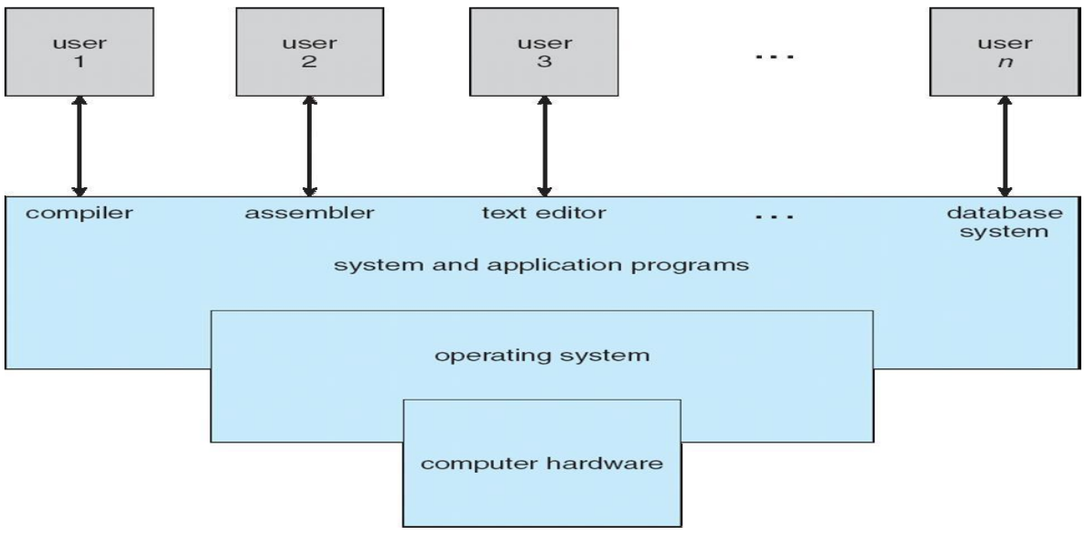
Introduction to Operating Systems

An **operating system** is a program that manages the computer hardware. It also acts as an intermediary between the computer user and the computer hardware. Mainframe operating systems are designed primarily to optimize utilization of hardware. Personal computer (PC) operating systems support complex games, business applications, and everything in between. Operating systems for handheld computers are designed to provide an environment in which a
user can easily interface with the computer to execute programs. Thus, some operating systems are designed to be *convenient,* others to be *efficient,* and others some combination of the two.

1.1 What Operating Systems Do

We begin our discussion by looking at the operating system's role in theoverall computer system. A computer system can be divided roughly intofour components: the *hardware,* the *operating system,* the *application programs,*and the *users*



Abstract view of the components of a computer system.

The **hardware—**the **central processing unit (CPU),** the **memory,** and the**input/output (I/O) devices**—provides the basic computing resources for thesystem. The **application programs**—such as word processors, spreadsheets,compilers, and web browsers—define the ways in which these resources areused to solve users' computing problems. The operating system controls andcoordinates the use of the hardware among the various application programs
for the various users.

We can also view a computer system as consisting of hardware, software,and data. The operating system provides the means for proper use of theseresources in the operation of the computer system.To understand more fully the operating system's role, we next explore
operating systems from two viewpoints: that of the user and that of the system.

User View

The user's view of the computer varies according to the interface being used.

Most computer users sit in front of a PC, consisting of a monitor, keyboard, mouse, and system unit. Such a system is designed for one user to monopolize its resources. The goal is to maximize the work (or play) that the user is performing. In this case, the operating system is designed mostly for ease of **use,** with some attention paid to performance

In other cases, a user sits at a terminal connected to a **mainframe** or **minicomputer.** Other users are accessing the same computer through other terminals. These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization to assure

that all available CPU time, memory, and I/O are used efficiently and that no individual user takes more than her fair share.

In still other cases, users sit at **workstations** connected to networks of other workstations and **servers.** These users have dedicated resources at their disposal, but they also share resources such as networking and servers—file, compute, and print servers.

System View

From the computer's point of view, the operating system is the program most intimately involved with the hardware. In this context, we can view an operating system as a **resource allocator.** A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on. The operating system acts as the manager of these resources.

An operating system is a control program. A **control program** manages the execution of user programs to prevent errors and improper use of the computer. It is especially concerned with the operation and control of I/O devices.

2. Storage Structure

Computer programs must be in main memory (also called **random-access memory** or **RAM)** to be executed. Main memory is the only large storage area (millions to billions of bytes) that the processor can access directly. It commonly is implemented in a semiconductor technology called **dynamic random-access memory (DRAM),** which forms an array of memory words. Each word has its own address. Interaction is achieved through a sequence of load or store instructions to specific memory addresses. The load instruction moves a word from main memory to an internal register within the CPU, whereas the store instruction moves the content of a register to main memory. Aside from explicit loads and stores, the CPU automatically loads instructions from main memory for execution.

A typical instruction-execution cycle, as executed on a system with a **von Neumann** architecture, first fetches an instruction from memory and stores that instruction in the **instruction register.** The instruction is then decoded and may cause operands to be fetched from memory and stored in some internal register. After the instruction on the operands has been executed, the result may be stored back in memory. Notice that the memory unit sees only a stream of memory addresses; it does not know how they are generated (by the instruction counter, indexing, indirection, literal addresses, or some other means) or what they are for (instructions or data). Accordingly, we can ignore *hoio* a memory address is generated by a program. We are interested only in the sequence of memory addresses generated by the running program. Ideally, we want the programs and data to reside in main memory permanently. This arrangement usually is not possible for the following tworeasons:

1. Main memory is usually too small to store all needed programs and data permanently.
2. Main memory is a *volatile* storage device that loses its contents when power is turned off or otherwise lost.

