Basic Concepts of Operating Systems

Introduction

- **Without software**, a computer is basically a **useless** equipment. With software, a computer can store, process, and retrieve information and engage in many other valuable activities.
  - Computer software can be divided roughly into two parts: **system programs, which manage the operation** of the computer itself,
  - and **application programs, which perform the actual work the user wants.**
  - The most important system program is the operating system (OS) that **controls all the computer resources and provides the base upon which the application program can be written.**

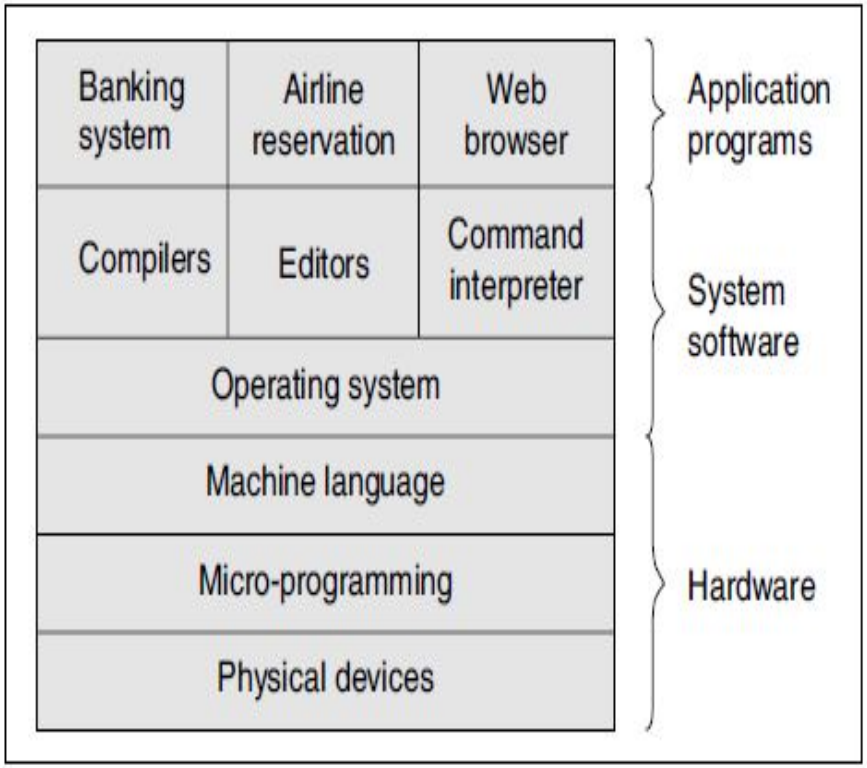


Fig. The software–hardware layers in a computer

Introduction : Operating System

- An operating system is a collection of programs that acts as an interface between the user of a computer and the computer hardware.
  - **Without software, a modern-day computer is unusable**.
  - Software comprising operating system, programming language compilers, etc. are essential to provide an 'user-friendly' interface to the user.
- A n operating system is an important part of almost every computer system that **comprises three main components**:
  - **The hardware** (memory, CPU, arithmetic-logic unit, various storage devices, I/O, peripheral devices, etc.)
  - **Systems programs** (operating system, compilers, editors, loaders, utilities, etc.)
  - **Application programs** (database systems, business programs, etc.)

- **A software that acts as an interface between the users and hardware of the computer system**
- A software that provides a working environment for the users' applications

- A software in which all common functions required to work on the computer system have been put together

- A resource manager that manages the resources needed for all the applications in the background

  - keep track of the resources
  - Enforce polices that determine who gets what, when and how much
  - allocate resources
  - Reclaim the resources

NEED OF OS
WHY WE NEED OS

- **How are the files as a logical concept mapped to the physical disk?**

- **Today, we are able to open many windows at a time; who is managing all these windows** ***despite a single processor?***

- **Who ensures that the CPU is not sitting idle or busy forever?**

- Sometimes we see some messages like memory error or power failure, connection failure in network, paper jam in printer, etc. Who is detecting these errors and displaying error messages?

- Who manages the limited memory despite the large size of user programs?
- Our processes can also communicate and cooperate via some synchronization mechanisms.
- Who provides the communication and synchronization mechanisms?
- Who schedules tasks to the CPU for execution?

- What happens to a task when the CPU is already busy in processing some other task?

- Despite a single processor, it seems that many jobs are being executed in parallel. How does this happen?

- Suppose, some users are working in a LAN with a single printer and more than one user gives a print command. How are the requests of multiple users on a single printer managed?

- Who protects one user's area from unauthorized access by another user's task?

- Why is it that sometimes our system hangs?

Functions of an Operating System

- **Manages the computer's resources**:
  The OS controls and efficiently utilizes hardware components such as CPU, memory and I/O devices.

- **Provides a user interface**:
  The OS enables users to easily interact with the computer hardware. For example the Windows operating systems displays icons, using which the users can interact with the system.

- **Process management**:
  the OS enables a user to execute more than one job at the same time to enhance productivity. Multiple processes being executed at the same time calls for efficient utilization of the system's resources by the operating system.

- **Memory management:**
  Finding vacant spaces in the primary memory, loading the appropriate data and programs in the located space, executing them and removing them from the memory is all done by the operating system.

- **File management**:
  The OS allows users to create copy, delete and rename files.

- **Security management**:
- The OS protects stored information from malicious users. It ensures that the data and files stored cannot be accessed by unauthorized users.

- **Device management**:
- The operating system manages and controls all I/O devices such as disks, tapes, terminal, printer and keyboard to ensure correct data transmission to and from devices. It also provides an intuitive interface so that the users can easily work with them.

- **Booting Services**:

Booting means loading an operating system into the computer's main memory. After the operating system is loaded, it becomes ready for users to run their applications. During the boot process, the computer performs a self-diagnostic test, also known as a POST (Power on Self Test) to ensure that all components are operational. It also loads necessary drivers and programs that help the computer and devices communicate with each other.

Common tasks performed by operating systems

Task	When performed
Maintain information for security	While registering new users
Construct a list of resources in the system	During booting
Verify identity of a user	At login time
Initiate execution of programs	At user commands
Maintain information for protection	When users specify or change protection information
Check protection information and perform resource allocation	At user/program request
Maintain current status of all resources	Continually during OS operation
Maintain current status of all programs and perform scheduling	Continually during OS operation

