Chapter 1

Basic Concepts of Operating Systems

1.1 Introduction

The operating system is an essential part of a computer system; it is an intermediary component between the application programs and the hardware. The ultimate purpose of an operating system is twofold: (1) to provide various services to users' programs and (2) to control the functioning of the computer system hardware in an efficient and effective manner.

This chapter presents the basic concepts and the *abstract* views of operating systems. This preliminary material is important in understanding more clearly the complexities of the structure of operating systems and how the various services are provided. General and brief references to Unix and Windows are included throughout the chapter. Appendix A presents a detailed explanation of the Linux commands.

1.1.1 Software Components

A *program* is a sequence of instructions that enables a computer to execute a specific task. Before a program executes, it has to be translated from its original text form (source program) into a machine language program. The program then needs to be linked and loaded into memory.

The *software components* are the collection of programs that execute in the computer. These programs perform computations and control, manage, and carry out other important tasks. There are two general types of software components:

- System software
- Application software

1

System software is the set of programs that control the activities and functions of the various hardware components, programming tools and abstractions, and other utilities to monitor the state of the computer system. This software forms an environment for the programmers to develop and execute their programs (collectively known as application software). There are actually two types of users: application programmers and end users.

The most important part of system software is the operating system (OS) that directly controls and manages the hardware resources of the computer. The OS also provides a set of services to the user programs. The most common examples of operating systems are Linux, Unix, Windows, MacOS, and OS/2.

Application software are the user programs and consists of those programs that solve specific problems for the users and execute under the control of the operating system. Application programs are developed by individuals and organizations for solving specific problems.

1.2   Operating Systems

An operating system (OS) is a large and complex set of system programs that control the various operations of a computer system and provide a collection of services to other (user) programs. The purpose of an operating system involves two key goals:

- Availability of a convenient, easy-to-use, and powerful set of services that are provided to the users and the application programs in the computer system

- Management of the computer resources in the most efficient manner

Application and system programmers directly or indirectly communicate with the operating system in order to request some of these services.

The services provided by an operating system are implemented as a large set of system functions that include scheduling of programs, memory management, device management, file management, network management, and other more advanced services related to protection and security. The operating system is also considered a huge resource manager and performs a variety of services as efficiently as possible to ensure the desired performance of the system.

Figure 1.1 shows an external view of a computer system. It illustrates the programmers and end users communicating with the operating system and other system software.

The most important active resource in the computer system is the CPU. Other important resources are memory and I/O devices. Allocation and deallocation of all resources in the system are handled by various resource managers of the operating system.

General-purpose operating systems support two general modes of operation: (1) user mode and (2) kernel mode, which is sometimes called the supervisor, protected, or privilege mode. A user process will normally execute in user mode. Some instructions, such as low-level I/O functions and memory access to special areas where the OS maintains
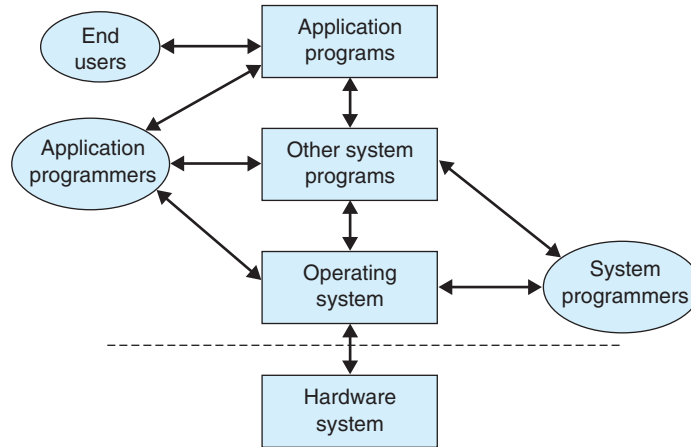
Figure 1.1   An external view of a computer system.

its data structures, can execute only in kernel mode. As such, the OS in kernel mode has direct control of the computer system.