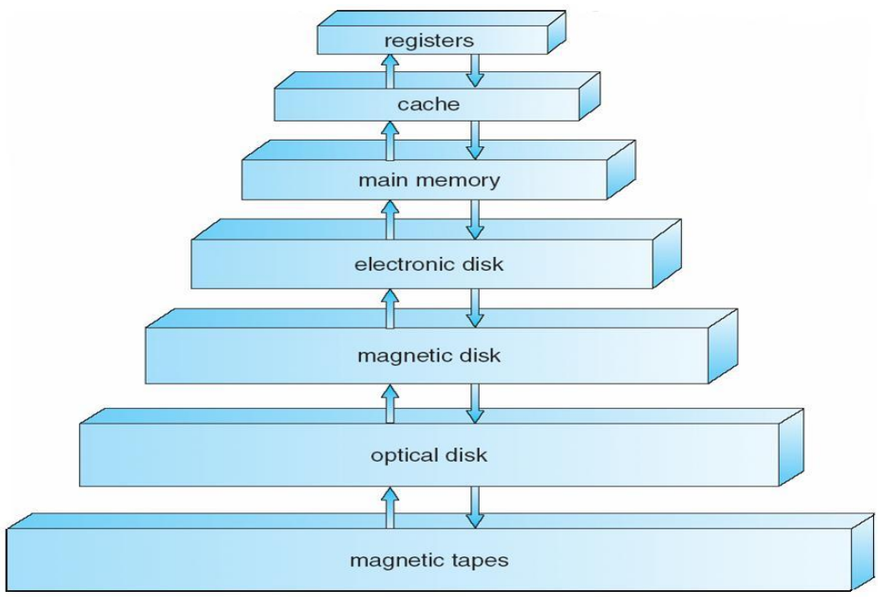
Thus, most computer systems provide **secondary storage** as an extension of main memory. The main requirement for secondary storage is that it be able to hold large quantities of data permanently.

The most common secondary-storage device is a **magnetic disk,** which provides storage for both programs and data. Most programs (web browsers, compilers, word processors, spreadsheets, and so on) are stored on a disk until they are loaded into memory. Many programs then use the disk as both a source and a destination of the information for their processing

In a larger sense, however, the storage structure that we have described consisting of registers, main memory, and magnetic disks—is only one of many possible storage systems. Others include cache memory, CD-ROM, magnetic tapes, and so on. Each storage system provides the basic functions of

storing a datum and of holding that datum until it is retrieved at a later time. The main differences among the various storage systems lie in speed, cost, size,and volatility.

The wide variety of storage systems in a computer system can be organized in a hierarchy according to speed and cost. The higher levels are expensive, but they are fast. As we move down the hierarchy, the cost per bit generally decreases, whereas the access time generally increases. This trade-off is reasonable; if a given storage system were both faster and less expensive than another—other properties being the same—then there would be no reason to use the slower, more expensive memory. In fact, many early storage devices, including paper tape and core memories, are relegated to museums now that magnetic tape and **semiconductor memory** have become faster and cheaper. The top four levels of memory in Figure 1.4 may be constructed using semiconductor memory..



In addition to differing in speed and cost, the various storage systems are either volatile or nonvolatile. As mentioned earlier, **volatile storage** loses its contents when the power to the device is removed. In the absence of expensive battery and generator backup systems, data must be written to **nonvolatile storage** for safekeeping. In the hierarchy shown in Figure the storage systems above the electronic disk are volatile, whereas those below are nonvolatile. An **electronic disk** can be designed to be either volatile or nonvolatile. During normal operation, the electronic disk stores data in a large DRAM array, which is volatile. But many electronic-disk devices contain a hidden magnetic hard disk and a battery for backup power. If external power is interrupted, the electronic-disk controller copies the data from RAM to the magnetic disk. When external power is restored, the controller copies the data back into the RAM. Another form of electronic disk is flash memory, which is popular in cameras and **personal digital assistants (PDAs),** in robots, and increasingly as removable storage on general-purpose computers. Flashmemory is slower than DRAM but needs no power to retain its contents. Another form of nonvolatile storage is NVRAM, which is DRAM with battery backup power. This memory can be as fast as DRAM but has a limited duration in which it is nonvolatile.

3. Computer-System Architecture

A computer system may be organized in a number of different ways, which we can categorize roughly according to the number of general-purpose processors used.

3.1. Single-Processor Systems

Most systems use a single processor. On a single-processor system, there is one main CPU capableof executing a general-purpose instruction set, including instructions fromuser processes. Almost all systems have other special-purpose processors aswell. They may come in the form of device-specific

processors, such as disk,keyboard, and graphics controllers; or, on mainframes, they may come in theform of more general-purpose processors, such as I/O processors that movedata rapidly among the components of the system.

For example, a disk-controller microprocessorreceives a sequence of requests from the main CPU and implements its own diskqueue and scheduling algorithm. This arrangement relieves the main CPU ofthe overhead of disk scheduling. PCs contain a microprocessor in the keyboardto convert the keystrokes into codes to be sent to the CPU. The use of special-purposemicroprocessors is common and does not turn a single-processor system intoa multiprocessor.

3.2. Multiprocessor Systems

Although single-processor systems are most common, **multiprocessor systems**(also known as **parallel systems** or **tightly coupled systems)** are growingin importance. Such systems have two or more processors in close communication,sharing the computer bus and sometimes the clock, memory, andperipheral devices.Multiprocessor systems have three main advantages:

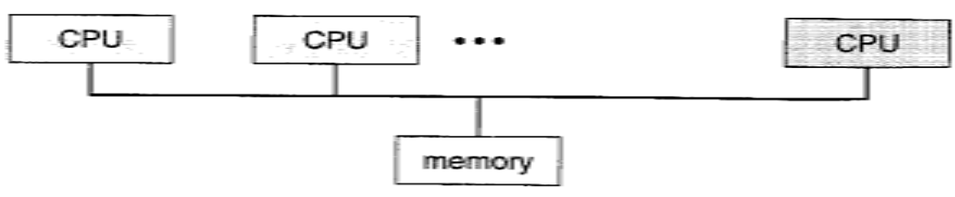
1. Increased throughput. By increasing the number of processors, we expectto get more work done in less time. The speed-up ratio with *N* processorsis not *N*, however; rather, it is less than N. When multiple processorscooperate on a task, a certain amount of overhead is incurred in keepingall the parts working correctly. This overhead, plus contention for sharedresources, lowers the expected gain from additional processors.

