Operating System Structure

Introduction

- **Multiprogramming**
- Operating System has an important feature i.e. multiprogramming. A single user cannot keep CPU and I/O devices busy which reduces utilization of CPU and I/O devices
- In multiprogramming CPU & I/O devices are kept busy by assigning different jobs to them. The jobs are kept in memory, which are executed one by one.
- If one job is busy in I/O operations, instead of being waiting or idle CPU switches to next job. So CPU is kept busy. This way OS manages resources and jobs by switching technique.
- Multiprogramming ensures efficiency; it organizes jobs in such a way that CPU and I/O devices are kept busy at all times.

- **Time Sharing**
- Time sharing is another feature which is logical extension of multiprogramming & multitasking. In time sharing system CPU executes multiple jobs.
- The switching time among different jobs is too less that user does not understand that CPU executes only one job at a time. So switching speed is very fast.
- Time sharing system ensures reasonable response time which is achieved using the technique of Swapping. In swapping processes are swapped in and out of memory to the disk.
- Time sharing system provides File System. A file system is collection of file related utilities present on the disk. File system is present on disks and needs disk management.

- It provides mechanism for protecting resources from inappropriate use, mechanism for job synchronization and communication and checks that deadlock condition should not occur. It creates interactive computing environment.
- OS provides quick access to users by providing very short response time. This increases CPU efficiency. It also does job scheduling and CPU scheduling.
- For multiprogramming & time sharing systems job pools are created on disk. Job pool consists of all processes residing on disk and awaiting allocation of main memory.

- **Memory Management**
- OS does memory management by providing Virtual memory which allows execution of process that is not completely in memory.
- The benefit is that this technique allows user to run program whose size is more than main memory.

2.1. Operating System Services

- A set of operating-system services provides functions that are helpful to the user
- Followings are the major services provided by operating system
  1. User Interface
  2. Program Execution
  3. Input Output operations
  4. File System manipulation
  5. Communications
  6. Error Detection
  7. Resource allocation
  8. Accounting
  9. Protection and Security

A) User Interface

- Almost all operating systems have a user interface (UI)
- Varies between Command-Line (CLI), Graphics User Interface (GUI), Voice UI, Batch
- **Command-Line Interface (CLI)**
    - CLI allows direct command entry
    - Sometimes implemented in kernel, sometimes by systems program
    - Sometimes multiple flavors implemented – shells
    - Primarily fetches a command from user and executes it
    - Sometimes commands built-in, sometimes just names of programs
    - If the latter, adding new features don't require shell modification
- **Graphics User Interface (GUI)**
    - User-friendly desktop metaphor interface. Usually uses mouse, keyboard, and monitor.
    - Icons represent files, programs, actions, etc
    - Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder)
    - Invented at Xerox PARC. Many systems now include both CLI and GUI interfaces. Microsoft Windows is GUI with CLI "command" shell.
    - Apple Mac OS X as "Aqua" GUI interface with UNIX kernel underneath and shells available. Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)
- **Batch Interface**
    - Batch allows command environment which executes commands one by one
    - A batch of commands is created by user contains multiple executable commands in a sequence
    - This batch is executed to execute the set of commands included in it.

B) Program Execution

- OS provides an environment where user can conveniently run programs.
- User does not worry about memory allocation or multitasking or anything
- The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)

C) Input Output Operations

- A running program may require I/O, which may involve a file or an I/O device.
- OS hides details of underlying hardware of an I/O device from user
- For better efficiency and protection, the users are not allowed to directly interact with I/O devices
- OS becomes bridge between user and I/O devices

D) File System Manipulation

- The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
- OS handles issues related to permissions like read, write, execute, access deny depending on ownership of file
- It involves creating, deleting and searching of file by its name also

E) Communications

- Processes may exchange information, on the same computer or between computers over a network
- Communications done may be via shared memory or through message passing (packets moved by the OS)
- User does not have to worry about passing messages to each other this work is done by OS

F) Error Detection

- OS needs to be constantly aware of possible errors
- May occur in the CPU and memory hardware, in I/O devices, in user program
- For each type of error, OS should take the appropriate action to ensure correct and consistent computing
- User program can't control this service because it involves monitoring and in case altering of memory or de allocation of memory for faulty programs
- Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

G) Resource allocation

- When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
- Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code

H) Accounting

- To keep track of which users use how much and what kinds of computer resources
- It helps in finding out usage statistics
- It is useful for researchers to find out usage requirements and requirements to reconfigure the system