1.2.1   Operating System Interfaces

Users and application programmers can communicate with an operating system through its interfaces. There are three general levels of interfaces provided by an operating system:

- Graphical user interface (GUI)

- Command line interpreter (also called the shell)

- System-call interface

Figure 1.2 illustrates the basic structure of an operating system and shows the three levels of interfaces, which are found immediately above the kernel. The highest level is the graphical user interface (GUI), which allows I/O interaction with a user through intuitive icons, menus, and other graphical objects. With these objects, the user interacts with the operating system in a relatively easy and convenient manner—for example, using the click of a mouse to select an option or command. The most common GUI examples are the Windows desktop and the X-Window in Unix.

The user at this level is completely separated from any intrinsic detail about the underlying system. This level of operating system interface is not considered an essential part of the operating system; it is rather an add-on system software component.
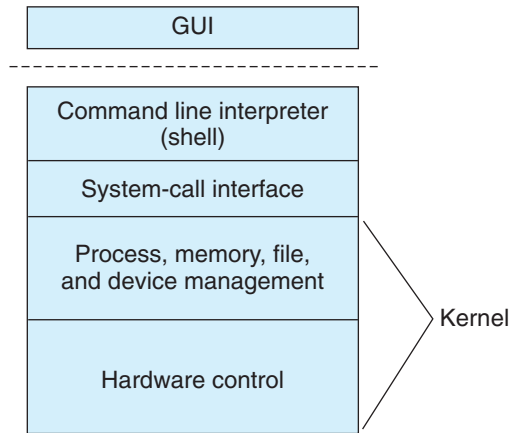
```
┌─────────────────────────────────────┐
│                 GUI                  │
└─────────────────────────────────────┘
- - - - - - - - - - - - - - - - - - - - - - -
┌─────────────────────────────────────┐
│      Command line interpreter        │
│               (shell)                │
├─────────────────────────────────────┤
│         System-call interface        │
├─────────────────────────────────────┤
│       Process, memory, file,         │
│      and device management           │
├─────────────────────────────────────┤
│                                      │      Kernel
│           Hardware control           │
│                                      │
└─────────────────────────────────────┘
```

Figure 1.2   Basic structure of an operating system.

The second level of interface, the command line interpreter, or shell, is a text-oriented interface. Advanced users and application programmers will normally directly communicate with the operating system at this level. In general, the GUI and the shell are at the same level in the structure of an operating system.

The third level, the system-call interface, is used by the application programs to request the various services provided by the operating system by invoking or calling system functions.

1.2.2   Multi-Level Views of an Operating System

The overall structure of an operating system can be more easily studied by dividing it into the various software components and using a top-down (layered) approach. This division includes the separation of the different interface levels, as previously discussed, and the additional components of the operating system. The higher the level of the layer, the simpler the interaction with the system. The top layer provides the easiest interface to the human operators and to users interacting with the system. Any layer uses the services or functions provided by the next lower layer.

As mentioned previously, the main goal of an operating system is to provide services to the various user and system programs. The operating system is structured into several components, each one providing its own group of services. For a more clear understanding of the complexities of how the various services are provided, two abstract views of the OS are presented: (1) the external view and (2) the internal view.

1.2.2.1   External View of an Operating System

The external view of an operating system is the set of interfaces discussed previously. These are sometimes referred to as the *abstract views*. Users of a computer system

directly or indirectly communicate with the operating system in order to request and receive some type of service provided by the operating system. Users and application programs view the computer system at the higher level(s) of the operating system interface. The low-level OS structures (internal view) and the hardware details are hidden behind the OS interface.

