

Chapter 1 - Introduction

Operating System

An Operating System (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs. An Operating System is a program that acts as an interface between applications and the computer hardware.

Operating system goals:

- Execute user programs and make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner

Gary Arlen Kildall is known as Father of Operating System. He developed, CP/M Disk Operating system. In 1973, Kildall wrote a software control program CP/M, which allowed files to be read and written to and from an eight inch floppy disk -- the first disk operating system for a microcomputer.

History of Operating System

- The earliest computers were mainframes that lacked any form of operating system. Each user had sole use of the machine for a scheduled period of time and would arrive at the computer with a program and data, often on punched paper cards and magnetic or paper tape.
- First operating System GMOS was developed by General motors in 1950's for IBM computer. GMOS was based on single stream batch processing system, because it collects all similar jobs in groups or batches and then submits the jobs to the operating system using a punch card to complete all jobs in a machine.
- UNIX was developed by AT&T Corporation's Bell Laboratories in the late 1960s as a result of efforts to create a time-sharing computer system. In 1969 a team led by computer scientists Ken Thompson and Dennis Ritchie created the first version of UNIX on a PDP-7 minicomputer.
- American computer programmer Timothy Paterson, wrote the original operating system for the Intel Corporation's 8086 microprocessor in 1980, initially calling it QDOS (Quick and Dirty Operating System), which was soon renamed 86-DOS. A year later, Microsoft owned and renamed Microsoft Disk Operating System (MS-DOS) as PC Operating System.
- Microsoft's Windows operating system was first introduced in 1985.

- While still a student at the University of Helsinki, Linus Torvalds started developing an operating system. In 1991 he released version 0.02; Version 1.0 of the Linux kernel, the core of the operating system, was released in 1994.
- BharOS is a mobile operating system designed by IIT Madras in 2023. It is an Indian government-funded project to develop a free and open-source operating system (OS) for use in government and public systems.

Flavours of Operating Systems

Non-Unix Operating Systems

- DOS
- Windows
- Novell Netware - computer network operating system developed by Novell
- MacOS (Apple)

Unix / Linux Varieties

- Sun Solaris
- Digital Unix
- IBM AIX
- HP-UX
- SCO unix
- Fedora
- Linux Mint
- openSUSE
- Debian

Mobile OS

- Symbian OS,
- iPhone OS,
- RIM's BlackBerry,
- Windows Mobile
- Palm WebOS
- Android
- Maemo (Android, WebOS, and Maemo are all derived from Linux).
- Android - until recently, all Android operating systems are named after desserts. First version of Android was developed in 23rd September 2008. Some dessert name for android versions is given in Table 1.

Table 1. Android Version Names

Ver.No	Name
1	Apple pie
1.5	Cupcake
1.6	Donut
2	Éclair
2.2	Froyo
2.3	Gingerbread
3	Honeycomb
4	Ice Cream Sandwich
4.1	Jelly Bean
4.4	KitKat
5	Lollipop
6	Marshmallow
7	Nougat
8	Oreo
9	Pie
10	Quince Tart
11	Redvelvet Cake
12	Snow Cone
13	Tiramisu (Coffee Flavoured Italian Desert)
14	Upside Down cake
15	Vanilla Ice cream

Computer-System Organization

Computer system consists of one or more CPUs, peripheral devices (Peripherals are commonly divided into three kinds: input devices, output devices, and storage devices) and a number of device controllers connected through a common bus that provides access to shared memory is shown in Figure 1.1.

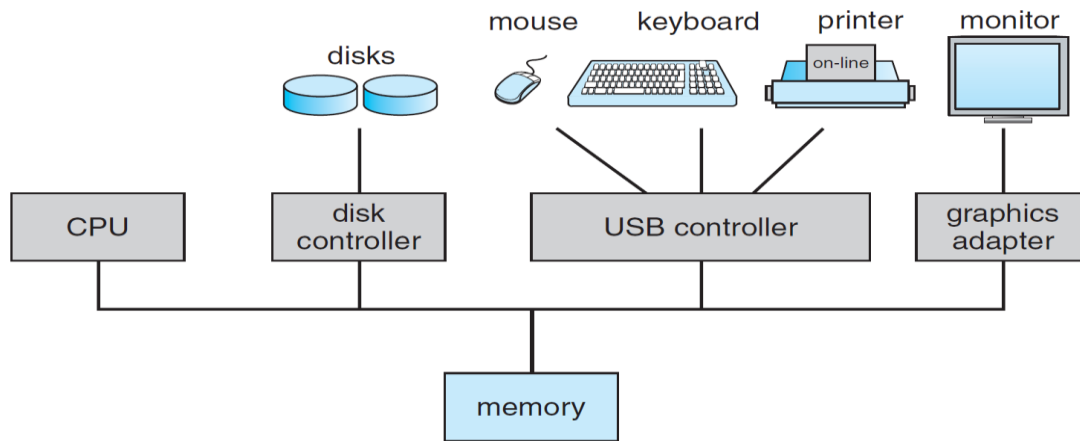


Figure 1.1. A modern computer system.

The salient features are

- Computer system can be divided into four components: Hardware, Operating system, Application programs, and Users
- The I/O devices and the CPU both execute concurrently. Some of the processes are scheduled for the CPU and at the same time, some processes undergo input/output operations.
- There are multiple device controllers, each in charge of a particular device such as keyboard, mouse, printer etc. Example: SCSI (Small Computer System Interface) and IDE (Integrated Drive Electronics) are both computer data storage interfaces.
- There is buffer available for each of the devices. The Device Controller receives the data from a connected device and stores it temporarily in some special purpose registers (i.e. local buffer) inside the controller. The input and output data can be stored in these buffers.
- The data is moved from memory to the respective device buffers by the CPU for I/O operations and then this data is moved back from the buffers to memory.
- The device controllers use an interrupt to inform the CPU that I/O operation is completed.
- When a computer system is powered up or rebooted, it needs to have an initial program to run. This initial program, or bootstrap program, tends to be simple. Typically, it is stored within the computer hardware in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM), known by the general term firmware. Operating can be installed in Hard Disk Drive (HDD) or Solid State Drive (SSD). SSD is a type of hard disk that uses flash memory, read / write operation is faster than HDD, Hence SSD has Shortest Boot time.
- The bootstrap program must locate the operating-system kernel and load it into memory. Once the kernel is loaded and executing, it can start providing services to the system and its users. System programs that are loaded into memory at boot time to become system processes.

- The occurrence of an event is usually signalled by an interrupt from either the hardware or the software. Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus. Software may trigger an interrupt by executing a special operation called a system call (also called a monitor call).
- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location. The fixed location usually contains the starting address where the service routine for the interrupt is located. The interrupt service routine executes; on completion, the CPU resumes the interrupted computation.

Storage Structure

- A Register is a fast memory used to accept, store, and transfer data and instructions that are being used immediately by the CPU.
- The cache is a smaller and faster memory that stores copies of the data from frequently used main memory locations.
- The CPU can load instructions only from memory, so any programs to run must be stored there. General-purpose computers run most of their programs from rewritable memory, called main memory.
- Main memory is usually too small to store all needed programs and data. 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.

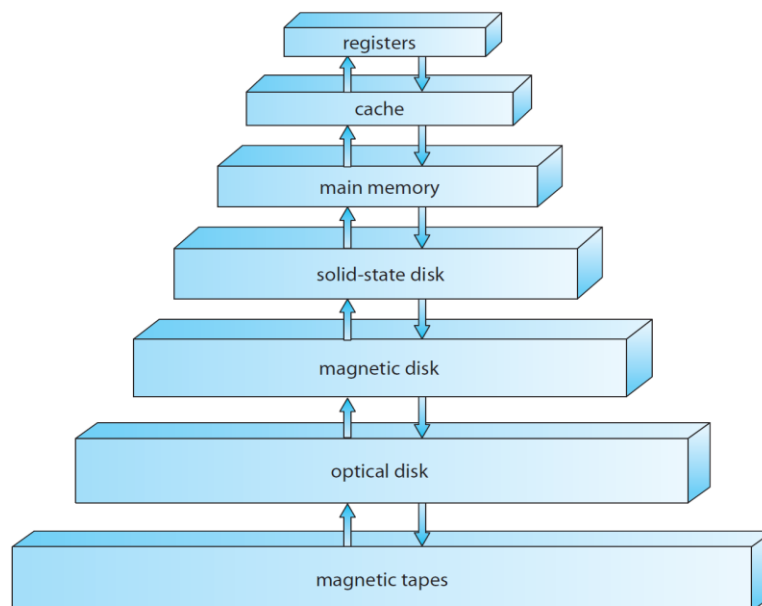


Figure 1.2. Storage Structures

- The Figure 1.2 describes Storage Structures. The basic unit of computer storage is the bit. A bit can contain one of two values, 0 and 1. A byte is 8 bits, and on most computers, it is the smallest convenient chunk of storage. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time. Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.

A kilobyte, or KB, is 1,024 bytes

A megabyte, or MB, is 1,024 KB

A gigabyte, or GB, is 1,024 MB

A terabyte, or TB, is 1,024 GB

A petabyte, or PB, is 1,024 TB

Computer-System Architecture

A computer system can be organized in a number of different ways. Here it is categorized according to the number of general-purpose processors used. They are classified as Single-Processor Systems, Multiprocessor Systems, and Clustered Systems. Working of Modern Computer is shown in Figure 1.3.

