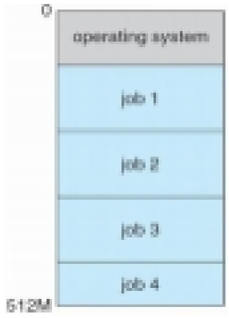
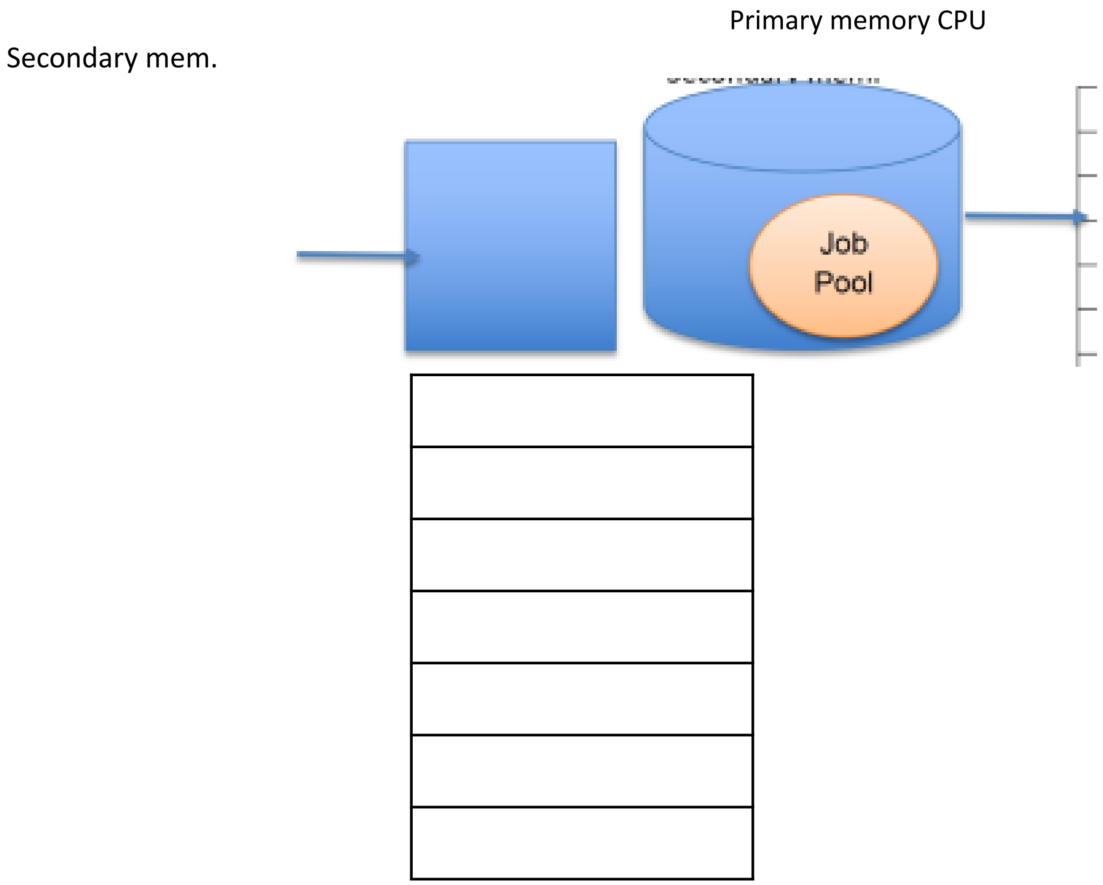
Operating-System Structure



One of the most important aspects of operating systems is the ability to multiprogram. Asingleuser cannot keep either the CPU or the I/O devices busy at all times. **Multiprogramming**increases CPU utilization by organizing jobs, so that the CPU always has onetoexecute.

The operating system keeps several jobs in memory simultaneouslyas shown in figure. This set of jobs is a subset of the jobs kept in the job pool. Sincethe number of jobs that can be kept simultaneously in memory is usuallysmallerthan the number of jobs that can be kept in the job pool(in secondary memory). The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/Ooperation, tocomplete. In a non-multiprogrammed system, the CPU would sit idle. Ina multiprogrammed system, the operating system simply switches to, andexecutes, another job. When *that* job needs to wait, the CPUis switchedto*another* job, and so on. Eventually, the first job finishes waiting andgetstheCPU back. Thus the CPU is never idle.



Multiprogrammed systems provide an environment in which the various systemresources (forexample, CPU, memory, and peripheral devices) are utilized effectively, but they do not providefor user interaction with the computer system.

In **Time sharing** (or **multitasking) systems,** a single CPU executes multiple jobsbyswitching among them, but the switches occur so frequently that the users can interact witheachprogram while it is running. The user feels that all the programs are being executed at thesametime. Time sharing requires an **interactive** (or **hands-on) computer system,** which providesdirect communication between the user and the system. The user gives instructions totheoperating system or to a program directly, using a input device such as a keyboard or a mouse, and waits for immediate results on an output device. Accordingly, the **response time** shouldbeshort—typically less than one second.

A time-shared operating system allows many users to share the computer simultaneously. As the system switches rapidly from one user to the next, each user is given the impressionthat the entire computer system is dedicated to his use only, even though it is being sharedamongmany users.

A **multiprocessor system** is a computer system having two or more CPUs withina singlecomputer system, each sharing main memory and peripherals. Multiple programs are executedby multiple processors parallel.

Distributed Systems

Individual systems that are connected and share the resource available in network is calledDistributed system. Access to a shared resource increases computation speed, functionality, dataavailability, and reliability.

A **network** is a communication path between two or more systems. Distributedsystemsdepend on networking for their functionality. Networks vary by the protocols used, the distancesbetween nodes, and the transport media. TCP/IP is the most common network protocol. Most operating systems support TCP/IP.

Networks are characterized based on the distances between their nodes. A**local-areanetwork (LAN)** connects computers within a room, a floor, or a building. A **wide-areanetwork(WAN)** usually links buildings, cities, or countries. A global company may have a WANtoconnect its offices worldwide. A **metropolitan-area network** (MAN) links buildings withinacity. A **small-area network** connects systems within a several feet using wireless technology. Eg. BlueTooth and 802.11.

The media to carry networks also vary - 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 filesharing across the network and that allows different processes on different computers to exchange messages.A computer running a network operating system acts autonomously fromall other computers on the network, although it is aware of the network and is able to communicate
with other networked computers.

Operating-System Operations

Modern operating systems are **interrupt driven.** If there are no processes to execute, noI/O devices to service, and no users to whom to respond, an operating systemwill sit quietly, waiting for something to happen. Events are signaled by the occurrence of an interrupt or a trap. A **trap (or** an **exception)** is a software-generated interrupt. For each type of interrupt, separatesegments 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.

a) Dual-Mode Operation

Since the operating system and the user programs share the hardware and software resourcesof the computer system, it has to be made sure that an error in a user programcannot causeproblems to other programs and the Operating System running in the system.

The approach taken is to use a hardware support that allows us to differentiate amongvariousmodes of execution.

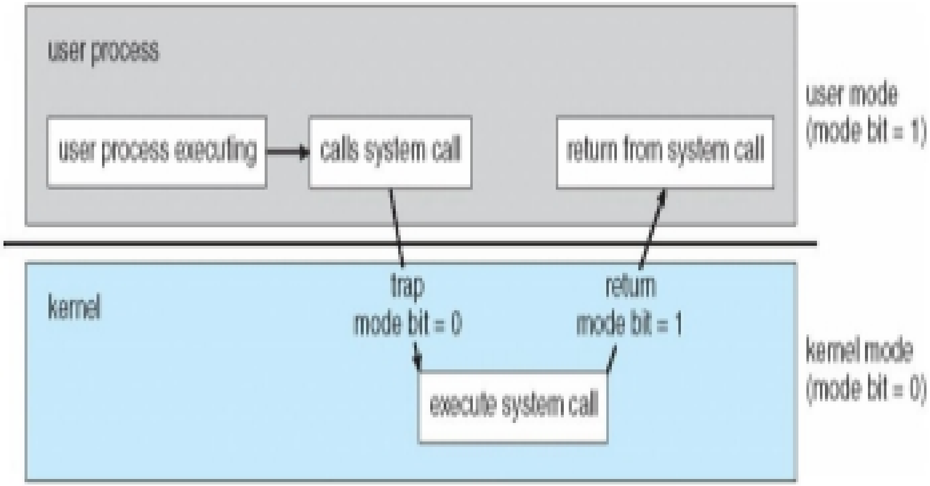
The system can be assumed to work in two separate **modes** of operation: • **user**

mode and

- **kernel mode (supervisor mode, system mode, or privileged mode).**

A hardware bit of the computer, called the **mode bit**, is used to indicate the current mode: kernel (0) or user (1). With the mode bit, we are able to distinguish between a task that is executedbythe operating system and one that is executed by the user.

When the computer system is executing a user application, the systemis in user mode. When a user application requests a service from the operating system (via a systemcall), thetransition from user to kernel mode takes place.