Figure 1.3 illustrates the abstract views of a computer system. Only the upper-level abstract views are important to users of the OS. The low-level OS view and the hardware view are relevant only to system specialists.
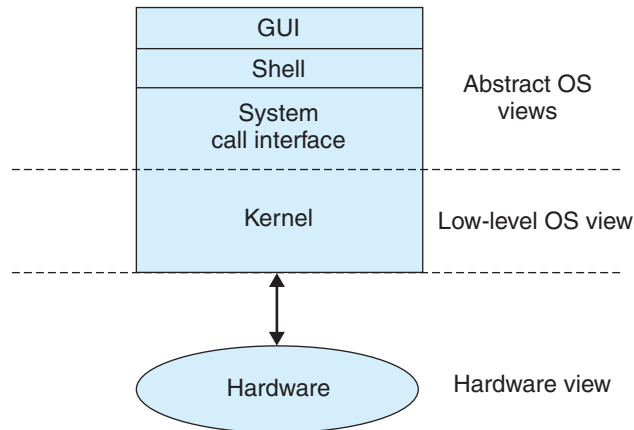

Figure 1.3   Abstract views of a computer system.

One of the main principles applied in the design of the external view concept is simplicity. The system interfaces hide the details of how the underlying (lower-level) machinery operates. The operational model of the computer system presented to the user should be relatively easy to use and is an abstract model of the operations of the system.

The abstraction provided by the OS interfaces gives users and application programs the appearance of a simpler machine. The various programs also convey the illusion that each one executes on its own machine; this abstraction is sometimes called the *abstract machine view.*

1.2.2.2   Internal View of an Operating System

The system services interface (or the system call interface) separates the *kernel* from the application layer. Located above the hardware, the kernel is the core and the most critical part of the operating system and needs to be always resident in memory. A detailed knowledge about the different components of the operating system, including these lower-level components, corresponds to an internal view of the system.

In studying the internal view of the operating system, it becomes important to identify the various components of the OS that establish policies for the various services they support and the mechanisms used to implement these policies.

It was previously noted that the various services provided by the operating systems include process management, memory management, device management, file management, network management, and other more advanced services related to protection and security. The most important functional components of the operating system are as follows:

- Process manager

- Memory manager

- Resource manager

- File manager

- Device manager

These components are actually modules of the operating system and are strongly related. In addition to these basic functions, most operating systems have some type of network management, security management, and other more advanced functions.

Process Manager

The *process manager* can be considered the most important component of the operating system. It carries out several services such as creating, suspending (blocking), executing, terminating, and destroying processes.

With multiprogramming, several active processes are stored and kept in memory at the same time. If there is only one CPU in the system, then only one process can be in execution at a given time; the other processes are waiting for CPU service. The decision regarding which process to execute next is taken by the scheduler. The dispatcher allocates the CPU to a process and deallocates the CPU from another process. This switching of the CPU from one process to another process is called *context switching* and is considered overhead time.

Memory Manager

The *memory manager* controls the allocation and deallocation of memory. It imposes certain policies and mechanisms for memory management. This component also includes policies and mechanisms for memory protection. Relevant memory management schemes are paging, segmentation, and virtual memory.

Resource Manager

The *resource manager* facilitates the allocation and deallocation of resources to the requesting processes. Functions of this component include the maintenance of resource tables with information such as the number of resources of each type that are available and the number of resources of each type that have been allocated to specific processes. The access mode for the various resources is another type of function.

File Manager

The *file manager* allows users and processes to create and delete files and directories. In most modern operating systems, files are associated with mass storage devices such as magnetic tapes and disks. Data can be read and/or written to a file using functions such as open file, read data from file, write data to file, and close file. The Unix operating system also uses files to represent I/O devices (including the main input device and a main output device such as the console keyboard and video screen).

Device Manager

For the various devices in the computer system, the *device manager* provides an appropriate level of abstraction to handle system devices. For every device type, a device driver program is included in the OS. The device manager operates in combination with the resource manager and file manager. Usually, user processes are not provided direct access to system resources. Instead, processes request services to the file manager and/or the resource manager.

1.3   Categories of Operating Systems

A common way to classify operating systems is based on how user programs are processed. These systems can be classified into one of the following categories:

- *Batch systems*, in which a set of jobs is submitted in sequence for processing. These systems have a relatively long average turnaround period (interval from job arrival time to completion time) for jobs.

