CHAPTER 2 Operating - System Structures

Review Questions

Section 2.1

2.1 List at least three operating system services that are useful to users.

2.2 List at least three operating system functions that maintain efficient operation of the system.

Section 2.2

2.3 What are the two different approaches for providing a user interface?

Section 2.3

2.4 What is a system call?

2.5 What is an API?

2.6 What kernel data structure can be used for one technique of passing parameters to system calls?

Section 2.4

2.7 List at least three of the major categories of system calls.

2.8 A program that has been loaded and executing is called a .

2.9 What part of the operating system makes the decision with regards to which job will run?

Section 2.5

2.10 List at least three of the categories of system programs.

2.11 True or False? The viewmost users see of the operating system is defined by application and system programs rather than system calls.

Section 2.6

2.12 What are the two basic goal groups that must be considered when designing an operating system?

2.13 What is the difference between policy and mechanism?

Section 2.7

2.14 List at least three different ways for structuring an operating system.

2.15 List at least two different hybrid operating systems.

2.16 What are the two devices that run the iOS operating system?

2.17 What technique domicrokernels use to communicate between services?

2.18 Provide an example of an operating system that uses the simple structure.

Section 2.8

2.19 True or False? Performance tuning is a type of debugging.

2.20 True or False? DTrace is available for Windows systems.

2.21 Name two activities the operating system is responsible for in connection with disk management.

2.22 Name at least two activities the operating system is responsible for in

connection with disk management.
2.23 Of the following 5 forms of storage, rank them from fastest to slowest in terms of access time: (1) main memory, (2) magnetic disk, (3) registers, (4) solid state disk, (5) cache.
Section 2.9
2.24 What does the term SYSGEN refer to?
Section 2.10
2.25 What is the name of the small piece of code that locates the kernel and loads it into main memory?

CHAPTER 3

Processes

Review Questions

Section 3.1
3.1 What are the four components of a process?
3.2 Provide at least three possible states a process may be in.
3.3 What is a Process Control Block (PCB) ?
3.4 What is another term for process?
3.5 True or False? Most operating systems allow a process to have multiple threads.
Section 3.2
3.6 What is the role of the process scheduler?
3.7 What is the degree of multiprogramming?
3.8 What is the term that describes saving the state of one process, and restoring the state of another?
3.9 What is the term that describes saving the state of one process, and restoring the state of another?
Section 3.3
3.10 What is a process identifier (PID)?
3.11 What system call creates a process on UNIX systems?
3.12 What system call creates a process on Windows systems?
3.13 What system call terminates a process on UNIX systems?
3.14 What is the name of the process that UNIX and Linux systems assign as the new parent of orphan processes?

Section 3.4
3.15 What are the two fundamental models of interprocess communication?
3.16 What are the two system calls used with message-passing systems?
3.17 True or False? Message passing is typically faster than shared memory.
3.18 How must shared memory behave for a rendezvous to occur?
Section 3.5
3.19 What system call is used to create a POSIX shared memory object?
3.20 What system call is used to configure the size of a POSIX shared memory

object?
3.21 What term does Mach use to describe mailboxes?
3.22 What system call does Mach use to create a new mailbox?
3.23 What term doesWindows use to name its message passing facility?
Section 3.6
3.24 Provide at least two types of communication mechanisms in client-server systems.
3.25 TCP sockets are (a) connection-oriented or (b) connection-less?
3.26 UDP sockets are (a) connection-oriented or (b) connection-less?
3.27 abstract procedure calls for use between systems with network connections.
3.28 What is parameter marshaling?
3.29 What are the two types of pipes?

CHAPTER 4

Threads

Review Questions

Section 4.1
4.1 How many threads does a traditional, heavyweight process have?
4.2 Provide at least three benefits of multithreaded programming.
Section 4.2
4.3 True or False? Concurrency is only possible with parallelism.
4.4 True or False? Amdahl's Law addresses the disproportionate effect of the serial portion of a program.
4.5 List at least three challenges when designing programming formulticore systems.
4.6 What are the two general types of parallelism?
Section 4.3
4.7 List the three common ways of mapping user threads to kernel threads.
4.8 True or False? Only Linux and Windows implement the one-to-one model.
Section 4.4
4.9 What are the two approaches for implementing a thread library?
4.10 What are the three main thread libraries in use?
4.11 True or False? PThreads is typically only implemented on UNIX-like systems.
4.12 True or False? PThreads is only a specification, not an implementation.
4.13 What is the PThread API for creating a thread?
4.14 What is theWindows API for creating a thread?