2. Economy of scale. Multiprocessor systems can cost less than equivalentmultiple single-processor systems, because they can share peripherals,mass storage, and power supplies. If several programs operate on thesame set of data, it is cheaper to store those data on one disk and to haveall the processors share them than to have many computers with localdisks and many copies of the data.

3. **Increased reliability.** If functions can be distributed properly amongseveral processors, then the failure of one processor will not halt thesystem, only slow it down. If we have ten processors and one fails, theneach of the remaining nine processors can pick up a share of the work ofthe failed processor. Thus, the entire system runs only 10 percent slower,rather than failing altogether.

The multiple-processor systems in use today are of two types. Somesystems use **asymmetric multiprocessing,** in which each processor is assigneda specific task. A master processor controls the system; the other processorseither look to the master for instruction or have predefined tasks. This schemedefines a master-slave relationship. The master processor schedules andallocates work to the slave processors.

The most common systems use **symmetric multiprocessing (SMP),** inwhich each processor performs all tasks within the operating system. SMPmeans that all processors are peers; no master-slave relationship existsbetween processors. Figure illustrates a typical SMP architecture. An example of the SMP system is Solaris.



Symmetric multiprocessing architecture

The benefit of this model is that many processescan run simultaneously—*N* processes can run if there are *N* CPUs—withoutcausing a significant deterioration of performance. However, we must carefullycontrol I/O to ensure that the data reach the appropriate processor. Also, sincethe CPUs are separate, one may be sitting idle while another is overloaded,resulting in inefficiencies. These

inefficiencies can be avoided if the processorsshare certain data structures. A multiprocessor system of this form will allowprocesses and resources—such as memory—to be shared dynamically amongthe various processors and can lower the variance among the processors. Allmodern operating systems—including Windows, Windows XP, Mac OS X, andLinux—now provide support for SMP.

The difference between symmetric and asymmetric multiprocessing mayresult from either hardware or software. Special hardware can differentiate themultiple processors, or the software can be written to allow only one master andmultiple slaves.

3.3. Clustered Systems

Another type of multiple-CPU system is the **clustered system.** Like multiprocessorsystems, clustered systems gather together multiple CPUs to accomplishcomputational work. Clustered systems differ from multiprocessor systems,however, in that they are composed of two or more individual systemscoupled together. The definition of the term *clustered* is not concrete. The generally accepted definition is that clusteredcomputers share storage and are closely linked via a **local-area network (LAN)**Clustering is usually used to provide **high-availability** service; that is,service will continue even if one or more systems in the cluster fail.

Clustering can be structured asymmetrically or symmetrically.

In **asymmetric clustering,** one machine is in **hot-standby mode** while the other is running the applications. The hot-standby host machine does nothing but monitor the active server. If that server fails, the hot-standby host becomes the active server.

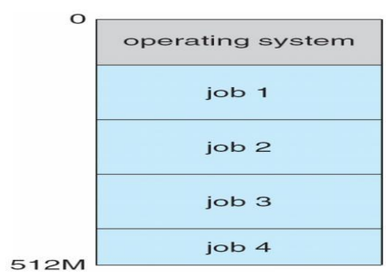
In **symmetric mode,** two or more hosts are running applications, and are monitoring each other. This mode is obviously more efficient, as it uses all of the available hardware. It does require that more than one application be available to run.

Other forms of clusters include parallel clusters and clustering over a wide-area network (WAN) .Parallel clusters allow multiple hosts to access the same data on the shared storage.

4 Operating-System Structure

An operating system provides the environment within which programs areexecuted. One of the most important aspects of operating systems is the ability tomultiprogram. A single user cannot, in general, keep either the CPU or theI/O devices busy at all times. **Multiprogramming** increases CPU utilization byorganizing jobs (code and data) so that the CPU always has one to execute.

The idea is as follows: The operating system keeps several jobs in memorysimultaneously (Figure 1.7). This set of jobs can be a subset of the jobs kept inthe job pool—which contains all jobs that enter the system—since the numberof jobs that can be kept simultaneously in memory is usually smaller thanthe number of jobs that can be kept in the job pool. The operating systempicks and begins to execute one of the jobs in memory. Eventually, the jobmay have to wait for some task, such as an I/O operation, to complete.



Memory layout for a multiprogramming system.

In a non-multiprogrammed system, the CPU would sit idle. In a multiprogrammedsystem, the operating system simply switches to, and executes, another job.When *that* job needs to wait, the CPU is switched to *another* job, and so on.Eventually, the first job finishes waiting and gets the CPU back. As long as atleast one job needs to execute, the CPU is never idle.

Multiprogrammed systems provide an environment in which the varioussystem resources (for example, CPU, memory, and peripheral devices) areutilized effectively, but they do not provide for user interaction with thecomputer system.

Time sharing (or **multitasking)** is a logical extension ofmultiprogramming. In time-sharing systems, the CPU executes multiple jobsby switching among them, but the switches occur so frequently that the userscan interact with each program while it is running.Time sharing requires an **interactive** (or **hands-on) computer system,**which provides direct communication between the user and the system. Theuser gives instructions to the operating system or to a program directly, using ainput device such as a keyboard or a mouse, and waits for immediate results onan output device. Accordingly, the **response time** should be short—typicallyless than one second.

A time-shared operating system allows many users to share the computersimultaneously. Since each action or command in a time-shared system tendsto be short, only a little CPU time is needed for each user. As the system switchesrapidly from one user to the next, each user is given the impression that theentire computer system is dedicated to his use, even though it is being sharedamong many users.

Time-sharing and multiprogramming require several jobs to be keptsimultaneously in memory. Since in general main memory is too small toaccommodate all jobs, the jobs are kept initially on the disk in the **job pool.**This pool consists of all processes residing on disk awaiting allocation of mainmemory. If several jobs are ready to be brought into memory, and if there isnot enough room for all of them, then the system must choose among them.Making this decision is **job scheduling,** Whenthe operating system selects a job from the job pool, it loads that job intomemory for execution. if several jobs are ready to run at the same time, the systemmust choose among them. Making this decision is **CPU scheduling.** Finally, running multiple jobs concurrently requiresthat their ability to affect one another be limited in all phases of the operatingsystem, including process scheduling, disk storage, and memory management.