- *Interactive systems*, which support interactive computing for users connected to the computer system via communication lines. The most common type of operating systems that support interactive computing is *time-sharing*. Figure 1.4 illustrates the general structure of a time-sharing system. A time-sharing system is an example of a multiuser system, which means that multiple users are accessing the system at the same time.

- *Real-time systems*, which support application programs with very tight timing constraints.

- *Hybrid systems*, which support batch and interactive computing.

The most general classification of operating systems is regarding the application environment and includes the following categories:

- *General purpose*, which are used mainly to develop programs and execute these on most types of platforms.
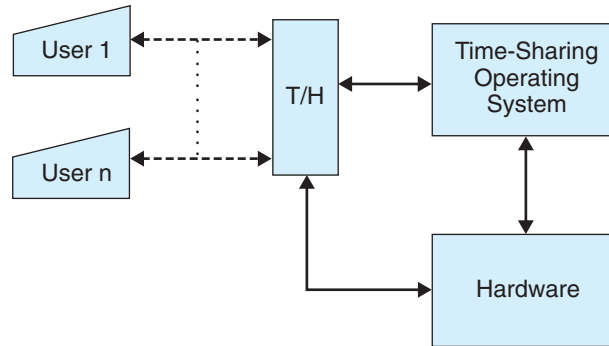
Figure 1.4   General structure of a time-sharing system.

- *Application-dependent*, which are designed for a special purpose. An example of this type is a real-time operating system that controls a power plant for the generation of electricity.

Systems can also be classified as being single user or multiuser. In single-user systems, the single user has access to all resources of the system all the time. In multiuser systems, several users request access to the resources of the system simultaneously.

The term *job* refers to a unit of work submitted by a user to the operating system. A typical job consists of the following parts:

- A sequence of commands to the operating system

- A program either in a source language or in binary form

- A set of input data used by the program when it executes

The sequence of commands in a job includes commands for compiling the source program provided (with an appropriate compiler), linking the program with appropriate libraries, loading the linked program, and executing the program with the input data provided. The term *job* was used in the early batch operating systems. The term *process* basically refers to an execution instance of a program.

1.4   Small and Specialized Operating Systems

A mobile operating system, also known as a mobile OS, a mobile platform, or a handheld operating system, is the operating system that controls a mobile device. Compared to the standard general-purpose operating systems, mobile operating systems are currently somewhat simpler, and focus on wireless versions of broadband and local connectivity, mobile multimedia formats, and different input methods.

Mobile operating systems that can be found on smartphones include Nokia's Symbian OS, Apple's IOS, RIM's BlackBerry OS, Windows Phone, Linux, Palm WebOS, Google's Android, and Nokia's Maemo. Android, WebOS, and Maemo are in turn built on top of Linux, and the iPhone OS is derived from the BSD and NeXTSTEP operating systems, which are all related to Unix.

An embedded operating system is an operating system for embedded computer systems. These operating systems are designed to be very compact and efficient, with similar functions that non-embedded computer operating systems provide, but which are more specialized depending on the applications they run. Most embedded operating systems are real-time operating systems.

Examples of systems that use embedded operating systems include: Automated Teller Machines, cash registers, CCTV systems, a TV box set, a GPS, jukeboxes, and missiles.

1.5   Brief History of Operating Systems

The development of operating systems can be summarized as follows:

1. Early computer systems had no operating systems and the programs had to directly control all necessary hardware components.

2. The first types of operating systems were the simple batch operating systems in which users had to submit their jobs on punched cards or tape. The computer operator grouped all the jobs in batches and stacked these on the main input device, which was a fast card reader or tape reader.

   One of the most important concepts for these systems was automatic job sequencing. An important performance metric for these systems was the CPU idle time, which affected the CPU utilization. The most important performance metric from the user point of view was the turnaround time, which is the interval from the submission of a job until the output was generated.

3. The next types of operating systems developed were batch systems with multiprogramming. These systems were capable of handling several programs active in memory at any time and required more advanced memory management. When a program stopped to wait for I/O in these systems, the OS was able to switch very rapidly from the currently executing program to the next. The short interval was called *context switch time*. Multiprogramming normally improves the processor and device utilization.

