Threads

The process model introduced in Chapter 3 assumed that a process was an executing program with a single thread of control. Many modern operating systems now provide features for a process to contain multiple threads of control. This chapter introduces many concepts associated with multithreaded computer systems and covers how to use Java to create and manipulate threads. We have found it especially useful to discuss how a Java thread maps to the thread model of the host operating system.

Exercises

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

Answer: (1) Any kind of sequential program is not a good candidate to be threaded. An example of this is a program that calculates an individual tax return. (2) Another example is a "shell" program such as the C-shell or Korn shell. Such a program must closely monitor its own working space such as open files, environment variables, and current working directory.

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

Answer: Context switching between user threads is quite similar to switching between kernel threads, although it is dependent on the threads library and how it maps user threads to kernel threads. In general, context switching between user threads involves taking a user thread of its LWP and replacing it with another thread. This act typically involves saving and restoring the state of the registers.

4.9 Under what circumstances does a multithreaded solution using multiple kernel threads provide better performance than a single-threaded solution on a single-processor system?

Answer: When a kernel thread suffers a page fault, another kernel thread can be switched in to use the interleaving time in a useful manner. A single-threaded process, on the other hand, will not be capable of

performing useful work when a page fault takes place. Therefore, in scenarios where a program might suffer from frequent page faults or has to wait for other system events, a multithreaded solution would perform better even on a single-processor system.

4.10 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

Answer: The threads of a multithreaded process share heap memory and global variables. Each thread has its separate set of register values and a separate stack.

4.11 Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a single-processor system?
Answer: A multithreaded system comprising of multiple user-level threads cannot make use of the different processors in a multiprocessor system simultaneously. The operating system sees only a single process and will not schedule the different threads of the process on separate processors. Consequently, there is no performance benefit associated with executing multiple user-level threads on a multiprocessor system.

4.12 As described in Section 4.5.2, Linux does not distinguish between processes and threads. Instead, Linux treats both in the same way, allowing a task to be more akin to a process or a thread depending on the set of flags passed to the `clone()` system call. However, many operating systems—such as Windows XP and Solaris—treat processes and threads differently. Typically, such systems use a notation wherein the data structure for a process contains pointers to the separate threads belonging to the process. Contrast these two approaches for modeling processes and threads within the kernel.
Answer: On one hand, in systems where processes and threads are considered as similar entities, some of the operating system code could be simplified. A scheduler, for instance, can consider the different processes and threads on an equal footing without requiring special code to examine the threads associated with a process during every scheduling step. On the other hand, this uniformity could make it harder to impose process-wide resource constraints in a direct manner. Instead, some extra complexity is required to identify which threads correspond to which process and perform the relevant accounting tasks.

4.13 The program shown in Figure 4.14 uses the Pthreads API. What would be output from the program at `LINE C` and `LINE P`?
Answer: Output at `LINE C` is 5. Output at `LINE P` is 0.

4.14 Consider a multiprocessor system and a multithreaded program written using the many-to-many threading model. Let the number of user-level

threads in the program be greater than the number of processors in the system. Discuss the performance implications of the following scenarios.

a. The number of kernel threads allocated to the program is less than the number of processors.

b. The number of kernel threads allocated to the program is equal to the number of processors.

c. The number of kernel threads allocated to the program is greater than the number of processors but less than the number of user-level threads.

Answer: When the number of kernel threads is less than the number of processors, then some of the processors would remain idle since the scheduler maps only kernel threads to processors and not user-level threads to processors. When the number of kernel threads is exactly equal to the number of processors, then it is possible that all of the processors might be utilized simultaneously. However, when a kernel-thread blocks inside the kernel (due to a page fault or while invoking system calls), the corresponding processor would remain idle. When there are more kernel threads than processors, a blocked kernel thread could be swapped out in favor of another kernel thread that is ready to execute, thereby increasing the utilization of the multiprocessor system.

4.15 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.
Answer: Please refer to the supporting Web site for source code solution.

4.16 Modify the socket-based date server (Figure 3.19) in Chapter 3 so that the server services each client request in a separate thread.
Answer: Please refer to the supporting Web site for source code solution.

4.17 The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8, ... Formally, it can be expressed as:

$$fib_0 = 0$$
$$fib_1 = 1$$
$$fib_n = fib_{n-1} + fib_{n-2}$$

Write a multithreaded program that generates the Fibonacci series using either the Java, Pthreads, or Win32 thread library. This program should work as follows: The user will enter on the command line the number of Fibonacci numbers that the program is to generate. The program will then create a separate thread that will generate the Fibonacci numbers, placing the sequence in data that is shared by the threads (an array is probably the most convenient data structure). When the thread finishes execution, the parent thread will output the sequence generated by the child thread. Because the parent thread cannot begin outputting the Fibonacci sequence until the child thread finishes, this will require having

the parent thread wait for the child thread to finish using the techniques described in Section 4.3.
Answer: Please refer to the supporting Web site for source code solution.

4.18 Exercise 3.18 in Chapter 3 specifies designing an echo server using the Java threading API. However, this server is single-threaded, meaning the server cannot respond to concurrent echo clients until the current client exits. Modify the solution to Exercise 3.18 so that the echo server services each client in a separate request
Answer: Please refer to the supporting Web site for source code solution.

4.19 A naming service such as DNS (for domain name system) can be used to resolve IP names to IP addresses. For example, when someone accesses the host `www.westminstercollege.edu`, a naming service is used to determine the IP address that is mapped to the IP name `www.westminstercollege.edu`. This assignment consists of writing a multithreaded naming service in Java using sockets (see Section 3.6.1.)

The `java.net` API provides the following mechanism for resolving IP names:

```
InetAddress hostAddress =
 InetAddress.getByName("www.westminstercollege.edu");

String IPaddress = hostAddress.getHostAddress();
```

where `getByName()` throws an `UnknownHostException` if it is unable to resolve the host name.

The Server

The server will listen to port 6052 waiting for client connections. When a client connection is made, the server will service the connection in a separate thread and will resume listening for additional client connections. Once a client makes a connection to the server, the client will write the IP name it wishes the server to resolve—such as `www.westminstercollege.edu`—to the socket. The server thread will read this IP name from the socket and either resolve its IP address or, if it cannot locate the host address, catch an `UnknownHostException`. The server will write the IP address back to the client or, in the case of an `UnknownHostException`, will write the message "`Unable to resolve host <host name>`." Once the server has written to the client, it will close its socket connection.

The Client

Initially, write just the server application and connect to it via telnet. For example, assuming the server is running on the localhost, a telnet session would appear as follows. (Client responses appear in blue.)

```
telnet localhost 6052
Connected to localhost.
Escape character is '^]'.
www.westminstercollege.edu
146.86.1.17
Connection closed by foreign host.
```

By initially having telnet act as a client, you can more accurately debug any problems you may have with your server. Once you are convinced your server is working properly, you can write a client application. The client will be passed the IP name that is to be resolved as a parameter. The client will open a socket connection to the server and then write the IP name that is to be resolved. It will then read the response sent back by the server. As an example, if the client is named `NSClient`, it is invoked as follows:

```
java NSClient www.westminstercollege.edu
```

and the server will respond with the corresponding IP address or "unknown host" message. Once the client has output the IP address, it will close its socket connection.

Answer: NO ANSWER