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 of memory 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 from the user or from a file of commands and executes them, usually by turning them into one or more system calls. It is usually not part of the kernel since the command interpreter 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 into memory and begins its execution. A user-level program could not be trusted to properly allocate CPU time.

b. **I/O operations**. Disks, tapes, serial lines, and other devices must 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 and must 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 communications medium, 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 process-independent (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 user with 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.

Exercises

2.12 The services and functions provided by an operating system can be divided into two main categories. Briefly describe the two categories and discuss how they differ.

Answer: One class of services provided by an operating system is to enforce protection between different processes running concurrently in the system. Processes are allowed to access only those memory locations that are associated with their address spaces. Also, processes are not allowed to corrupt files associated with other users. A process is also not allowed to access devices directly without operating system intervention. The second class of services provided by an operating system is to provide new functionality that is not supported directly by the underlying hardware. Virtual memory and file systems are two such examples of new services provided by an operating system.

2.13 Describe three general methods for passing parameters to the operating system.

Answer:

a. Pass parameters in registers

b. Registers pass starting addresses of blocks of parameters

c. Parameters can be placed, or *pushed*, onto the *stack* by the program, and *popped* off the stack by the operating system.

2.14 Describe how you could obtain a statistical profile of the amount of time spent by a program executing different sections of its code. Discuss the importance of obtaining such a statistical profile.

Answer: One could issue periodic timer interrupts and monitor what instructions or what sections of code are currently executing when the interrupts are delivered. A statistical profile of which pieces of code were active should be consistent with the time spent by the program in different sections of its code. Once such a statistical profile has been obtained, the programmer could optimize those sections of code that are consuming more of the CPU resources.

2.15 What are the five major activities of an operating system in regard to file management?

Answer:

• The creation and deletion of files

• The creation and deletion of directories

• The support of primitives for manipulating files and directories

• The mapping of files onto secondary storage

• The backup of files on stable (nonvolatile) storage media

2.16 What are the advantages and disadvantages of using the same system call interface for manipulating both files and devices?
Answer: Each device can be accessed as though it was a file in the file system. Since most of the kernel deals with devices through this file interface, it is relatively easy to add a new device driver by implementing the hardware-specific code to support this abstract file interface. Therefore, this benefits the development of both user program code, which can be written to access devices and files in the same manner, and device driver code, which can be written to support a well-defined API. The disadvantage with using the same interface is that it might be difficult to capture the functionality of certain devices within the context of the file access API, thereby either resulting in a loss of functionality or a loss of performance. Some of this could be overcome by the use of ioctl operation that provides a general purpose interface for processes to invoke operations on devices.

2.18 What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?
Answer: The two models of interprocess communication are message passing model and the shared-memory model. The message-passing model controls the

2.19 Why is the separation of mechanism and policy desirable?
Answer: Mechanism and policy must be separate to ensure that systems are easy to modify. No two system installations are the same, so each installation may want to tune the operating system to suit its needs. With mechanism and policy separate, the policy may be changed at will while the mechanism stays unchanged. This arrangement provides a more flexible system.

2.20 It is sometimes difficult to achieve a layered approach if two components of the operating system are dependent on each other. Identify a scenario in which it is unclear how to layer two system components that require tight coupling of their functionalities.
Answer: The virtual memory subsystem and the storage subsystem are typically tightly-coupled and requires careful design in a layered system due to the following interactions. Many systems allow files to be mapped into the virtual memory space of

an executing process. On the other hand, the virtual memory subsystem typically uses the storage system to provide the backing store for pages that do not currently reside in memory. Also, updates to the file system are sometimes buffered in physical memory before it is flushed to disk, thereby requiring careful coordination of the usage of memory between the virtual memory subsystem and the file system.

2.21 What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?
Answer: Benefits typically include the following (a) adding a new service does not require modifying the kernel, (b) it is more secure as more operations are done in user mode than in kernel mode, and (c) a simpler kernel design and functionality typically results in a more reliable operating system. User programs and system services interact in a microkernel architecture by using interprocess communication mechanisms such asmessaging. These messages are conveyed by the operating system. The primary disadvantage of the microkernel architecture are the overheads associated with interprocess communication and the frequent use of the operating system's messaging functions in order to enable the user process and the system service to interact with each other.

2.22 In what ways is the modular kernel approach similar to the layered approach? In what ways does it differ from the layered approach?
Answer: The modular kernel approach requires subsystems to interact with each other through carefully constructed interfaces that are typically narrow (in terms of the functionality that is exposed to external modules). The layered kernel approach is similar in that respect. However, the layered kernel imposes a strict ordering of subsystems such that subsystems at the lower layers are not allowed to invoke operations corresponding to the upper-layer subsystems. There are no such restrictions in the modular-kernel approach, wherein modules are free to invoke each other without any constraints.

2.23 What is the main advantage for an operating-system designer of using virtual-machine architecture? What is the main advantage for a user?
Answer: The system is easy to debug, and security problems are easy to solve. Virtual machines also provide a good platform for operating system research since many different operating systems may run on one physical system.

2.24 Why is a just-in-time compiler useful for executing Java programs?

Answer: Java is an interpreted language. This means that the JVM interprets the bytecode instructions one at a time. Typically, most interpreted environments are slower than running native binaries, for the interpretation process requires converting each instruction into native machine code. A just-in-time (JIT) compiler compiles the bytecode for a method into native machine code the first time the method is encountered. This means that the Java program is essentially running as a native application (of course, the conversion process of the JIT takes time as well but not as much as bytecode interpretation.) Furthermore, the JIT caches compiled code so that it may be reused the next time the method is encountered. A Java program that is run by a JIT rather than a traditional interpreter typically runs much faster.

2.25 What is the relationship between a guest operating system and a host operating system in a system like VMware? What factors need to be considered in choosing the host operating system?

Answer: A guest operating system provides its services by mapping them onto the functionality provided by the host operating system. A key issue that needs to be considered in choosing the host operating system is whether it is sufficiently general in terms of its system-call interface in order to be able to support the functionality associated with the guest operating system.

2.26 The experimental Synthesis operating system has an assembler incorporated within the kernel. To optimize system-call performance, the kernel assembles routines within kernel space to minimize the path that the system call must take through the kernel. This approach is the antithesis of the layered approach, in which the path through the kernel is extended to make building the operating system easier. Discuss the pros and cons of the Synthesis approach to kernel design and to system-performance optimization.

Answer: Synthesis is impressive due to the performance it achieves through on-the-fly compilation. Unfortunately, it is difficult to debug problems within the kernel due to the fluidity of the code. Also, such compilation is system specific; making Synthesis difficult to port (a new compiler must be written for each architecture).