Chapter 1 Introduction

Practice Exercises

1.1 What are the three main purposes of an operating system?

1.2 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?

1.3 What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?

1.4 Keeping in mind the various definitions ofoperating system,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.

1.5 How does the distinction betweenkernel mode and user mode function as a rudimentary form of protection (security) system?

1.6 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. AccessI/Odevice.

1.7 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.

1.8 SomeCPUs provide for more than two modes of operation. What are two possible uses of these multiple modes?

1.9 Timers could be used to compute the current time. Provide a short description of how this could be accomplished.

1.10 Give two reasons why caches are useful. What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?

Chapter 2 Operating-System Structures

Practice Exercises

2.1 What is the purpose of system calls?

2.2 What are the five major activities of an operating system with regard to process management?

2.3 What are the three major activities of an operating system with regard to memory management?

2.4 What are the three major activities of an operating system with regard to secondary-storage management?

2.5 What is the purpose of the command interpreter? Why is it usually separate from the kernel?

Exercises 95

2.6 What system calls have to be executed by a command interpreter or shell in order to start a new process?

2.7 What is the purpose of system programs?

2.8 What is the main advantage of the layered approach to system design?

What are the disadvantages of the layered approach?

2.9 List five services provided by an operating system, and explain how each

creates convenience for users. In which cases would it be impossible for

user-level programs to provide these services? Explain your answer.

2.10 Why do some systems store the operating system in firmware, while

others store it on disk?

Chapter 3 Processes

Practice Exercises

3.1 Using the program shown in Figure 3.30, explain what the output will

be atLINE A.

3.2 Including the initial parent process, how many processes are created by

the program shown in Figure 3.31?

150 Chapter 3 Processes

```
#include<stdio.h>

#include<unistd.h>

int main()

{

/* fork a child process */

fork();

/* fork another child process */

fork();

/* and fork another */
```

```
fork();
return 0;
}
```

Figure 3.31 How many processes are created?

3.3 Original versions of Apple's mobile iOSoperating system provided no means of concurrent processing. Discuss three major complications that concurrent processing adds to an operating system.

3.4 The SunUltraSPARCprocessor 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?

3.5 When a process creates a new process using thefork()operation, which of the following states is shared betweenthe parent process and the child process?

a. Stack

b. Heap

c. Shared memory segments

3.6 Consider the"exactly once"semantic with respect to theRPCmechanism. Does the algorithm for implementing this semantic execute correctly even if theACKmessage sent back to the client is lost due to a network problem? Describe the sequence of messages, and discuss whether "exactly once"is still preserved.

3.7 Assume that a distributed system is susceptible to server failure. What mechanisms would be required to guarantee the"exactly once"semantic for execution ofRPCs?

Exercises

3.8 Describe the differences among short-term, medium-term, and long-term scheduling.

Exercises 151

```
#include<stdio.h>

#include<unistd.h>

int main()

{

int i;

for (i = 0; i < 4; i++)

fork();

return 0;

}
```

Figure 3.32 How many processes are created?

3.9 Describe the actions taken by a kernel to context-switch between

processes.

3.10 Construct a process tree similar to Figure 3.8. To obtain process infor-mation for theUNIXor Linux system, use the commandps -ael.

```
#include<sys/types.h>

#include<stdio.h>

#include<unistd.h>

int main()

{

pidt pid;

/* fork a child process */

pid = fork();
```

```
if (pid < 0){ /* error occurred */

fprintf(stderr, "Fork Failed");

return 1;

}

else if (pid == 0){ /* child process */

execlp("/bin/ls","ls",NULL);

printf("LINE J");

}

else { /* parent process */

/* parent will wait for the child to complete */

wait(NULL);

printf("Child Complete");

}

return 0;

}
```

Figure 3.33 When willLINE Jbe reached?

Use the commandman psto get more information about the pscom-mand. The task manager on Windows systems does not provide the

parent processID,but theprocess monitortool, available from tech-net.microsoft.com, provides a process-tree tool.

Chapter 4 Threads

Practice Exercises

4.1 Provide two programming examples in which multithreading provides

better performance than a single-threaded solution.

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

4.3 Describe the actions taken by a kernel to context-switch between kernel-level threads.

4.4 What resources are used when a thread is created? How do they differ

from those used when a process is created?

192 Chapter 4 Threads

4.5 Assume that an operating system maps user-level threads to the kernel

using the many-to-many model and that the mapping is done through

LWPs. Furthermore, the system allows developers to create real-time

threads for use in real-time systems. Is it necessary to bind a real-time

thread to anLWP? Explain.

Exercises

4.6 Provide two programming examples in which multithreading doesnot

provide better performance than a single-threaded solution.

4.7 Under what circumstances does a multithreaded solution using multi-ple kernel threads provide better performance than a single-threaded

solution on a single-processor system?

4.8 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

4.9 Can a multithreaded solution using multiple user-level threads achieve

better performance on a multiprocessor system than on a single-processor system? Explain.

4.10 In Chapter 3, we discussed Google’s Chrome browser and its practice

of opening each new website in a separate process. Would the same benefits have been achieved if instead Chrome had been designed to

open each new website in a separate thread? Explain.

Chapter 5 Process Synchronization

Practice Exercises

5.1 In Section 5.4, we mentioned that disabling interrupts frequently can

affect the system's clock. Explain why this can occur and how such

effects can be minimized.

5.2 Explain why Windows, Linux, and Solaris implement multiple locking

mechanisms. Describe the circumstances under which they use spin-locks, mutex locks, semaphores, adaptive mutex locks, and condition

variables. In each case, explain why the mechanism is needed.

Exercises 243

5.3 What is the meaning of the termbusy waiting? What other kinds of

waiting are there in an operating system? Can busy waiting be avoided

altogether? Explain your answer.

5.4 Explain why spinlocks are not appropriate for single-processor systems

yet are often used in multiprocessor systems.

5.5 Show that, if thewait()andsignal()semaphore operations are not

executed atomically, then mutual exclusion may be violated.

5.6 Illustrate how a binary semaphore can be used to implement mutual

exclusion amongnprocesses.

Exercises

5.7 Race conditions are possible in many computer systems. Consider a

banking system that maintains an account balance with two functions:

deposit(amount)andwithdraw(amount). These two functions are

passed theamountthat is to be deposited or withdrawn from the bank account balance. Assume that a husband and wife share a bank account.

Concurrently, the husband calls thewithdraw()function and the wife

callsdeposit(). Describe how a race condition is possible and what

might be done to prevent the race condition from occurring.

5.8 The first known correct software solution to the critical-section problem

for two processes was developed by Dekker. The two processes,P0and

P1, share the following variables:

boolean flag[2]; /* initially false */

int turn;

The structure of processPi (i== 0 or 1) is shown in Figure 5.21. The

other process isPj (j== 1 or 0). Prove that the algorithm satisfies all

three requirements for the critical-section problem.

5.9 The first known correct software solution to the critical-section problem

for nprocesses with a lower bound on waiting ofn−1 turns was

presented by Eisenberg and McGuire.The processes share the following

variables:

enum pstate{idle, wantin, incs};

pstate flag[n];

int turn;

All the elements offlagare initiallyidle. The initial value of turnis

immaterial (between 0 andn-1). The structure of processPi is shown in

Figure 5.22. Prove that the algorithm satisfies all three requirements for

the critical-section problem.

5.10 Explain why implementing synchronization primitives by disabling

interrupts is not appropriate in a single-processor system if the syn-chronization primitives are to be used in user-level programs.