I) Protection and Security

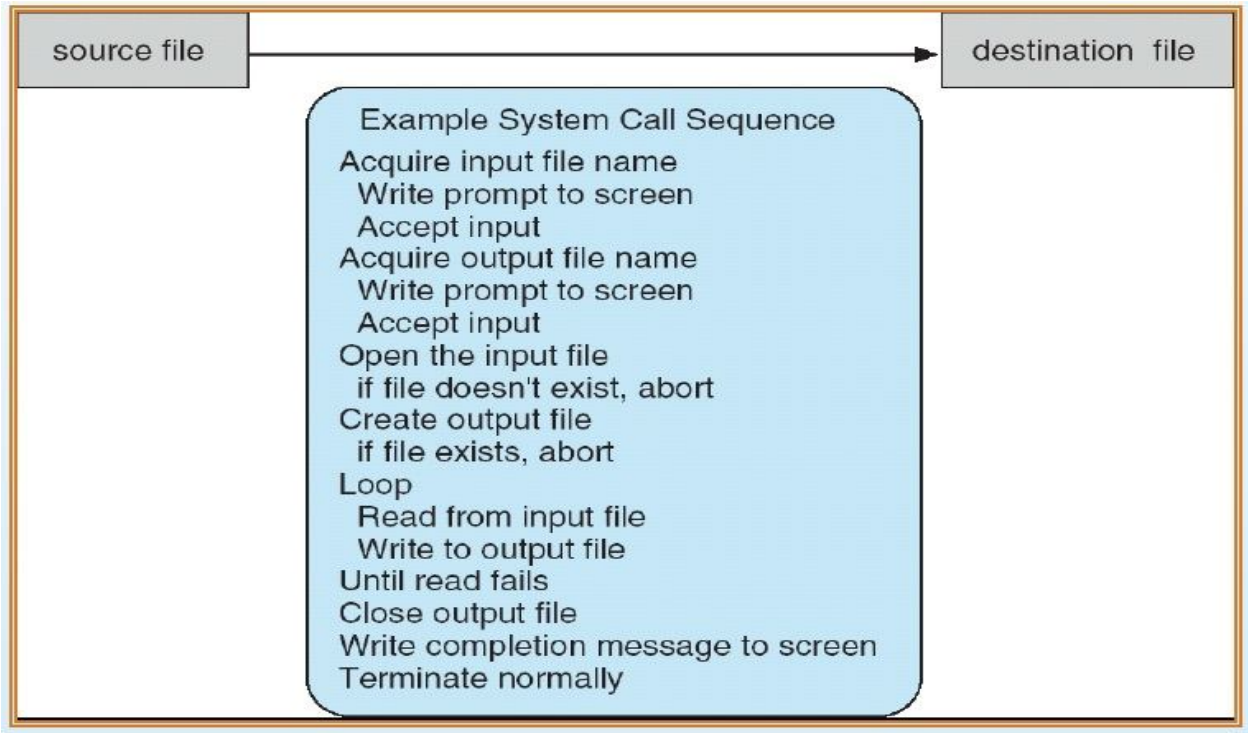
- The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other

- 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
- If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link

2.2. System Calls

- System Call is a communication made by an operating system with processes.
- System call provides an interface between process and operating system
- Operating system provides services to user level processes because of system calls. To implement system calls operating system enters kernel mode to provide service to user level process
- It is a programming interface to the services provided by the OS
- It is typically written and used in a high-level language (C or C++)
- It is mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- System programs provide basic functioning to users so that they do not need to write their own environment for program development e.g. editors, compilers and for execution e.g. shell. System programs are the bundles of system calls
- When any process request for a particular service to the operating system, respective system calls are made.
- Ex. if one file is copied into another. For this copy operation following operations are required-
- names of source and destination files should be known

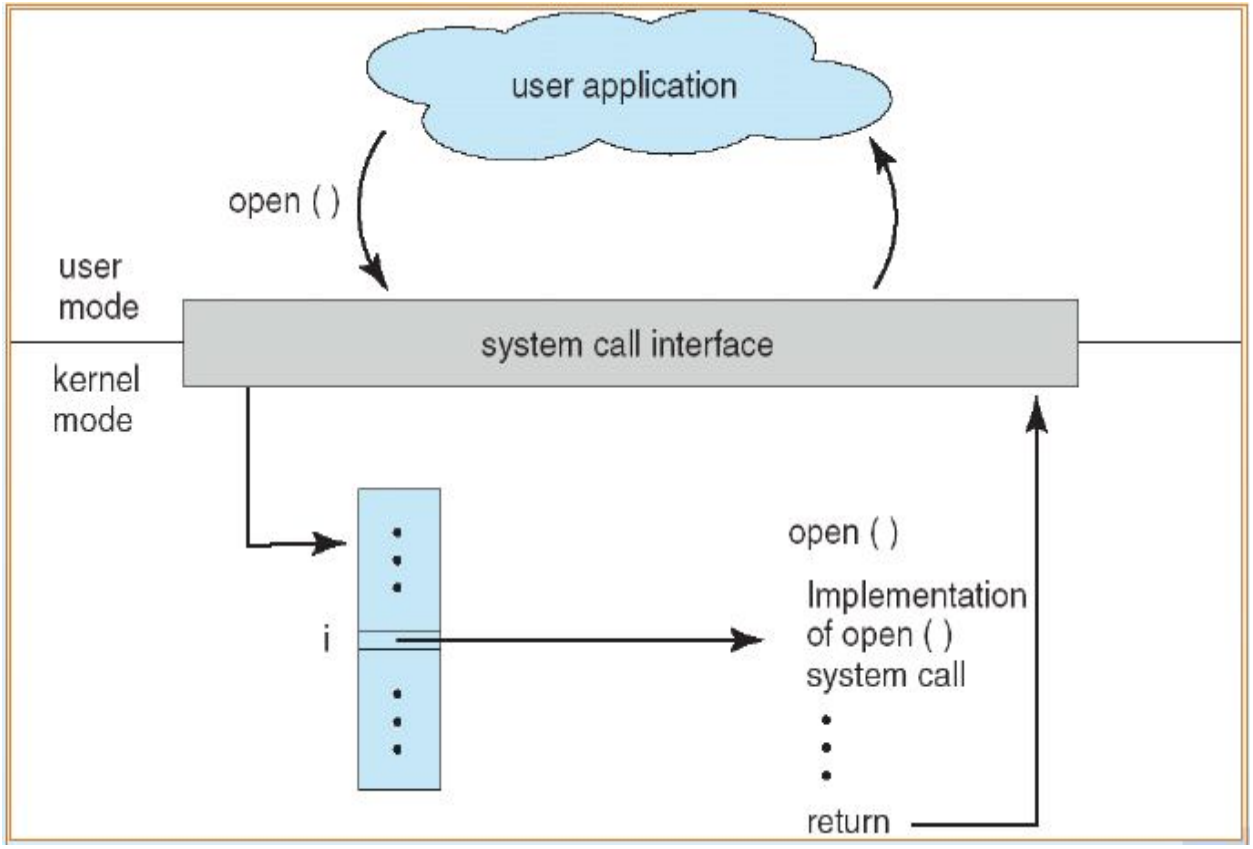
- checking that whether given files exists or do it requires access permission; if yes read the contents from source file
- To perform these operations, system calls are made by operating system to provide services to user program of copy operation