At system boot time, the hardware starts in kernel mode. The operating systemis then loadedandstarts user applications in user mode. Whenever a trap or interrupt occurs, the hardware switchesfrom user mode to kernel mode (that is, changes the mode bit from 1 to 0). Thus, whenever theoperating system gains control of the computer, it is in kernel mode.

The dual mode of operation provides us with the means for protecting the operatingsystem from errant users—and errant users from one another.

The hardware allows privileged instructions to be executed only in kernel mode. If anattempt is made to execute a privileged instruction in user mode, the hardware does not executethe instruction but rather treats it as illegal and traps it to the operating system. The instructiontoswitch to user mode is an example of a privileged instruction.

Initial control is within the operating system, where instructions are executed inkernel mode. When control is given to a user application, the mode is set to user mode. Eventually, control is switched back to the operating system via an interrupt, a trap, or a systemcall.

b) Timer

Operating system uses timer to control the CPU. A user program cannot hold CPUfor along time, this is prevented with the help of timer.

A timer can be set to interrupt the computer after a specified period. The period maybe**fixed** (for example, 1/60 second) or **variable** (for example, from 1 millisecond to 1 second).

Fixed timer – After a fixed time, the process under execution is interrupted.

Variable timer – Interrupt occurs after varying interval. This is implementedusingafixed-rate clock and a counter. The operating system sets the counter. Every time the clockticks, the counter is decremented. When the counter reaches 0, an interrupt occurs.

Before changing to the user mode, the operating system ensures that the timer is set to interrupt. If the timer interrupts, control transfers automatically to the operating system, which maytreat the interrupt as a fatal error or may give the program more time.

Process Management

A program under execution is a process. A process needs resources like CPU time, memory, files, and I/O devices for its execution. These resources are given to the process when it is createdor at run time. When the process terminates, the operating system reclaims the resources.

The program stored on a disk is a **passive entity** and the program under executionis an**active entity**. A single-threaded process has one **program counter** specifying the next instruction to execute. The CPU executes one instruction of the process after another, until theprocess completes. A multithreaded process has multiple program counters, each pointingtothenext instruction to execute for a given thread.

The operating system is responsible for the following activities in connection with process management:

- Scheduling process and threads on the CPU
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication

Memory Management

Main memory is a large array of words or bytes. Each word or byte has its own address. Main memory is the storage device which can be easily and directly accessed by the CPU. As theprogram executes, the central processor reads instructions and also reads and writes data frommain memory.

To improve both the utilization of the CPU and the speed of the computer's responsetoits users, general-purpose computers must keep several programs in memory, creating a needfor memory management.

The operating system is responsible for the following activities in connection with memorymanagement:

- Keeping track of which parts of memory are currently being used by user. • Deciding which processes and data to move into and out of memory.
- Allocating and deallocating memory space as needed.

Storage Management

There are three types of storage management i) File system management ii) Mass-storagemanagement iii) Cache management.

File-System Management

File management is one of the most visible components of an operating system. Computerscanstore information on several different types of physical media. Magnetic disk, optical disk, andmagnetic tape are the most common. Each of these media has its own characteristics andphysical organization. Each medium is controlled by a device, such as a disk drive or tapedrive, that also has its own unique characteristics.

A file is a collection of related information defined by its creator. Commonly, filesrepresent programs and data. Data files may be numeric, alphabetic, alphanumeric, or binary. Files may be free-form (for example, text files), or they may be formatted rigidly (for example, fixed fields).

The operating system implements the abstract concept of a file by managing mass storagemedia. Files are normally organized into directories to make them easier to use. Whenmultipleusers have access to files, it may be desirable to control by whom and in what ways (read, write, execute) files may be accessed.

The operating system is responsible for the following activities in connection with file management:

- Creating and deleting files
- Creating and deleting directories to organize files
- Supporting primitives for manipulating files and directories
- Mapping files onto secondary storage
- Backing up files on stable (nonvolatile) storage media

Mass-Storage Management

As the main memory is too small to accommodate all data and programs, and as thedatathat it holds are erased when power is lost, the computer system must provide secondarystorageto back up main memory. Most modern computer systems use disks as the storage mediumforboth programs and data.

Most programs—including compilers, assemblers, word processors, editors, andformatters—are stored on a disk until loaded into memory and then use the disk as boththesource and destination of their processing. Hence, the proper management of disk storageisofcentral importance to a computer system. The operating system is responsible for the followingactivities in connection with disk management:

- Free-space management
- Storage allocation
- Disk scheduling

As the secondary storage is used frequently, it must be used efficiently. The entirespeedof operation of a computer may depend on the speeds of the disk. Magnetic tape drives andtheirtapes, CD, DVD drives and platters are **tertiary storage** devices. The functions that operatingsystems provides include mounting and unmounting media in devices, allocating and freeingthedevices for exclusive use by processes, and migrating data from secondary to tertiary storage.

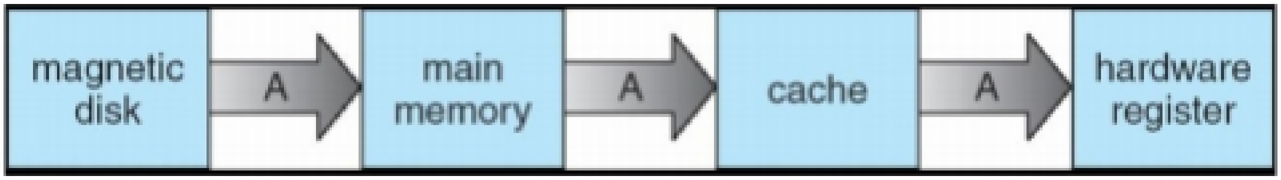
Caching

Caching is an important principle of computer systems. Information is normally kept in somestorage system (such as main memory). As it is used, it is copied into a faster storage system—the cache—as temporary data. When a particular piece of information is required, first we checkwhether it is in the cache. If it is, we use the information directly from the cache; if it is not incache, we use the information from the source, putting a copy in the cache under the assumptionthat we will need it again soon.

Because caches have limited size, **cache management** is an important design problem. Careful selection of the cache size and page replacement policy can result in greatly increasedperformance.

The movement of information between levels of a storage hierarchy may be eitherexplicit or implicit, depending on the hardware design and the controlling operating-systemsoftware. For instance, data transfer from cache to CPU and registers is usually a hardwarefunction, with no operating-system intervention. In contrast, transfer of data fromdisktomemory is usually controlled by the operating system.

In a hierarchical storage structure, the same data may appear in different levels of thestorage system. For example, suppose to retrieve an integer A from magnetic disktotheprocessing program. The operation proceeds by first issuing an I/O operation to copythediskblock on which A resides to main memory. This operation is followed by copying Ato thecacheand to an internal register. Thus, the copy of A appears in several places: on the magnetic disk, inmain memory, in the cache, and in an internal register.



In a multiprocessor environment, in addition to maintaining internal registers, each of theCPUsalso contains a local cache. In such an environment, a copy of A may exist simultaneouslyinseveral caches. Since the various CPUs can all execute concurrently, any update done tothevalue of A in one cache is immediately reflected in all other caches where Aresides. Thissituation is called **cache coherency,** and it is usually a hardware problem(handled belowtheoperating-system level).

I/O Systems

One of the purposes of an operating system is to hide the peculiarities of specific hardware devices from the user. The I/O subsystem consists of several components: • A memory- management component that includes buffering, caching, and spooling • A general device-driver interface

• Drivers for specific hardware devices

Only the device driver knows the peculiarities of the specific device to which it is assigned.

Protection and Security

If a computer system has multiple users and allows the concurrent execution of multipleprocesses, 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.

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

For example, memory-addressing hardware ensures that a process can executeonlywithin its own address space. The timer ensures that no process can gain control of the CPUfor along time. Device-control registers are not accessible to users, so the integrity of the variousperipheral devices is protected.

Protection is a mechanism for controlling the access of processes or users totheresources defined by a computer system. This mechanism must provide means for specificationof the controls to be imposed and means for enforcement.

Protection improves reliability. A protection-oriented system provides a means todistinguish between authorized and unauthorized usage. A system can have adequate protectionbut still be prone to failure and allow inappropriate access.

Consider a user whose authentication information is stolen. Her data could be copiedordeleted, even though file and memory protection are working. It is the job of **security** todefendasystem from external and internal attacks. Such attacks spread across a huge range andincludeviruses and worms, denial-of service attacks etc.

Protection and security require the system to be able to distinguish among all its users. Most operating systems maintain a list of user names and associated **user identifiers (user IDs).** When a user logs in to the system, the authentication stage determines the appropriate user IDforthe user.

Distributed Systems

A distributed system is a collection of systems that are networked to provide the users withaccess to the various resources in the network. Access to a shared resource increasescomputation speed, functionality, data availability, and reliability.

A **network** is a communication path between two or more systems. Networks varybytheprotocols used(TCP/IP,UDP,FTP etc.), the distances between nodes, and the transport media(copper wires, fiber-optic,wireless).

TCP/IP is the most common network protocol. The operating systems support ofprotocols also varies. Most operating systems support TCP/IP, including the Windows andUNIXoperating systems.