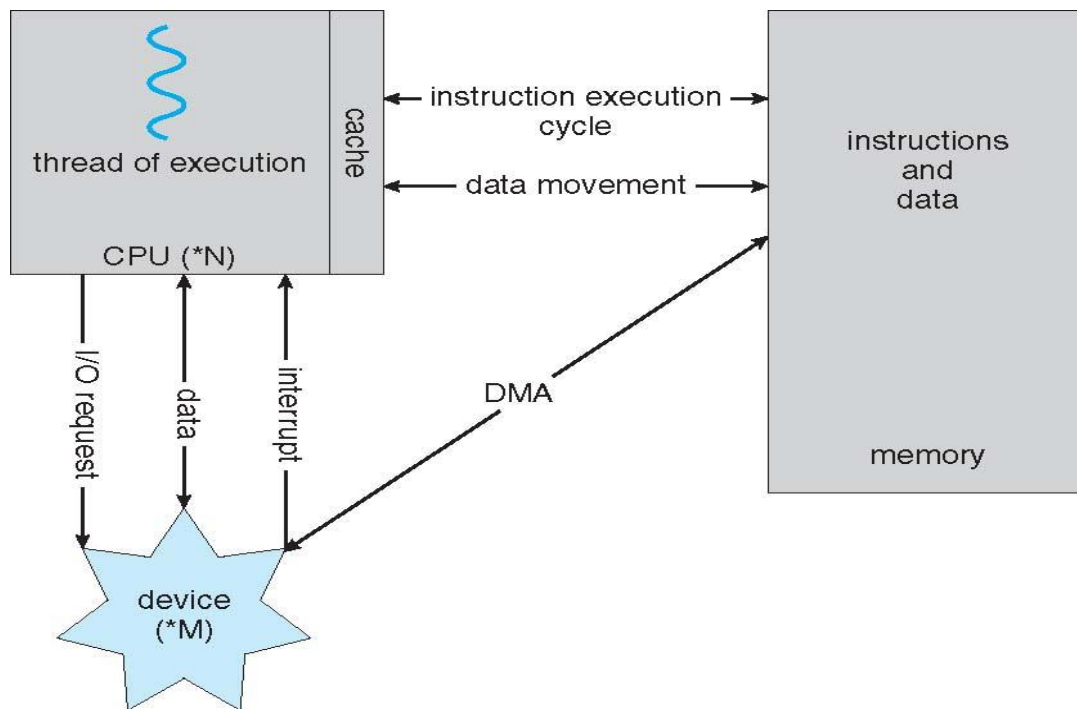


Figure 1.3. A von Neumann architecture

Single-Processor Systems: A single processor system contains only one processor. So only one process can be executed at a time and then the process is selected from the ready queue. Until recently, Most general purpose computers contain the single processor systems as shown in figure 1.4. Example: Most of Modern PCs.

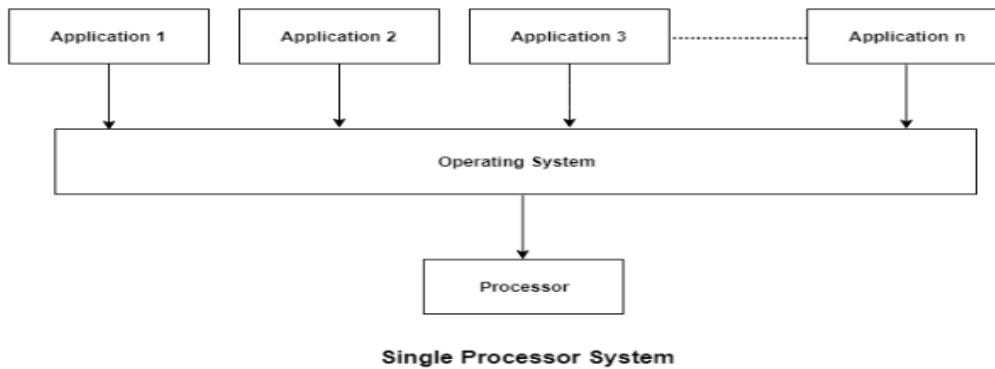


Figure 1.4. Single Processor System

Multiprocessor Systems:

- If a System contains two or more than two processors for processing than it is called multiprocessor system as shown in figure 1.5.
- In Multi Processor Systems Two Types of approaches are Used:
 - 1) Symmetric Multiprocessing (SMP)
 - 2) Asymmetric Multiprocessing (AMP)
- In Asymmetric Multiprocessing one Processor works as Master and Second Processor act as Slave and in Symmetric Multiprocessing each processor performs all the tasks within the operating system.
- Example: Blade Servers. The Blade servers consists of multiple processor boards, I/O boards, and networking boards that are placed in the same chassis. The difference between these and traditional multiprocessor systems is that each blade-processor board boots independently and runs its own operating system.
- Multiprocessor systems have three main advantages:
 1. Throughput: Throughput of Multiprocessor systems is greater than single processor systems. If a System Contains N Processors than its throughput will be slightly less than N because synchronization must be maintained between two processors and they also share resource which increases certain amount of overhead.
 2. Economy of scale: Multiprocessor Systems cost less than equivalent multiple single processor systems because they use same resources on sharing basis.
 3. Increased reliability: More reliable because failure of one processor does not halt the entire system but only speed will be slow down. No of Cores and their names is shown in Table 2.

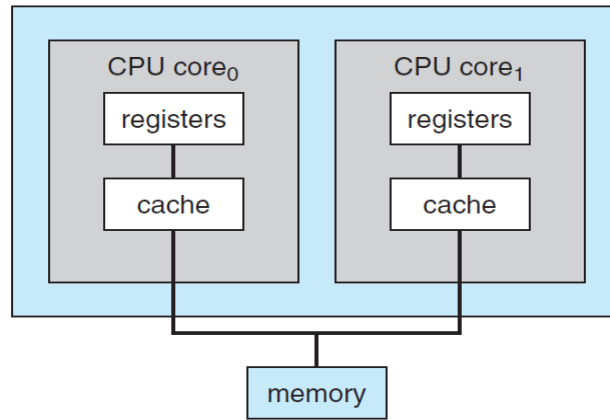


Figure 1.5. A dual-core design with two cores placed on the same chip.

Table 2. No of Cores and their names

No. of Core	Name
1	Single core
2	Dual Core
4	Quad Core
6	Hexa Core
8	Octa
10	Deca
12	Do deca
13	Tri deca
14	Tetra deca
15	Penta deca

Clustered Systems:

Another type of multiprocessor system is a clustered system, which gathers together multiple CPUs. Clustered systems differ from the multiprocessor systems in that they are composed of two or more individual systems or nodes joined together. Such systems are considered loosely coupled. Each node may be a single processor system or a multicore system. Figure 1.6. Shows General structure of a clustered system.

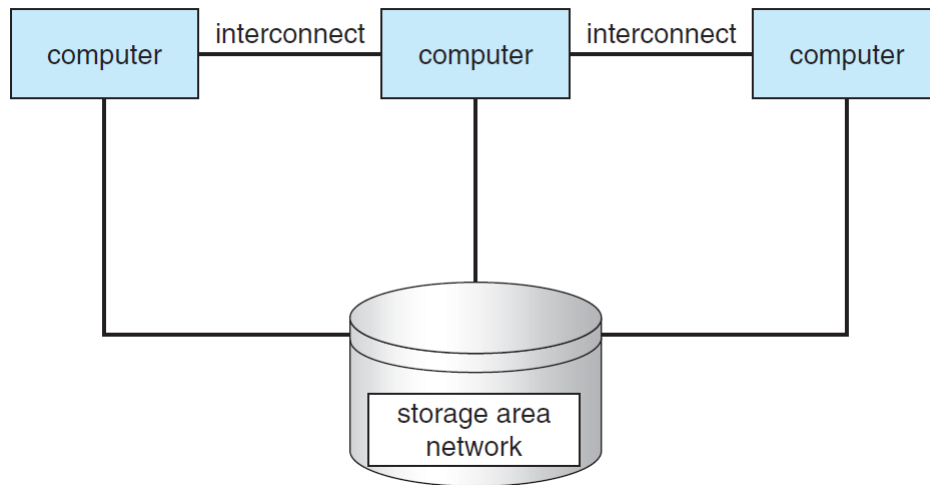


Figure 1.6. General structure of a clustered system.

- Clustering is usually used to provide high-availability service—that is, service will continue even if one or more systems in the cluster fail. A layer of cluster software runs on the cluster nodes. Each node can monitor one or more of the others (over the LAN). If the monitored machine fails, the monitoring machine can take ownership of its storage and restart the applications that were running on the failed machine.
- 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 clustering, two or more hosts are running applications and are monitoring each other. This structure is obviously more efficient, as it uses all of the available hardware. However it does require that more than one application be available to run.
- Since a cluster consists of several computer systems connected via a network, clusters can also be used to provide high-performance computing environments. However, this involves a technique known as parallelization, which divides a program into separate components that run in parallel on individual computers in the cluster.
- Parallel clusters allow multiple hosts to access the same data on shared storage. To provide this shared access, the system must also supply access control and locking to ensure that no conflicting operations occur. This function, commonly known as a distributed lock manager (DLM), is included in some cluster technology.
- Some cluster products support dozens of systems in a cluster, as well as clustered nodes that are separated by miles. Many of these improvements are made possible by storage-area networks (SANs).

- Storage-area networks (SANs), which allow many systems to attach to a pool of storage. If the applications and their data are stored on the SAN, then the cluster software can assign the application to run on any host that is attached to the SAN. If the host fails, then any other host can take over. In a database cluster, dozens of hosts can share the same database, greatly increasing performance and reliability.

Operating-System Structure

An operating system provides the environment within which programs are executed. One of the most important aspects of operating systems is the ability to multiprogram. A single program cannot, in general, keep either the CPU or the I/O devices busy at all times. Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.

A subset of total jobs in system is kept in memory is shown in figure 1.7. If processes don't fit in memory, OS moves them in and out from primary to secondary (swapping). Virtual memory allows execution of processes not completely in memory. When more than one job is ready for execution, one job selected through job scheduling. When that job needs to perform I/O operation, CPU switches to another job. Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing.