Computer System with Operating System

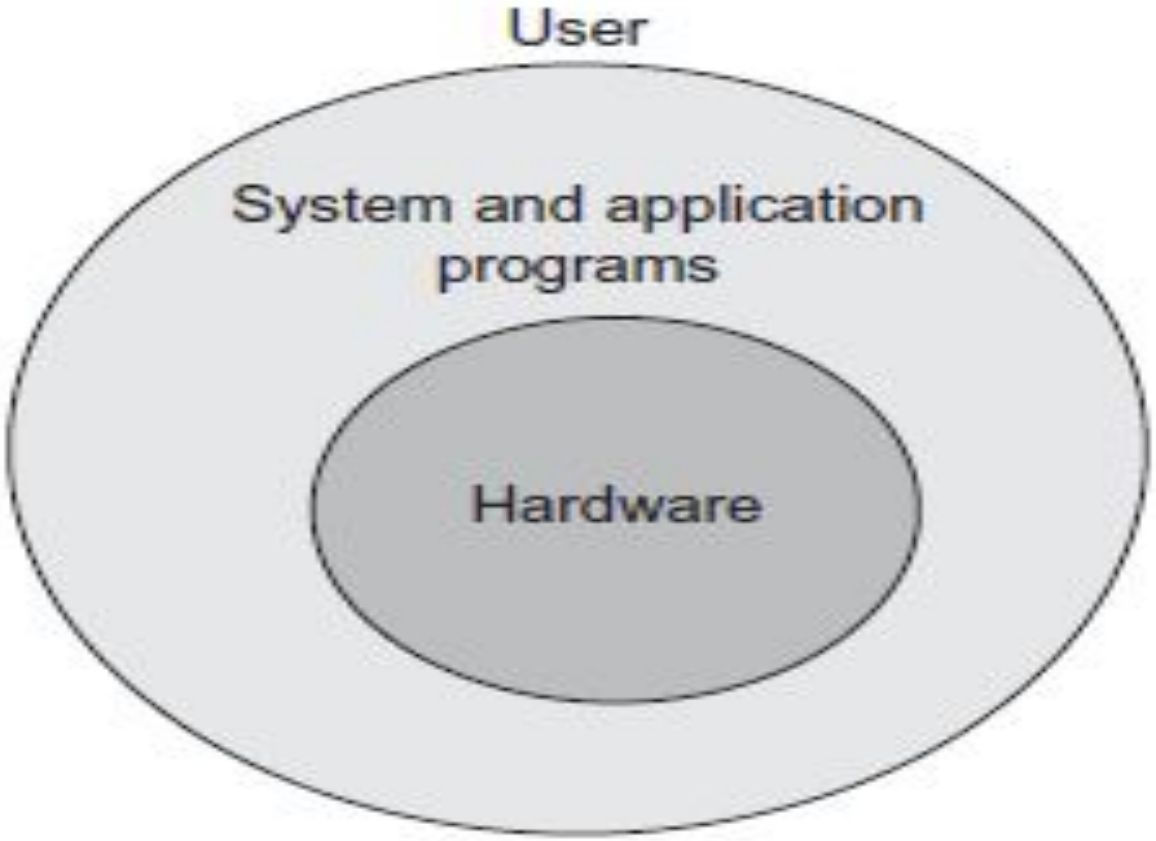
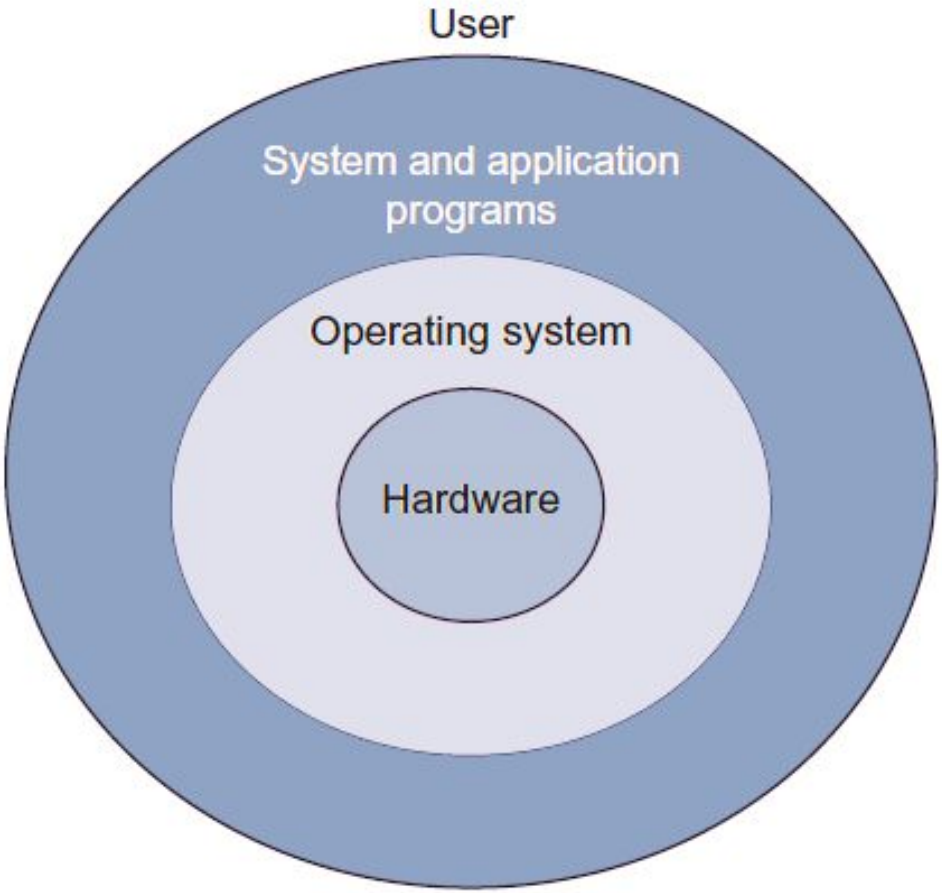


Fig. 1.1 Computer system

Computer System with Operating System



Goals of an Operating System

- Convenience

Resource utilization/management

Protection

Goals of an OS

- Two primary goals of an OS are
  - **Efficient use of the computer's resources**
    - To ensure cost-effectiveness of the computer
  - **User convenience**
    - A user should find it easy to use the computer
- These two goals sometimes conflict
  - Prompt service can be provided through exclusive use of a computer; however, efficient use requires sharing of a computer's resources among many users
  - An OS designer decides which of the two goals is more important under what conditions
    - That is why we have so many operating systems!

User convenience

- The *computing environment* influences the notion of user convenience
  - The computing environment is comprised of
    - The computer system
    - Its interfaces with other systems
    - Nature of computations performed by its users
  - Computing environments change with the architecture and use of a computer, e.g. personal computing, online applications, embedded applications
  - Hence the notion of user convenience has several facets that depend on the computing environment

User convenience

- User convenience has several facets
  - Fulfillment of a necessity
    - Use of programs and files
  - Good service
    - Speedy response
  - Ease of Use
    - User friendliness
  - New programming model
    - e.g., Concurrent programming
  - Web-oriented features
    - e.g., Web-enabled servers
  - Evolution
    - Addition of new features, use of new computers

Operation of an OS

- During operation, an OS performs two primary functions
  - **Program management**
    - Perform program initiation/termination
    - Provide means through which many programs can work toward a common goal
      - Helps in more convenient or faster fulfillment of user requirements
  - **Resource management**
    - Ensure efficient use of resources
      - Use of CPU, I/O devices

Operation of an OS

- Program management and resource management require three fundamental tasks
  - **Scheduling**
    - Deciding which program should be serviced by the CPU and for how long
  - **Resource allocation**
    - Sharing of resources among user programs
  - **Security and protection**
    - Ensuring non-interference between programs
      - *Security:* guarding against interference by external entities
      - *Protection:* guarding against interference by internal entities

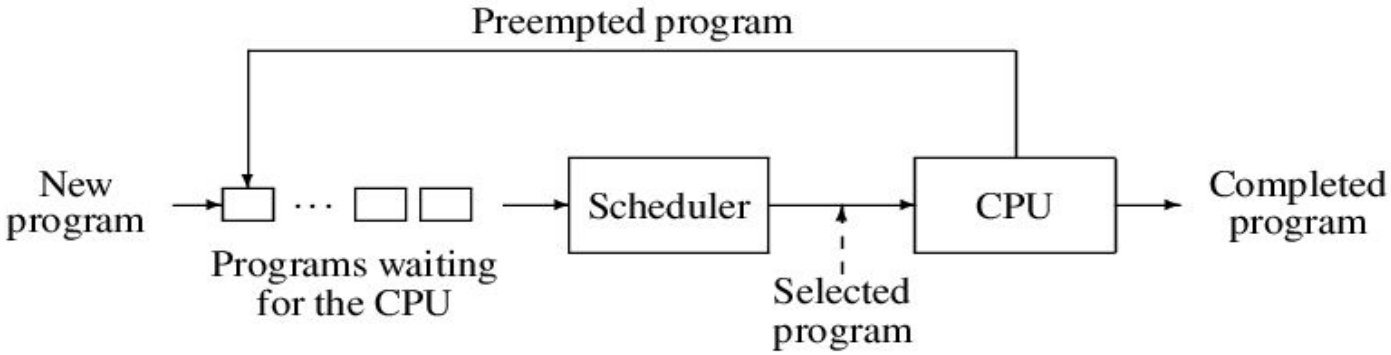
Operation of an OS

- To facilitate scheduling, resource management, and security and protection
  - OS maintains various kinds of information to facilitate operation
    - Names of registered users
    - identity of a user who is executing a specific program
    - status of a program
    - status of a resource
    - which users can access a resource
  - OS performs several tasks repeatedly to implement scheduling, resource management, and security and protection

Program management

- *A computational structure* is a configuration of several programs. Three typical computational structures are:
  - A single program
    - Execution of a program on a given set of data
  - A sequence of single programs
    - A program should be executed only if previous programs in the sequence executed successfully
    - Programs should pass their results to other programs through files
  - Co-executing programs
    - The OS must execute these programs at the same time
    - The OS must provide an interface between co-executing programs so that their results are available to one another

A schematic of scheduling



Programs waiting for CPU attention are entered in a pool

The scheduler picks one program from the pool for execution on the CPU

The CPU may be forcibly taken away from a program. This action is called *pre-emption*