5 Operating-System Operations

As mentioned earlier, modern operating systems are **interrupt driven.** If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an operating system will sit quietly, waiting for something to happen. Events are almost always signaled by the occurrence of an interrupt or a trap. A **trap (or** an **exception)** is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed. The interrupt-driven nature of an operating system defines that system's general structure. For each type of interrupt, separate segments of code in the operating system determine what action should be taken. An interrupt service routine is provided that is responsible for dealing with the interrupt.

Since the operating system and the users share the hardware and software resources of the computer system, With sharing, many processes could be adversely affected by a bug in one program. For example, if a process gets stuck in an infinite loop, this loop could prevent the correct operation of many other processes. More subtle errors can occur in a multiprogramming system, where one erroneous program might modify another program, the data of another program, or even the operating system itself.

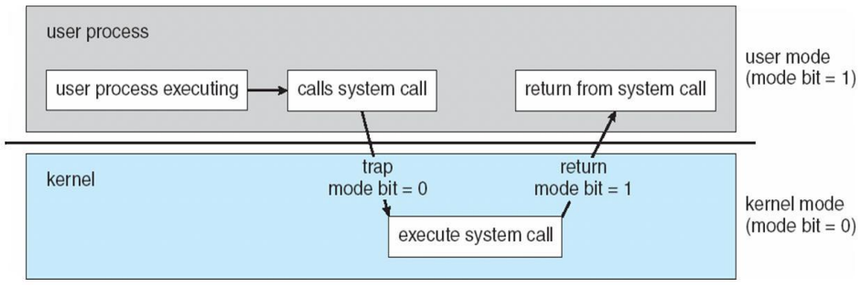
A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other programs to execute incorrectly.

5.1 Dual-Mode Operation

In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and userdefined code. The approach taken by most

computer systems is to provide hardware support that allows us to differentiate among various modes of execution.

At the very least, we need two separate **modes** of operation: **user mode**and **kernel mode** (also called **supervisor mode, system mode, or privilegedmode).** A bit, called the **mode bit,** is added to the hardware of the computer toindicate the current mode: kernel (0) or user (1). With the mode bit, we are ableto distinguish between a task that is executed on behalf of the operating systemand one that is executed on behalf of the user. When the computer system isexecuting on behalf of a user application, the system is in user mode. However,when a user application requests a service from the operating system (via asystem call), it must transition from user to kernel mode to fulfill the request.



This is shown in Figure. As we shall see, At system boot time, the hardware starts in kernel mode. The operatingsystem is then loaded and starts user applications in user mode. Whenever atrap or interrupt occurs, the hardware switches from user mode to kernel mode
(that is, changes the state of the mode bit to 0). Thus, whenever the operatingsystem gains control of the computer, it is in kernel mode. The system alwaysswitches to user mode (by setting the mode bit to 1) before passing control toa user program.

The dual mode of operation provides us with the means for protecting theoperating system from errant users—and errant users from one another. Weaccomplish this protection by designating some of the machine instructions that may cause harm as **privileged instructions.** The hardware allows privileged instructions to be executed only in kernel mode. If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.

5.2 Timer

We must prevent a user program from getting stuck in an infinite loop or not calling system services and never returning control to the operating system.To accomplish this goal, we can use a **timer.** A timer can be set to interrupt the computer after a specified period. The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second). A **variable timer** is generally implemented by a fixed-rate clock and a counter. The operating system sets the counter. Every time the clock ticks, the counter is decremented. When the counter reaches 0, an interrupt occurs

Before turning over control to the user, the operating system ensures that the timer is set to interrupt. If the timer interrupts, control transfers automatically to the operating system, which may treat the interrupt as a fatal error or may give the program more time. Clearly, instructions that modify thecontent of the timer are privileged.Thus, we can use the timer to prevent a user program from running toolong.

6 OPERATING SYSTEM FUNCTIONS

Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a *passive entity*, process is an *active entity*.
- Process needs resources to accomplish its task
- CPU, memory, I/O, files

- Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
- Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
- Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

- The operating system is responsible for the following activities in connection with process management:
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
- Optimizing CPU utilization and computer response to users
- **Memory management activities**
- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes (or parts thereof) and data to move into and out of memory
- Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
- Abstracts physical properties to logical storage unit - **file**
- Each medium is controlled by device (i.e., disk drive, tape drive)
- Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
- Files usually organized into directories
- Access control on most systems to determine who can access what
- **OS activities include**
- Creating and deleting files and directories
- Primitives to manipulate files and dirs
- Mapping files onto secondary storage
- Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a –long‖ period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- **MASS STORAGE activities**
- Free-space management
- Storage allocation
- Disk scheduling
- Some storage need not be fast
- Tertiary storage includes optical storage, magnetic tape
- Still must be managed
- Varies between WORM (write-once, read-many-times) and RW (read-write)

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
- Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
- General device-driver interface
- Drivers for specific hardware devices

Protection and Security

Protection – any mechanism for controlling access of processes or users to resources defined by the OS

Security – defense of the system against internal and external attacks

- Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
- User identities (**user IDs**, security IDs) include name and associated number, one per user
- User ID then associated with all files, processes of that user to determine access control
- Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
- **Privilege escalation** allows user to change to effective ID with more rights

7 . Protection and Security

If a computer system has multiple users and allows the concurrent execution of multiple rocesses, then access to data must be regulated. For that purpose, mechanisms ensure that files, memory segments, CPU, and other resources can be operated on by only those processes that have gained proper authorization from the operating system.

Protection, then, is any mechanism for controlling the access of processes or users to the resources defined by a computer system. This mechanism must provide means for specification of the controls to be imposed and means for enforcement. Protection can improve reliability by detecting latent errors at the interfaces between component subsystems.

A system can have adequate protection but still be prone to failure and allow inappropriate access. Consider a user whose authentication information (her means of identifying herself to the system) is stolen. Her data could be copied or deleted, even though file and memory protection are working. It is the job of **security** to defend a system from external and internal attacks. Such attacks spread across a

huge range and include viruses and worms, denial-ofservice attacks , identity theft, and theft of service (unauthorized use of a system).