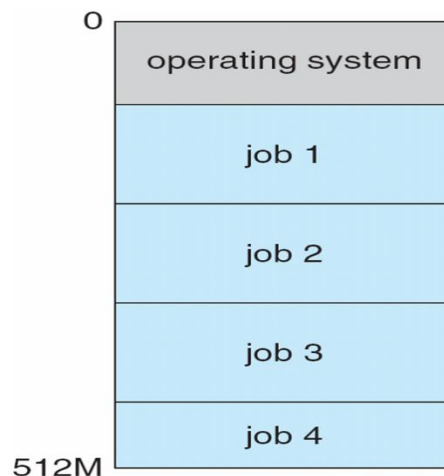


Figure 1.7. Memory Layout for Multiprogrammed System

Operating-System Operations

Modern operating systems are interrupt driven. Events are almost always signalled 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. 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 to deal with the interrupt.

Dual-Mode and Multimode Operation:

Since the operating system and users share the hardware and software resources of the computer system, The dual mode operation in the OS ensures the proper execution of the operating system and shields it from illegal users. We accomplish 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. At the very least, we need two separate modes of operation: user mode and kernel mode (also called supervisor mode, system mode, or privileged mode).

A bit, called the mode bit, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1). With the mode bit, we can distinguish between a task that is executed on behalf of the operating system and one that is executed on behalf of the user. When the computer system is executing 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 a system call), the system must transition from user to kernel mode to fulfil the request. This is shown in Figure 1.8. At system boot time, the hardware starts in kernel mode. The operating system is then loaded and starts user applications in user mode. Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode. Increasingly CPUs support multi-mode operations i.e. virtual machine manager (VMM) mode for guest VMs.

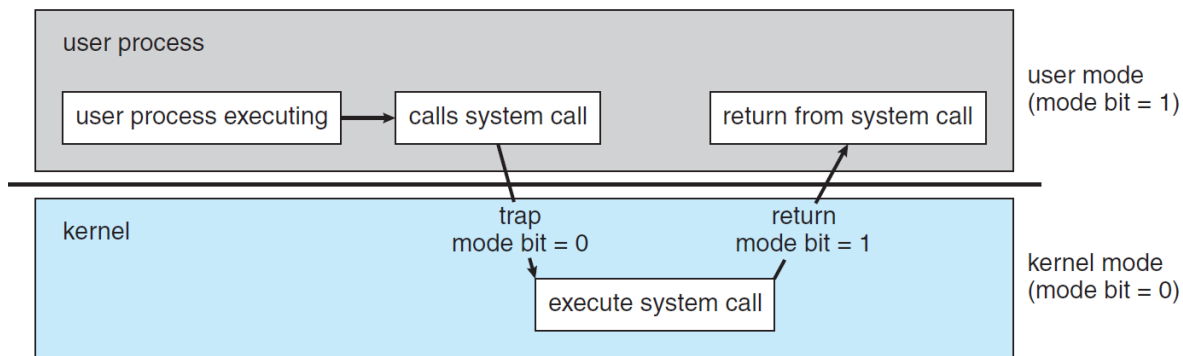


Figure 1.8. Transition from user to kernel mode.

The major operations of the operating system are process management, memory management, device management and file management. It is shown in figure 1.9.

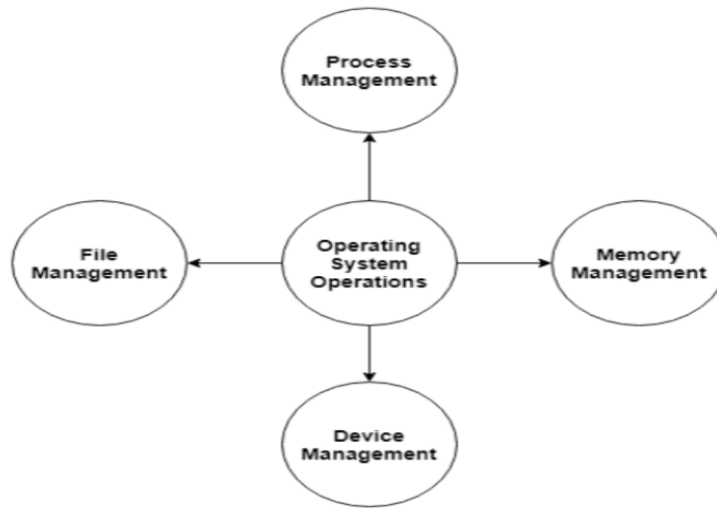


Figure 1.9. Operating-System Operations.

Process Management

The operating system is responsible for managing the processes i.e assigning the processor to a process at a time. This is known as process scheduling. The different algorithms used for process scheduling are FCFS (first come first served), SJF (shortest job first), priority scheduling, round robin scheduling etc.

A program in execution, is a process. A time-shared user program such as a compiler is a process. A word-processing program being run by an individual user on a PC is a process. A process needs certain resources—including CPU time, memory, files, and I/O devices—to accomplish its task. These resources are either given to the process when it is created or allocated to it while it is running.

A program is a passive entity, like the contents of a file stored on disk, whereas a process is an active entity. A single-threaded process has one program counter specifying the next instruction to execute. The execution of such a process must be sequential. The CPU executes one instruction of the process after another, until the process completes. A multithreaded process has multiple program counters, each pointing to the next instruction to execute for a given thread.

A process is the unit of work in a system. A system consists of a collection of processes, some of which are operating-system processes (those that execute system code) and the rest of which are user processes (those that execute user code). All these processes can potentially execute concurrently—by multiplexing on a single CPU, for example.

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

- Scheduling processes and threads on the CPUs
- 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 bytes, ranging in size from hundreds of thousands to billions. Memory Management deals with moving of processes from disk to primary memory for execution and back again.

The activities performed by the operating system for memory management are

- The operating system assigns memory to the processes as required. This can be done using best fit, first fit and worst fit algorithms.
- Keeping track of which parts of memory are currently being used and who is using them.
- Deallocating memory space as needed. This may happen when a process has been terminated or if it no longer needs the memory.

Storage Management

Storage devices define a logical storage unit known as file. Files are used to provide a uniform view of data storage by the operating system. All the files are mapped onto physical devices that are usually non volatile, so data is safe in the case of system failure.

File Management

File management is one of the most visible components of an operating system. Computers can store information on several different types of physical media. Magnetic disk, optical disk, and magnetic tape are the most common. Each of these media has its own characteristics and physical organization. The files can be accessed by the system in two ways i.e. sequential access and direct access.

- Sequential Access: The information in a file is processed in order using sequential access. The files records are accessed one after another. Most of the file systems such as editors, compilers etc. use sequential access.
- Direct Access: In direct access or relative access, the files can be accessed in random for read and write operations. The direct access model is based on the disk model of a file, since it allows random accesses.

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

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.

The operating system is responsible for the following activities in connection with disk management / Mass-Storage Management

- Free-space management
- Storage allocation
- Disk scheduling

Caching

Cache Memory is a special very high-speed memory. The cache is a smaller and faster memory that stores copies of the data from frequently used main memory locations. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. Because caches have limited size, cache management is important. Performance of various levels of storage is shown in Table 3.

Table 3. Performance of various levels of storage.

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

I/O Device Management

There are many I/O devices handled by the operating system such as mouse, keyboard, disk drive etc. There are different device drivers that can be connected to the operating system to handle a specific device. The device controller is an interface between the device and the device driver. The user applications can access all the I/O devices using the device drivers, which are device specific codes.

Protection and Security

If a computer system has multiple users and allows the concurrent execution of multiple processes, then access to data must be regulated. Protection, is any mechanism for controlling the access of processes or users to the resources defined by a computer system. Protection tackles the system's internal threats. In simple words, It specifies which files a specific user can access or view and modify to maintain the proper functioning of the system. It allows the safe sharing of common physical address space or common logical address space which means that multiple users can access the memory due to the physical address space.

Security, defend a system from external and internal attacks. Such attacks spread across a huge range and include viruses and worms, denial-of service attacks (which use all of a system's resources and so keep legitimate users out of the system), identity theft, and theft of service (unauthorized use of a system). Prevention of some of these attacks is considered an operating-system function on some systems, while other systems leave it to policy or additional software.

Some Methods to Ensure Protection and Security in Operating System

- 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 unique user identifiers (user IDs one per user).
- The authentication stage determines the appropriate user ID for the user. That user ID is associated with all of the user's processes and threads.
- In some circumstances, a group name and the set of users belonging to that group can be defined. Group functionality can be implemented as a system-wide list of group names and group identifiers. A user can be in one or more groups, depending on operating-system design decisions. The user's group IDs are also included in every associated process and thread.
- However, a user sometimes needs to escalate privileges to gain extra permissions for an activity. Operating systems provide various methods to allow privilege escalation. On UNIX, for instance, the `setuid` attribute on a program causes that program to run with the user ID of the owner of the file, rather than the current user's ID. The process runs with this effective UID until it turns off the extra privileges or terminates.

Kernel Data Structures

Following data structures are used as Kernel Data Structures in operating system.

1. Lists, Stacks, and Queues
2. Trees
3. Hash Functions and Maps

4. Bitmaps

Lists, Stacks, and Queues

An array is a simple data structure in which each element can be accessed directly. For example, main memory is constructed as an array. If the data item being stored is larger than one byte, then multiple bytes can be allocated to the item, and the item is addressed as $\text{item number} \times \text{item size}$. But what about storing an item whose size may vary? And what about removing an item if the relative positions of the remaining items must be preserved? In such situations, arrays give way to other data structures.

The most common method for implementing this structure is a linked list, in which items are linked to one another. Linked lists are of several types: singly linked list, doubly linked list, and circularly linked list. In a singly linked list, each item points to its successor. It is shown in figure 1.10 (a).