4.15 What Java method is used for allocating and initializing a new thread in the JVM?
Section 4.5

4.16 Provide at least two techniques for supporting implicit threading.
4.17 True or False? Grand Central Dispatch only works for Apple'sMac OS X and iOS operating systems.
Section 4.6
4.18 True or False? The semantics of the fork() system call can vary on multithreaded systems.
4.19 What are the two scenarios for canceling a target thread?
4.20 What is the PThreads API for thread cancellation?
Section 4.7
4.21 True or False?Windows threads provide both user and kernel stacks.
4.22 What term does Linux use to refer to a process or a thread?

CHAPTER 2 Operating-System Structures

Practice Exercises

2.1 What is the purpose of system calls?
Answer:
System calls allow user-level processes to request services of the operating system.
2.2 What are the five major activities of an operating system with regard to process management?
Answer:
The five major activities are:
a. The creation and deletion of both user and system processes
b. The suspension and resumption of processes
c. The provision of mechanisms for process synchronization
d. The provision of mechanisms for process communication
e. The provision of mechanisms for deadlock handling
2.3 What are the three major activities of an operating system with regard to memory management?
Answer:
The three major activities are:
a. Keep track of which parts ofmemory are currently being used and by whom.
b. Decide which processes are to be loaded into memory when memory space becomes available.
c. Allocate and deallocate memory space as needed.
2.4 What are the three major activities of an operating system with regard to secondary-storage management?
Answer:

The three major activities are:

- Free-space management.
- Storage allocation.
- Disk scheduling.

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

Answer:

It reads commands fromthe user or fromafile ofcommands and executes them, usually by turning them into one or more system calls. It is usually not part of the kernel since thecommandinterpreter is subject to changes.

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

Answer:

In Unix systems, a *fork* system call followed by an *exec* system call need to be performed to start a new process. The *fork* call clones the currently executing process, while the *exec* call overlays a new process based on a different executable over the calling process.

2.7 What is the purpose of system programs?

Answer:

System programs can be thought of as bundles of useful system calls. They provide basic functionality to users so that users do not need to write their own programs to solve common problems.

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

Answer:

As in all cases of modular design, designing an operating system in a modular way has several advantages. The system is easier to debug and modify because changes affect only limited sections of the system rather than touching all sections of the operating system. Information is kept only where it is needed and is accessible only within a defined and restricted area, so any bugs affecting that data must be limited to a specific module or layer.

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.

Answer:

The five services are:

a. **Program execution**. The operating system loads the contents (or sections) of a file intomemory and begins its execution.Auser-level program could not be trusted to properly allocate CPU time.

b. **I/O operations**.Disks, tapes, serial lines, and other devicesmust be communicated with at a very low level. The user need only specify the device and the operation to perform on it, while the system converts that request into device- or controller-specific commands. User-level programs cannot be trusted to access only devices they

should have access to and to access them only when they are otherwise unused.

c. **File-system manipulation**. There are many details in file creation, deletion, allocation, and naming that users should not have to perform. Blocks of disk space are used by files andmust be tracked.

Deleting a file requires removing the name file information and freeing the allocated blocks. Protections must also be checked to assure proper file access. User programs could neither ensure adherence to protection methods nor be trusted to allocate only free blocks and deallocate blocks on file deletion.

d. **Communications**. Message passing between systems requires messages to be turned into packets of information, sent to the network controller, transmitted across a communicationsmedium, and reassembled by the destination system. Packet ordering and data correction must take place. Again, user programs might not coordinate access to the network device, or they might receive packets destined for other processes.

e. **Error detection**. Error detection occurs at both the hardware and software levels. At the hardware level, all data transfers must be inspected to ensure that data have not been corrupted in transit.All data on media must be checked to be sure they have not changed since they were written to the media. At the software level, media must be checked for data consistency; for instance, whether the number of allocated and unallocated blocks of storage match the total number on the device. There, errors are frequently processindependent (for instance, the corruption of data on a disk), so there must be a global program (the operating system) that handles all types of errors. Also, by having errors processed by the operating system, processes need not contain code to catch and correct all the errors possible on a system.

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

Answer:

For certain devices, such as handheld PDAs and cellular telephones, a disk with a file system may be not be available for the device. In this situation, the operating system must be stored in firmware.

2.11 How could a system be designed to allow a choice of operating systems from which to boot? What would the bootstrap program need to do?