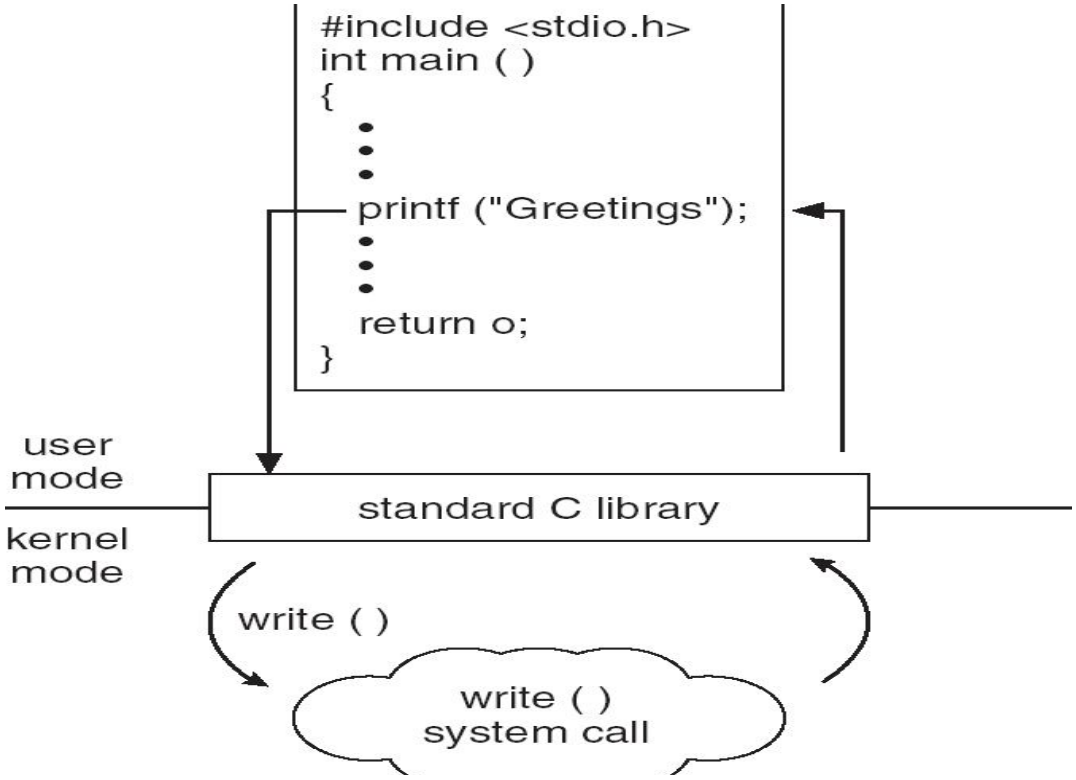
- Linux/UNIX has about 60 system calls
- The most calls are written in C.
- These calls can be accessed from C programs.
- There are similar system programs that provide similar system call features/services
- Ex, Basic I/O, Process control (creation, termination, execution), File operations and permission, System status.

System Call Implementation

- During implementation a number is assigned with each system call. It is used to number the system calls
- System call interface maintains a table indexed according to these numbers
- System call interface invokes system call in OS kernel and returns status of system call and any return values
- Caller does not know how system call is implemented. Caller uses API(Application Programming Interface) and understands what OS will do as a result
- Operating system details are hidden form programmer by API which is managed by runtime library. Compiler includes set of functions built in to libraries.



- **Standard C library example-**



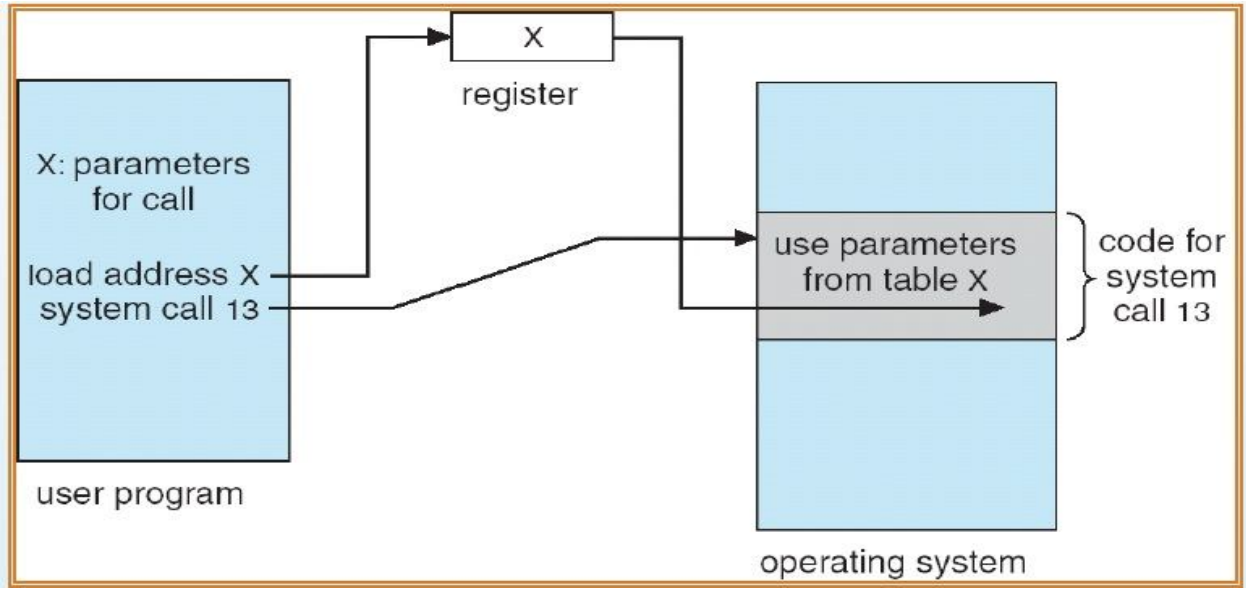
System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
- Exact type and amount of information vary according to OS and call
- Two general methods used to pass parameters to the OS"