A pre-empted program is again entered into the pool

Resource management

- Criteria and techniques of resource management
  - **Criterion**: Resource utilization efficiency
    - Share the resources wherever possible
    - Avoid idling of resources
    - Speedy resource allocation and de-allocation is desirable
  - **Technique 1**: Partitioning of resources
    - A *resource partition* is a collection of resources
      - A resource partition is allocated to a process
      - Simplifies resources allocation and also speeds it up
  - **Technique 2**: Allocation from a pool
    - Resources are allocated individually when needed

Virtual resources

- A *virtual resource* is an illusion presented to a program
  - A virtual resource is an abstraction; its implementation is hidden from a program
  - Use of virtual resources simplifies resource allocation
    - Virtual resources may be created if several programs need a resource. It permits more programs to execute at the same time
  - A virtual I/O device is a virtual resource
    - The OS implements a virtual device as follows:
      - When a program performs an operation on a virtual device, the operation may be actually performed on some real resource
    - Example: virtual printer
      - When a program 'prints' on a virtual printer, the data is actually stored in memory; printing takes place sometime later

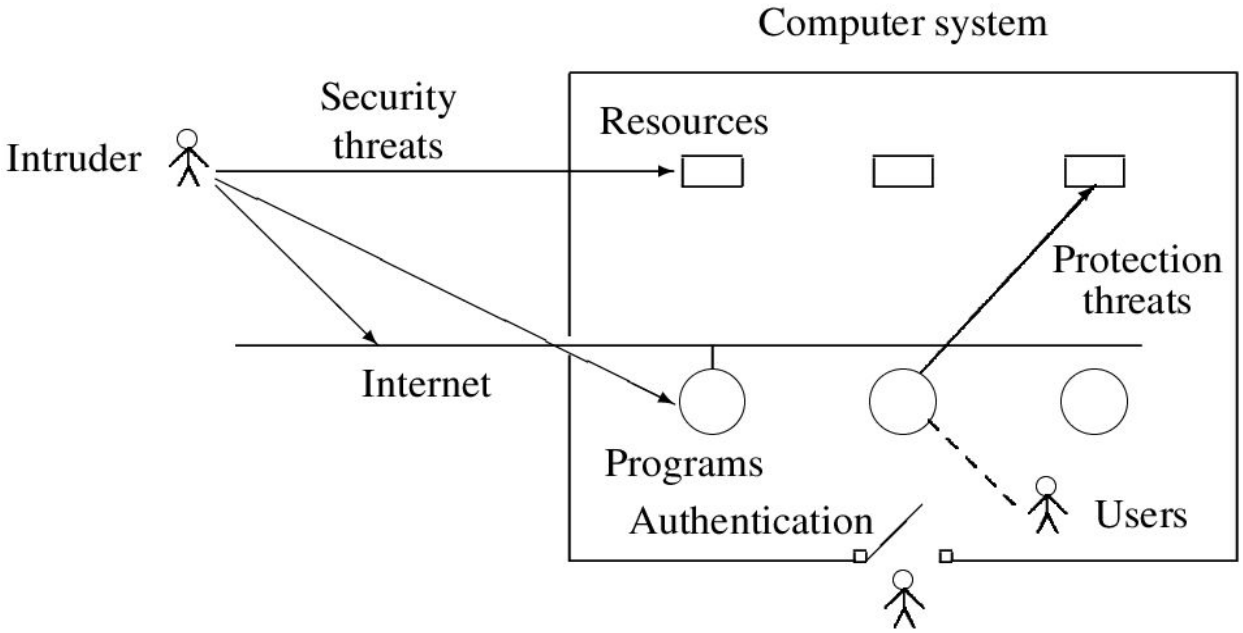
Security and Protection

- An OS must ensure non-interference with users' programs and data
  - Aspects of non-interference
    - Resources must not be used by un-authorized persons
    - There should be no interference with use of resources by authorized persons and their programs
  - Terminology
    - An *intruder* is a person or program that causes interference
    - *Security* measures avoid interference by non-users and their programs
    - *Protection* measures avoid interference by users and their programs

Security and protection techniques

- Two key techniques of security and protection
  - *Authentication*
    - Verification of a user's identity
    - Performed when a person logs in (typically through passwords)
  - *Authorization:* has two aspects
    - The owner of a resource selectively permits other users to access them
    - When a user/program tries to access a resource, the OS verifies whether the user has the permission to perform the access

Overview of security and protection threats



- Authentication checks whether a person is a registered user
- Threats posed by non-users are security threats
- Threats posed by a user are protection threats

Components of an Operating System

- In general there are two main components of an operating system:
  - **command interpreter** &
  - **Kernel**
- ***Command interpreter***
  - Command interpreter is one of the most important components of an operating system. **It is the primary interface between the user and the rest of the system.**
- ***Kernel***
  - **Kernel is a core part** of the operating system and is loaded on the main memory when it starts up. It is the core library of functions; the operating system 'knows'.
  - **In the kernel, there are the functions and streams to communicate with the system's hardware resources.**

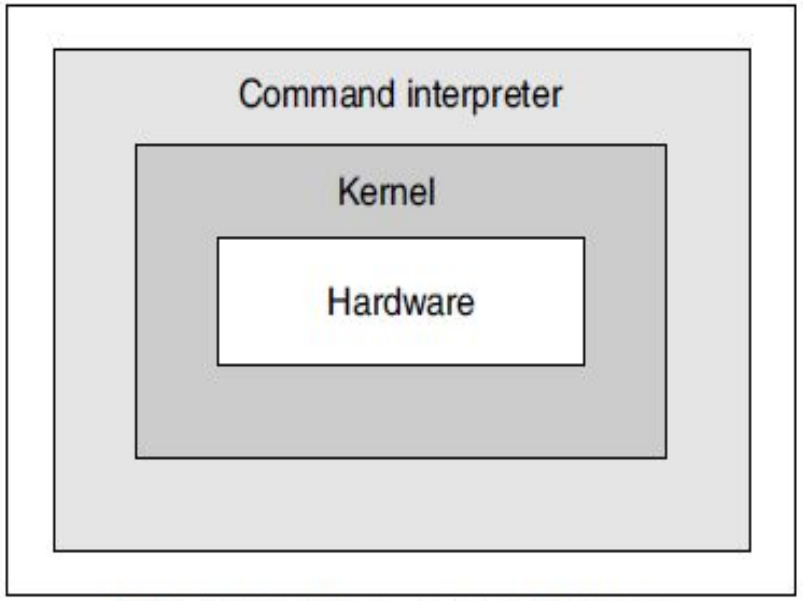


Fig. Operating system structure

Components of an Operating System

- In general there are two main components of an operating system:
  - command interpreter &
  - Kernel
- ***Command interpreter***

  Command interpreter is one of the most important components of an operating system. It is the primary interface between the user and the rest of the system.
- ***Kernel***
  - Kernel is a core part of the operating system and is loaded
  - on the main memory when it starts up. It is the core
  - library of functions; the operating system 'knows'.
  - In the kernel, there are the functions and streams to
  - communicate with the system's hardware resources.

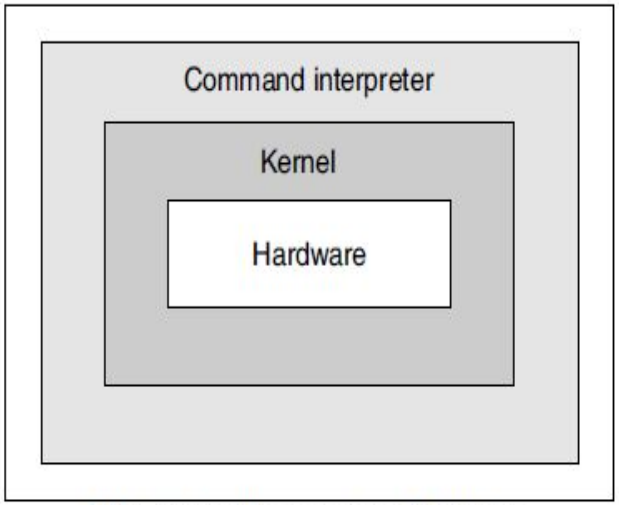


Fig. Operating system structure

Kernel

- Provide a mechanism for creation and deletion of processes
- Provide CPU scheduling ,memory management and device management for these processes
- Provide communication tools so that processes can communicate with each other.