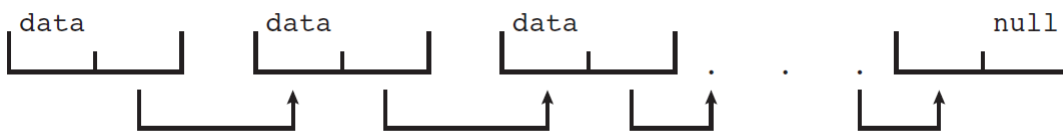


Figure 1.10. (a) Singly Linked List

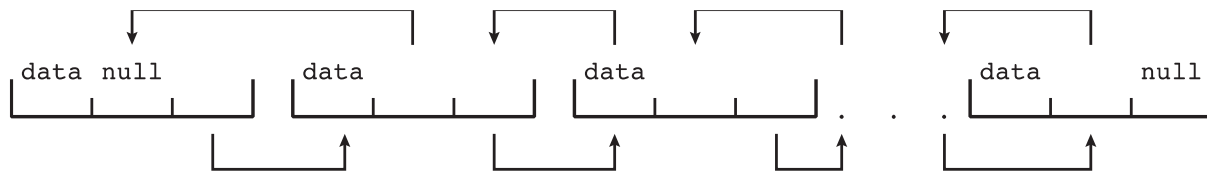


Figure 1.10. (b) Doubly Linked List

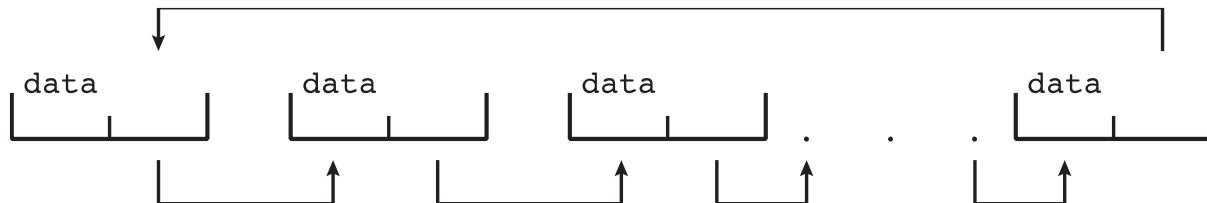


Figure 1.10. (c) Circular Linked List

In a doubly linked list, a given item can refer either to its predecessor or to its successor. It is shown in Figure 1.10 (b). In a circularly linked list, the last element in the list refers to the first element, rather than to null. It is shown in Figure 1.10 (c).

A stack is a sequentially ordered data structure that uses the last in, first out (LIFO) principle for adding and removing items, meaning that the last item placed onto a stack is the first item removed. The operations for inserting and removing items from a stack are known as push and pop, respectively. An operating system often uses a stack when invoking function calls. Parameters, local variables, and the return address are pushed onto the stack when a function is called; returning from the function call pops those items off the stack.

A queue, in contrast, is a sequentially ordered data structure that uses the first in, first out (FIFO) principle: items are removed from a queue in the order in which they were inserted. There are many everyday examples of queues, including shoppers waiting in a checkout line at a store and cars waiting in line at a traffic signal. Queues are also quite common in operating systems—jobs that are sent to a printer are typically printed in the order in which they were submitted.

Trees

A tree is a data structure that can be used to represent data hierarchically. Data values in a tree structure are linked through parent–child relationships. In a general tree, a parent may have an unlimited number of children

Hash Functions and Maps

A hash function takes data as its input, performs a numeric operation on this data, and returns a numeric value. This numeric value can then be used as an index into a table (typically an array) to quickly retrieve the data. Whereas searching for a data item through a list of size n can require up to $O(n)$ comparisons in the worst case, using a hash function for retrieving data from table can be as good as $O(1)$ in the worst case, depending on implementation details. Because of this performance, hash functions are used extensively in operating systems.

Bitmaps

A bitmap is a string of n binary digits that can be used to represent the status of n items. For example, suppose we have several resources, and the availability of each resource is indicated by the value of a binary digit: 0 means that the resource is available, while 1 indicates that it is unavailable (or vice-versa).

Computing Environments

In computing environments computer devices are arranged in different ways and they exchange information in between them to process and solve problem. The different types of computing technologies that are used in OS are as follows,

1. Traditional computing
2. Mobile Computing
3. Distributed Computing
4. Client-Server computing
5. Peer to peer network
6. Virtualization
7. Cloud computing
8. Real-Time Embedded Systems

Traditional computing

In traditional computing the user can use a traditional method like static memory allocation and it is mainly useful in single user operating systems. In this technique there will be a particular operating system that is going to perform all tasks to that particular computer system.

It is like one task is performed by the CPU at a given time and the CPU utilizes the memory that is used only for one task. Now-a-days the traditional computing uses the CPU scheduling method and it is used in desktop, server and other different computers so that the user can manage and give a slice of computer memory to other processes.

Mobile computing

Mobile computing refers to computing on handheld smartphones and tablet computers. These devices share the distinguishing physical features of being portable and lightweight. Historically, compared with desktop and laptop computers, mobile systems gave up screen size, memory capacity, and overall functionality in return for handheld mobile access to services such as e-mail and web browsing.

Distributed Computing

In a distributed computing environment multiple nodes are connected together using network but physically they are separated. A single task is performed by different functional units of different nodes of distributed unit. Here different programs of an application run simultaneously on different nodes, and communication happens in between different nodes of this system over network to solve task..

Client-server computing

In client-server computing there is a technology available by using a client and server that are connected with each other with the help of a network. When some process or program transfers from client to server or server to client then the memory and I/O calculations is called client-server computing. Basically in the internet the user can use this technology and a network is used to connect the server with the client. So, that the client and server are connected with each other for client-server computing. The figure 1.11 shows client-server computing.

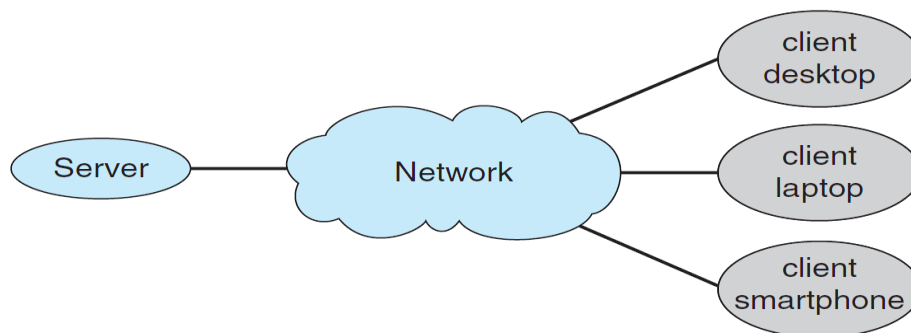


Figure 1.11. Client-Server Computing

Peer-to-Peer Computing

All the nodes at the Network have an equal relationship with others and have a similar type of software that helps the sharing of resources. A Peer to Peer computing System allows two or more computers to share their resources, along with printers, scanners, CD-ROM, etc., to be accessible from each computer. It is shown in figure 1.12.

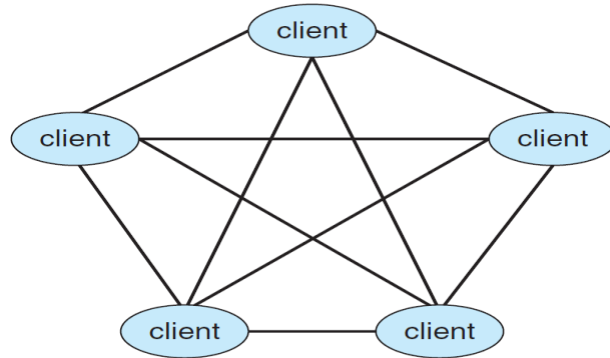


Figure 1.12. Peer-to-Peer Computing

Virtualization

Virtualization is a technology that allows operating systems to run as applications within other operating systems. Virtualization is one member of a class of software that also includes emulation. Emulation is used when the source CPU type is different from the target CPU type. A common example of emulation occurs when a computer language is not compiled to native code but instead is either executed in its high-level form or translated to an intermediate form. This is known as interpretation.

Cloud computing

In cloud computing environment on demand availability of computer system resources like processing and storage are availed. Here computing is not done in individual technology or computer rather it is computed in cloud of computers where all required resources are provided by cloud vendor. There are actually many types of cloud computing. 1. Public cloud: A cloud available via the Internet to anyone willing to pay for the services. 2. Private cloud: A cloud run by a company for that company's own use. 3. Hybrid cloud: A cloud that includes both public and private cloud components. This environment primarily comprised of three services i.e software-as-a-service (SaaS), infrastructure-as-a-service (IaaS), and platform-as-a-service (PaaS).

Real-Time Embedded Systems

Real time systems are those systems that work within strict time constraints and provide a worst case time estimate for critical situations. Embedded systems provide a specific function in a much larger system. When there is an embedded component in a real time system, it is known as a real time embedded system.

Types of Real Time Embedded Systems are 1. Hard Real Time Embedded System, and 2. Soft Real Time Embedded System

Hard Real Time Embedded System

This type of system makes sure that all critical processes are completed within the given time frame. This means that all the delays in the system are strictly time bound. Also, there is little to no secondary memory and data is stored in short term memory or read only memory. Hard real time systems are used in various areas such as missiles, airplanes etc.

Soft Real Time Embedded System

These are much less constrictive than hard real time systems but the basic premise is the same i.e critical processes need to be completed within the given time frame. However, this time frame can be a little flexible. Soft real time systems are used in various areas such as multimedia, scientific projects etc.

Open-Source Operating Systems

Open-source operating systems are those available in source-code format rather than as compiled binary code. Linux is the most famous opensource operating system, while Microsoft Windows is a well-known example of the opposite closed-source approach. Apple's Mac OS X and iOS operating systems comprise a hybrid approach. They contain an open-source kernel named Darwin yet include proprietary, closed-source components as well.

Digital Rights Management (DRM) refers to a set of technologies that enable the control of what the reader can do with the content. Also commonly known as "copy protection" or "copy prevention", DRM software enables the sender to prevent the unintended spread of such information. The operating system works together with drivers to enforce these rules.