4. Time-sharing operating systems were the next important generation of systems developed. The most significant advantage of these systems was the capability to provide interactive computing to the users connected to the system. The basic technique employed on these systems was that the processor's time was uniformly shared by the user programs. The operating system provided CPU service during a small and fixed interval to a program and then switched to the next program.

5. Variations of multiprogramming operating systems were developed with more advanced techniques. These included improved hardware support for interrupt mechanisms and allowed implementation of better scheduling techniques based on priorities. Real-time operating systems were developed.

6. Advances in hardware and memory management techniques allowed development of operating systems with newer and more powerful features, such as paging and virtual memory, multi-level cache, and others.

Most of the development of modern operating systems has focused on networking, distribution, reliability, protection, and security. Several widely used operating systems are available today:

- Microsoft Windows, a family of systems that includes 98, Me, CE, 2000, XP, Vista, Windows 7, and others

- Linux (Linus Torvalds, FSF GNU)

- OSF-1 (OSF, DEC)

- Solaris (Sun Microsystems)

- IRIX (Silicon Graphics)

- OS2 (IBM)

- OS/390 (IBM)

- VMS (Dec/Compaq/HP)

- MacOS X (Apple)

1.6   Contemporary Operating Systems

The three most widely used operating systems are Linux, Unix, and Windows. These actually represent two different families of operating systems that have evolved over the years: Unix/Linux and Microsoft Windows.

1.6.1   Unix

Unix was originally introduced in 1974 by Dennis Ritchie and Ken Thompson while working at AT&T Bell Labs. The operating system was developed on a small computer and had two design goals: small size of the software system and portability. By 1980, many of the users were universities and research labs. Two versions became the best known systems: System V Unix (AT&T) and BSD Unix (University of California at Berkeley).

Unix became the dominant time-sharing OS used in small and large computers. It is one of the first operating systems written almost entirely in C, a high-level programming language.

The Unix family includes Linux, which was primarily designed and first implemented by Linus Torvals and other collaborators in 1991. Torvals released the source code on the Internet and invited designers and programmers to contribute their modifications and enhancements. Today Linux is freely available and has been implemented in many small and large computers. This operating system has also become important in commercial software systems.

1.6.2   Microsoft Windows

Windows operating systems are considered the most widely used family of operating systems released by Microsoft. These systems were developed for personal computers that use the Intel microprocessors. This group of operating systems consists of Windows 95, 98, ME, and CE, which can be considered the smaller members of the family. The larger and more powerful members of the Windows family of operating systems are Windows NT, 2000, XP, Vista, and Windows 7.

The larger Windows operating systems include a comprehensive security model, more functionality facility for network support, improved and more powerful virtual memory management, and full implementation of the Win32 API functions.

1.6.3   Mac OS

Mac OS X is an operating system that is really a very good development platform. Mac OS X is based on BSD (Berkeley System Distribution) UNIX and provides support for many POSIX, Linux, and System V APIs. Apple integrated the widely used FreeBSD 5 UNIX distribution with the Mach 3.0 microkernel.

This OS supports multiple development technologies including UNIX, Java, the proprietary Cocoa and Carbon runtime environments, and several open source, web, scripting, database, and development technologies. MacOS X provides several runtime environments integrated under a single desktop environment. The system provides an object-oriented application framework, procedural APIs, a highly-optimized and tightly integrated implementation of Java Standard Edition (JSE), BSD UNIX APIs and libraries, and X11.

1.7   64-bit Operating Systems

These are operating systems for 64-bit processors and systems with 64-bit computer architecture. They appeared more than three decades ago for the large mainframes of the '70s. Less than 10 years ago, microprocessors with 64-bit architecture were introduced.