Interaction with Operating System

- Broadly speaking, there are two ways to interact with an operating system:
  - **By means of operating *system calls in a program***
  - **Directly by means of *operating system commands***

- ***System calls***
  - **System calls provide the interface between a running program and the operating system**.
  - **These calls are generally available as assembly language instructions, and are usually listed in the manuals used by assembly language programmers**.
  - **Some systems may allow system calls to be made directly from a high-level language program, in which case the calls normally resemble predefined function or subroutine calls**.
  - **System calls can be roughly grouped into five major categories : *process control,***
  - ***file manipulation,***
  - ***device manipulation,***
  - ***information maintenance***
  - ***communications***.

Interaction with Operating System

- ***Operating system commands***

  - **Apart from system calls, users may interact with the operating system directly by means of commands.**
  - For example, if the user wants to list files or sub-directories in MSDOS, the DIR command is invoked.
  - In either case, the operating system acts as an interface between users and the hardware of a computer system.
  - The fundamental goal of a computer system is to solve user problems. The computer hardware is designed towards this goal.

  - The command function of controlling and allocating resources are then brought together into one piece of software, the operating system.

History of Operating Systems

- By tracing that evolution, the common elements of operating systems can be identified as well as how and why they developed as they are now.
  - Operating systems and computer architecture have a great deal of influence on each other. Operating systems were developed to facilitate the use of the hardware.
  - **First Generation (1945–55)**
  - **Second Generation (1956–63)—Transistors and Batch System**
  - **Third Generation (1964–80)—Integrated Chips and Multiprogramming**
  - **Fourth Generation (1980–present)— Personal Computers**

Evolution of Operating Systems

Generation	Period	Computer architecture	Problems and development of Operating systems
First	1940s–1950s	Vacuum tubes based technology, plug boards and punched cards, magnetic core memories	No operating system

Evolution of Operating Systems

Generation	Period	Computer architecture	Problems and development of Operating systems
Second	1950s–1960s	Transistors based technology, Mainframe computers, line printers, magnetic tapes, assemblers, linkers, loaders, compilers, FORTRAN, COBOL	Set up delay problem due to loading and unloading of tapes in earlier computer systems. CPU was idle. Jobs of users prepared with same programming language were batched together. Automated job sequencing Resident monitor Batch systems Mismatch between the speed of CPU and I/O devices Offline operation with magnetic tapes Tapes were sequential access devices

Evolution of Operating Systems

Generation	Period	Computer architecture	Problems and development of Operating systems
Third	1960s–1980s	IC based technology, Minicomputer Magnetic disk	Hard disks came into existence Spooling Multiprogramming Multi programmed batch systems Lack of user/programmer interaction with their jobs in multi programmed batch systems Timesharing multiuser systems CTSS MULTICS UNICS UNIX Unix written in C

Evolution of Operating Systems

Generation	Period	Computer architecture	Problems and development of Operating systems
Fourth	1980s–Present	LSI and VLSI based technology, Microcomputer	CP/M for PCs MS-DOS Multiuser facilities were not there in DOS XENIX OS/2 No user friendliness and convenience due to command driven and complex file systems Apple Macintosh Windows Multitasking Multithreading X-windows Motif Network operating systems Distributed operating systems

Types of Operating System

Type of operating system	Features/benefits	Example	Applicable to which type of application
Batch systems	More than one job can be stored in main memory Batches of same type of jobs can be executed quickly	FMS (FORTRAN monitor system), IBM's operating system for 7094	Background jobs in which the user interaction is not necessary
Multiuser systems	Jobs of different users who are connected to a main computer are executed through the multiprogramming Interaction of jobs with the user is possible Debugging is easy	CTSS by MIT, TSS by IBM, MULTICS, UNIX	When multiple users need to share a single system

Types of Operating System

Type of operating system	Features/benefits	Example	Applicable to which type of application
Multitasking systems	Multiple tasks of a single user can be opened on the system through multiprogramming	Windows	When a user wants to open and work simultaneously on many windows on the system
Network systems	The user is able to connect to another machine and perform many operations The user is aware of the location of the network node where he/she wants to connect	Novell Netware, Windows NT, Windows 2000, Windows XP, Sun Solaris	When a user wants to remote login on a system, wants to transfer a file, etc. on a network system

Types of Operating System

Type of operating system	Features/benefits	Example	Applicable to which type of application
Distributed systems	When multiple nodes of a wide network realized as a powerful machine sharing the resources on the network. The users are not aware where their processes are being sent and executed.	Amoeba, V system, Chorus	When computational speed and resource sharing is required and implemented through various full computer systems in a network
Real-time systems	Used to handle time-bound responses to the applications	pSOS, VxWorks, RTLinux, etc.	Applicable to systems which require time-bound response, i.e., for the real-time processing systems

Types of Operating System

Type of operating system	Features/benefits	Example	Applicable to which type of application
Embedded systems	Specialized systems with size, memory and power restrictions	Palm Pilot, Toshiba Pocket PC, Palm OS, Symbian OS, iPhone OS, RIM's BlackBerry, Windows Phone, Linux, Palm WebOS, Android and Maemo.	Used in consumer electronics items, mobile phones, smart cards, etc.

Types of Operating Systems

- Modern computer operating systems may be classified into three groups according to the nature of interaction that takes place between the computer user and user's program during its processing.
  - The three groups are called batch process, time-shared, and real-time operating systems.
- **Batch Process Operating System**
  - In a batch process operating system, environment users submit jobs to a central place where these jobs are collected in batch, and subsequently placed in an input queue in the computer where they are run.

Batch Processing

- A typical computer in the 1960s and ‘70s was a large machine
- Its processing was managed by a human *operator*
- The operator would organize various jobs from multiple users into *batches*