To counter the move to limit software use and redistribution Richard Stallman formed the Free Software Foundation (FSF) with the goal of encouraging the free exchange of software source code and the free use of that software. Rather than copyright its software, the FSF "copyleft" the software to encourage sharing and improvement. The GNU General Public License (GPL) codifies copylefting and is a common license underwhich free software is released. Examples include GNU/Linux and BSD UNIX (including core of Mac OS X), and many more Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - <http://www.virtualbox.com>)

GNU/ LINUX

GNU/Linux.

The GNU project produced many UNIX-compatible tools, including compilers, editors, and utilities, but never released a kernel. In 1991, a student in Finland, Linus Torvalds, released a UNIX-like kernel (Linux

operating system)using the GNU compilers and tools and invited contributions worldwide. Major distributions forms include RedHat, SUSE, Fedora, Debian, Slackware, and Ubuntu. Distributions vary in function, utility, installed applications, hardware support, user interface, and purpose. For example, RedHat Enterprise Linux is geared to large

commercial use. PCLinuxOS is a LiveCD—an operating system that can be booted and run from a CD-ROM without being installed on a system’s hard disk. One variant of PCLinuxOS—called “PCLinuxOS Supergamer DVD”—is a LiveDVD that includes graphics drivers and games. A gamer can run it on any compatible system simply by booting from the DVD. When the gamer is finished, a reboot of the system resets it to its installed operating system.

You can run Linux on a Windows system using the following simple, free approach:

1. Download the free “VMware Player” tool from <http://www.vmware.com/download/player/> and install it on your system.
2. Choose a Linux version from among the hundreds of “appliances,” or virtual machine images, available from VMware at <http://www.vmware.com/appliances/>. These images are preinstalled with operating systems and applications and include many flavors of Linux.
3. Boot the virtual machine within VMware Player.

BSD UNIX

BSD UNIX has a longer and more complicated history than Linux. It started in 1978 as a derivative of AT&T’s UNIX. Releases from the University of California at Berkeley (UCB) came in source and binary form, but they were not opensource because a license from AT&T was required. BSD UNIX’s development was slowed by a lawsuit by AT&T, but eventually a fully functional, open-source version, 4.4BSD-lite, was released in 1994.

Solaris is the commercial UNIX-based operating system of Sun Microsystems. Originally, Sun’s SunOS operating system was based on BSD UNIX. Sun moved to AT&T’s System V UNIX as its base in 1991. In 2005, Sun open-sourced most of the Solaris code as the Open Solaris project.

Operating-System Services

The various operating system services are [A view of operating system services are shown in figure 1.13]:

1. Program Execution
2. I/O Operations
3. File System Manipulation
4. Communications
5. Error Detection

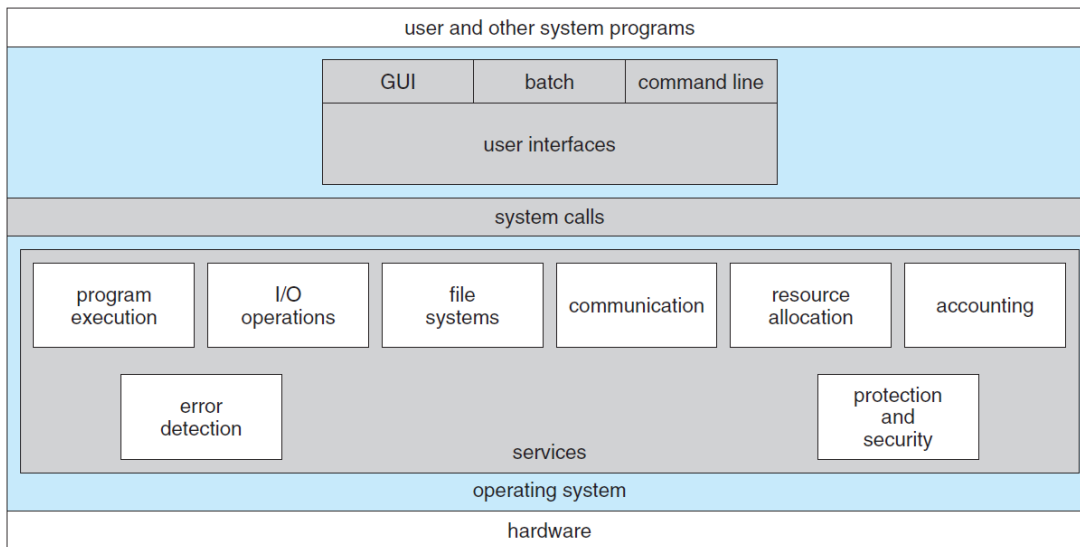


Figure 1.13. A view of operating system services

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

Each program requires an input and produces output. This involves the use of I/O. The operating systems hides the user the details of underlying hardware for the I/O. All the user sees is that the I/O has been performed without any details. So the operating systems by providing I/O makes it convenient for the users to run programs. For efficiency and protection users cannot control I/O so this service cannot be provided by user-level programs.

File System Manipulation

The output of a program may need to be written into new files or input taken from some files.

The operating systems provides this service. The user does not have to worry about secondary storage management. User gives a command for reading or writing to a file and sees his/her task accomplished. Thus operating systems makes it easier for user programs to accomplish their task. This service involves secondary storage management. The speed of I/O that depends on secondary storage management is critical to the speed of many programs.

Communications

There are instances where processes need to communicate with each other to exchange information. It may be between processes running on the same computer or running on the different computers. By providing this service the operating system relieves the user of the worry of passing messages between processes. In case where the messages need to be passed to processes on the other computers through a network it can

be done by the user programs. The user program may be customized to the specific hardware through which the message transits and provides the service interface to the operating system.

Error Detection

An error is one part of the system may cause malfunctioning of the complete system. To avoid such a situation the operating system constantly monitors the system for detecting the errors. This relieves the user of the worry of errors propagating to various part of the system and causing malfunctioning. This service cannot allowed to be handled by user programs because it involves monitoring and in cases altering area of memory or deallocation of memory for a faulty process. Or may be relinquishing the CPU of a process that goes into an infinite loop. These tasks are too critical to be handed over to the user programs. A user program if given these privileges can interfere with the correct (normal) operation of the operating systems.

Another set of OS functions exists for ensuring the efficient operation of the system itself.

Resource allocation: - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them. Many types of resources - CPU cycles, main memory, file storage, I/O devices.

Accounting: - To keep track of which users use how much and what kinds of computer resources.

Protection and security: - Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts.

User and Operating-System Interface

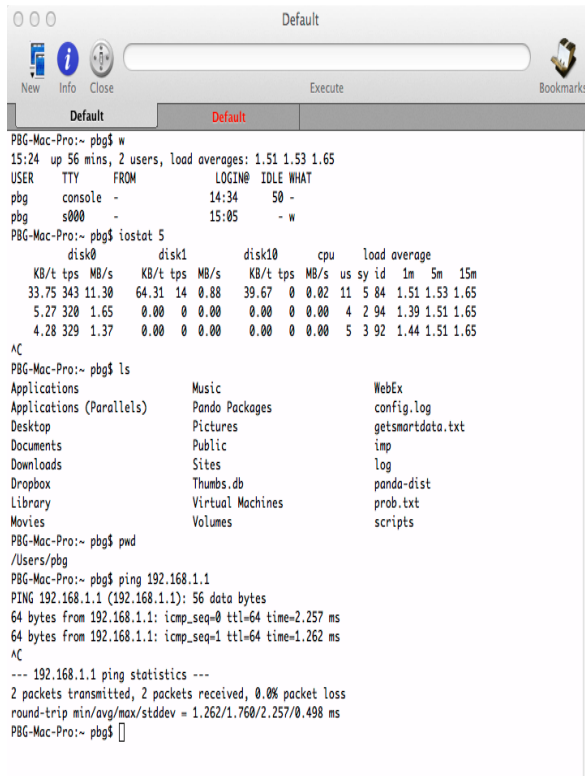
There are several ways for users to interface with the operating system as shown in figure 1.14.

1. Command-line interface, or command interpreter, that allows users to directly enter commands to be performed by the operating system.
2. Graphical User Interface(GUI) - that allows users to interface with the operating system via the functions are carried out by clicking or moving buttons, icons and menus by means of a pointing device.

The basic components of GUIs are –

- Start menu with program groups
- Taskbar which showing running programs
- Desktop screen

- Different icons and shortcuts.



```

PBG-Mac-Pro:~ pbg$ w
15:24 up 56 mins, 2 users, load averages: 1.51 1.53 1.65
USER      TTY      FROM          LOGIN@  IDLE WHAT
pbg       console  -             14:34   50  -
pbg       s000    -             15:05   -  w

PBG-Mac-Pro:~ pbg$ iostat 5

disk0      disk1      disk10     cpu        load average
KB/t tps MB/s  KB/t tps MB/s  KB/t tps MB/s  us sy id  1m  5m  15m
33.75 343 11.30 64.31 14 0.88 39.67 0 0.02 11 5 84 1.51 1.53 1.65
5.27 320 1.65 0.00 0 0.00 0.00 0 0.00 4 2 94 1.39 1.51 1.65
4.28 329 1.37 0.00 0 0.00 0.00 0 0.00 5 3 92 1.44 1.51 1.65

AC
PBG-Mac-Pro:~ pbg$ ls
Applications                               Music                               WebEx
Applications (Parallels)                  Pando Packages                     config.log
Desktop                                   Pictures                           getsmartdata.txt
Documents                                Public                             imp
Downloads                               Sites                             log
Dropbox                                 Thumbs.db                         panda-dist
Library                                Virtual Machines                  prob.txt
Movies                                 Volumes                          scripts

PBG-Mac-Pro:~ pbg$ pwd
/Users/pbg
PBG-Mac-Pro:~ pbg$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=2.257 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.262 ms
AC
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.262/1.760/2.257/0.498 ms
PBG-Mac-Pro:~ pbg$

```



Figure 1.14. Bourne Shell Command Interpreter and Touchscreen Interfaces

System Calls

System calls provide an interface between a running program and operating system. System calls are generally available as assembly language instructions. Several higher level languages such as C also allow to make system calls directly. It is as shown in figure 1.15.