8. Distributed Systems

A distributed system is a collection of physically separate, possibly heterogeneous computer systems that are networked to provide the users with access to the various resources that the system maintains. Access to a shared resource increases computation speed, functionality, data availability, and reliability.

A **network,** in the simplest terms, is a communication path between two or more systems. Distributed systems depend on networking for their functionality. Networks vary by the protocols used, the distances between nodes, and the transport media. TCP/IP is the most common network protocol,although ATM and other protocols are in widespread use.

Networks are characterized based on the distances between their nodes. A **local-area network (LAN)** connects computers within a room, a floor, or a building. A **wide-area network (WAN)** usually links buildings, cities, or countries. A global company may have a WAN to connect its offices worldwide. **metropolitan-area network** (MAN) could link buildings within a city. BlueTooth and 802.11 devices use wireless technology to communicate over a distance of several feet, in essence creating a **small-area network** such as might be found in a home.

The media to carry networks are equally varied. They include copper wires, fiber strands, and wireless transmissions between satellites, microwave dishes, and radios.

A **network operating system** is an operating system that provides features such as file sharing across the network and that includes a communication scheme that allows different processes on different computers to exchange messages. A computer running a network operating system acts autonomously from all other computers on the network, although it is aware of the network and is able to communicate with other networked computers.

9. Special-Purpose Systems

The discussion thus far has focused on general-purpose computer systems that we are all familiar with. There are, however, different classes of computer systems whose functions are more limited and whose objective is to deal with limited computation domains.

9.1 Real-Time Embedded Systems

Embedded computers are the most prevalent form of computers in existence. These devices are found everywhere, from car engines and manufacturing robots to VCRs and microwave ovens. They tend to have very specific tasks. The use of embedded systems continues to expand.

Embedded systems almost always run **real-time operating systems.** A real-time system is used when rigid time requirements have been placed on the operation of a processor or the flow of data; thus, it is often used as a control device in a dedicated application. Sensors bring data to the computer. The computer must analyze the data and possibly adjust controls to modify the sensor inputs. Systems that control scientific experiments, medical imaging systems, industrial control systems, and certain display systems are realtime systems. Some automobile-engine fuel-injection systems, home-appliancecontrollers, and weapon systems are also real-time systems.

A real-time system has well-defined, fixed time constraints. Processing *mustbe* done within the defined constraints, or the system will fail. A real-time system functions correctly only if it returns the correct result within its time constraints. Contrast this system with a time-sharing system, where it is desirable (but not mandatory) to respond quickly, or a batch system, which may have no time constraints at all.

9.2 Multimedia Systems

Most operating systems are designed to handle conventional data such as text files, programs, word-processing documents, and spreadsheets. However, a recent trend in technology is the incorporation of **multimedia data** into computer systems. Multimedia data consist of audio and video files as well as conventional files. These data differ from conventional data in that multimedia data—such as frames of video—must be delivered (streamed) according to certain time restrictions (for example, 30 frames per second).

Multimedia describes a wide range of applications that are in popular use today. These include audio files such as MP3 DVD movies, video conferencing, and short video clips of movie previews or news stories downloaded over theInternet. Multimedia applications may also include live webcasts Multimedia applications need not be either audio or video; rather, amultimedia application often includes a combination of both

9.3 Handheld Systems

Handheld systems include personal digital assistants (PDAs), such as Palmand Pocket-PCs, and cellular telephones, many of which use special-purposeembedded operating systems. Developers of handheld systems and applicationsface many challenges, most of which are due to the limited size of suchdevices. For example, a PDA is typically about 5 inches in height and 3 inchesin width, and it weighs less than one-half pound. Because of their size, mosthandheld devices have a small amount of memory, slow processors, and smalldisplay screens.

The amount of physical memory in a handheld depends upon the device,but typically is is somewhere between 512 KB and 128 MB. (Contrast this with atypical PC or workstation, which may have several gigabytes of memory!)As a result, the operating system and applications must manage memoryefficiently.

A second issue of concern to developers of handheld devices is the speedof the processor used in the devices. Processors for most handheld devicesrun at a fraction of the speed of a processor in a PC. Faster processors requiremore power. To include a faster processor in a handheld device would requirea larger battery, which would take up more space and would have to bereplaced (or recharged) more frequently. Most handheld devices use smaller,slower processors that consume less power. Therefore, the operating systemand applications must be designed not to tax the processor.

The last issue confronting program designers for handheld devices is I/O.A lack of physical space limits input methods to small keyboards, handwritingrecognition, or small screen-based keyboards. The small display screens limitoutput options. Whereas a monitor for a home computer may measure up to30 inches, the display for a handheld device is often no more than 3 inchessquare. Familiar tasks, such as reading e-mail and browsing web pages, mustbe condensed into smaller displays. One approach for displaying the contentin web pages is **web clipping,** where only a small subset of a web page isdelivered and displayed on the handheld device.

10. Operating-System Services

An operating system provides certain services to the users of those programs.. These operating-system services are provided for the convenience of the programmer, to make the programming task easier.

One set of operating-system services provides functions that are helpful to the user.

• **User interface.** Almost all operating systems have a **user interface** (UI). This interface can take several forms. One is a **command-line interface (CLI),** which uses text commands and a method for entering them (say, a program to allow entering and editing of commands **graphical user interface (GUI)** is used.

Here, the interface is a window system with a pointing device to direct I/O, choose from menus, and make selections and a keyboard to enter text. Some systems provide two or all three of these variations.

• **Program execution.** The system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally (indicating error).

• **I/O operations.** A running program may require I/O, which may involve a file or an I/O device. For specific devices, special functions may be desired (such as recording to a CD or DVD drive or blanking a CRT screen). For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide a means to do I/O.

• **File-system manipulation.** The file system is of particular interest. Obviously, programs need to read and write files and directories. They also need to create and delete them by name, search for a given file, and list file information. Finally, some programs include permissions management to allow or deny access to files or directories based on file ownership.