Networks are characterized based on the distances between their nodes. A**local-areanetwork (LAN)** connects computers within a room, a floor, or a building. A **wide-areanetwork(WAN)** usually links buildings, cities, or countries. A global company may have a WANtoconnect its offices worldwide. These networks may run one protocol or several protocols. A**metropolitan-area network** (MAN) connects buildings within a city. BlueTooth and802.11devices use wireless technology to communicate over a distance of several feet, inessencecreating a **small-area network** such as might be found in a home.

The transportation media to carry networks are also varied. They include copper wires, fiber strands, and wireless transmissions between satellites, microwave dishes, and radios. Whencomputing devices are connected to cellular phones, they create a network.

Special-Purpose Systems

There are different classes of computer systems, whose functions are more limited andspecificand it deal with limited computation domains. The systems can be classified as Real-TimeEmbedded Systems, Multimedia Systems and Handheld Systems.

Real-Time Embedded Systems

Embedded computers are the most prevalent form of computers in existence. Thesedevices are found everywhere, from car engines and manufacturing robots to VCRs andmicrowave ovens. They tend to have very specific tasks. Usually, they have little user interface, and more time is spent for monitoring and managing hardware devices, eg. automobile enginesand robotic arms.

The Operating Systems, in these embedded systems vary considerably. Some systemshave standard operating systems—such as UNIX—with special-purpose applications. Othershave special-purpose embedded operating system providing just the functionality desired.

Embedded systems always run **real-time operating systems.** A real-time systemis used whenthere is restricted time for an operation or for the flow of data. A real-time systemfunctions

correctly only if it returns the correct result within its time constraints. Sensors bring data tothecomputer. The computer must analyze the data and perform certain action.

Some medical imaging systems, automobile-engine fuel-injection systems, home appliance controllers, and weapon systems are real-time systems. A real-time systemhas well defined, fixed time constraints. Processing **mustbe** done within the defined constraints, or thesystem will fail. For instance, the robot arm should be halted before it has smashed into the car, it was building.

Entire houses can be computerized, so that a computer —can control heating and lighting, alarmsystems, and even coffee makers. Web access can enable a home owner to tell the house toheat up before she arrives home.

Multimedia Systems

Multimedia data consist of audio and video files as well as conventional files. These datadifferfrom 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 like audio files - MP3, DVDmovies, videoconferencing, and short video clips of movie previews or news. Multimedia applications mayalso include live webcasts of speeches or sporting events and even live webcams. Multimediaapplications can be either audio or video or combination of both. For example, a moviemayconsist of separate audio and video tracks.

Handheld Systems

Handheld systems include personal digital assistants (PDAs), such as Palmand Pocket-PCs, andcellular

telephones. Developers of these systems face many challenges, due to the limitedmemory, slow processors and small screens in such devices.
The amount of physical memory in a handheld depends upon the device, the operatingsystemand applications must manage memory efficiently. This includes returning all allocatedmemoryback to the memory manager when the memory is not being used. A second issue of concerntodevelopers of handheld devices is the speed of the processor used in the devices. Processorsformost handheld devices run at faster speed than the processor in a PC. Faster processors requiremore power and so, a larger battery is required. Another issue is the usage of I/Odevices.

Generally, the limitations in the functionality of PDAs are balanced by their convenience and portability. Their use continues to expand as network connections become more available and other options, suchasdigital cameras and MP3 players, expand their utility.

Computing Environments

The different computing environments are -

Traditional Computing

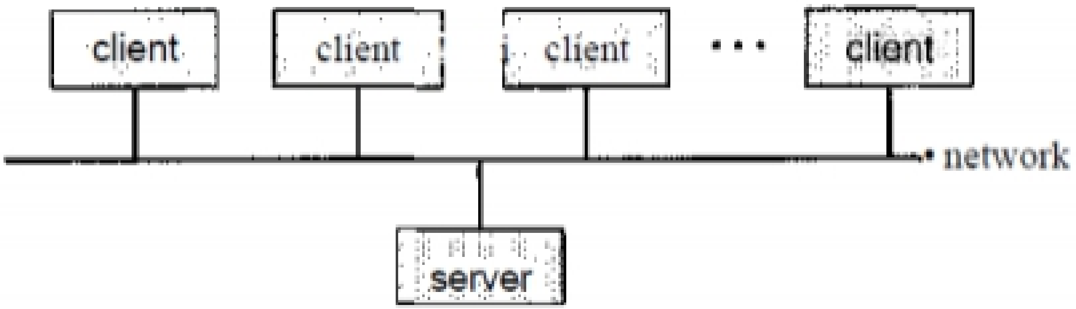
The current trend is toward providing more ways to access these computing environments. Web technologies are stretching the boundaries of traditional computing. Companies establish**portals,** which provide web accessibility to their internal servers. **Network computers**areessentially terminals that understand web-based computing. Handheld computers cansynchronize with PCs to allow very portable use of company information. Handheld PDAs canalso connect to **wireless networks** to use the company's web portal. The fast data connectionsare allowing home computers to serve up web pages and to use networks. Some homes evenhave **firewalls** to protect their networks.

In the latter half of the previous century, computing resources were scarce. Years before, systemswere either batch or interactive. Batch system processed jobs in bulk, with predeterminedinput (from files or other sources of data). Interactive systems waited for input fromusers. Tooptimizethe use of the computing resources, multiple users shared time on these systems. Time-sharingsystems used a timer and scheduling algorithms to rapidly cycle processes through theCPU, giving each user a share of the resources.

Today, traditional time-sharing systems are used everywhere. The same scheduling techniqueisstill in use on workstations and servers, but frequently the processes are all owned by thesameuser (or a single user and the operating system). User processes, and systemprocesses that provide services to the user, are managed so that each frequently gets a slice of computer time.

Client-Server Computing

Designers shifted away from centralized system architecture to - terminals connected to centralized systems. As a result, many of today’s systems act as **server systems to** satisfyrequests generated by **client systems.** This form of specialized distributed system, called **client server** system.



General

Structure of Client – Server System

Server systems can be broadly categorized as compute servers and file servers: • The **compute- server system** provides an interface to which a client can send a request to performanaction(for example, read data); in response, the server executes the action and sends back resultstothe client. A server running a database that responds to client requests for data is an exampleof such a svstem.

- The **file-server system** provides a file-system interface where clients can create, update, read, and delete files. An example of such a system is a web server that delivers filestoclients running the web browsers.

Peer-to-Peer Computing

In this model, clients and servers are not distinguished from one another; here, all nodes withinthe system are considered peers, and each may act as either a client or a server, dependingonwhether it is requesting or providing a service.

In a client-server system, the server is a bottleneck, because all the services must beserved by the server. But in a peer-to-peer system, services can be provided by several nodesdistributed throughout the network.

To participate in a peer-to-peer system, a node must first join the network of peers. Once anodehas joined the network, it can begin providing services to—and requesting services from—othernodes in the network. Determining what services are available is accomplished in one of twogeneral ways:

- When a node joins a network, it registers its service with a centralized lookup service onthenetwork. Any node desiring a specific service first contacts this centralizedlookupservice to determine which node provides the service. The remainder of thecommunication takes place between the client and the service provider.
- A peer acting as a client must know, which node provides a desired service by broadcastinga request for the service to all other nodes in the network. The node (or nodes) providingthat service responds to the peer making the request. To support this approach, a*discovery protocol* must be provided that allows peers to discover services providedbyother peers in the network.

Web-Based Computing

Web computing has increased the importance on networking. Devices that werenot previously networked now include wired or wireless access. Devices that were networkednowhave faster network connectivity.

The implementation of web-based computing has given rise to new categories of devices, suchas**load balancers,** which distribute network connections among a pool of similar servers. Operating systems like Windows 95, which acted as web clients, have evolved into LinuxandWindows XP, which can act as web servers as well as clients. Generally, the Web has increasedthe complexity of devices, because their users require them to be web-enabled.

The design of an operating system is a major task. It is important that the goals of thenewsystem be well defined before the design of OS begins. These goals formthe basis for choicesamong various algorithms and strategies.

2.1 Operating-System Services

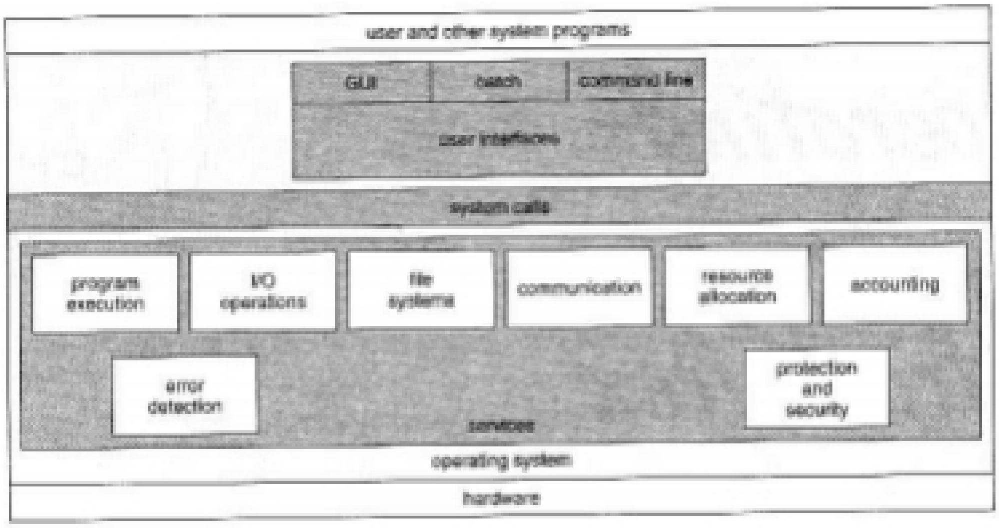


Figure 2.1 A view of operating system services.