  1. Simplest: pass the parameters in *registers,* in some cases, may be more parameters than registers. Parameters stored in a *block,* or table, in memory, and address of block passed as a parameter in a register. This approach taken by Linux and Solaris

  2. Parameters placed, or *pushed,* onto the *stack* by the program and *popped* off the stack by the operating system. Block and stack methods do not limit the number or length of parameters being passed

- Ex, Parameter Passing via Table-



Types of System Calls

Followings are the types of system calls

1. Process control
2. File management
3. Device management
4. Information maintenance
5. Communications

A) Process Control

- If a program is running the OS may either halt program execution normally or abnormally i.e. abort operation has to be performed
- If currently running process is terminated or if a problem arises, then error message is generated
- This memory is taken up by debugger to check the errors
- If a running program discovers and error, it has to define the error level

- If errors are more high level error can be define and if no errors are there 0 level error is defined
- For executing multiple commands, command executer executes number of programs given by processor
- To understand that when process execution is completed, where the control should be returned, it is necessary to find out status of program whether it is lost, loaded, saved, completely executed etc.
- For controlling processes in multitasking system it is necessary to determine and reset process attributes of a job or process
- The process is terminated when it is no longer needed. Sometimes processes get the signals to stop at particular event. To wait for particular event process has to wait. When program gets execute at each and every occurrence of the timer interrupt, value of program counter is recorded for further use.
- If there is requirement to start a new process, in UNIX the shell executes fork() system call. Exec() system call is used to load a process from memory and executes it
- Process control system calls include end, abort, load, execute, create process, terminate process, get/set process attributes, wait for time or event, signal event, and allocate and free 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. When processes stop abnormally it may be necessary to provide core dumps and/or other diagnostic or recovery tools.
- Various system calls related to process control are as follows:-

1. end, abort
2. load, execute
3. create process, terminate process
4. get process attributes, set process attributes

5. wait for time
6. wait event, signal event
7. allocate and free memory

B) File management

- File management system calls include create file, delete file, open, close, read, write, reposition, get file attributes, and set file attributes.
- These operations may also be supported for directories as well as ordinary files.
- Various system calls related to file management are as follows:-

1. create file, delete file
2. open, close
3. read, write, reposition
4. get file attributes, set file attributes

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.
- Devices may be physical (e.g. disk drives), or virtual / abstract (e.g. files, partitions, and RAM disks).
- Some systems represent devices as special files in the file system, so that accessing the "file" calls upon the appropriate device drivers in the OS.
- Various system calls related to Device management are as follows:-

1. request device, release device
2. read, write, reposition
3. get file attributes, set file attributes
4. Logically attach or detach devices

D) Information maintenance

- Information maintenance system calls include calls to get/set the time, date, system data, and process, file, or device attributes.
- Systems may also provide the ability to dump memory at any time, single step programs pausing execution after each instruction, and tracing the operation of programs, all of which can help to debug programs.
- Various system calls related to Information maintenance are as follows:-

1. get time or date, set time or date
2. get system data, set system data
3. get process, file or device attributes
4. set process, file or device attributes

E) Communications

- Communication system calls are create/delete communication connection, send/receive messages, transfer status information, and attach/detach remote devices.
- Two models-
- **Message passing:-**
- 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.
- Delete the connection when no longer needed.
- **Shared Memory:-**
- The shared memory model must support calls to:
- Create and access memory that is shared amongst processes (and threads.)

- Provide locking mechanisms restricting simultaneous access.
- 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.
- Shared memory is faster, and is generally the better approach where large amounts of data are to be shared
- Various system calls related to communication are as follows:-

1. Create, delete communication connection
2. Send or receive messages
3. Transfer status information
4. Attach or detach remote devices

Uses of System Calls

- System calls provide an interface between the running program i.e. process and operating system.
- System calls allow user-level processes to request some services from the operating system which the process can't do itself.
- System calls provide basic functioning to users so that they do not need to write their own environment for program development and program execution.
- System calls are used to perform input/ output operations which involve reading or writing a particular area and this request is satisfied by the operating system. It is because of the critical nature operations, the operating system does itself every time they are needed.