Batch Processing

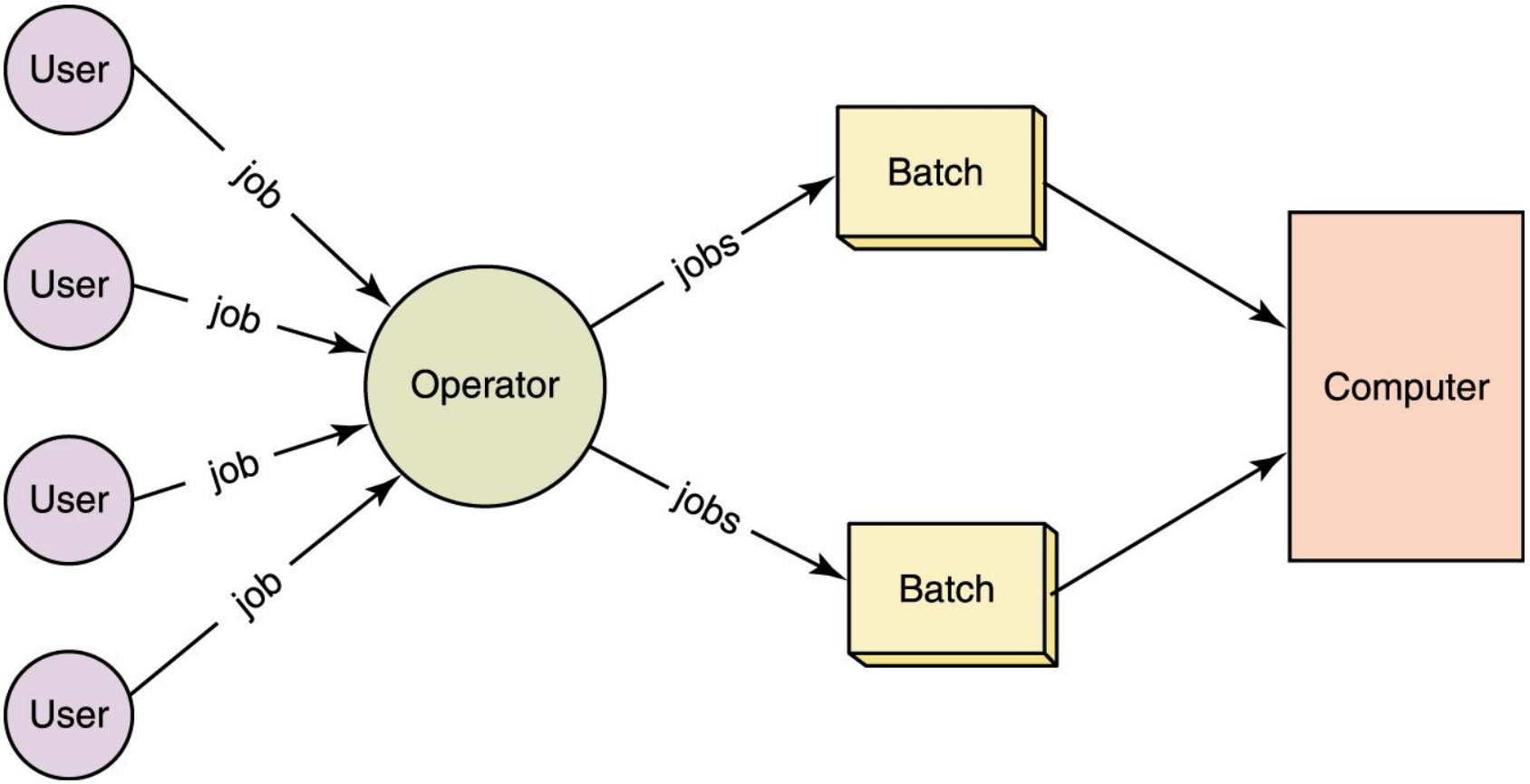


Figure 10.2 In early systems, human operators would organize jobs into batches

Two major disadvantages and they are as follows:

Non-interactive environment

Offline debugging

Types of Operating Systems

- **Multiprogramming Operating System**
  - A multiprogramming operating system allows more than one active user program (or part of user program) to be stored in the main memory simultaneously.
  - Compared to batch operating systems, multiprogramming operating systems are fairly sophisticated.
    - ***Multitasking operating systems***
    - ***Multi-user operating system***
    - ***Multiprocessing system***

- **Time-sharing Operating Systems**
  - Another mode for delivering computing services is provided by time-sharing operating systems.
  - In this environment a computer provides computing services to several or many users concurrently online.

Types of Operating Systems

- **Real-time Operating Systems**
  - The fourth class of operating systems, real-time operating systems, are designed to service those applications where response time is of essence in order to prevent error, misrepresentation, or even disaster.

- **Network Operating System**
  - A networked computing system is a collection of physically interconnected computers.
  - The operating system of each of the interconnected computers must contain, in addition to its own stand-alone functionality, provisions for handling communication and transfer of program and data among the other computers with which it is connected.

Types of Operating Systems

- **Distributed Operating System**
  - A distributed computing system consists of a number of computers that are connected and managed so that they automatically share the job-processing load among the constituent computers, or separate the job load, as appropriate, to particularly configured processors.
- ***Advantages of distributed operating systems :***
  - **Major breakthrough in microprocessor technology**
  - **Incremental growth**
  - **Reliability**
  - ***File system***
  - ***Protection***
  - ***Program execution***

Overview of UNIX Operating System

- UNIX is an operating system. It was created in the late 1960s,in an effort to provide a multi-user, multitasking system for use by programmers.

- The philosophy behind the design of UNIX was to provide simple, yet powerful utilities that could be pieced together in a flexible manner to perform a wide variety of tasks.

Reasons for Success of UNIX

- During the past 30 years, UNIX has evolved into a powerful, flexible, and versatile operating system.

- It is used on

  (a) single user personal computers,
  (b) engineering workstations,
  (c) multi-user microcomputers,
  (d) minicomputers,
  (e) mainframes, and
  (f) Supercomputers

Features of Unix

- The reasons for this are the characteristics of UNIX, enumerated as follows:
- **Portability:**
  - Because the UNIX operating system is written mostly in C, it is highly portable.
  - It runs on a range of computers from microprocessors to the largest mainframe , provided the system has two components: a C compiler, and a modest amount of machine-dependent coding (machine dependent I/O hardware service routines).
- **Open system :**
  - It easily adapts to particular requirements.
  - This openness has led to the introduction of a wide range of new features and versions customized to meet special needs.
  - The code for UNIX is straightforward, modular, and compact.
  - This has fostered the evolution of the UNIX system.
- **Rich and productive programming environment :**
  - UNIX provides users with powerful tools and utilities. Some of these tools are simple commands that can be used to carry out specific tasks.
  - Other tools and utilities are really small programmable languages that may be used to build scripts to solve problems.
  - More importantly, the tools are intended to work together, like machine parts or building blocks.

Features of Unix

- **Communication:**
  - The UNIX system provides an excellent environment for networking.
  - It offers programs and utilities that provide the services needed to build networked applications, the basis for distributed network computing.
- **Multi-user capability:**
  - More than one user can access the same data at the same time.
  - A computer system that can support multiple users is generally less expensive than the equivalent number of single-user machines.
- **Multitasking:**
  - A given user can perform more than one task at the same time. One could update the client's database while printing the monthly sales report.
  - The limit is about 20 simultaneous tasks per user and depending on the computer system, a system-wide limit of 50 or more tasks can be performed, which slows the response.

Components of UNIX

- UNIX carries out various functions through three separate, but closely integrated parts: kernel, command interpreter, and file system.
- ***Kernel :*** Known as the base operating system, kernel manages and allocates resources, interacts with I/O devices, and controls access to the processor. It controls the computer's resources.
- In short, it provides the following functions:
  - Process scheduling (process representation—structure, scheduling, and dispatching)
  - Memory management
  - Device management
  - File management
  - System call interface
  - Process synchronization and inter-process communication
  - Operator console interface

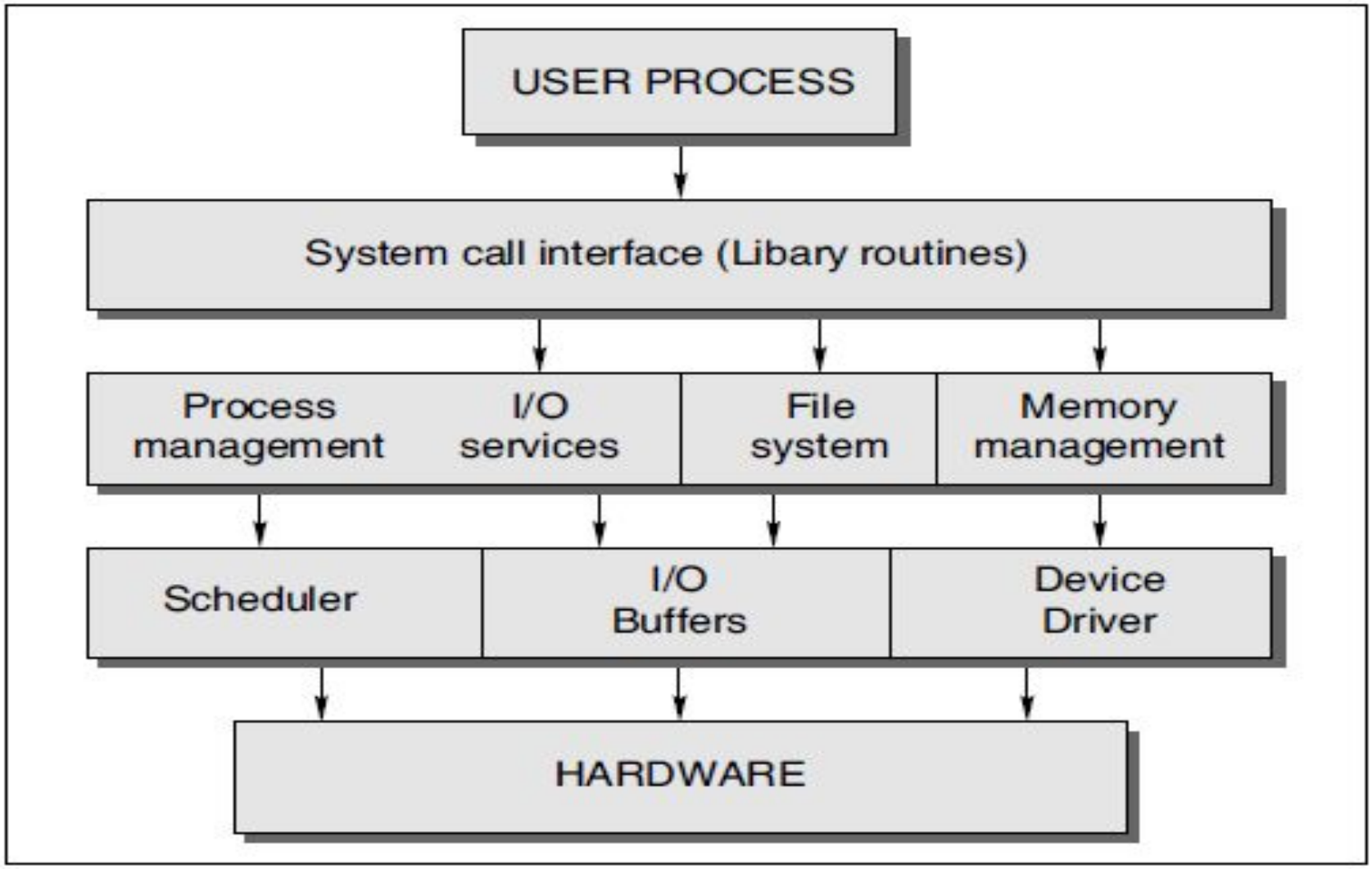


Fig. Functional layer model of the `unix` kernel

Command Interpreter

- This is a utility program and is called the *shell. It interacts* with the user and translates the user's request into actions on the part of the kernel and the other utility . Each user opens one shell on logging on.