An operating system provides an environment for the execution of programs. It provides certain services to programs and to the users of those programs. OS provide services for the users of the system, including:

- **User Interfaces** - Means by which users can issue commands to the system. Dependingonthe operating system these may be a **command-line interface** (e.g. sh, csh, ksh, tcsh, etc.), a **Graphical User Interface** (e.g. Windows, X-Windows, KDE, Gnome, etc.), or a**batch command systems**. In Command Line Interface(CLI)- commands are giventothesystem. In Batch interface – commands and directives to control these commands areput in a file and then the file is executed. In GUI systems- windows with pointing devicetoget inputs and keyboard to enter the text.
- **Program Execution** - The OS must be able to load a program into RAM, run the program, and terminate the program, either normally or abnormally.
- **I/O Operations** - The OS is responsible for transferring data to and fromI/Odevices, including keyboards, terminals, printers, and files. For specific devices, special functionsare provided(device drivers) by OS.

- **File-System Manipulation** – Programs need to read and write files or directories. Theservices required to create or delete files, search for a file, list the contents of a fileandchange the file permissions are provided by OS.
- **Communications** - Inter-process communications, IPC, either between processes runningon the same processor, or between processes running on separate processors or separatemachines. May be implemented by using the service of OS- like shared memoryormessage passing.
- **Error Detection** - Both hardware and software errors must be detected andhandledappropriately by the OS. Errors may occur in the CPU and memory hardware (suchaspower failure and memory error), in I/O devices (such as a parity error ontape, aconnection 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).

OS provide services for the efficient operation of the system, including:

- **Resource Allocation** – Resources like CPU cycles, main memory, storage space, andI/Odevices must be allocated to multiple users and multiple jobs at the same time. • **Accounting**– There are services in OS to keep track of system activity and resource usage, either for billing purposes or for statistical record keeping that can be used to optimize future performance.
- **Protection and Security** – The owners of information(file) in multiuser or networkedcomputer system may want to control the use of that information. When several separateprocesses execute concurrently,

one process should not interfere with other or withOS. Protection involves ensuring that all access to system resources is controlled. Securityofthe system from outsiders must also be done, by means of a password.

2.2 User Operating-System Interface

There are several ways for users to interface with the operating system.

1) Command-line interface, or command interpreter, allows users to directly enter commands to be performed by the operating system.
2) Graphical user interface(GUI), allows users to interface with the operating systemusing pointer device and menu system.

Command Interpreter

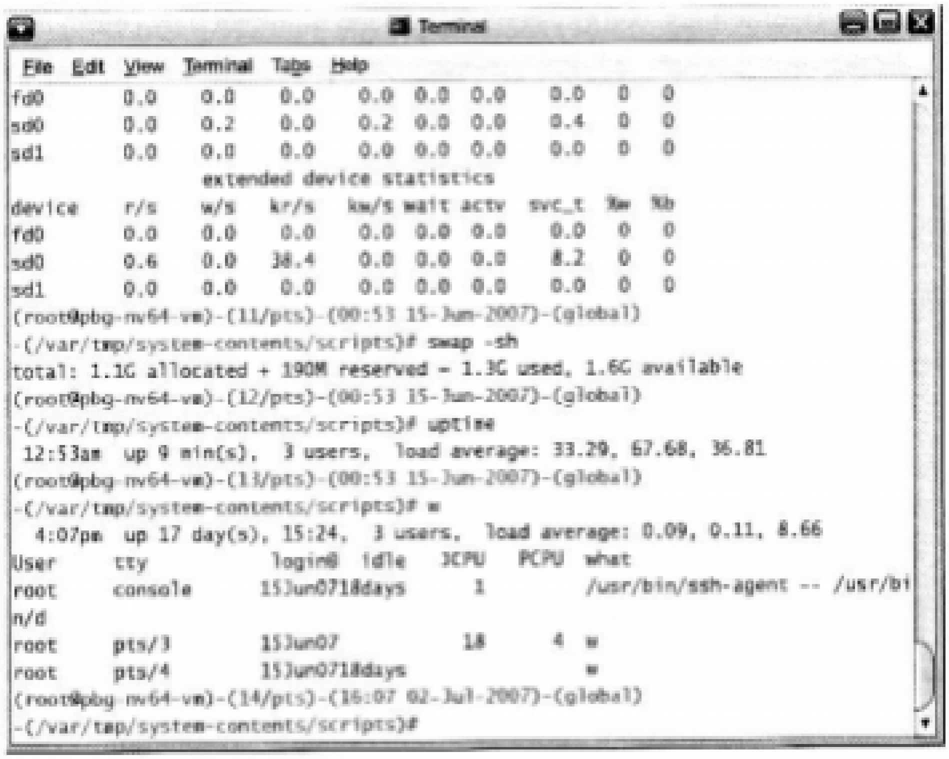
Command Interpreters are used to give commands to the OS. There are multiplecommand interpreters known as shells. In UNIX and Linux systems, there are several different shells, like the *Bourne shell, C shell, Bourne-Again shell, Korn shell,* and others.

The main function of the command interpreter is to get and execute the user-specifiedcommand. Many of the commands manipulate files: create, delete, list, print, copy, execute, andso on.

The commands can be implemented in two general ways

1) The command interpreter itself contains the code to execute the command. For example, acommand to delete a file may cause the command interpreter to jump to a particular sectionof its code that sets up the parameters and makes the appropriate system call.

2) The code to implement the command is in a function in a separate file. The interpretersearches for the file and loads it into the memory and executes it by passing the parameter. Thusby adding new functions new commands can be added easily to the interpreter without disturbingit.



```
fd0         0.0     0.0     0.0     0.0  0.0  0.0     0.0   0   0
sd0         0.0     0.2     0.0     0.2  0.0  0.0     0.4   0   0
sd1         0.0     0.0     0.0     0.0  0.0  0.0     0.0   0   0
                   extended device statistics
device      r/s     w/s    kr/s    kw/s wait actv  svc_t  %w  %b
fd0         0.0     0.0     0.0     0.0  0.0  0.0     0.0   0   0
sd0         0.6     0.0    38.4     0.0  0.0  0.0     8.2   0   0
sd1         0.0     0.0     0.0     0.0  0.0  0.0     0.0   0   0
(root@pbg-nv64-vm)-(11/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@pbg-nv64-vm)-(12/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# uptime
 12:53am  up 9 min(s),  3 users,  load average: 33.29, 67.68, 36.81
(root@pbg-nv64-vm)-(13/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# w
  4:07pm  up 17 day(s), 15:24,  3 users,  load average: 0.09, 0.11, 8.66
User     tty           login@  idle   JCPU   PCPU  what
root     console      15Jun0718days      1         /usr/bin/ssh-agent -- /usr/bi
n/d
root     pts/3        15Jun07           18      4  w
root     pts/4        15Jun0718days                w
(root@pbg-nv64-vm)-(14/pts)-(16:07 02-Jul-2007)-(global)
-(/var/tmp/system-contents/scripts)#
```

Figure 2.2 The Bourne shell command interpreter in Solaris 10.

Graphical User Interface, GUI

Another way of interfacing with the operating system is through a user friendly graphical userinterface, or GUI. Here, rather than entering commands directly via a command-line interface, users employ a mouse-based window and menu system. The user moves the mouse to positionits

pointer on images, or icons on the screen (the desktop) that represent programs, files, directories, and system functions. Depending on the mouse pointer's location, clicking a button on the mousecan invoke a program, select a file or directory-known as a folder-or pull down a menuthat contains commands.

Graphical user interfaces first appeared on the Xerox Alto computer in 1973.