2.3. Operating System Structure

Followings are the types of operating system structure

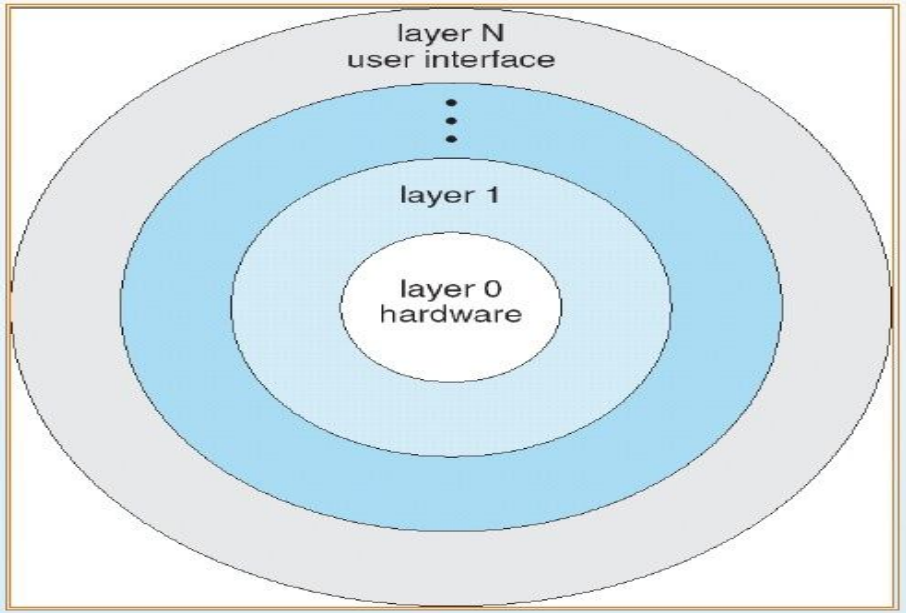
1. Simple Structure
2. Layered Structure
3. Micro kernel
4. Monolithic System

A) Simple Structure

- Earliest operating system MS DOS has the simple operating system structure
- This OS provides most functionality with less space hence this OS is not divided into different modules
- MS DOS has not created interfaces, functional or access levels, due to which the structure of MS DOS is not complicated. It uses basic I/O functions to execute its input and output operations.
- Due to this simple structure and use of basic functions MS DOS is vulnerable to malicious programs which causes whole system crash when user program fails.
- The reason behind this was Intel 8080 was used as hardware and it was not able to process dual mode and does not provide protection against access to hardware.

B) Layered Structure

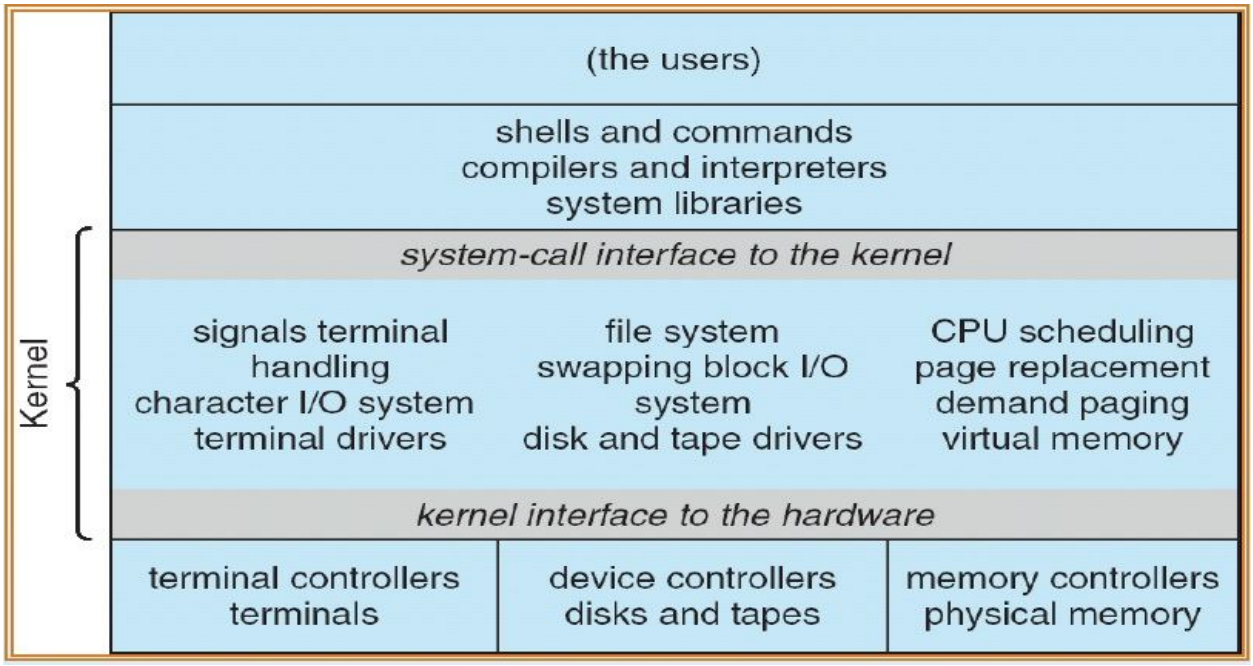
- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers



- There are mainly two operating systems which use layered structure, i.e. Unix and MS DOS

