MODULE 1

Overview of operating system

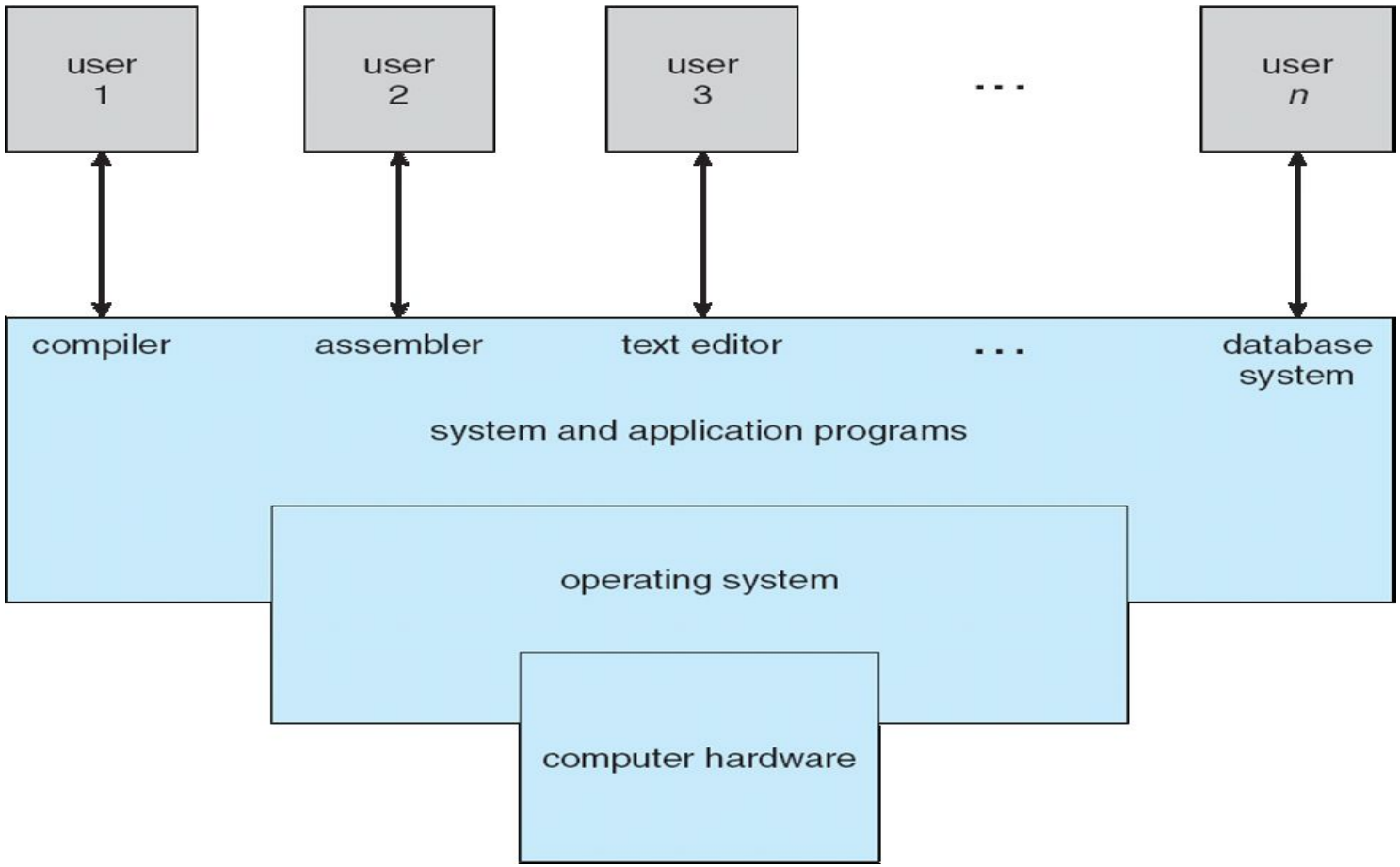
Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware

Operating System Objectives

- Convenience
  - Makes the computer more convenient to use
- Efficiency
  - Allows computer system resources to be used in an efficient manner
- Ability to evolve
  - Permit effective development, testing, and introduction of new system functions without interfering with existing services.