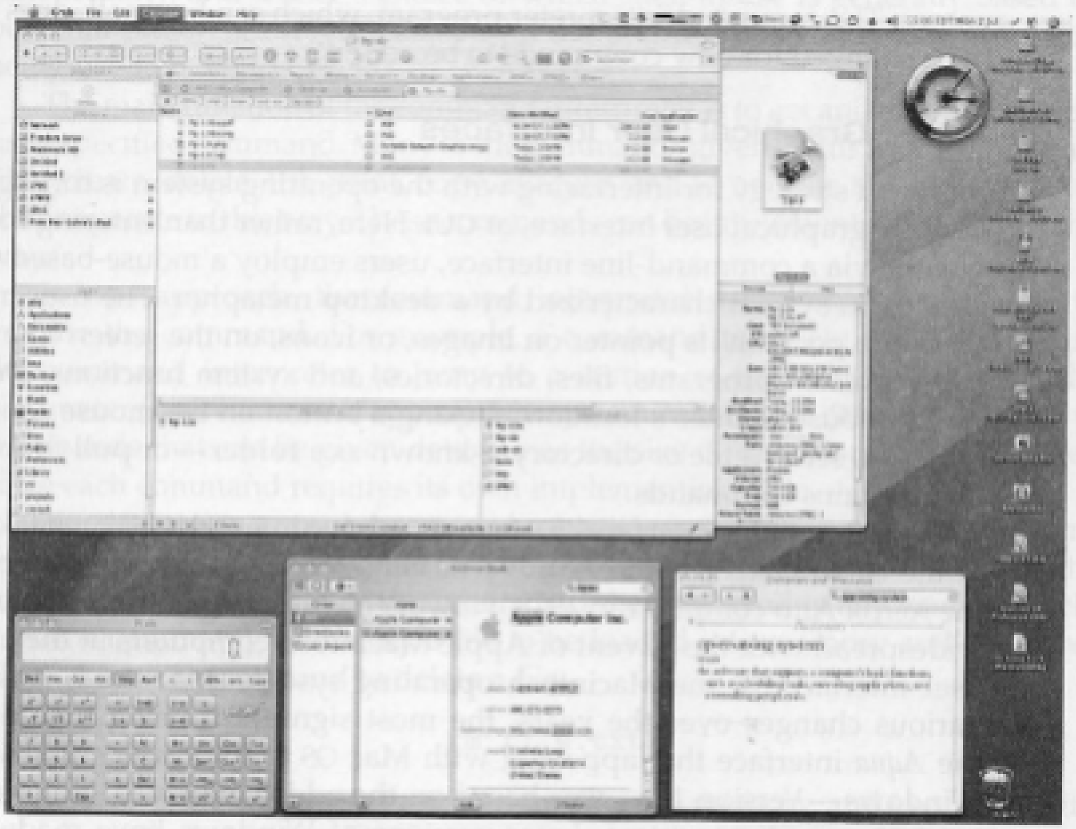


Figure 2.3 The Mac OS X GUI.

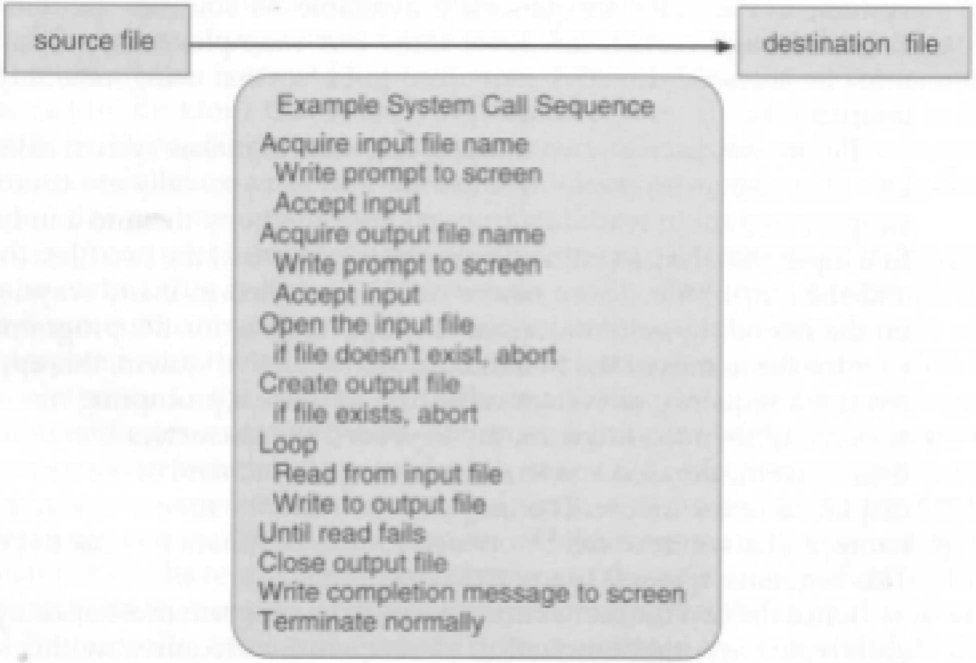
Most modern systems allow individual users to select their desired interface, and to customizeitsoperation, as well as the ability to switch between different interfaces as needed.

2.3 System Calls

- System calls is a means to access the services of the operating system.

- Generally written in C or C++, although some are written in assembly for optimal performance.

- The below figure illustrates the sequence of system calls required to copy a file content from one file(input file) to another file (output file).



Example System Call Sequence

```
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
```

There are number of system calls used to finish this task. The first systemcall is towriteamessage on the screen (monitor). Then to accept the input filename. Then another systemcall towrite message on the screen, then to accept the output filename. When the programtries toopenthe input file, it may find that there is no file of that name or that the file is protectedagainst access. In these cases, the program should print a message on the console(another systemcall)and then terminate abnormally (another system call) and create a new one (another systemcall).

Now that both the fileas are opened, we enter a loop that reads from the input file(anothersystem call) and writes to output file (another system call).

Finally, after the entire file is copied, the program may close both files (another systemcall), write a message to the console or window(system call), and finally terminate normally(final system call).

- Most programmers do not use the low-level system calls directly, but instead use an"Application Programming Interface", API.

- The APIs instead of direct system calls provides for greater programportabilitybetweendifferent systems. The API then makes the appropriate system calls through the systemcall interface, using a system call table to access specific numbered systemcalls, asshown in Figure 2.6.
- Each system call has a specific numbered system call. The system call table (consistingofsystem call number and address of the particular service) invokes a particular serviceroutine for a specific system call.
- The caller need know nothing about how the system call is implemented or what it does during execution.

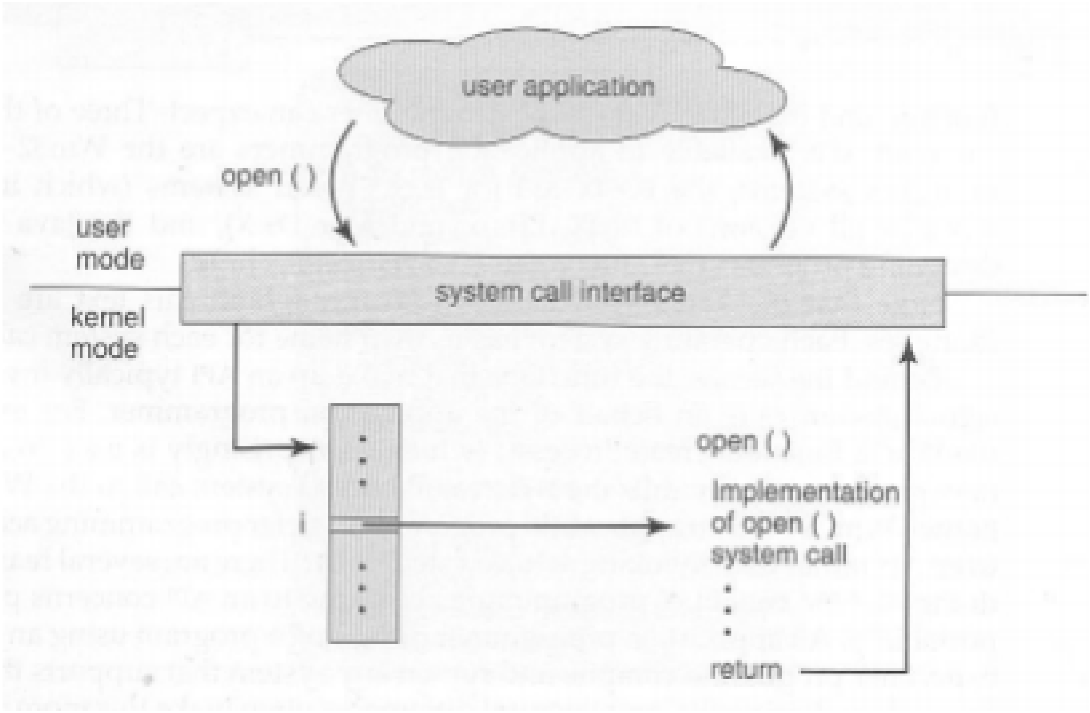
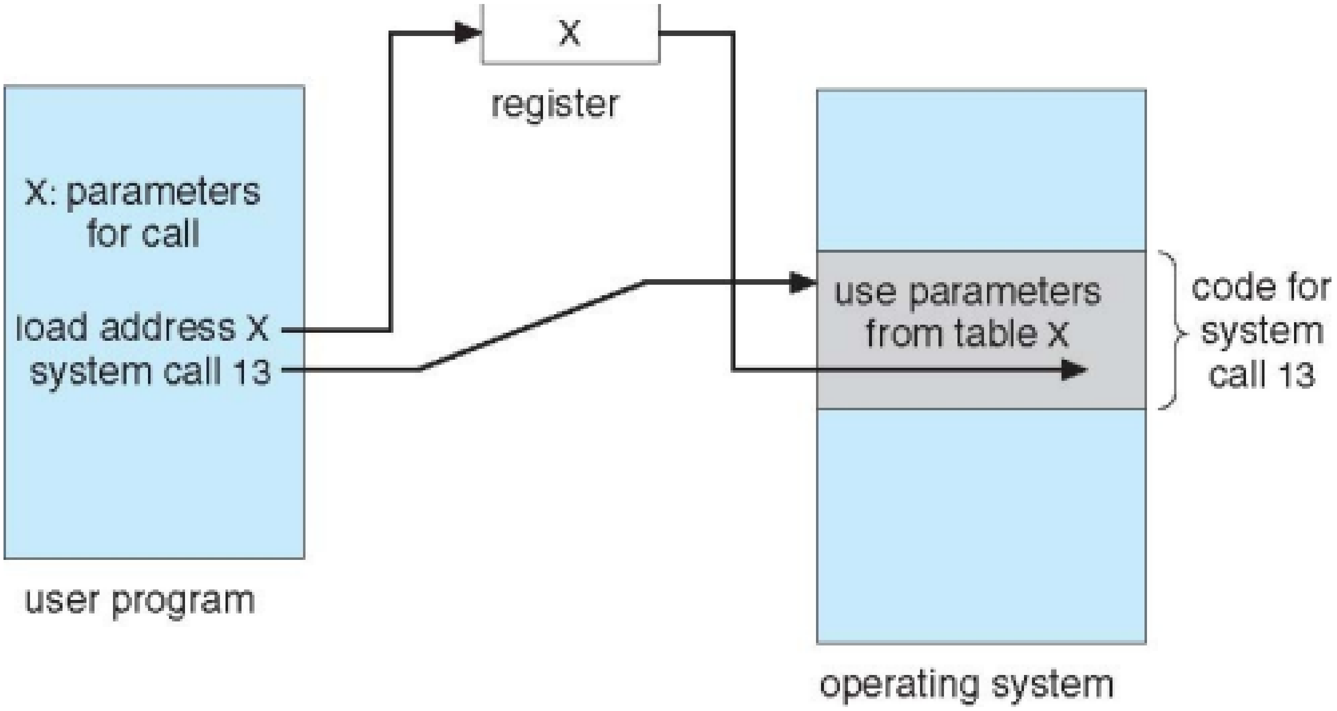