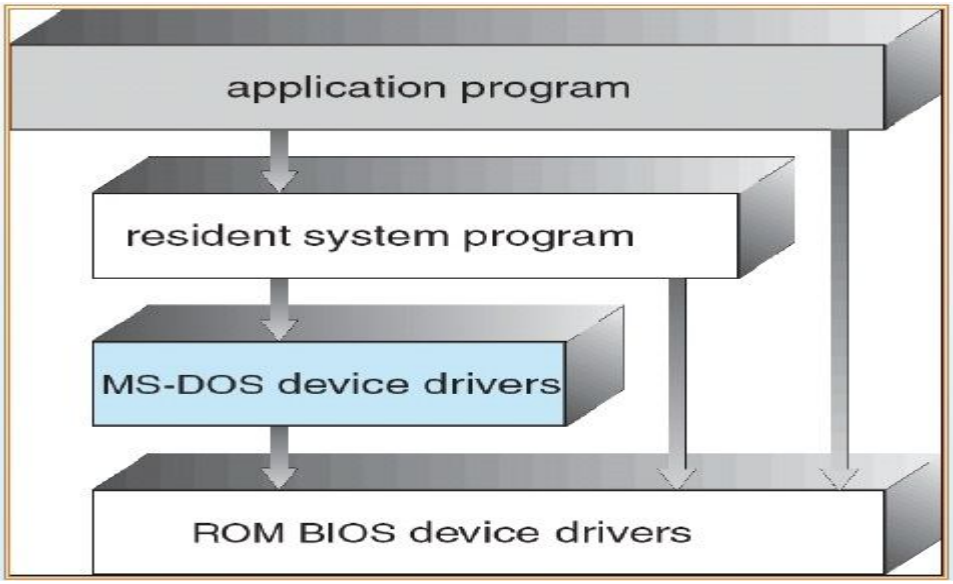
A. UNIX Layered Structure

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts viz. Systems programs and The kernel
- Consists of everything below the system-call interface and above the physical hardware
- Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level
- Kernel is the most important component of UNIX OS which is heart of UNIX.
- Kernel is further divided into series of interfaces and device drivers.
- These interfaces and device drivers are added on need basis.



- There is specific position of kernel. Everything which is below system call interface and above physical hardware is kernel
- Kernel provides main functions which are performed by OS itself
- Some of the functions are file system, CPU scheduling, memory management and other operating system functions through system calls

B. MS DOS Layered Structure/ Layered approach



C) Micro kernel

- The basic idea behind micro kernels is to remove all non-essential services from the kernel, and implement them as system applications instead, thereby making the kernel as small and efficient as possible.
- Most micro kernels provide basic process and memory management, and message passing between other services, and not much more.
- Security and protection can be enhanced, as most services are performed in user mode, not kernel mode.
- 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.
- Windows NT was originally microkernel, but suffered from performance problems relative to Windows 95. NT 4.0 improved performance by moving more services into the kernel, and now XP is back to being more monolithic.
- Another microkernel example is QNX, a real-time OS for embedded systems.

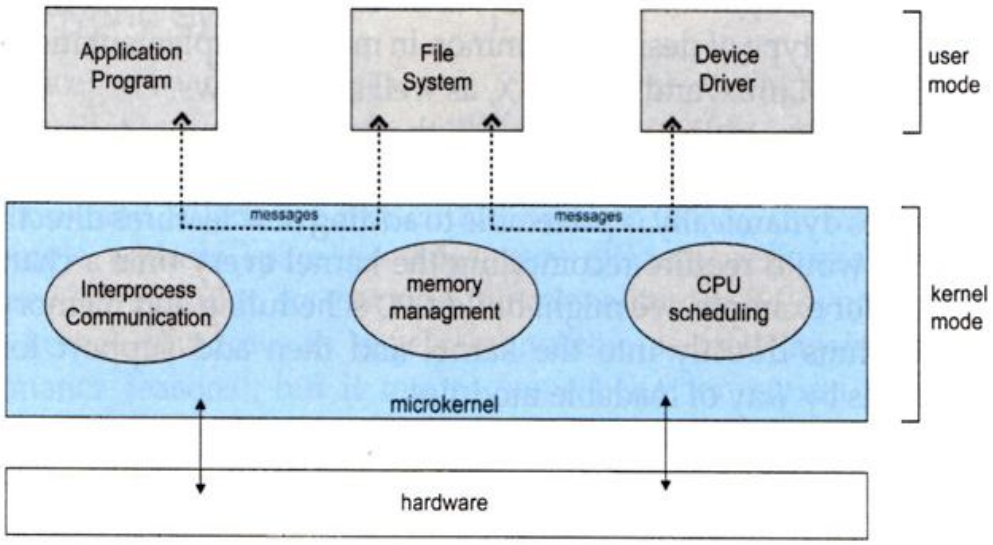


Figure 2.14 Architecture of a typical microkernel.

- Moves as much from the kernel into "*user*" space
- Communication takes place between user modules using message passing
- **Benefits:**

1. Easier to extend a microkernel
2. Easier to port the operating system to new architectures
3. More reliable (less code is running in kernel mode)
4. More secure
5. Performance overhead of user space to kernel space Communication

D) Monolithic System

- Modern OS development is object-oriented, with a relatively small core kernel and a set of procedures/modules which can be linked in dynamically.
- OS is collection of procedures each of which can call each other whenever it needs
- Each procedure in system has well defined interface in terms of parameters and results
- 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 going through multiple intermediary layers.
- The kernel is relatively small in this architecture, similar to micro kernels, but the kernel does not have to implement message passing since modules are free to contact each other directly.
- Details of procedures are hidden thus supporting information hiding
- Only official entry points are can be called from outside modules
- System calls provided by OS are requested by user programs by arranging parameters in registers or stacks.
- Then a special trap instruction is executed known as kernel call or supervisor call

- Monolithic organization propose following basic structure
  1. Main procedure invokes requested system call
  2. Set of service procedures carry out system calls. For each system call separate service procedure is assigned
  3. Set of utility procedures help service procedures ex. Fetching data from user program