• **Communications.** There are many circumstances in which one process needs to exchange information with another process. Such communication may occur between processes that are executing on the same computer or between processes that are executing on different computer systems tied together by a computer network. Communications may be implemented via *shared memory* or through *message passing,* in which packets of information are moved between processes by the operating system.

• **Error detection.** The operating system needs to be constantly aware of possible errors. Errors may occur in the CPU and memory hardware (such as a memory error or a power failure), in I/O devices (such as a parity error on tape, a connection failure on a network, or lack of paper in the printer), and in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location, or a too-great use of CPU time). For each type of error, the operating system should take the appropriate action to ensure correct and consistent computing. Debugging facilities can greatlyenhance the user's and programmer's abilities to use the system efficiently.

Another set of operating-system functions exists not for helping the user but rather for ensuring the efficient operation of the system itself. Systems with multiple users can gain efficiency by sharing the computer resources among the users.

Resource allocation. When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of Them. Many different types of resources are managed by the operating system. For instance, in determining how best to use the CPU, operating systems have CPU-scheduling routines that take into account the speed of the CPU, the jobs that must be executed, the number of registers available, and other factors. There may also be routines to allocate printers, modems, USB storage drives, and other peripheral devices.

Accounting. We want to keep track of which users use how much and what kinds of computer resources. This record keeping may be used for accounting (so that users can be billed) or simply for accumulating usage statistics. Usage statistics may be a valuable tool for researchers who wish to reconfigiire the system to improve computing services.

Protection and security. The owners of information stored in a multiuser or networked computer system may want to control use of that information. When several separate processes execute concurrently, it should not be possible for one process to interfere with the others or with the operating system itself. Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important. Such security starts with requiring each user to authenticate himself or herself to the system, usually by means of a password, to gain access to system resources. It extends to defending external I/O devices, including modems and network adapters, from invalid access attempts and to

recording all such connections for detection of break-ins. If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

11. System Calls

System calls provide an interface to the services made available by an operating system.

System calls can be grouped roughly into five major categories:

Processcontrol, file manipulation, device manipulation, information maintenance,and **communications.**

Most ofthese system calls support, or are supported by, concepts and functions thatare discussed in later chapters

. Process control

o end, abort

o load, execute

o create process, terminate process

o get process attributes, set process attributes

o wait for time

o wait event, signal event

o allocate and free memory

• File management

° create file, delete file

o open, close

° read, write, reposition

o get file attributes, set file attributes

• Device management

o request device, release device

° read, write, reposition

o get device attributes, set device attributes

° logically attach or detach devices

• Information maintenance

o get time or date, set time or date

o get system data, set system data

o get process, file, or device attributes

o set process, file, or device attributes

• Communications

0 create, delete communication connection

° send, receive messages

o transfer status information

o attach or detach remote devices

Process Control

A running program needs to be able to halt its execution either normally (end)or abnormally (abort). If a system call is made to terminate the currentlyrunning program abnormally, or if the program runs into a problem andcauses an error trap, and an errormessage generated.

File Management

We first need to be able to create and delete files. Either system call requires the name of the file and perhaps some of the file's attributes. Once the file is created, we need to open it and to use it. We may also read, write, or reposition (rewinding or skipping to the end of the file, for example). Finally, we need

to close the file, indicating that we are no longer using it. We may need these same sets of operations for directories if we have a

directory structure for organizing files in the file system. In addition, for either files or directories, we need to be able to determine the values of various attributes and perhaps to reset them if necessary. File attributes include the file name, a file type, protection codes, accounting information, and so on. At least two system calls, get file attribute and set file attribute, are required for this function. Some operating systems provide many more

calls, such as calls for file move and copy.

Device Management

A process may need several resources to execute—main memory, disk drives,access to files, and so on. If the resources are available, they can be granted,and control can be returned to the user process. Otherwise, the process willhave to wait until sufficient resources are available.

The various resources controlled by the operating sysstem can be thoughtof as devices. Some of these devices are physical devices (for example, tapes),while others can be thought of as abstract or virtual devices (for example,files). If there are multiple users of the system, the system may require us tofirst request the device, to ensure exclusive use of it. After we are finishedwith the device, we release it. These functions are similar to the open andclose system calls for files. Other operating systems allow unmanaged access

to devices.

Once the device has been requested (and allocated to us), we can read,write, and (possibly) reposition the device, just as we can with files. In fact,the similarity between I/O devices and files is so great that many operatingsystems, including UNIX, merge the two into a combined file-device structure.

Information Maintenance

Many system calls exist simply for the purpose of transferring informationbetween the user program and the operating system. For example, mostsystems have a system call to return the current t i m e and d a t e . Other systemcalls may return information about the system, such as the number of currentusers, the version number of the operating system, the amount of free memoryor disk space, and so on.

In addition, the operating system keeps information about all its processes,and svstem calls are used to access this information. Generally, calls arealso used to reset the process information (get process a t t r i b u t e s ands e t process a t t r i b u t e s) . In Section 3.1.3, we discuss what information isnormally kept.

Communication

There are two common models of interprocess communication:

the messagepassingmodel and the shared-memory model. In the message-passing model,the communicating processes exchange messages with one another to transferinformation. Messages can be exchanged between the processes either directlyor indirectly through a common mailbox. Before communication can takeplace, a connection must be opened. The name of the other communicator

must be known, be it another process on the same system or a process onanother computer connected by a communications network. Each computerin a network has a *host name* by which it is commonly known. A host alsohas a network identifier, such as an IP address. Similarly, each process hasa *process name*, and this name is translated into an identifier by which theoperating system can refer to the process. The get host id and get processed system calls do this translation. The identifiers are then passed to the generalpurposeopen and close calls provided by the file system or to specificopen connection and close

connection system calls, depending on thesystem's model of communication. The recipient process usually must give itspermission for communication to take place with an accept connection call.

In the shared-memory model, processes use shared memory create andshared memory attach system calls to create and gain access to regions ofmemory owned by other processes. Recall that, normally, the operating systemtries to prevent one process from accessing another process's memory. Sharedmemory requires that two or more processes agree to remove this restriction.They can then exchange information by reading and writing data in the sharedareas.