Figure 2.6 The handling of a user application invoking the open() systemcall.



Three general methods used to pass parameters to OS are –

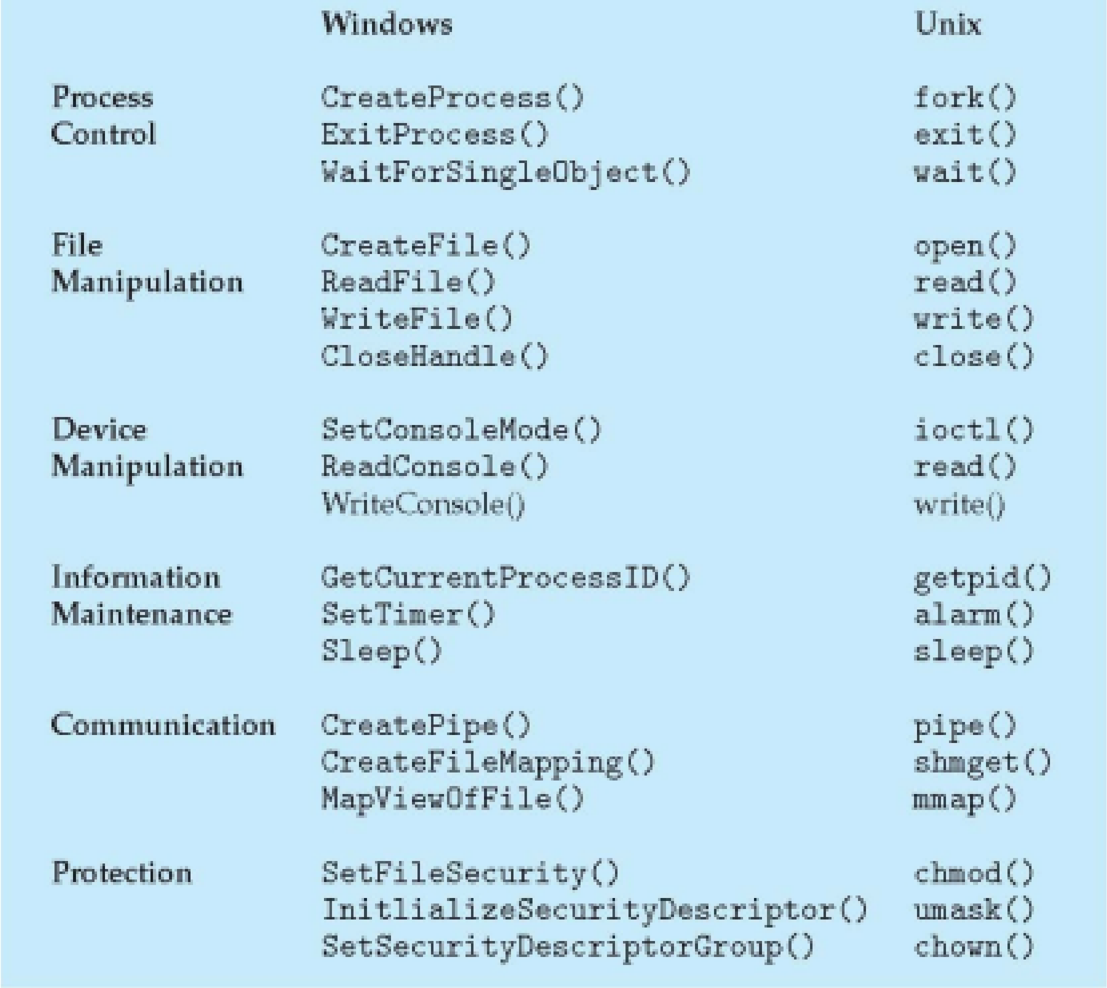
- To pass parameters in registers
- If parameters are large blocks, address of block (where parameters are stored in memory) issent to OS in the register. (Linux & Solaris).
- Parameters can be pushed onto the stack by program and popped off the stack by OS.

2.3.1 Types of System Calls

The system calls can be categorized into six major categories:

- Process Control
- File management

- Device management
- Information management
- Communications
- Protection



	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Communication	CreatePipe()	pipe()
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitlializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()

a) Process Control

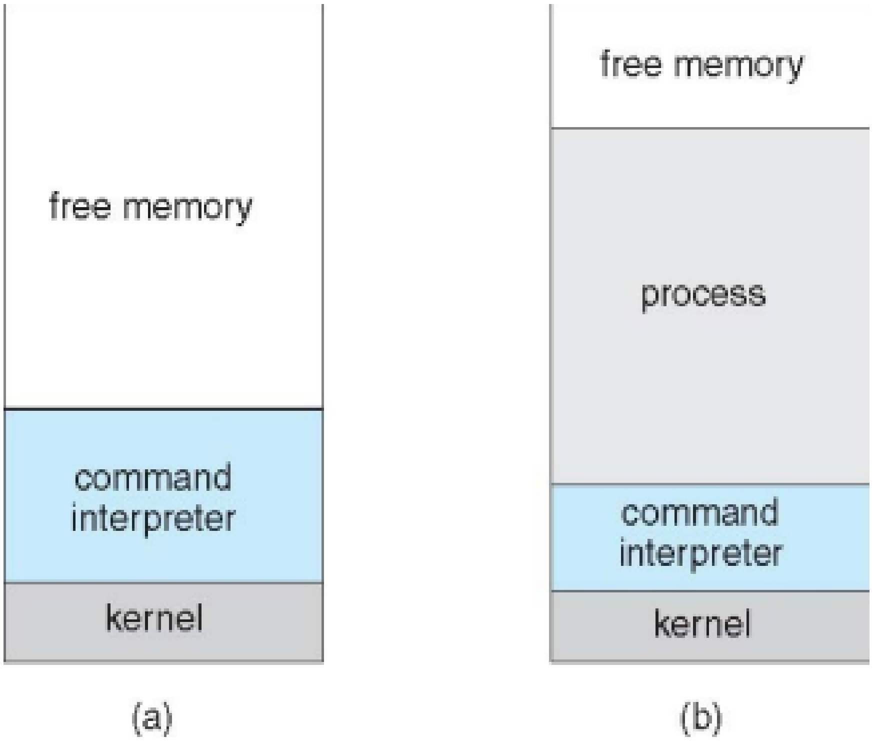
• Process control system calls include end, abort, load, execute, create process, terminateprocess, get/set process attributes, wait for time or event, signal event, and allocateandfree memory.

• Processes must be created, launched, monitored, paused, resumed, and eventually stopped. • When one process pauses or stops, then another must be launched or resumed • Process attributes like process priority, max. allowable execution time etc. are set andretrieved by OS.

• After creating the new process, the parent process may have to wait (wait time), or wait foran event to occur(wait event). The process sends back a signal when the event hasoccurred (signal event).