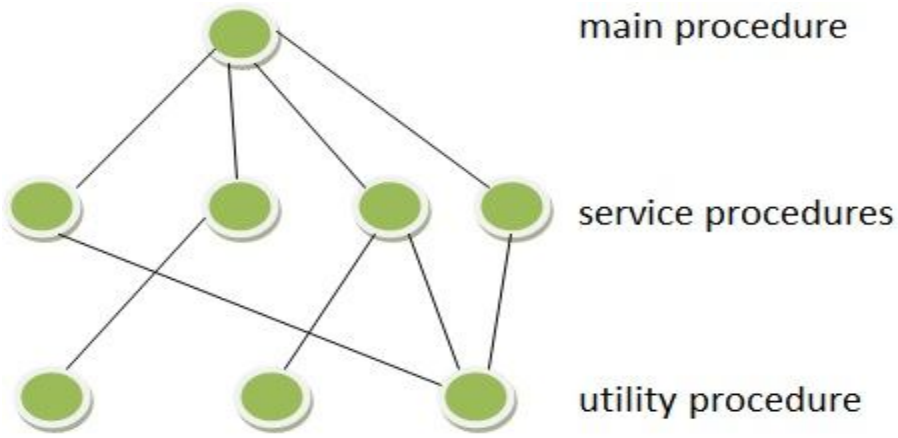


Fig. Structure of Monolithic System

2.4. Components of Operating System

Followings are the operating system components-

1. Process management
2. Main memory management
3. File management
4. I/O management
5. Secondary storage management

A) Process Management

- Process is any task executed by CPU. It could be a program in execution (user process) or it could be internal systems task executed by CPU (system process).
- Process may be one instance or moment of a program in execution. It is a unit to measure work done by CPU.

- OS manages many kinds of activities of user programs to system programs; each of these is called as process.
- During program execution different resources are required ex. I/O devices, data, program memory etc. which are part of some process. A process requires input that includes complete execution components like code, data, Program Counter (PC), registers, OS resources in use etc.
- As CPU executes tasks and for every task number of process are working so process management is required throughout execution.
- In process management, CPU does following activities-
- **Creation and deleting of System & User processes**

  This activity creates as well as deletes system & user processes
- **Suspension & resumption of processes**

  This activity suspends i.e. temporary stops and continues process as per situation occurs in the system.
- **Mechanism for process synchronization**

  Synchronization of process is required when one process may give some input to other process and execution of second process proceeds.

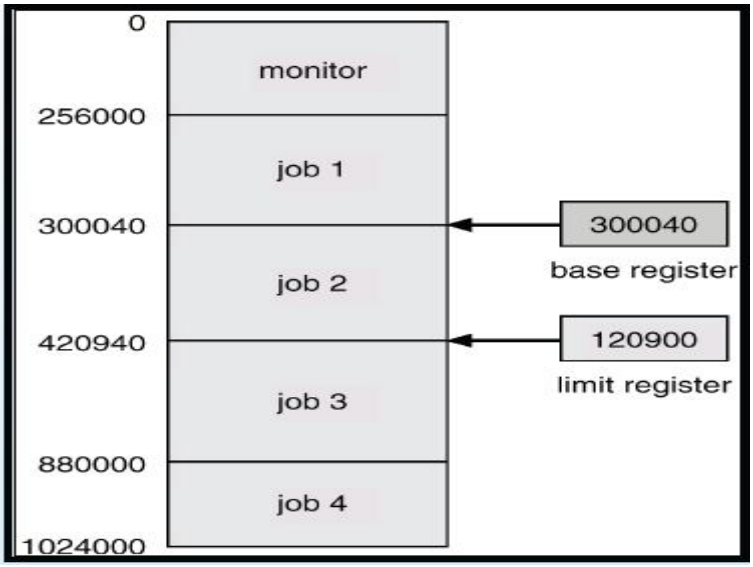
  Process synchronization ensures coordination of different processes which runs at a time.
- **Mechanism for process communication**

  Processes need communication with each other to ensure its full execution and smooth running of multiple processes by using different resources at a time.
- **Mechanism for deadlock handling**

  When multiple processes run at a time wait for each other, a deadlock situation occurs. OS handles this deadlock situation.

B) Main Memory Management

- In computer system, there are two types of memories available. Primary memory includes RAM, ROM, PROM, EPROM etc and Secondary memory includes mass storage like hard disk, floppy drives, CD, DVD etc.
- RAM is major part of primary i.e. main memory which is large array of words or bytes. Each word or byte has its own memory address by which it is referred.
- For execution of any task or instructions and calls to I/O devices takes memory from main memory. Means main memory provides storage that can be accessed directly by CPU.
- **Memory Protection**
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
- Base register – holds the smallest legal physical memory address.
- Limit register – contains the size of the range



- CPU does following activities for main memory management -
- To keep track of each and every memory location
- To decide which process to be loaded in main memory
- Allocation and de allocation of memory space

C) File Management

- A file is defined as collection of related data that could be a collection of executable statements or simply a data file.
- These files are stored on secondary storages such as disks which provides long term storage
- The storage devices are categorized on the basis of their characteristics like speed, capacity data transfer rate and access methods
- Operating system decides where and how to store data on disks. Data is stored on storage devices in the locations such as tracks, sector and cluster.
- Directories are memory locations which store or organize files in it. Directories provide ease to use files. These directories may contain files or other directories in it.
- Operating System does following activities for file management -

1. Creation and deletion of files
2. Creation and deletion of directories
3. Support for primitives for manipulating files and directories
4. Mapping of files on to secondary storage
5. Backup of files on secondary storage

D) Input/ Output System Management

- Input output system consists of buffer caching system, hardware, I/O devices with device drivers, a general device-driver interface, drivers for specific hardware devices and supporting systems. Operating system manages this input output system.
- I/O subsystem hides complexity of hardware devices from the user. Only device drivers directly interact with the I/O hardware.
- Operating system tries to utilize I/O devices maximum for maximum efficiency.
- Operating system does following activities for I/O management

1. To keep track of all I/O devices to assign jobs to them.
2. To keep I/O devices continuously busy
3. To manage various I/O operations
4. To provide buffers to store those processes which are waiting due to busy I/O device.
5. To use techniques like spooling for waiting processes waiting for I/O devices.