1.7.1   Microsoft 64-bit Windows 7

With Windows 7 operating system, more computers are able to use the 64-bit edition. One of the important benefits of running a 64-bit OS is that it can address more memory. A 32-bit version of Windows can address only up to 4GB of RAM, and effectively use only around 3GB, since that extra gigabyte is reserved. A 64-bit OS can theoretically address around 17 billion gigabytes of RAM. In practice, the more expensive and advanced versions of 64-bit Windows 7 can handle up to 192GB, while the lower-end editions are more limited.

Of course, 64-bit operating systems still pose some challenges. The 64-bit flavors of Windows 7 and Vista need specific hardware drivers written for them—their 32-bit counterparts won't work. And although manufacturers have been developing 64-bit drivers for their newer peripherals, users with older printers, scanners, and other hardware face a tougher time trying to find 64-bit drivers. Microsoft's Windows 7 Compatibility page lets you browse or search for different hardware and software to determine whether it will run under 64-bit Windows.

1.7.2   Mac OS X

Mac OS X is considered not only the world's most advanced operating system but also extremely secure, compatible, and easy to use. Mac OS X Snow Leopard incorporates new technologies that offer immediate improvements while also smartly setting it up for the future.

64-bit systems provide all users the tools to apply the power of 64-bit to speed up everything from everyday applications to the most demanding scientific computations. Although Mac OS X is already 64-bit capable in many ways, Snow Leopard takes the next big step by rewriting nearly all system applications in 64-bit code and by enabling the Mac to address massive amounts of memory. Mac OS X is faster, more secure, and completely ready to support future applications.

1.8   Summary

The two basic components of a computer system are the hardware and the software components. An operating system is a large and complex software component of a computer system that controls the major operations of user programs and manages all the hardware resources. The main type of system studied in this book is the general-purpose operating system. This chapter has presented the fundamental concepts of operating systems. Discussions on the general structure, user interfaces, and functions of an operating system have also been presented. Two examples of well-known operating systems are Linux and Microsoft Windows.

This chapter also described various views of the operating system, the high-level external view, also called the abstract machine view, and the low-level or internal view. The abstract machine view is seen by the programs that request some type of service

from the operating system. The internal details of the OS are hidden below the interfaces. The internal view includes the components that represent the modules of the OS for carrying out various functions: the process manager, memory manager, resource manager, file manager, and device manager. The process manager is considered the most important component of the OS.

Key Terms		
abstract view	file manager	OS interface
access time	hardware component	process
application software	input	process manager
batch system	interactive system	processing
command	job	real-time system
computer system	kernel	software components
context switching	Linux	system software
CPU	Mac OS	Unix
external view	memory manager	Windows

Exercises and Questions

1. Explain the main facilities provided by an operating system. Give examples.

2. What type of computing do you normally use with your personal computer: batch or interactive? Explain and give examples.

3. Explain when you would prefer time-sharing processing instead of batch processing.

4. Describe the general type of work that you regularly do with Windows. Explain the type or level of operating system interface that you normally use. Describe any other possible interface level that you could use with Windows.

5. Get a Unix/Linux account and log in to the system; start experimenting with a small set of commands. Get acquainted with the system's prompt, which is normally a dollar sign. Investigate the use and purpose of the following commands: `who`, `cd`, `pwd`, `ls`, `finger`, `mkdir`, `cp`, `more`, `cat`, and `pico`.

6. From the user point of view, list and explain the main differences between Windows and Linux.

7. From your point of view, what are the main limitations or disadvantages of Windows?

8. From your point of view, what are the main limitations or disadvantages of Linux?

9. Search the Internet and, from the user point of view, list and explain the main differences between Unix and Linux.

10. Explain the main advantages and disadvantages of using internal and external views for the operations of a computer system, such as the one presented in this chapter. Give good arguments.

11. Log in to Linux. At the command prompt, type `man ps` to read the online manual documentation for the `ps` command. After reading this documentation, type `ps`. How many programs are active? How are they identified? Are these all the programs in the OS? Investigate with the options in the `ps` command.

12. Search the Internet and write a brief report on the differences between Windows 7 Home Edition and the Professional Edition.

13. Search the Internet to find data on the commercial support and commercial applications developed for Linux.