- Different types of shells are available such as Bourne shell, C shell, and Korn shell.
  - Protection of file data
  - The treatment of peripheral devices as files

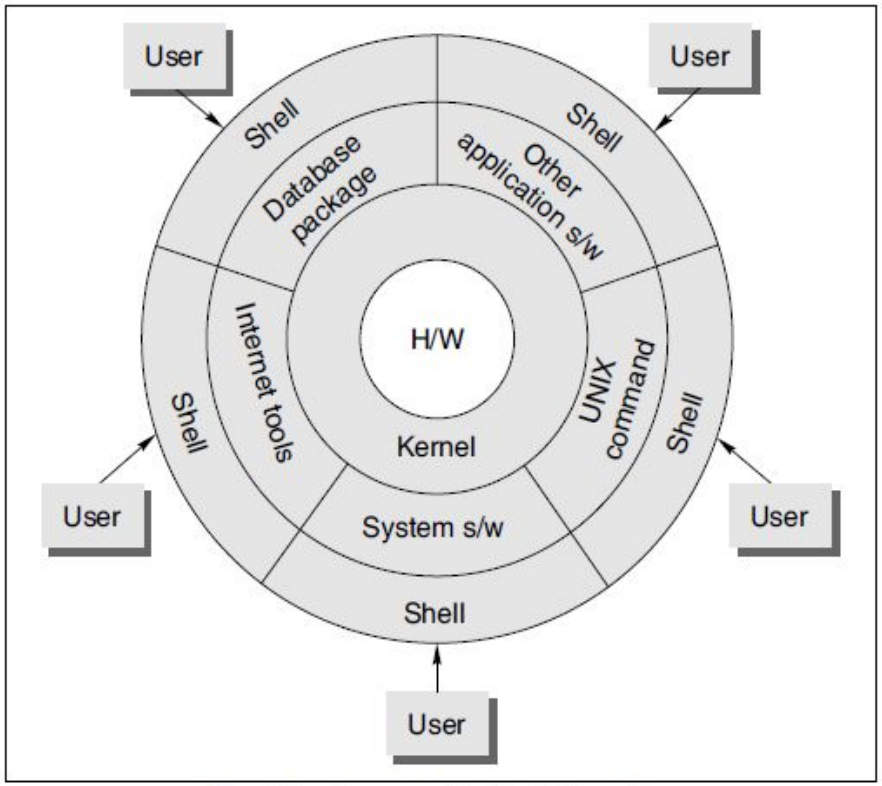


Fig. The kernel–shell relationship

The UNIX File System

- The file system is one of the major subsystems of the operating system.
  - It is responsible for storing information on disk drives and retrieving and updating this information as directed by the user or by a program.
  - The UNIX operating system regards practically every assemblage of information as a file. The formal definition of a file is a string of characters.
  - Often, it is desirable to organize UNIX files as a set of lines. Every line is terminated by a new line character.

The UNIX File System

- The UNIX file system is characterized by the following:
  - A hierarchical structure
  - Consistent treatment of file data
  - The ability to create and delete files
  - Dynamic growth of files
- ***Types of files:*** The UNIX system has the following types of files:
  - Ordinary files
  - Directory files
  - Special files

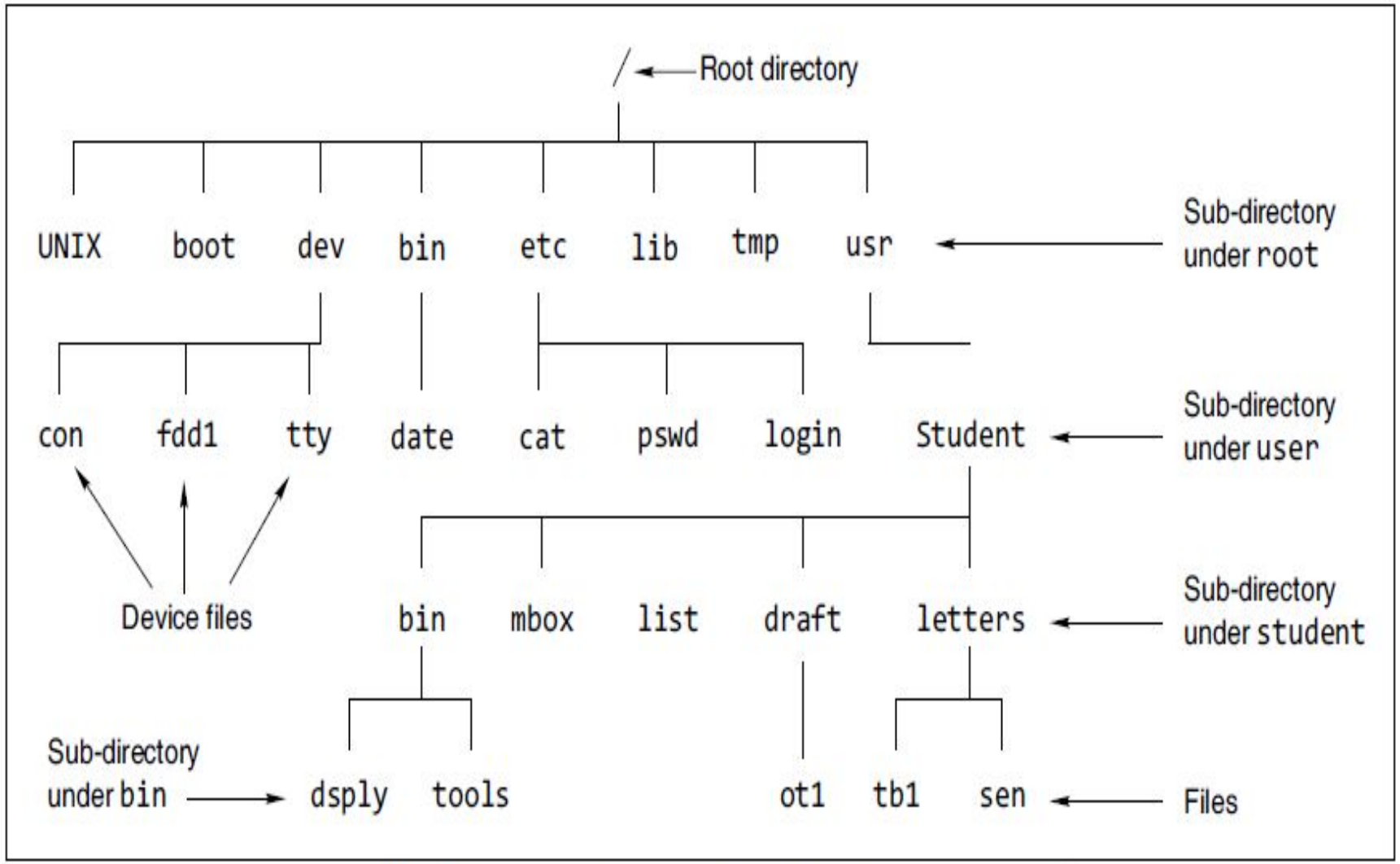


Fig. The typical tree structure of the file system in UNIX

Example

- **Problem :-** Using a UNIX command, return to home directory.

- **Solution:**

  - Cd..
  - Issuing the cd command without any arguments moves the choice to the home directory. This is very useful if the user is lost in the file system.