- In DOS, the command interpreter loaded first.Then loads the process andtransferscontrol to it. The interpreter does not resume until the process has completed, asshown in Figure

At system startup running a program



- Because UNIX is a multi-tasking system, the command interpreter remains completely resident when executing a process, as shown in Figure 2.11 below. ▪ The user can switch back to the command interpreter at any time, and can placethe running process in the background even if it was not originally launchedasabackground process.
  - In order to do this, the command interpreter first executes a "fork" systemcall, which creates a second process which is an exact duplicate (clone)of the original command interpreter. The original process is knownasthe parent, and the cloned process is known as the child, with its ownunique process ID and parent ID.
  - The child process then executes an "exec" system call, which replaces its code with that of the desired process.
  - The parent (command interpreter) normally waits for the child to completebefore issuing a new command prompt, but in some cases it can alsoissuea new prompt right away, without waiting for the child process tocomplete. (The child is then said to be running "in the background", or"as a background process".)

b) File Management

The file management functions of OS are –

- File management system calls include create file, delete file, open, close, read, write, reposition, get file attributes, and set file attributes.
- After **creating** a file, the file is **opened**. Data is **read** or **written** to a file. • The file pointer may need to be **repositioned** to a point.
- The file **attributes** like filename, file type, permissions, etc. are set and retrieved usingsystem calls.
- These operations may also be supported for directories as well as ordinary files. **c) Device**

Management

- Device management system calls include **request device**, **release** device, **read**, **write**, **reposition**, **get/set** device attributes, and logically **attach** or **detach** devices. • When a process needs a resource, a request for resource is done. Then the control is granted to the process. Ifrequested resource is already attached to some other process, the requesting process has towait.
- In multiprogramming systems, after a process uses the device, it has to be returned toOS, so that another process can use the device.
- Devices may be physical (e.g. disk drives), or virtual / abstract (e.g. files, partitions, andRAM disks).

d) Information Maintenance

- Information maintenance system calls include calls to get/set the time, date, systemdata, and process, file, or device attributes.
- These system calls care used to transfer the information between user and theOS. Information like current time & date, no. of current users, version no. of OS, amount offree memory, disk space etc. are passed from OS to the user.

e) Communication

- Communication system calls create/delete communication connection, send/receive messages, transfer status information, and attach/detach remote devices. • The **message passing** model must support calls to:
    - Identify a remote process and/or host with which to communicate.
    - Establish a connection between the two processes.
    - Open and close the connection as needed.
    - Transmit messages along the connection.
    - Wait for incoming messages, in either a blocking or non-blocking state. o Delete the connection when no longer needed.
- The **shared memory** model must support calls to:
    - Create and access memory that is shared amongst processes (and threads.) o Free up shared memory and/or dynamically allocate it as needed.
- Message passing is simpler and easier, (particularly for inter-computer communications), and is generally appropriate for small amounts of data. It is easy to implement, but thereare system calls for each read and write process.
- Shared memory is faster, and is generally the better approach where large amounts of dataare to be shared. This model is difficult to implement, and it consists of only fewsystemcalls.

f) Protection

- Protection provides mechanisms for controlling which users / processes have access towhich system resources.
- System calls allow the access mechanisms to be adjusted as needed, and for non privelegedusers to be granted elevated access permissions under carefully controlled temporarycircumstances.

2.4 System Programs

A collection os programs that provide a convenient environment for programdevelopment andexecution (other than OS) are called system programs or system utilities.

- It is not a part of the kernel or command interpreters.
- System programs may be divided into five categories:
    - **File management** - programs to create, delete, copy, rename, print, list, andgenerally manipulate files and directories.
    - **Status information** - Utilities to check on the date, time, number of users, processes running, data logging, etc. System **registries** are used to store andrecall configuration information for particular applications.
    - **File modification** - e.g. text editors and other tools which can change file contents. o **Programming-language support** - E.g. Compilers, linkers, debuggers, profilers, assemblers, library archive management, interpreters for common languages, andsupport for make.
    - **Program loading and execution** - loaders, dynamic loaders, overlay loaders, etc., as well as interactive debuggers.

- o **Communications** - Programs for providing connectivity between processes andusers, including mail, web browsers, remote logins, file transfers, andremote command execution.

2.5 Operating-System Design and Implementation

2.5.1 Design Goals

Any system to be designed must have its own goals and specifications. SimilarlytheOSto be built will have its own goals depending on the type of systemin which it will beused, the type of hardware used in the system etc.

- **Requirements** define properties which the finished system must have, and are a necessarysteps in designing any large complex system. The requirements may be of twobasicgroups:

1. User goals (User requirements)
2. System goals (system requirements)

- o **User requirements** are features that users care about and understand like systemshould be convenient to use, easy to learn,reliable, safe and fast.
- o **System requirements** are written for the developers, ie. People who designtheOS. Their requirements are like easy to design, implement and maintain, flexible, reliable, error free and efficient.

2.5.2 Mechanisms and Policies

- Policies determine *what* is to be done. Mechanisms determine *how* it is to be implemented. • Example: in timer, counter and decrementing counter is the mechanismand decidinghowlong the time has to be set is the policies.
- Policies change overtime. In the worst case, each change in policy would require a changein the underlying mechanism.
- If properly separated and implemented, policy changes can be easily adjusted without rewriting the code, just by adjusting parameters or possibly loading new data / configuration files.

2.5.3 Implementation

- Traditionally OS were written in assembly language.
- In recent years, Os are written in C, or C++. Critical sections of code are still writteninassembly language.
- The first OS that was not written in assembly language was the Master Control Program(MCP).
- The advantages of using a higher-level language for implementing operating systems are: The code can be written faster, more compact, easy to port to other systems andis easierto understand and debug.
- The only disadvantages of implementing an operating system in a higher-level languageareduced speed and increased storage requirements.

2.7 Operating-System Structure

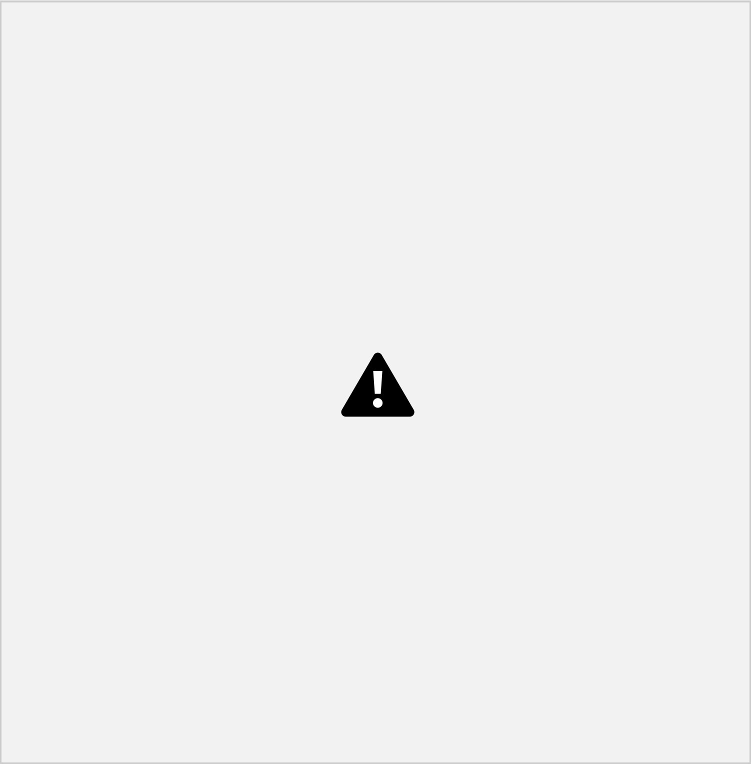
OS structure must be carefully designed. The task of OS is divided into small components andthen interfaced to work together.

2.7.1 Simple Structure

Many operating systems do not have well-defined structures. They started as small, simple, andlimited systems and then grew beyond their original scope. Eg: MS-DOS.

In MS-DOS, the interfaces and levels of functionality are not well separated. Applicationprograms can access basic I/O routines to write directly to the display and disk drives. Suchfreedom leaves MS-DOS in bad state and the entire system can crash down when user programsfail.

UNIX OS consists of two separable parts: the kernel and the system programs. The kernel is further separated into a series of interfaces and device drivers. The kernel provides thefilesystem, CPU scheduling, memory management, and other operating-systemfunctions throughsystem calls.



MS-DOS Layer Structure

Figure 2.13 UNIX System Structure

2.7.2 Layered Approach

- The OS is broken into number of layers (levels). Each layer rests on the layer belowit, andrelies on the services provided by the next lower layer.

- Bottom layer(layer 0) is the hardware and the topmost layer is the user interface. • Atypical layer, consists of data structure and routines that can be invoked by higher-level layer.

Advantage of layered approach is simplicity of construction and debugging.

The layers are selected so that each uses functions and services of only lower-level layers. Sosimplifies debugging and system verification. The layers are debugged one by one fromthelowest and if any layer doesn't work, then error is due to that layer only, as the lower layersarealready debugged. Thus the design and implementation is simplified.

A layer need not know how its lower level layers are implemented. Thus hides the operations from higher layers.