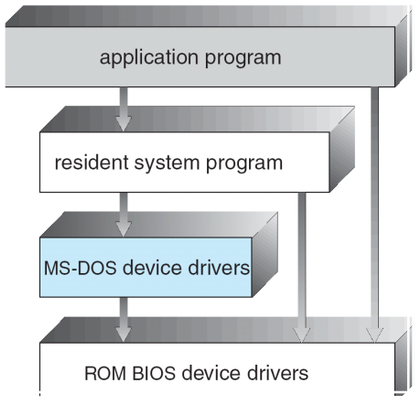
12. Operating-System Structure

A system as large and complex as a modern operating system must beengineered carefully if it is to function properly and be modified easily. Acommon approach is to partition the task into small components rather than

have one monolithic system.

12.1 Simple Structure

Many commercial systems do not have well-defined structures. Frequently,such operating systems started as small, simple, and limited systems and thengrew beyond their original scope. MS-DOS is an example of such a system. It wasoriginally designed and implemented by a few people who had no idea that itwould become so popular. It was written to provide the most functionality in the least space, so it was not divided into modules carefully.

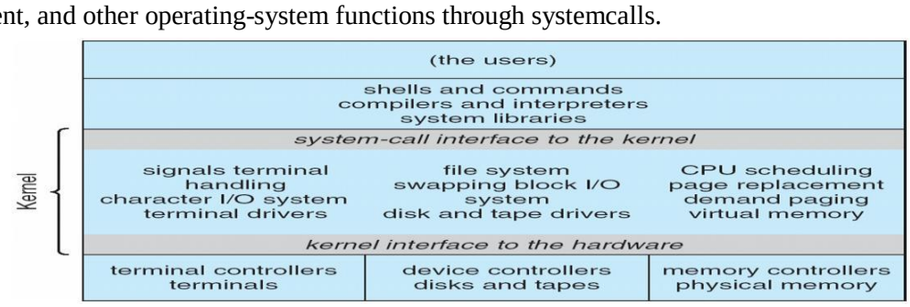


MS-DOS layer structure.

In MS-DOS, the interfaces and levels of functionality are not well separated.For instance, application programs are able to access the basic I/O routinesto write directly to the display and disk drives. Such freedom leaves MS-DOSvulnerable to errant (or malicious) programs, causing entire system crasheswhen user programs fail.

Another example of limited structuring is the original UNIX operatingsystem. UNIX is another system that initially was limited by hardware functionality.It consists of two separable parts: the kernel and the system programs.

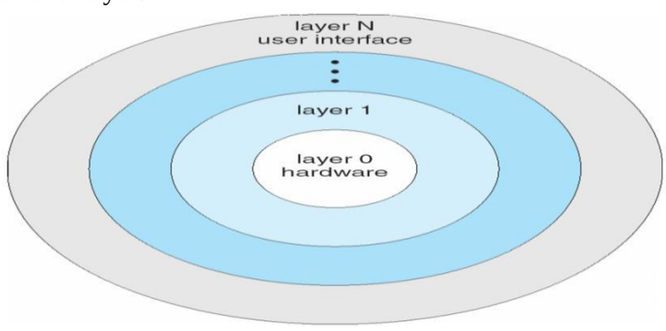
The kernel is further separated into a series of interfaces and device drivers,which have been added and expanded over the years as UNIX has evolved. Wecan view the traditional UNIX operating system as being layered, as shown inFigure. Everything below the system call interface and above the physicalhardware is the kernel. The kernel provides the file system, CPU scheduling,memory management, and other operating-system functions through systemcalls.



12.2 Layered Approach

With proper hardware support, operating systems can be broken into piecesthat are smaller and more appropriate than those allowed by the originalMS-DOS or UNIX systems. The operating system can then retain much greater
control over the computer and over the applications that make use of thatcomputer.

A system can be made modular in many ways. One method is the **layeredapproach,** in which the operating system is broken up into a number of layers(levels). The bottom layer (layer 0) is the hardware; the highest (layer N) is theuser interface. This layering structure is depicted in Figure.An operating-system layer is an implementation of an abstract object madeup of data and the operations that can manipulate those data. A typicaloperating-system layer—say, layer M—consists of data structures and a setof routines that can be invoked by higher-level layers. Layer *M*, in turn, can
invoke operations on lower-level layers.



A layered operating system.

The main advantage of the layered approach is simplicity of constructionand debugging. The layers are selected so that each uses functions (operations)and services of only lower-level layers. This approach simplifies debuggingand system verification. The first layer can be debugged without any concernfor the rest of the system, because, by definition, it uses only the basic hardware(which is assumed correct) to implement its functions. Once the first layer isdebugged, its correct functioning can be assumed while the second layer isdebugged, and so on. If an error is found during the debugging of a particularlayer, the error must be on that layer, because the layers below it are alreadydebugged. Thus, the design and implementation of the system is simplified.Each layer is implemented with only those operations provided by lowerlevellayers.

The major difficulty with the layered approach involves appropriatelydefining the various layers. Because a layer can use only lower-level layers,careful planning is necessary. For example, the device driver for the backingstore (disk space used by virtual-memory algorithms) must be at a lowerlevel than the memory-management routines, because memory managementrequires the ability to use the backing store.

A final problem with layered implementations is that they tend to be lessefficient than other types. For instance, when a user program executes an I/Ooperation, it executes a system call that is trapped to the I/O layer, which callsthe memory-management layer, which in turn calls the CPU-scheduling layer,which is then passed to the hardware. At each layer, the parameters may bemodified, data may need to be passed, and so on. Each layer adds overhead tothe system call; the net result is a system call that takes longer than does oneon a nonlayered system.

12.3 Microkernels

In the mid-1980s, researchers at Carnegie MellonUniversity developed an operating system called **Mach** that modularizedthe kernel using the **microkernel** approach. This method structures theoperating system by removing all nonessential components from the kernel andimplementing them as system and user-level programs. The result is a smallerkernel. There is little consensus regarding which services

should remain in thekernel and which should be implemented in user space. Typically, however,microkernels provide minimal process and memory management, in additionto a communication facility.