- **The directories . and ..**

  - In UNIX, (.) means the current directory, so typing cd . means staying in the current directory. While (..) means the parent of the current directory, so typing cd .. will take the user one directory up the hierarchy, that is, back to the user's home directory.
  - Note that there is a space between cd and the dot. Entering cd/ moves the user to the root directory. / is the root directory.

Objectives

- General operation of an operating system
- Booting the OS
- System calls, their execution and types

BASIC INPUT OUTPUT SYSTEM (BIOS)

BIOS provides basic functionality to operate and control the hardware connected to or built into the computer.

- The BIOS is built into the computer and is the first code run by the computer when it is switched on.
- The key role of the BIOS is to load and start the operating system.
- When the computer starts, the first function that BIOS performs is to initialize and identify system devices such as the video display card, keyboard and mouse, hard disk, CD/DVD drive and other hardware.
- The code in the BIOS chip runs a series of tests called POST (Power On Self Test) to ensure that the system devices are working correctly.
- The BIOS then locates software held on a peripheral device such as a hard disk or a CD, and loads and executes that software, giving it control of the computer. This process is known as *booting*.

- BIOS is stored on a ROM chip built into the system.

General operation of an operating system

Booting/ Boot strapping

process to place the operating system in memory is known as booting or bootstrapping.

Boot Software/Boot Loader/Bootstrap Loader

The set of instructions needed for booting, i.e. to load the operating system in RAM is known as Boot Software/Boot Loader/Bootstrap Loader.

General operation of an operating system

Boot Device

The device that stores the operating system is called boot device.

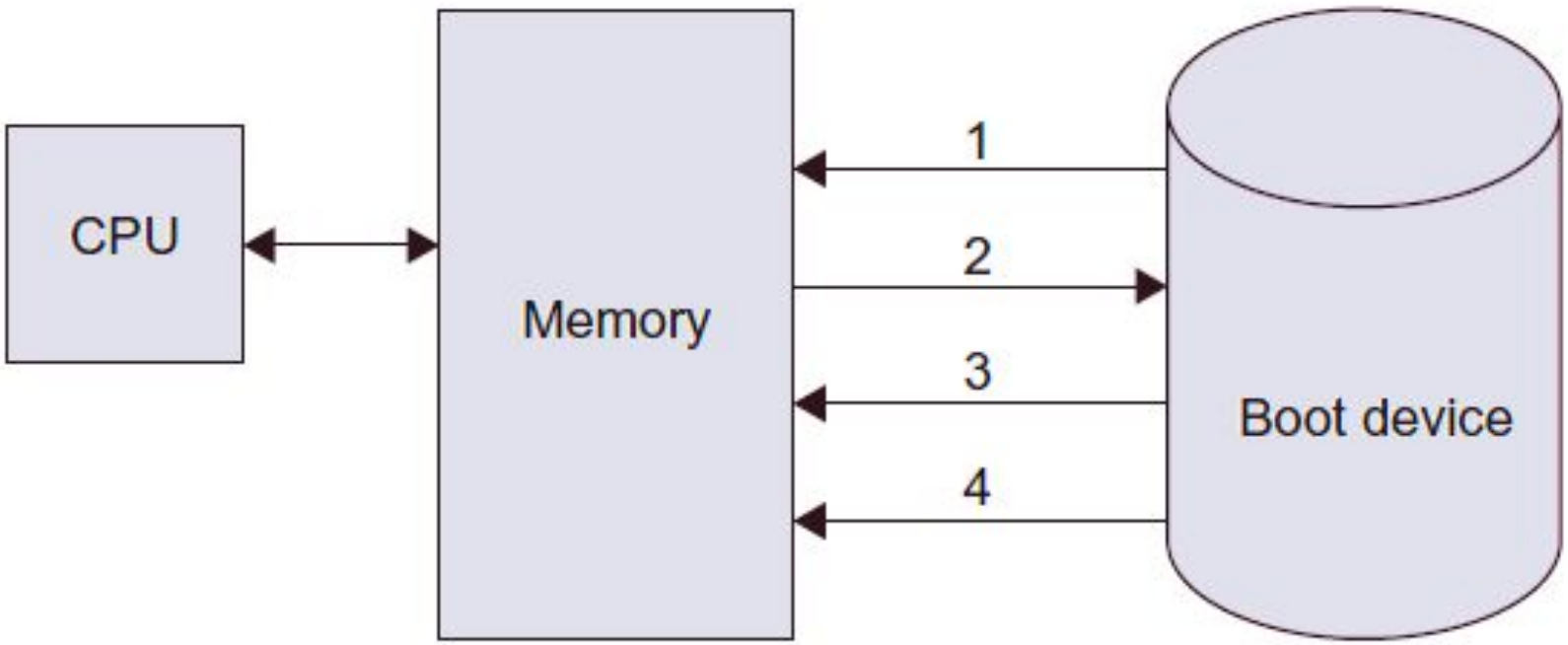
Privileged instructions

the instructions which are not directly executed by the user but need to be passed to the operating system are known as privileged instructions.

System Call

All the privileged instructions, i.e. the instructions which need to interact with hardware and other resources and therefore passed to the operating system for execution, are known as system calls.