E) Secondary Storage Management

- Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data
- Secondary storage consists of tapes, disks and other media that can hold information. This information is divided into bytes or words. Each location of in storage has its own address. The set of all addresses is called as address space
- Operating system does following activities for secondary storage management
- Free space management- management of free space available on secondary storage
- Storage allocation- allocate storage to new files for their creation or written
- Disk scheduling- scheduling requests for memory access

Booting

- The procedure of starting a computer by loading kernel is known as booting the system. Bootstrap is a smaller piece of code which finds out where kernel is and locates it in main memory and starts is execution.
- The general approach when most computers boot up goes something like this:
- When the system powers up, instruction registers are loaded in memory locations an execution starts.
- Bootstrap loader program exists there which is stored in ROM not in RAM as RAM is unknown in initial stage when system starts and ROM does not need initialization and not affected by viruses.
- Bootstrap program does diagnostic test that checks whether all devices are in proper state to continue system program or not. It also initializes CPU registers, device controllers etc and starts the OS in main memory.
- The bootstrap program then looks for a non-volatile storage device containing an OS. Depending on configuration, it may look for a floppy drive, CD ROM drive, or primary or secondary hard drives.
- Generally OS is stored on disk and when system starts OS get loaded in RAM not in ROM as ROM memory is small size due to which OS can't be stored in it.
- In case of cellular phones, PDA's and game console OS is stored in ROM.
- Assuming that OS is stored in disk it goes to a hard drive, it will find the first sector on the hard drive and load up the fdisk table, which contains information about how the physical hard drive is divided up into logical partitions, where each partition starts and ends, and which partition is the "active" partition used for booting the system.
- For a single-boot system, the boot program loaded off of the hard disk will then proceed to locate the kernel on the hard drive, load the kernel into memory, and then transfer control over to the kernel.

- For dual-boot or multiple-boot systems, the boot program will give the user an opportunity to specify a particular OS to load, with a default choice if the user does not pick a particular OS within a given time frame. The boot program then finds the boot loader for the chosen single-boot OS, and runs that program
- Once the kernel is running, it may give the user the opportunity to enter into single-user mode, also known as maintenance mode. This mode launches very few if any system services, 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.
- When the system enters full multi-user multi-tasking mode, it examines configuration files to determine which system services are to be started, and launches each of them in turn. It then spawns login programs (gettys) on each of the login devices which have been configured to enable user logins.

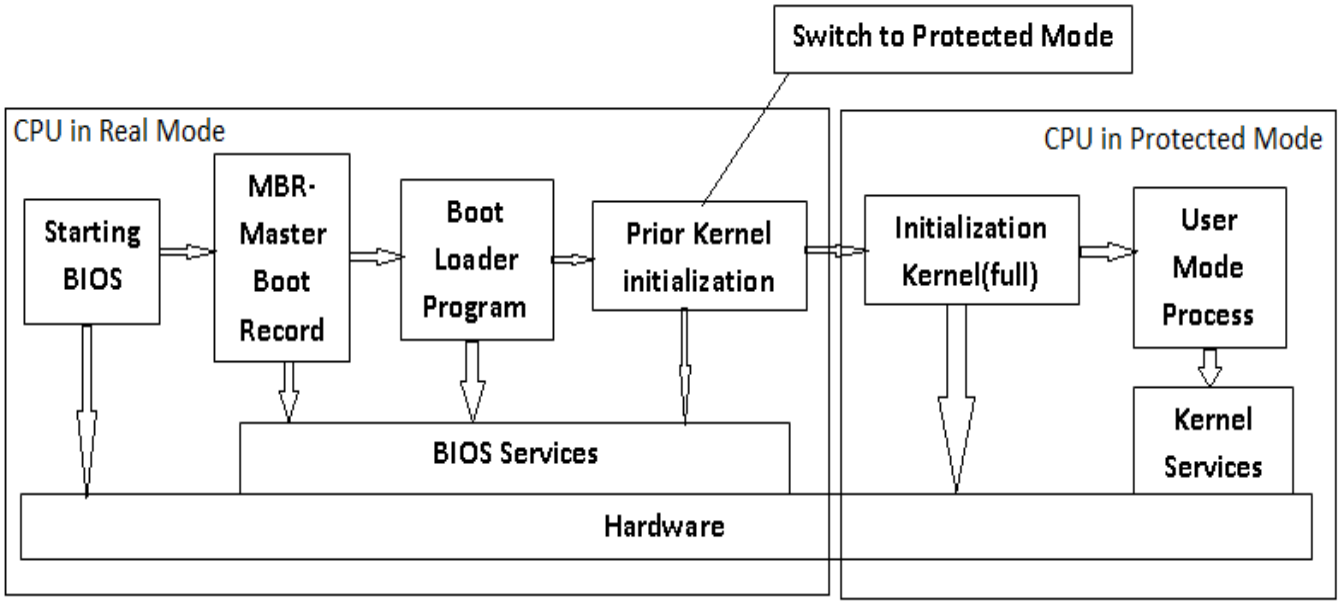


Fig. Booting Process