The main function of the microkernel is to provide a communication facilitybetween the client program and the various services that are also runningin user space. Communication is provided by *message passing*

For example, if the client program wishes to accessa file, it must interact with the file server. The client program and service neverinteract directly. Rather, they communicate indirectly by exchanging messageswith the microkernel.

One benefit of the microkernel approach is ease of extending the operating system. All new services are added to user space and consequently do notrequire modification of the kernel. When the kernel does have to be modified,the changes tend to be fewer, because the microkernel is a smaller kernel.The resulting operating system is easier to port from one hardware designto another.

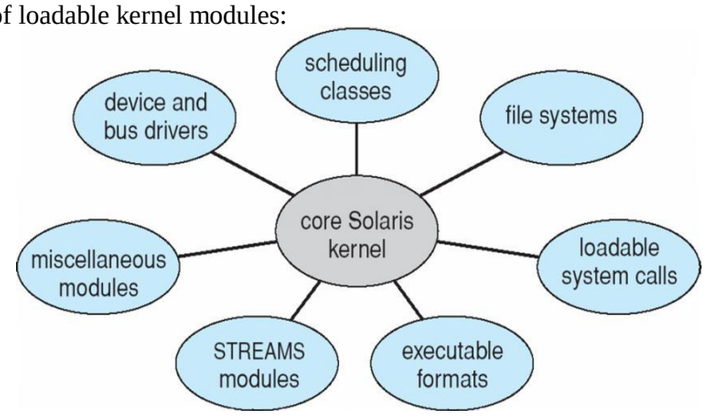
The microkernel also provides more security and reliability, sincemost services are running as user—rather than kernel—processes. If a servicefails, the rest of the operating system remains untouched.

Unfortunately, microkernels can suffer from performance decreases dueto increased system function overhead.

12.4 Modules

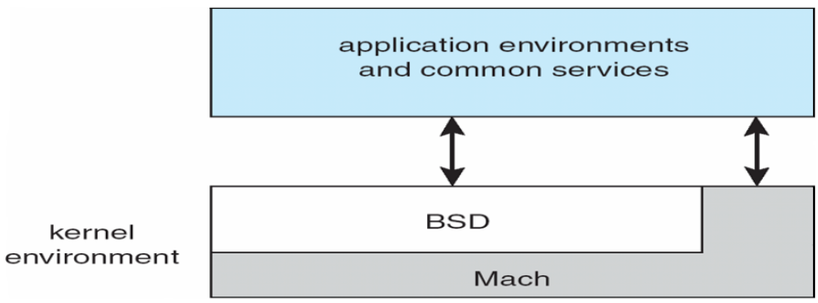
The best current methodology for operating-system design involvesusing object-oriented programming techniques to create a modular kernel.Here, the kernel has a set of core components and dynamically links inadditional services either during boot time or during run time. Such astrategy uses dynamically loadable modules and is common in modernimplementations of UNIX, such as Solaris, Linux, and Mac OS X. For example, theSolaris operating system structure is organized arounda core kernel with seven types of loadable kernel modules:

1. Scheduling classes
2. File systems



3. Loadable system calls
4. Executable formats
5. STREAMS modules
6. Miscellaneous
7. Device and bus drivers

Such a design allows the kernel to provide core services yet also allowscertain features to be implemented dynamically. For example, device andbus drivers for specific hardware can be added to the kernel, and supportfor different file systems can be added as loadable modules. The overallresult resembles a layered system in that each kernel section has defined,protected interfaces; but it is more flexible than a layered system in that anymodule can call any other module. Furthermore, the approach is like themicrokernel approach in that the primary module has only core functionsand knowledge of how to load and communicate with other modules; but itis more efficient, because modules do not need to invoke message passing inorder to communicate.



The Apple Macintosh Mac OS X operating system uses a hybrid structure.Mac OS X (also known as *Danvin)* structures the operating system using alayered technique where one layer consists of the Mach microkernel. Thestructure of Mac OS . The top layers include application environments and a set of servicesproviding a graphical interface to applications. Below these layers is the kernelenvironment, which consists primarily of the Mach microkernel and the BSDkernel. Mach provides memory management; support for remote procedurecalls (RPCs) and interprocess communication (IPC) facilities, including messagepassing; and thread scheduling. The BSD component provides a BSD commandline interface, support for networking and file systems. In addition to Mach and BSD, the kernelenvironment provides an I/O kit for development of device drivers anddynamically loadable modules (which Mac OS X refers to as **kernel extensions).**As shown in the figure, applications and common services can make use ofeither the Mach or BSD facilities directly

13. Operating-System Generation

It is possible to design, code, and implement an operating system specificallyfor one machine at one site. More commonly, however, operating systemsare designed to run on any of a class of machines at a variety of sites with
a variety of peripheral configurations.

The system must then be configuredor generated for each specific computer site, a process sometimes known as
system generation (SYSGEN).The operating system is normally distributed on disk or CD-ROM. Togenerate a system, we use a special program. The SYSGEN program reads froma given file, or asks the operator of the system for information concerning thespecific configuration of the hardware system, or probes the hardware directlyto determine what components are there. The following kinds of informationmust be determined.

• What CPU is to be used? What options (extended instruction sets, floatingpointarithmetic, and so on) are installed? For multiple CPU systems, eachCPU must be described.

• How much memory is available? Some systems will determine this valuethemselves by referencing memory location after memory location until an"illegal address" faul is generated. This procedure defines the final legaladdress and hence the amount of available memory.

• What devices are available? The system will need to know how to addresseach device (the device number), the device interrupt number, the device'stype and model, and any special device characteristics.

• What operating-system options are desired, or what parameter values areto be used? These options or values might include how many buffers ofwhich sizes should be used, what type of CPU-scheduling algorithm isdesired, what the maximum number of processes to be supported is, andso on.

Once this information is determined, it can be used in several ways. At one

extreme, a system administrator can use it to modify a copy of the source code ofthe operating system. The operating system then is completely compiled. Datadeclarations, initializations, and constants, along with conditional compilation,produce an output object version of the operating system that is tailored to thesystem described.