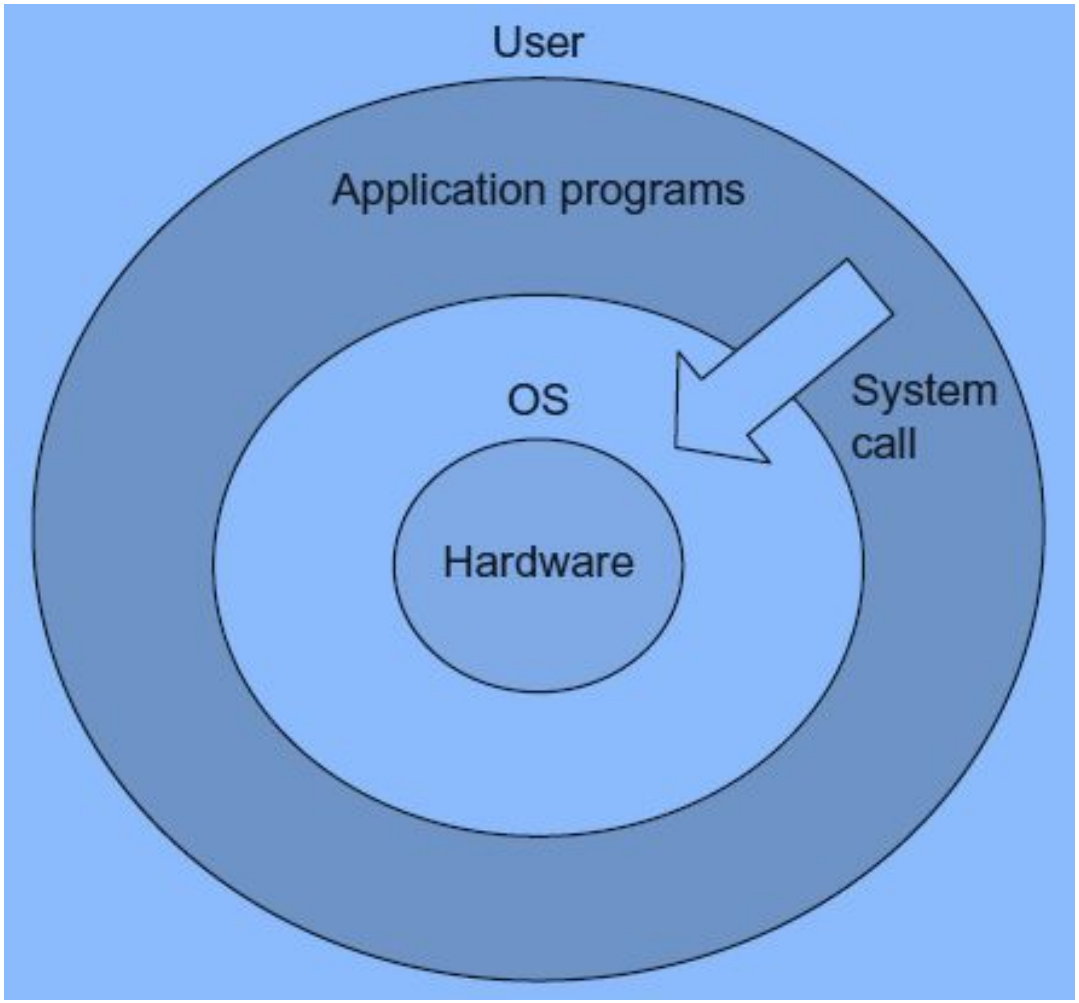
Booting Sequence



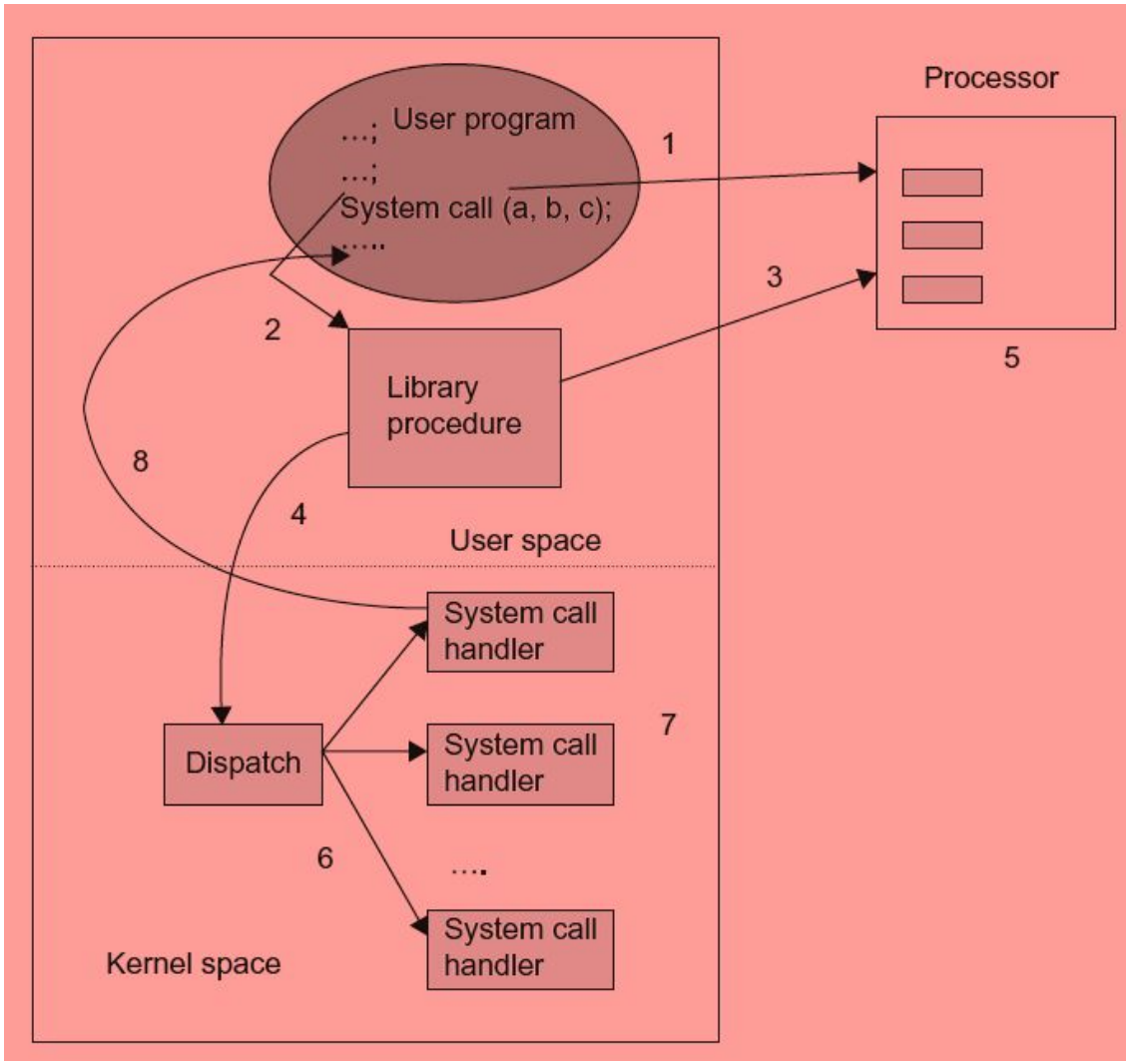
System calls

- System call is an instruction which requests the operating system to perform the desired operation which needs hardware access or other privileged operations.
- System call in fact generates an interrupt that causes the operating system to gain the control of the CPU.
- The operating system then finds out the type of the system call and the corresponding interrupt handler routine is executed to perform the operations desired by the user through the system call.
- Therefore, system call is just a bridge between the user programs and operating system for executing the privileged operations

System call interface



Steps to execute a system call



1. In the user program when the system call is executed, first of all, its parameters are pushed onto the stack and later on saved in the processor registers.
2. The corresponding library procedure for the system call is executed.
3. There is a particular code for every system call by which the kernel identifies which system call function or handler needs to be executed. Therefore, library procedure places the system call number in the processor register.
4. Then the library procedure traps to the kernel by executing interrupt instruction. With this interrupt execution, the user mode switches to kernel mode by loading Program Status Word (PSW) register to 0.

5. The hardware saves the current contents of CPU registers, so that after executing the system call, the execution of the rest of the program can be resumed.
6. The kernel identifies the system call by examining its number and dispatches the control to the corresponding system call handler.
7. The system call handler executes.
8. On completion of system call handler, the control is returned to the user program and it resumes its execution.

Type of System Calls

- Process Control System Calls

- File Management System Calls
- Device Management system Calls
- Information Maintenance system calls
- Communication system Calls

Process Control System Calls

System call	UNIX example
Create a process: Creating a new process	`fork()`
Terminate a process: When a process executes its operation, it exits normally.	`exit()`
Terminate a process abnormally: There may be situations in which you need to terminate the process in between; for example, there is hang situation, program has been stuck in an indefinite loop, and the performance of system has been affected such that no processing is being performed.	`kill()`
Increase the priority of a process: Some processes have got more importance than others. So their execution must get priority over others. This is done by setting and increasing the priority of the process.	`Nice()`
Suspend the process: There may be situations in which a process needs to be suspended but not terminated. It will resume again after receiving some signal.	`pause()`
Cause the process to sleep: A process may need to wait for I/O devices. In that period of time, the processor switches to some other process and the current process is blocked to wait or sleep for some time.	`wait()`

File Management System Calls

System call	UNIX example
Create a file: Creating a new file.	`Creat()`
Open a file: Opening a file that is already created.	`Open()`
Close a file: Closing a file that has been opened earlier.	`Close()`
Read a file: Reading a file that has been opened earlier.	`Read()`
Write a file: Writing into a file that has already been opened.	`Write()`
Change the position of the read-write pointer: There is a need to access any part of a file randomly. File pointer indicates the current position in the file. This call changes its position as desired.	`Lseek()`
Give another name to a file: This call allows a file to appear in different directories with different names. But the copy of the file is single. It means that if there is change in the file, it is visible in all the directories wherever it appears. This is done through the unique ID of the file (known as i-number), which is an index into a table of entries known as *i-nodes*. These entries in the table store the information of a file, such as who owns the file, its disk blocks, etc. So, the i-number is same for all entries of the file in different directories. However, there is a single file; only the name is different under different directories. The file is accessible through either name.	`Link()`
Delete a file in a directory: This call removes the entry of a file in one directory.	`Unlink()`
Make a directory: Create a new directory.	`Mkdir()`
Remove a directory: Delete an existing directory.	`Rmdir()`
Change the directory: When you are working in a directory, you can move to some other directory by using this call.	`Chdir()`
Change the mode: There are various modes and groups of users who will use the files. For a particular group, there may be different access permissions (modes) such as read, write, or execute. This call changes the access permissions of a file to the specified mode.	`Chmod()`
Change ownership of file: Changes the owner and group of the indicated file.	`Chown()`

Information Maintenance system calls

System call	UNIX example
Get process identification number: Every process has a unique identification number. If the user wants to see the same, this call is used.	`Getpid()`
Get status information of a file: The information regarding a file such as the file type and its permissions, size, time of last access, and so on.	`Stat()`
Set the system date and time	`Stime()`
Get statistics about a file system: Gets the statistics about a file system such as number of free blocks, file system name, and so on.	`Ustat()`

Communication system Calls

System call	UNIX example
Sending a message	`Msgsnd()`
Receiving a message	`Msgrcv()`