How System Calls Work

- The Applications run in an area of memory known as user space.
- A system call connects to the operating system's kernel, which executes in kernel space.
- When an application creates a system call, it obtains permission from the kernel by raising interrupt request, which pauses the current process and transfers control to the kernel.
- If the request is permitted, the kernel returns the results kernel performs the requested action, then control moved from kernel space to user space.

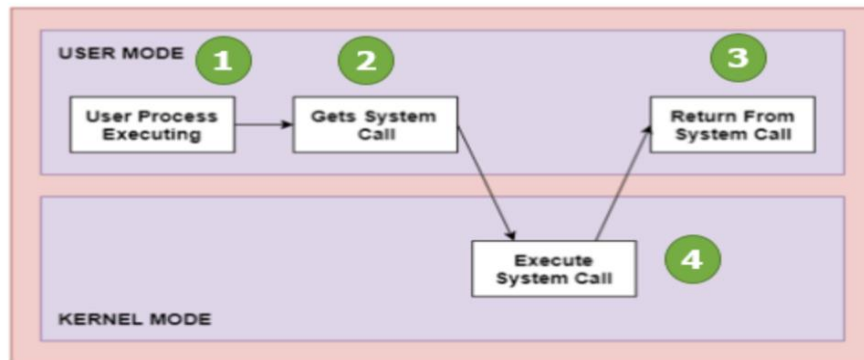


Figure 1.15. System call working model

There are three general methods that are used to pass information (parameters) between a running program and the operating system.

1. One method is to store parameters in registers.
2. Another is to store parameters in a table in memory and pass the address of table.
3. The third method is to push parameters on stack and allow operating system to pop the parameters off the stack.

Types of System Calls

System calls can be grouped roughly into six major categories:

- Process control
- File manipulation
- Device manipulation
- Information maintenance
- Communications
- Protection.

Process control

The following services are provided by Process Control System calls that are used to control a process.

- To forcefully abort the process, simply end it normally.
- Execute a process after loading it into the main memory.
- Terminate the current process before starting a new one.
- Wait for a process to complete running. Wait until a specific event happens, then announce it once it has.
- Allocate memory to a process, then release the memory if the process is terminated.

system calls are used to control the processes are

- end, abort
- load, execute
- create process, terminate process

- get process attributes, set process attributes
- wait for time
- wait event, signal event
- allocate and free memory

File manipulation

The following services are provided by system call for file management:

- Making and erasing files
- Open the file, then close it.
- Write to a specific file, read from a specific file.
- To obtain a file's attribute and to change a file's attribute

System calls for File manipulation are

- create file, delete file
- open, close
- read, write, reposition
- get file attributes, set file attributes

Device manipulation

The following services are offered by a system call that controls I/O devices:

- Devices might be needed while a process is running. such as access to the file system, I/O devices, main memory, etc.
- As a result, it can ask for a device and then release it once the task is complete.
- when a requested device is granted access by the process. It is capable of reading, writing, and repositioning operations.
- In order to obtain or modify a specific device's attribute.
- To detach a device from the processor that is currently executing a command, the call can be made.

System calls for Device manipulation are

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices

Information maintenance

The system calls for information maintenance call moves data from the user programme to the operating system. Considering this, the services offered by this type of system call are:

- Obtain the system's time or date. Set the system's time or date.
- Obtain system-related information. Configure the system data.

- Obtain the characteristics of a specific operating system process. Alternatively, of a specific file on the system or on any attached devices.
- Set the characteristics of a specific operating system process. Alternatively, of a specific file on the system or on any attached devices.

System calls for information maintenance are

- ◦ get time or date, set time or date
- ◦ get system data, set system data
- ◦ get process, file, or device attributes
- ◦ set process, file, or device attributes

Communications

Such a system call facilitates the system's network connection. The services that these system calls offer are:

- Open a fresh connection to send the data. After the transmission is finished, disconnect from the connection.
- On a particular connection, send a message. Obtain communication from a specific connection.
- Identify and connect a specific remote device to the network. Remove a specific remote computer or device from the network.

System calls for communications are

- create, delete communication connection
- send, receive messages
- transfer status information
- attach or detach remote devices

Protection.

Protection provides a mechanism for controlling access to the resources provided by a computer system. Historically, protection was a concern only on multiprogrammed computer systems with several users. Typically, system calls providing protection include set permission() and get permission(), which manipulate the permission settings of resources such as files and disks. The allow user() and deny user() system calls specify whether particular users can—or cannot—be allowed access to certain resources.

Table 4. System calls in windows and Linux OS

Types of System Calls	Windows	Linux
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Management	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()

Device Management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

System Programs

System programs provide a convenient environment for program development and execution. They can be divided into:

1. File manipulation
2. Status information sometimes stored in a File modification
3. Programming language support
4. Program loading and execution
5. Communications
6. Background services
7. Application programs

Most users' view of the operation system is defined by system programs, not the actual system calls.

File management - These programs create, delete, copy, rename, print, dump, list, and generally manipulate files and directories.

Status information - Some programs simply ask the system for the date, time, amount of available memory or disk space, number of users, or similar status information. Others are more complex, providing detailed performance, logging, and debugging information. Typically, these programs format and print the output to the terminal or other output devices or files or display it in a window of the GUI. Some systems also support a registry, which is used to store and retrieve configuration information.

File modification - Several text editors may be available to create and modify the content of files stored on disk or other storage devices. There may also be special commands to search contents of files or perform transformations of the text.

Programming-language support - Compilers, assemblers, debuggers, and interpreters for common programming languages (such as C, C++, Java, and PERL) are often provided with the operating system or available as a separate download.

Program loading and execution - Once a program is assembled or compiled, it must be loaded into memory to be executed. The system may provide absolute loaders, relocatable loaders, linkage editors, and

overlay loaders. Debugging systems for either higher-level languages or machine language are needed as well.

Communications - These programs provide the mechanism for creating virtual connections among processes, users, and computer systems. They allow users to send messages to one another's screens, to browse Web pages, to send e-mail messages, to log in remotely, or to transfer files from one machine to another.

Background services - background Programs that are used for system startup, terminate, and provide facilities like disk checking, process scheduling, error logging, printing. Constantly running system-program processes are known as services, subsystems, or daemons. They Run in user context not kernel context.

Application programs - most operating systems are supplied with programs that are useful in solving common problems or performing common operations. Such application programs include Web browsers, word processors and text formatters, spreadsheets, database systems, compilers, plotting and statistical-analysis packages, and games.

Operating-System Design and Implementation

Design Goals

The first problem in designing a system is to define goals and specifications. At the highest level, the design of the system will be affected by the choice of hardware and the type of system: batch, time sharing, single user, multiuser, distributed, real time, or general purpose.

There are basically two types of goals while designing an operating system. These are –

User Goals

The operating system should be convenient, easy to use, reliable, safe and fast according to the users. However, these specifications are not very useful as there is no set method to achieve these goals.

System Goals

The operating system should be easy to design, implement and maintain. These are specifications required by those who create, maintain and operate the operating system. But there is not specific method to achieve these goals as well.

Operating System Mechanisms and Policies

Mechanism shows how to do something and policy shows what to do. Policies may change over time and this would lead to changes in mechanism. So, it is better to have a general mechanism that would require few changes even when a policy change occurs.

For example - If the mechanism and policy are independent, then few changes are required in mechanism if policy changes. If a policy favours I/O intensive processes over CPU intensive processes, then a policy change to preference of CPU intensive processes will not change the mechanism.

Operating System Implementation

The operating system needs to be implemented after it is designed. Earlier they were written in assembly language but now higher level languages are used. The first system that was not written in assembly language was probably the Master Control Program (MCP) for Burroughs computers. MCP was written in a variant of ALGOL. MULTICS, developed at MIT, was written mainly in the system programming language PL/1.

Advantages of Higher Level Language

There are multiple advantages to implementing an operating system using a higher level language such as: the code is written more fast, it is compact and also easier to debug and understand. Also, the operating system can be easily moved from one hardware to another if it is written in a high level language (known as port).

Disadvantages of Higher Level Language

Using high level language for implementing an operating system leads to a loss in speed and increase in storage requirements. However in modern systems only a small amount of code is needed for high performance, such as the CPU scheduler and memory manager. Also, the bottleneck routines in the system can be replaced by assembly language equivalents if required.

Operating-System Structure

An operating system provides the environment within which programs are executed. System software is installed and utilized on top of the operating system. We can define operating system structure as to how different components of the operating system are interconnected. There are many structures of the operating system:

- Simple Structure – Ex: MS-DOS
- Monolithic Structure – Ex: UNIX
- Layered Structure – Ex: Windows XP, Linux
- Micro-kernels - Mac OS 8, Minix, L4Linux.
- Loadable kernel module or module – Ex: modern UNIX, Solaris, Mac OS X, Windows
- Hybrid Systems – Ex: Android

All these approaches or structures evolved from time to time, making the OS more and more improved than before.

Simple Structure - MS-DOS

It is the simplest Operating System Structure and is not well defined; It can only be used for small and limited systems. In this structure, the interfaces and levels of functionality are not well separated; hence user programs can access I/O routines which can cause unauthorized access to I/O routines. This structure is implemented in MS-DOS operating system: The MS-DOS operating System is made up of various layers, each with its own set of functions. A Diagram of the simple structure is shown in figure 1.16.