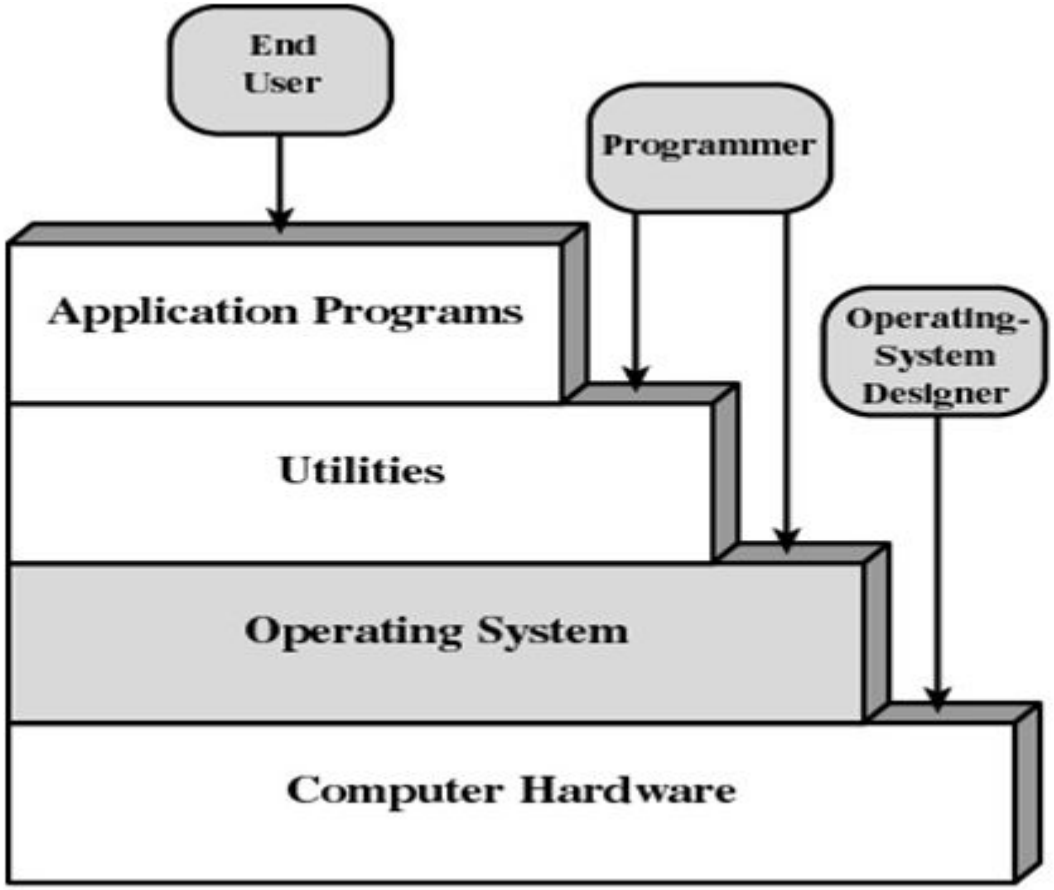
Four Components of a Computer System



Computer System Structure

- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Layers and Views of Computer System



Layers and Views of a Computer System

Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Operating System Definition (Cont.)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel.**
  - Everything else is either a system program or an application program.

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Operating System

- Functions same way as ordinary computer software
  - It is program that is executed
- Operating system relinquishes control of the processor to execute other programs

OS Components and Functions

Services Provided by the Operating System

- Program development
  - Editors and debuggers
- Program execution
- Access to I/O devices
- Controlled access to files
- System access

Services Provided by the Operating System

- Error detection and response
  - internal and external hardware errors
    - memory error
    - device failure
  - software errors
    - arithmetic overflow
    - access forbidden memory locations

Services Provided by the Operating System

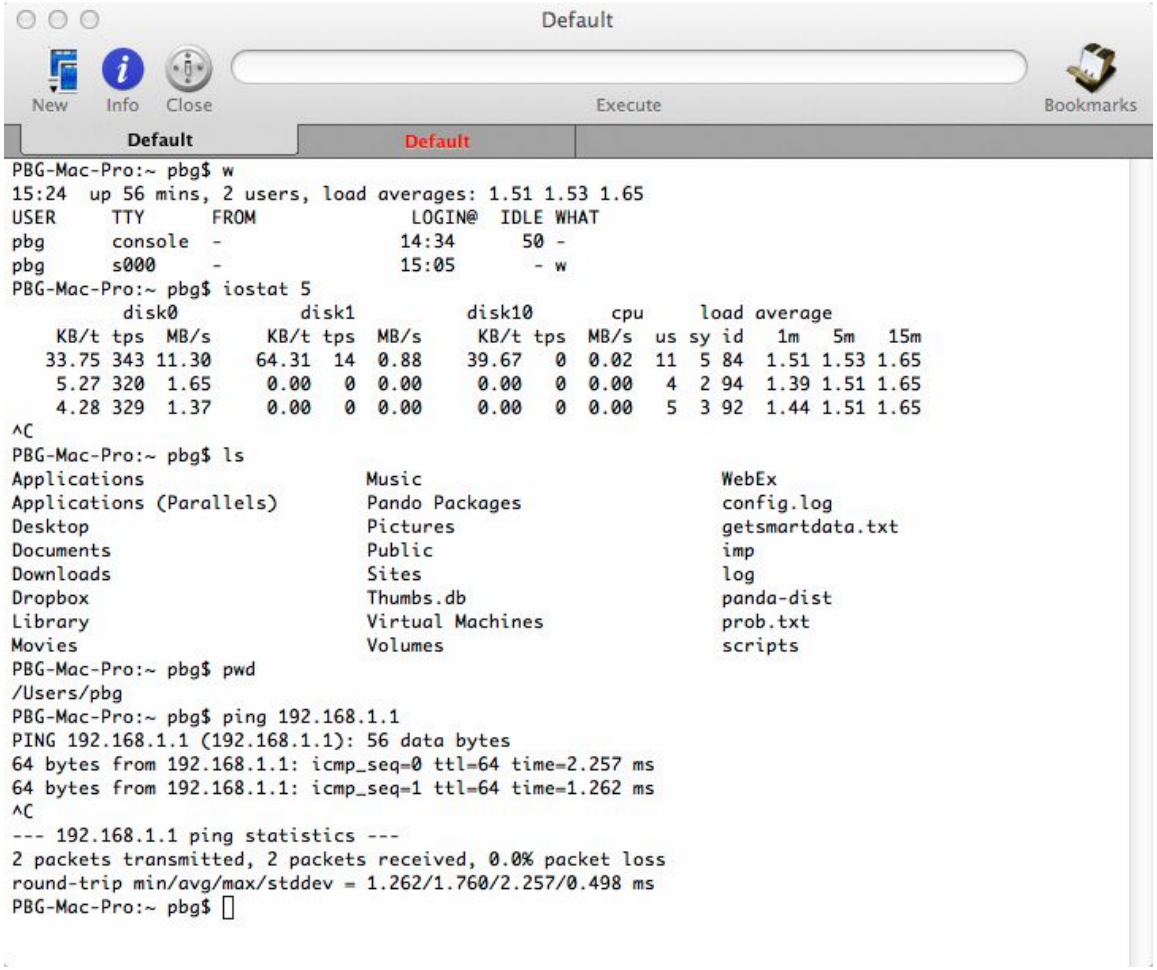
- Accounting
  - collect statistics
  - monitor performance
  - used to anticipate future enhancements
  - used for billing users (potentially)

