Chapter 2  Operating-System Structures

The types of requests vary according to level. The system-call level must provide the basic functions, such as process control and file and device manipulation. Higher-level requests, satisfied by the command interpreter or system programs, are translated into a sequence of system calls. System services can be classified into several categories: program control, status requests, and I/O requests. Program errors can be considered implicit requests for service.

The design of a new operating system is a major task. It is important that the goals of the system be well defined before the design begins. The type of system desired is the foundation for choices among various algorithms and strategies that will be needed.

Throughout the entire design cycle, we must be careful to separate policy decisions from implementation details (mechanisms). This separation allows maximum flexibility if policy decisions are to be changed later.

Once an operating system is designed, it must be implemented. Operating systems today are almost always written in a systems-implementation language or in a higher-level language. This feature improves their implementation, maintenance, and portability.

A system as large and complex as a modern operating system must be engineered carefully. Modularity is important. Designing a system as a sequence of layers or using a microkernel is considered a good technique. Many operating systems now support dynamically loaded modules, which allow adding functionality to an operating system while it is executing. Generally, operating systems adopt a hybrid approach that combines several different types of structures.

Debugging process and kernel failures can be accomplished through the use of debuggers and other tools that analyze core dumps. Tools such as DTrace analyze production systems to find bottlenecks and understand other system behavior.

To create an operating system for a particular machine configuration, we must perform system generation. For the computer system to begin running, the CPU must initialize and start executing the bootstrap program in firmware. The bootstrap can execute the operating system directly if the operating system is also in the firmware, or it can complete a sequence in which it loads progressively smarter programs from firmware and disk until the operating system itself is loaded into memory and executed.

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?
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?

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?

---

**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.

2.13 Describe three general methods for passing parameters to 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.

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

2.16 What are the advantages and disadvantages of using the same system-call interface for manipulating both files and devices?

2.17 Would it be possible for the user to develop a new command interpreter using the system-call interface provided by the operating system?

2.18 What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?

2.19 Why is the separation of mechanism and policy desirable?

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.

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?

2.22 What are the advantages of using loadable kernel modules?
Chapter 2  Operating-System Structures

The types of requests vary according to level. The system-call level must provide the basic functions, such as process control and file and device manipulation. Higher-level requests, satisfied by the command interpreter or system programs, are translated into a sequence of system calls. System services can be classified into several categories: program control, status requests, and I/O requests. Program errors can be considered implicit requests for service.

The design of a new operating system is a major task. It is important that the goals of the system be well defined before the design begins. The type of system desired is the foundation for choices among various algorithms and strategies that will be needed.

Throughout the entire design cycle, we must be careful to separate policy decisions from implementation details (mechanisms). This separation allows maximum flexibility if policy decisions are to be changed later.

Once an operating system is designed, it must be implemented. Operating systems today are almost always written in a systems-implementation language or in a higher-level language. This feature improves their implementation, maintenance, and portability.

A system as large and complex as a modern operating system must be engineered carefully. Modularity is important. Designing a system as a sequence of layers or using a microkernel is considered a good technique. Many operating systems now support dynamically loaded modules, which allow adding functionality to an operating system while it is executing. Generally, operating systems adopt a hybrid approach that combines several different types of structures.

Debugging process and kernel failures can be accomplished through the use of debuggers and other tools that analyze core dumps. Tools such as DTrace analyze production systems to find bottlenecks and understand other system behavior.

To create an operating system for a particular machine configuration, we must perform system generation. For the computer system to begin running, the CPU must initialize and start executing the bootstrap program in firmware. The bootstrap can execute the operating system directly if the operating system is also in the firmware, or it can complete a sequence in which it loads progressively smarter programs from firmware and disk until the operating system itself is loaded into memory and executed.

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?