These layers are:

- Application Program
- System Program
- MS-DOS device drivers
- ROM BIOS device drivers

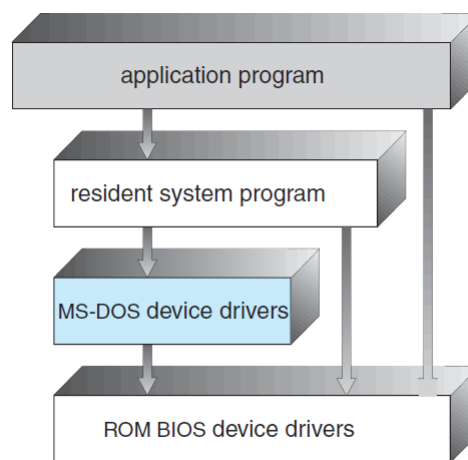


Figure 1.16. Simple OS structure.

Layering has an advantage in the MS-DOS operating system since all the levels can be defined separately and can interact with each other when needed.

Advantages of Simple Structure

- It is easy to develop because of the limited number of interfaces and layers.
- Offers good performance due to lesser layers between hardware and applications.

Disadvantages of Simple Structure

- If one user program fails, the entire operating system crashes.
- Abstraction or data hiding is not present as layers are connected and communicate with each other.
- Layers can access the processes going in the Operating System, which can lead to data modification and can cause Operating System to crash.
- Not efficient

Monolithic Approach – UNIX Operating System Structure

In Monolithic operating System, Kernel serves as the primary interface between the Operating System and the hardware. The main components of the Unix operating system structure are the kernel layer, the shell layer and the application layer. A Diagram of the Unix structure is shown in figure 1.17.

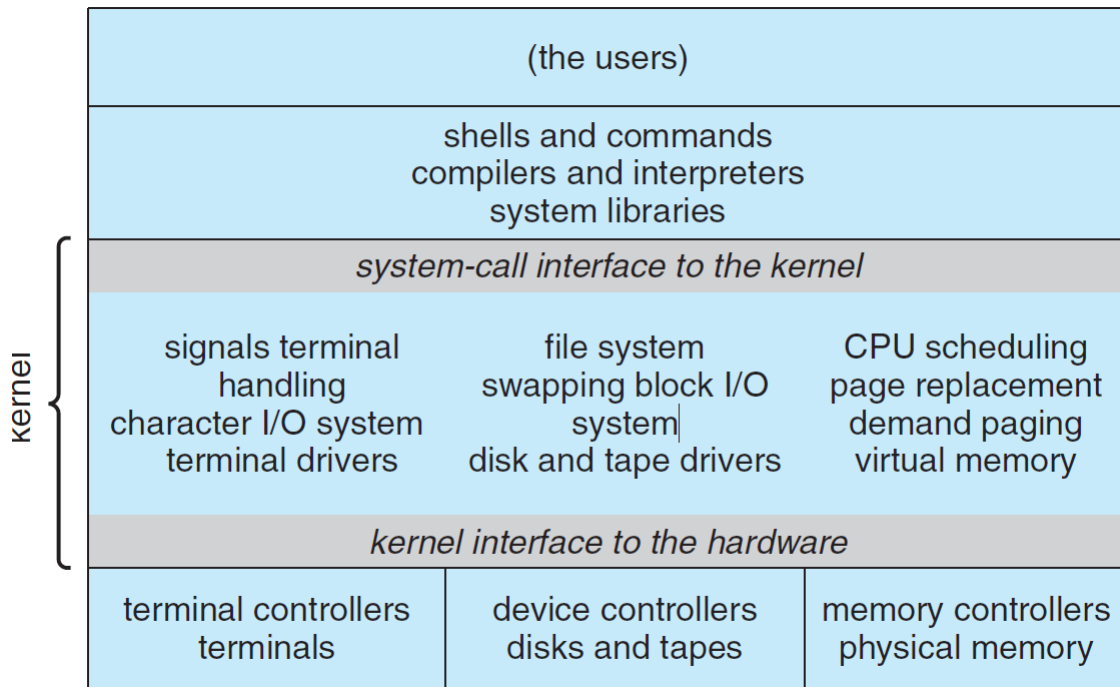


Figure 1.17. Monolithic – Unix OS structure

Kernel

The kernel provides a bridge between the hardware and the user. It is a software application that is central to the operating system. The kernel handles the files, memory, devices, processes and the network for the operating system. It is the responsibility of the kernel to make sure all the system and user tasks are performed correctly.

Shell

The program between the user and the kernel is known as the shell. It translates the many commands that are typed into the terminal session. These commands are known as the shell script. There are two major types of shells in Unix. These are Bourne shell and C Shell. The Bourne shell is the default shell for version 7 Unix. The character \$ is the default prompt for the Bourne shell. The C shell is a command processor that is run in a text window. The character % is the default prompt for the C shell.

Applications

The applications and utility layer in Unix includes the word processors, graphics programs, database management programs, commands etc. The application programs provide an application to the end users. For example, a web browser is used to find information while gaming software is used to play games. The requests for service and application communication systems used in an application by a programmer is known as an application program interface (API).

Advantages of Monolithic structure:

- It is simple to design and implement because all operations are managed by kernel only, and layering is not needed.
- As services such as memory management, file management, process scheduling, etc., are implemented in the same address space, the execution of the monolithic kernel is relatively fast as compared to normal systems. Using the same address saves time for address allocation for new processes and makes it faster.

Disadvantages of Monolithic structure:

- If any service in the monolithic kernel fails, the entire System fails because, in address space, the services are connected to each other and affect each other.
- It is not flexible, and to introduce a new service

Layered Approach

In this type of structure, OS is divided into layers or levels. The hardware is on the bottom layer (layer 0), while the user interface is on the top layer (layer N). These layers are arranged in a hierarchical way in which the top-level layers use the functionalities of their lower-level levels.

In this approach, functionalities of each layer are isolated, and abstraction is also available. In layered structure, debugging is easier as it is a hierarchical model, so all lower-level layered is debugged, and then the upper layer is checked. So all the lower layers are already checked, and the current layer is to be checked only. Example: The Windows XP, Linux operating system uses this layered approach. Layered structure in OS is shown in figure 1.18.

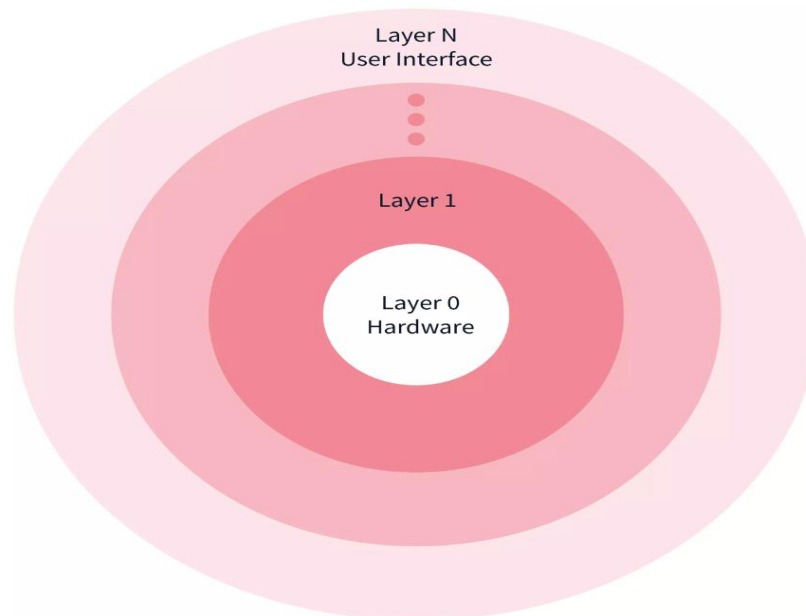


Figure 18. Layered structure

Advantages of Layered Structure

- Each layer has its functionalities, so work tasks are isolated, and abstraction is present up to some level.
- Debugging is easier as lower layers are debugged, and then upper layers are checked.

Disadvantages of Layered Structure

- In Layered Structure, layering causes degradation in performance.
- It takes careful planning to construct the layers since higher layers only utilize the functions of lower layers.

Micro-kernel

Micro-Kernel structure designs the Operating System by removing all non-essential components of the kernel. These non-essential components of kernels are implemented as systems and user programs. The result is a smaller kernel. The main function of the microkernel is to provide communication between the client program and the various services that are also running in user space. Communication is provided through message passing.

One benefit of the microkernel approach is that it makes extending the operating system easier. All new services are added to user space and consequently do not require modification of the kernel. Micro-Kernel structure is shown in Figure 1.19. Examples of microkernel OS are Mac OS 8, Minix, L4Linux.

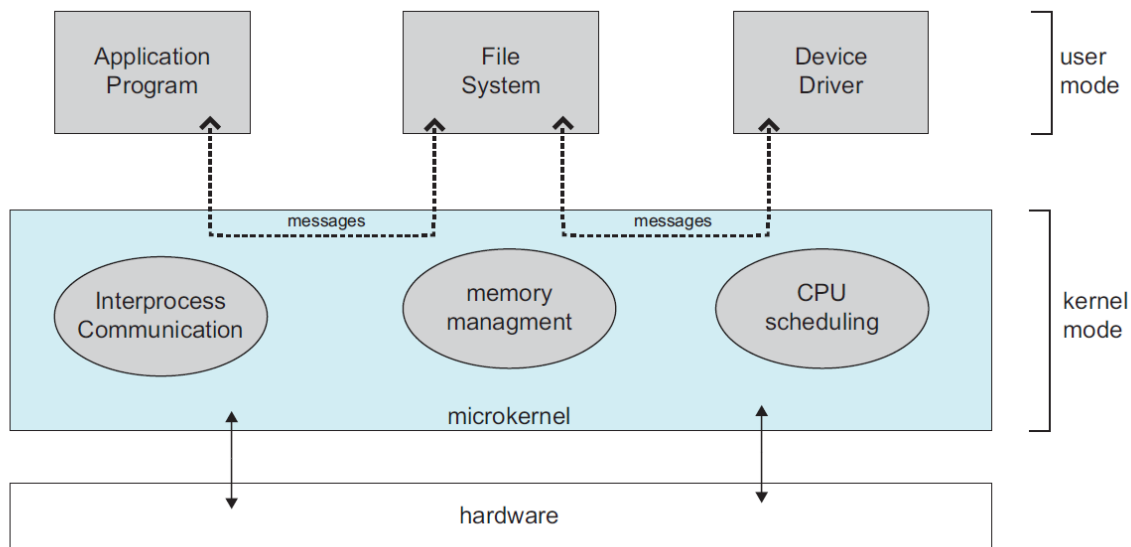


Figure 1.19. Micro-Kernel structure

Advantages of Micro-kernel structure:

- It functions better in spite of the fact that the architecture of the microkernel is small and isolated.
- The system can undergo easy expansion as additions can be done to the system application without interrupting the kernel.

