As a control program it serves two major functions: (1) supervision of the execution of user programs to prevent errors and improper use of the computer, and (2) management of the operation and control of I/O devices.

1.2 What are the main differences between operating systems for mainframe computers and personal computers?
Answer: Generally, operating systems for batch systems have simpler requirements than for personal computers. Batch systems do not have to be concerned with interacting with a user as much as a personal computer. As a result, an operating system for a PC must be concerned with response time for an interactive user. Batch systems do not have such requirements. A pure batch system also may have not to handle time sharing, whereas an operating system must switch rapidly between different jobs.

1.3 List the four steps that are necessary to run a program on a completely dedicated machine—a computer that is running only that program.
Answer: The four steps are:

a. Reserve machine time.

b. Manually load program into memory.

c. Load starting address and begin execution.

d. Monitor and control execution of program from console.

1.4 We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to "waste" resources? Why is such a system not really wasteful?
Answer: Single-user systems should maximize use of the system for the user. A GUI might "waste" CPU cycles, but it optimizes the user's interaction with the system.

1.5 What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?
Answer: The main difficulty is keeping the operating system within the fixed time constraints of a real-time system. If the system does not complete a task in a certain time frame, it may cause a breakdown of the entire system it is running. Therefore when writing an operating system for a real-time system, the writer must be sure that his scheduling schemes don't allow response time to exceed the time constraint.

1.6 Consider the various definitions of *operating system*. Next, consider whether the operating system should include applications such as Web browsers and mail programs. Argue both that it should and that it should not, and support your answers.
Answer: **Point**. Applications such as web browsers and email tools are performing an increasingly important role in modern desktop computer systems. To fulfill this role, they should be incorporated as part of the operating system. By doing so, they can provide better performance and better integration with the rest of the system. In addition, these important applications can have the same look-and-feel as the operating system software.
Counterpoint. The fundamental role of the operating system is to manage system resources such as the CPU, memory, I/O devices, etc. In addition, it's role is to run software applications such as web browsers and email applications. By incorporating such applications into the operating system, we burden the operating system with additional functionality. Such a burden may result in the operating system performing a less-than-satisfactory job at managing system resources. In addition, we increase the size of the operating system thereby increasing the likelihood of system crashes and security violations.

1.7 How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system?
Answer: The distinction between kernel mode and user mode provides a rudimentary form of protection in the following manner. Certain instructions could be executed only when the CPU is in kernel mode. Similarly, hardware devices could be accessed only when the program is executing in kernel mode. Control over when interrupts could be enabled or disabled is also possible only when the CPU is in kernel mode. Consequently, the CPU has very limited capability when executing in user mode, thereby enforcing protection of critical resources.

1.8 Which of the following instructions should be privileged?

a. Set value of timer.

b. Read the clock.

c. Clear memory.

d. Issue a trap instruction.

e. Turn off interrupts.

f. Modify entries in device-status table.

g. Switch from user to kernel mode.

h. Access I/O device.

Answer: The following operations need to be privileged: Set value of timer, clear memory, turn off interrupts, modify entries in device-status table, access I/O device. The rest can be performed in user mode.

1.9 Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.
Answer: The data required by the operating system (passwords, access controls, accounting information, and so on) would have to be stored in or passed through unprotected memory and thus be accessible to unauthorized users.

1.10 Some CPUs provide for more than two modes of operation. What are two possible uses of these multiple modes?
Answer: Although most systems only distinguish between user and kernel modes, some CPUs have supported multiple modes. Multiple modes could be used to provide a finer-grained security policy. For example, rather than distinguishing between just user and kernel mode, you could distinguish between different types of user mode. Perhaps users belonging to the same group could execute each other's code. The machine would go into a specified mode when one of these users was running code. When the machine was in this mode, a member of the group could run code belonging to anyone else in the group.
Another possibility would be to provide different distinctions within kernel code. For example, a specific mode could allow USB device drivers to run. This would mean that USB devices could be serviced without having to switch to kernel mode, thereby essentially allowing USB device drivers to run in a quasi-user/kernel mode.

1.11 Timers could be used to compute the current time. Provide a short description of how this could be accomplished.
Answer: A program could use the following approach to compute the current time using timer interrupts. The program could set a timer for some time in the future and go to sleep. When it is awakened by the interrupt, it could update its local state, which it is using to keep track of the number of interrupts it has received thus far. It could then repeat this process of continually setting timer interrupts and updating its local state when the interrupts are actually raised.

1.12 Is the Internet a LAN or a WAN?

Answer: The Internet is a WAN as the various computers are located at geographically different places and are connected by long-distance network links.