Chapter 1

1.1 How are network computers different from traditional personal computers? Describe some usage scenarios in which it is advantageous to use network computers

1.4 Under what circumstances would a user be better off using a timesharing system rather than a PC or a single-user workstation? 3

1.8 Describe a mechanism for enforcing memory protection in order to prevent a program from modifying the memory associated with other programs.

1.14 Discuss, with examples, how the problem of maintaining coherence of cached data manifests itself in the following processing environments:
a. Single-processor systems
b. Multiprocessor systems
c. Distributed systems

1.22 Describe the differences between symmetric and asymmetric multiprocessing. What are three advantages and one disadvantage of multiprocessor systems?

1.24 What is the purpose of interrupts? What are the differences between a trap and an interrupt? Can traps be generated intentionally by a user program? If so, for what purpose?

1.25 Consider an SMP system similar to what is shown in Figure 1.6. Illustrate with an example how data residing in memory could in fact have two different values in each of the local caches.

Chapter 2

2.1 What are the five major activities of an operating system with regard to file management?

2.6 Would it be possible for the user to develop a new command interpreter using the system-call interface provided by the operating system?

2.10 What is the main advantage of the layered approach to system design? What are the disadvantages of using the layered approach?

2.17 In what ways is the modular kernel approach similar to the layered approach? In what ways does it differ from the layered approach?

2.19 What are the advantages and disadvantages of using the same system call interface for manipulating both files and devices?

2.21 Why do some systems store the operating system in firmware, while others store it on disk?

Chapter 3

3.2 Consider the RPC mechanism. Describe the undesirable consequences that could arise from not enforcing either the "at most once" or "exactly once" semantic. Describe possible uses for a mechanism that has neither of these guarantees.

3.3 With respect to the RPC mechanism, consider the "exactly once" semantic. Does the algorithm for implementing this semantic execute correctly even if the ACK message back to the client is lost due to a network problem? Describe the sequence of messages and discuss whether "exactly once" is still preserved.

3.6 The Sun UltraSPARC processor has multiple register sets. Describe what happens when a context switch occurs if the new context is already loaded into one of the register sets. What happens if the new context is in memory rather than in a register set and all the register sets are in use?

3.7 Construct a process tree similar to Figure 3.9. To obtain process information for the UNIX or Linux system, use the command ps -ael. Use the command man ps to get more information about the ps command. On Windows systems, you will have to use the task manager.

3.9 Describe the differences among short-term, medium-term, and longterm scheduling.

3.10 Including the initial parent process, how many processes are created by the program below?

```
#include  <stdio.h>
#include <unistd.h>
int main(){
/* fork a child process*/
fork();
/* fork another child process*/
fork();
/*and fork another*/
fork();
return 0;
}
```

Chapter 4

4.1 Provide two programming examples in which multithreading does not provide better performance than a single-threaded solution.

4.2 Write a multithreaded Java, Pthreads, or Win32 program that outputs prime numbers. This program should work as follows: The user will run the program and will enter a number on the command line. The program will then create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user.

4.3 Which of the following components of program state are shared across threads in a multithreaded process?
a. Register values
b. Heap memory
c. Global variables
d. Stack memory

4.6 What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

4.10 What resources are used when a thread is created? How do they differ from those used when a process is created?

4.15 Describe the actions taken by a thread library to context-switch between user-level threads.