- As it is modular in nature, modules can be modified, reloaded or replaced without any modification of the kernel.
- The Microsystem kernel is said to be a versatile technique as the APIs implemented by several servers can coexist.
- New features can be added to the Microkernel without recompiling.

Disadvantages of Micro-kernel structure:

- The performance of the microkernel system may not be consistent and can cause issues.
- These services are expensive as compared to the traditional monolithic system.
- When drivers are run as processes a context switch is always required.

Loadable kernel modules (or) Modules

current methodology for operating-system design involves using loadable kernel modules. Here, the kernel has a set of core components and links in additional services via modules, either at boot time or during run time. This type of design is common in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X, as well as Windows. The idea of the design is for the kernel to provide core services while other services are implemented dynamically, as the kernel is running. Solaris loadable module is shown in Figure 1.20.

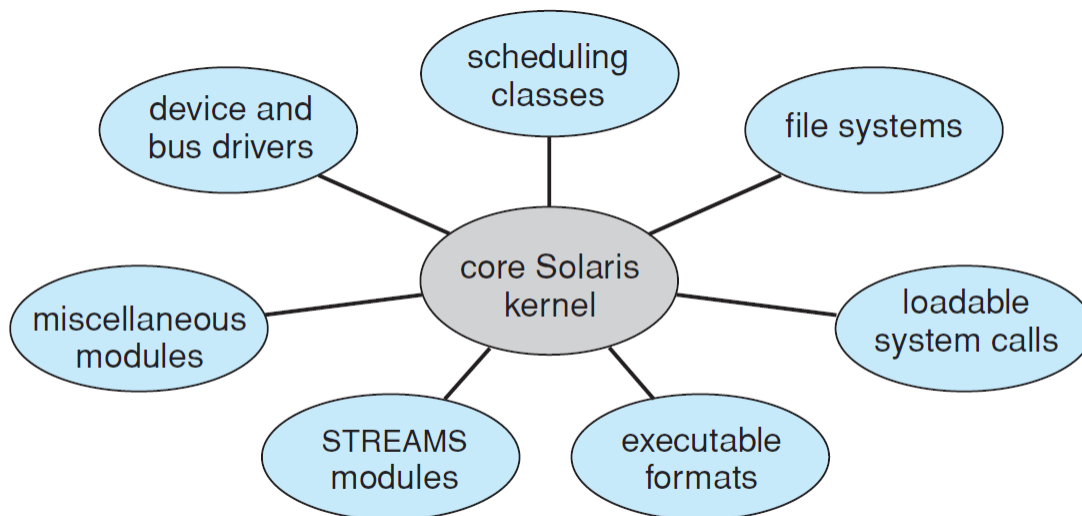


Figure 1.20. Solaris loadable modules.

Hybrid Structure

Hybrid architecture as the name suggests consists of a hybrid of all the architectures explained so far and hence it has properties of all of those architectures which makes it highly useful in present-day operating systems. Example of hybrid systems are the Apple Mac OS X operating system, iOS and Android.

Operating-System Debugging

Debugging

Debugging is finding and fixing errors, or bugs. OS generate log files containing error information. Failure of an application can generate core dump file capturing memory of the process which allows a programmer to explore the code and memory of a process. A failure in the kernel is called a crash. Operating system failure can generate crash dump file containing kernel memory.

Kernighan's Law: "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

Performance Tuning

Beyond crashes, performance tuning can optimize system performance

- Sometimes using trace listings of activities, recorded for analysis.
- Profiling is periodic sampling of instruction pointer to look for statistical trends.
- Improve performance by removing bottlenecks
- OS must provide means of computing and displaying measures of system behavior
- For example, "top" program or Windows Task Manager

DTrace

DTrace is a facility that dynamically adds probes to a running system, both in user processes and in the kernel. These probes can be queried via the D programming language to determine about kernel, system state, and process activities.

- DTrace tool in Solaris, FreeBSD, Mac OS X allows live instrumentation on production systems
- Probes fire when code is executed within a provider, capturing state data and sending it to consumers of those probes.

Operating-System Generation

Operating systems are designed to run on any of a class of machines. The system must then be configured or generated for each specific computer site, a process sometimes known as system generation SYSGEN. The SYSGEN program reads from a given file, or asks the operator of the system for information concerning the specific configuration of the hardware system, or probes the hardware directly to determine what components are there. The following kinds of information must be determined.

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

How much memory is available? Some systems will determine this value themselves by referencing memory location after memory location until an "illegal address" fault is generated. This procedure defines the final legal address and hence the amount of available memory.

What devices are available? The system will need to know how to address each device (the device number), the device interrupt number, the device's type and model, and any special device characteristics.

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

Once this information is determined, a system administrator can use it to modify a copy of the source code of the operating system. The operating system then is completely compiled. Data declarations, initializations, and constants, along with conditional compilation, produce an output object version of the operating system that is tailored to the system described. At a slightly less tailored level, the system description can cause the creation of tables and the selection of modules from a precompiled library.

System Boot

After an operating system is generated, it must be made available for use by the hardware. But how does the hardware know where the kernel is or how to load that kernel? The procedure of starting a computer by loading the kernel is known as booting the system. On most computer systems, a small piece of code known as the **bootstrap program or bootstrap loader** locates the kernel, loads it into main memory, and starts its execution.

When a CPU receives a reset event—for instance, when it is powered up or rebooted—the instruction register is loaded with a predefined memory location, and execution starts there. At that location is the initial bootstrap program. This program is in the form of read-only memory (ROM) / Erasable programmable read-only memory EPROM, because the RAM is in an unknown state at system startup. ROM is convenient because it needs no initialization and cannot be infected by a computer virus. All forms of ROM are also known as firmware.

For large operating systems (including most general-purpose operating systems like Windows, Mac OS X, and UNIX) or for systems that change frequently, the bootstrap loader is stored in firmware, and the operating system is on disk. In this case, the bootstrap runs diagnostics and has a bit of code that can read a single block at a fixed location (say block zero) from disk into memory and execute the code from that boot block. The program stored in the boot block may be sophisticated enough to load the operating system from disk into memory and begin its execution.

GRand Unified Bootloader (GRUB) is an example of an open-source bootstrap program for Linux systems. All of the disk-bound bootstrap, and the operating system itself, can be easily changed by writing new versions to disk. A disk that has a boot partition is called a **boot disk or system disk**.

Now that the full bootstrap program has been loaded, it can traverse the file system to find the operating system kernel, load it into memory, and start its execution. It is only at this point that the system is said to be running. System Boot Process is shown in figure 1.21.

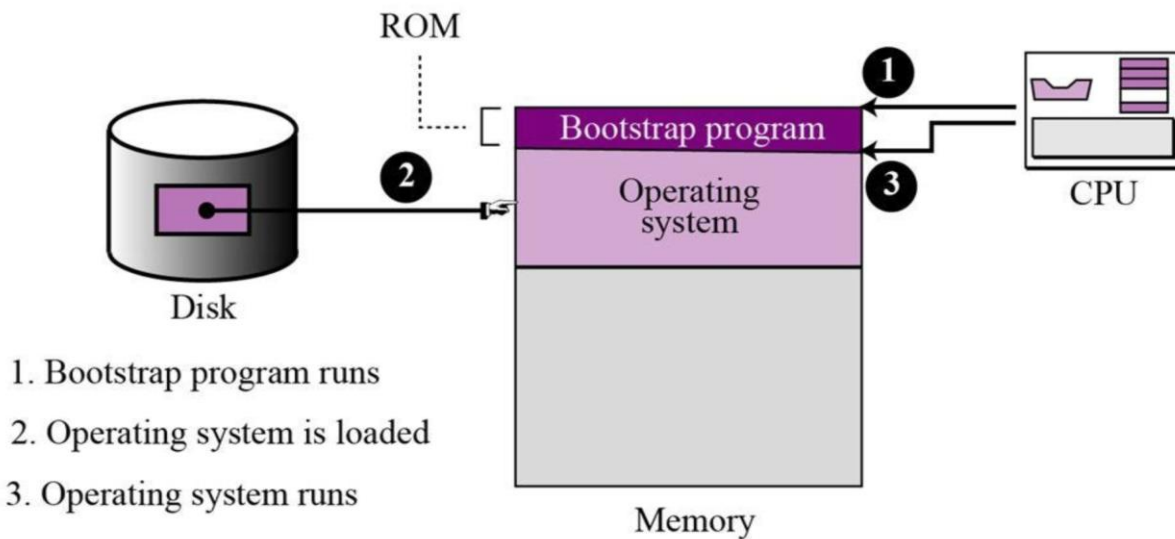


Figure 1.21. System Boot Process

Type of Booting:

There are two types of Booting available:

Cold Booting/ Hard Booting: Cold booting is the process when our computer system moves from shut down state to the start by pressing the power button. The system reads the BIOS from ROM and will eventually load the Operating System.

Warm Booting/ Soft Booting: Warm booting is the process in which the computer gets restart due to reasons like setting the configuration for newly installed software or hardware. Warm booting is called as rebooting.