

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 of operating 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.

50 Chapter 1 Introduction

1.5 How does the distinction between kernel 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. Access I/O device.

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 Some CPUs 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 at LINE 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 iOS operating system provided no means of concurrent processing. Discuss three major complications that concurrent processing adds to an operating system.

3.4 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?

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

- a. Stack
- b. Heap
- c. Shared memory segments

3.6 Consider the "exactly once" semantic with respect to the RPC mechanism. Does the algorithm for implementing this semantic execute correctly even if the ACK message 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 information for theUNIXor Linux

system, use the commandps -ael.

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

```
#include<stdio.h>
```

```
#include<unistd.h>
```

```
int main()
```

```
{
```

```
pid_t 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 will LINE J be reached?

152 Chapter 3 Processes

Use the command `ps` to get more information about the `ps` command. The task manager on Windows systems does not provide the parent process ID, but the `process monitor` tool, available from tech-net.microsoft.com, provides a process-tree tool.

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 term busy 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 the `wait()` and `signal()` 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 among processes.

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)` and `withdraw(amount)`. These two functions are passed the amount that 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 the `withdraw()` function and the wife calls `deposit()`. Describe how a race condition is possible and what might be done to prevent the race condition from occurring.

5.8 Describe how the `compareandswap()` instruction can be used to provide mutual exclusion that satisfies the bounded-waiting requirement

5.9 Explain why interrupts are not appropriate for implementing synchronization primitives in multiprocessor systems.

5.10 The Linux kernel has a policy that a process cannot hold a spinlock while attempting to acquire a semaphore. Explain why this policy is in place.