Answer:

Consider a system that would like to run both Windows XP and three different distributions of Linux (e.g., RedHat, Debian, and Mandrake). Each operating system will be stored on disk. During system boot-up, a special program (which we will call the **boot manager**) will determine which operating system to boot into. This means that rather initially booting to an operating system, the boot manager will first run during system startup. It is this boot manager that is responsible for determining which system to boot into. Typically boot managers must be stored at

certain locations of the hard disk to be recognized during system startup. Boot managers often provide the userwith a selection of systems to boot into; boot managers are also typically designed to boot into a default operating system if no choice is selected by the user.

CHAPTER 3

Processes

Practice Exercises

3.1 Using the program shown in Figure 3.30, explain what the output will be at Line A.

Answer:

The result is still 5 as the child updates its copy of value. When control returns to the parent, its value remains at 5.

3.2 Including the initial parent process, how many processes are created by the program shown in Figure 3.31?

Answer:

There are 8 processes 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.

Answer: FILL

3.4 The Sun UltraSPARC processor has multiple register sets.Describewhat 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?

Answer:

The CPU current-register-set pointer is changed to point to the set containing the new context, which takes very little time. If the context is in memory, one of the contexts in a register set must be chosen and be moved to memory, and the new context must be loaded from memory into the set. This process takes a little more time than on systems with one set of registers, depending on how a replacement victim is selected.

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

a. Stack

b. Heap

c. Shared memory segments

Answer:

Only the shared memory segments are shared between the parent process and the newly forked child process. Copies of the stack and the heap are made for the newly created process.

3.6 With respect to the RPCmechanism, 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.

Answer:

The "exactly once" semantics ensure that a remore procedure will be executed exactly once and only once. The general algorithm for ensuring this combines an acknowledgment (ACK) scheme combined with timestamps (or some other incremental counter that allows the server to distinguish between duplicate messages).

The general strategy is for the client to send the RPC to the server along with a timestamp. The client will also start a timeout clock. The client will then wait for one of two occurrences: (1) itwill receive an ACK from the server indicating that the remote procedure was performed, or (2) it will time out. If the client times out, it assumes the server was unable to perform the remote procedure so the client invokes the RPC a second time, sending a later timestamp. The client may not receive the ACK for one of two reasons: (1) the original RPC was never received by the server, or (2) the RPC was correctly received—and performed—by the server but the ACK was lost. In situation (1), the use of ACKs allows the server ultimately to receive and perform the RPC. In situation (2), the server will receive a duplicate RPC and it will use the timestamp to identify it as a duplicate so as not to perform the RPC a second time. It is important to note that the server must send a second ACK back to the client to inform the client the RPC has been performed.

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

Answer:

The server should keep track in stable storage (such as a disk log) information regarding what RPC operations were received, whether they were successfully performed, and the results associated with the operations. When a server crash takes place and a RPC message is

CHAPTER 4

Threads

Practice Exercises

4.1 Provide three programming examples in whichmultithreading provides better performance than a single-threaded solution.

Answer:

a. AWeb server that services each request in a separate thread.

b. A parallelized application such as matrix multiplication where different parts of the matrix may be worked on in parallel.

c. An interactive GUI program such as a debugger where a thread is used to monitor user input, another thread represents the running application, and a third thread monitors performance.

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

Answer:

a. User-level threads are unknown by the kernel, whereas the kernel is aware of kernel threads.

b. On systems using either M:1 or M:N mapping, user threads are scheduled by the thread library and the kernel schedules kernel threads.

c. Kernel threads need not be associatedwith a processwhereas every user thread belongs to a process. Kernel threads are generally more expensive to maintain than user threads as they must be represented with a kernel data structure.

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

Answer:

Context switching between kernel threads typically requires saving the value of the CPU registers from the thread being switched out and restoring the CPU registers of the new thread being scheduled.

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

Answer:

Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation. Creating a process requires allocating a process control block (PCB), a rather large data structure. The PCB includes a memory map, list of open files, and environment variables. Allocating and managing the memory map is typically the most time-consuming activity. Creating either a user or kernel thread involves allocating a small data structure to hold a register set, stack, and priority.

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 an LWP? Explain.

Answer:

Yes. Timing is crucial to real-time applications. If a thread is marked as real-time but is not bound to an LWP, the thread may have to wait to be attached to an LWP before running. Consider if a real-time thread is running (is attached to an LWP) and then proceeds to block (i.e. must perform I/O, has been preempted by a higher-priority real-time thread, is waiting for a mutual exclusion lock, etc.) While the real-time thread is blocked, the LWP it was attached to has been assigned to another thread. When the real-time thread has been scheduled to run again, it must first wait to be attached to an LWP. By binding an LWP to a real-time thread you are ensuring the thread will be able to run with minimal delay once it is scheduled.