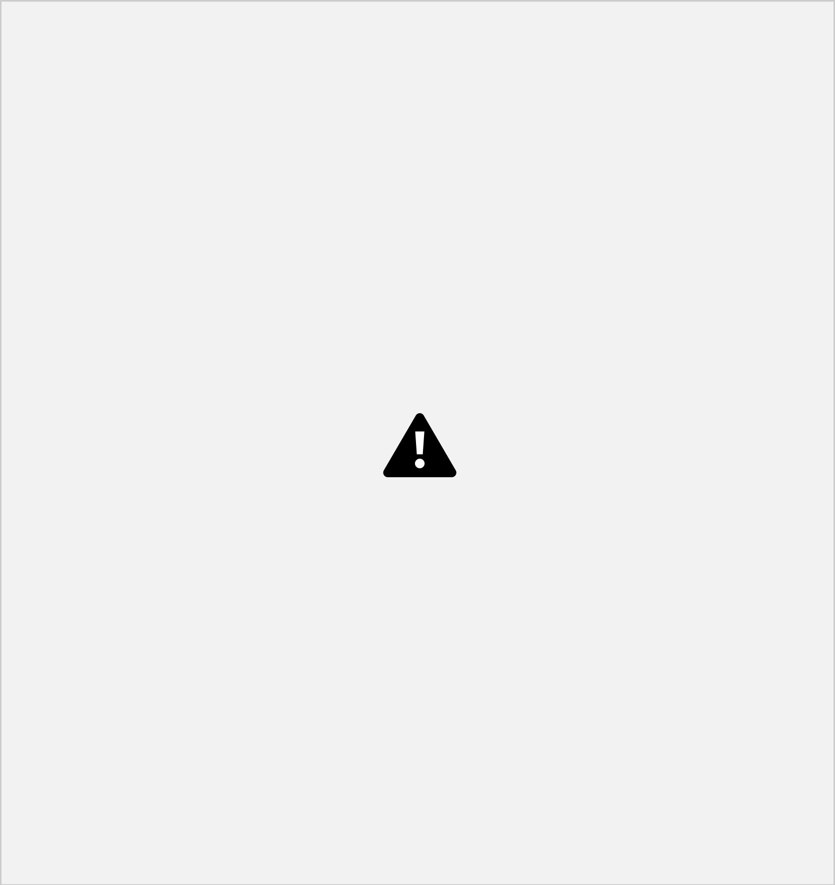


Figure 2.14 A layered Operating System

Disadvantages of layered approach:

- The various layers must be appropriately defined, as a layer can use only lower level layers.
- Less efficient than other types, because any interaction with layer 0 required fromtoplayer. The system call should pass through all the layers and finally to layer 0. This isanoverhead.

2.7.3 Microkernels

- The basic idea behind micro kernels is to **remove all non-essential** services fromthe kernel, thus making the kernel as small and efficient as possible.
- The removed services are implemented as system applications. • Most microkernels provide basic process and memory management, and message passingbetween other services.
- **Benefit** of microkernel - System expansion can also be easier, because it only involves adding more system applications, not rebuilding a new kernel. • Mach was the first and most widely known microkernel, and now forms a major component of Mac OSX.
- Disadvantage of Microkernel is, it suffers from reduction in performance due to increasessystem function overhead.

Application

Program System usermode

File DeviceDriver

messages messages

CPU

memory

Interprocess managment

scheduling

Communication microkernel

2.7.4 Modules

kernelmode

hardware

- Modern OS development is object-oriented, with a relatively small core kernel and a set of ***modules*** which can be linked in dynamically.
- Modules are similar to layers in that each subsystem has clearly defined tasks and interfaces, but any module is free to contact any other module, eliminating the problems of goingthrough multiple intermediary layers.
- The kernel is relatively small in this architecture, similar to microkernels, but the kernel does not have to implement message passing since modules are free to contact eachotherdirectly. Eg: Solaris, Linux and MacOSX.

Figure 2.15 Solaris loadable modules

- The Max OSX architecture relies on the Mach microkernel for basic systemmanagement services, and the BSD kernel for additional services. Application services anddynamically loadable modules (kernel extensions) provide the rest of theOSfunctionality.
- Resembles layered system, but a module can call any other module.
- Resembles microkernel, the primary module has only core functions and the knowledgeof how to load and communicate with other modules.

2.8 Virtual Machines

The fundamental idea behind a virtual machine is to abstract the hardware of a singlecomputer (the CPU, memory, disk drives, network interface cards, and so forth) intoseveral different execution environments, thereby creating the illusion that each separate executionenvironment is running its own private computer.

Creates an illusion that a process has its own processor with its own memory. Host OSisthe main OS installed in system and the other OS installed in the systemare called guest OS.

Benefits

• Able to share the same hardware and run several different execution environments(OS). • Host system is protected from the virtual machines and the virtual machines are protectedfrom one another. A virus in guest OS, will corrupt that OS but will not affect the other guest systems and host systems.

• Even though the virtual machines are separated from one another, software resources canbe shared among them. Two ways of sharing s/w resource for communication are: a)Toshare a file system volume(part of memory). b)To develop a virtual communication network to communicate between the virtual machines.

• The operating system runs on and controls the entire machine. Therefore, the current system must be stopped and taken out of use while changes are made and tested. This period is commonly called *system development time.* In virtual machines such problemiseliminated. User programs are executed in one virtual machine and systemdevelopment is done in another environment.

• Multiple OS can be running on the developer's system **concurrently**. This helps in rapidporting and testing of programmers code in different environments.

• **System consolidation** – two or more systems are made to run in a single system.

2.10 Operating-System Generation

• OS may be designed and built for a specific HW configuration at a specific site, but morecommonly they are designed with a number of variable parameters and components, which are then configured for a particular operating environment.

• Systems sometimes need to be re-configured after the initial installation, to add additional resources, capabilities, or to tune performance, logging, or security.

• At one extreme the OS source code can be edited, re-compiled, and linked into a newkernel. • More commonly configuration tables determine which modules to link into the newkernel, and what values to set for some key important parameters. This approach may requiretheconfiguration of complicated makefiles, which can be done either automaticallyorthrough interactive configuration programs; Then make is used to actually generatethenew kernel specified by the new parameters.

• At the other extreme a system configuration may be entirely defined by table data, inwhichcase the "rebuilding" of the system merely requires editing data tables. • Once a systemhasbeen regenerated, it is usually required to reboot the system to activate the newkernel. Because there are possibilities for errors, most systems provide some mechanismfor bootingto older or alternate kernels.

2.11 System Boot

The general approach when most computers boot up goes something like this:

- When the system powers up, an interrupt is generated which loads a memory address intothe program counter, and the system begins executing instructions found at that address. This address points to the "bootstrap" program located in ROMchips (or EPROMchips)on the motherboard.
- The ROM bootstrap program first runs hardware checks, determining what physical resources are present and doing power-on self tests (POST) of all HWfor whichthisisapplicable. Some devices, such as controller cards may have their own on-boarddiagnostics, which are called by the ROM bootstrap program.
- The user generally has the option of pressing a special key during the POST process, whichwill launch the ROM BIOS configuration utility if pressed. This utility allows the user tospecify and configure certain hardware parameters as where to look for anOSandwhether or not to restrict access to the utility with a password.
    - Some hardware may also provide access to additional configuration setup programs, such as for a RAID disk controller or some special graphics or networkingcards. • Assuming the utility has not been invoked, the bootstrap program then looks for a nonvolatile storage device containing an OS. Depending on configuration, it may lookfor afloppy drive, CD ROM drive, or primary or secondary hard drives, in the order specifiedby the HW configuration utility.
- Assuming it goes to a hard drive, it will find the first sector on the hard drive andloadupthe fdisk table, which contains information about how the physical hard drive is dividedup into logical partitions, where each partition starts and ends, and which partitionisthe"active" partition used for booting the system.
- There is also a very small amount of system code in the portion of the first disk blocknot occupied by the fdisk table. This bootstrap code is the first step that is not built intothehardware, i.e. the first part which might be in any way OS-specific. Generally this codeknows just enough to access the hard drive, and to load and execute a (slightly) largerboot program.
- For a single-boot system, the boot program loaded off of the hard disk will then proceedtolocate the kernel on the hard drive, load the kernel into memory, and then transfer control over to the kernel. There may be some opportunity to specify a particular kernel tobeloaded at this stage, which may be useful if a new kernel has just been generatedanddoesn't work, or if the system has multiple kernels available with different configurationsfor different purposes. (Some systems may boot different configurations automatically, depending on what hardware has been found in earlier steps.)
- For dual-boot or multiple-boot systems, the boot program will give the user an opportunityto specify a particular OS to load, with a default choice if the user does not pickaparticular OS within a given time frame. The boot program then finds the boot loader forthe chosen single-boot OS, and runs that program as described in the previous bullet point.
- Once the kernel is running, it may give the user the opportunity to enter into single-usermode, also known as maintenance mode. This mode launches very fewif anysystemservices, and does not enable any logins other than the primary log in on the console. This mode is used primarily for system maintenance and diagnostics.