Interfaces

User Operating System Interface - CLI

- CLI or command interpreter 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 doesn't require shell modification

Bourne Shell Command Interpreter



User Operating System Interface - GUI

- User-friendly desktop metaphor interface
  - Usually 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 is “Aqua” GUI interface with UNIX kernel underneath and shells available
  - Unix and Linux have CLI with optional GUI interfaces (CDE, KDE, GNOME)

Touchscreen Interfaces

- Touchscreen devices require new interfaces
  - Mouse not possible or not desired
  - Actions and selection based on gestures
  - Virtual keyboard for text entry
- Voice commands.

The Mac OS X GUI

System Programs

System Programs

- System programs provide a convenient environment for program development and execution. They can be divided into:
  - File manipulation
  - Status information sometimes stored in a File modification
  - Programming language support
  - Program loading and execution
  - Communications
  - Background services
  - Application programs
- Most users' view of the operation system is defined by system programs, not the actual system calls

System Programs

- Provide a convenient environment for program development and execution
  - Some of them are simply user interfaces to system calls; others are considerably more complex
- File management - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- Status information
  - Some ask the system for info - date, time, amount of available memory, disk space, number of users
  - Others provide detailed performance, logging, and debugging information
  - Typically, these programs format and print the output to the terminal or other output devices
  - Some systems implement a registry - used to store and retrieve configuration information

System Programs (Cont.)

- File modification
  - Text editors to create and modify files
  - Special commands to search contents of files or perform transformations of the text
- Programming-language support - Compilers, assemblers, debuggers and interpreters sometimes provided
- Program loading and execution- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language
- Communications - Provide the mechanism for creating virtual connections among processes, users, and computer systems
  - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

System Programs (Cont.)

- Background Services
  - Launch at boot time
    - Some for system startup, then terminate
    - Some from system boot to shutdown
  - Provide facilities like disk checking, process scheduling, error logging, printing
  - Run in user context not kernel context
  - Known as services, subsystems, daemons
- Application programs
  - Don't pertain to system
  - Run by users
  - Not typically considered part of OS
  - Launched by command line, mouse click, finger poke

System Calls

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Programming 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 Call

- A system call is how a process requests a service from an operating system's kernel.
- This may include
  - Hardware related services (e.g. accessing the hard disk)
  - Creating and executing new processes
  - Communicating with integral kernel services (like scheduling).
- System calls are interface between a user process and the system.

System Call Categories

- System calls can be roughly grouped into five major categories:
  - Process Control
  - File management
  - Device Management
  - Information Maintenance
  - Communication

System Call Categories

- Process Control
  - load
  - execute
  - create process (for example, fork on Unix-like systems or NtCreateProcess in the Windows NT Native API)
  - terminate process
  - get/set process attributes
  - wait for time, wait event, signal event
  - allocate, free memory

System Call Categories

- File management
  - create file, delete file
  - open, close
  - read, write, reposition
  - get/set file attributes
- Device Management
  - request device, release device
  - read, write, reposition
  - get/set device attributes
  - logically attach or detach devices

System Call Categories

- Information Maintenance
  - get/set time or date
  - get/set system data
  - get/set process, file, or device attributes
- Communication
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach or detach remote devices

OS Evolution

Evolution of an Operating System

- When
  - Hardware upgrades and new types of hardware
  - New services
  - Fixes

Serial Processing

- No operating system
- Machines run from a console with display lights and toggle switches, input device, and printer
- Schedule tome (book)
- Setup included loading the compiler, source program, saving compiled program, and loading and linking

Simple Batch Systems

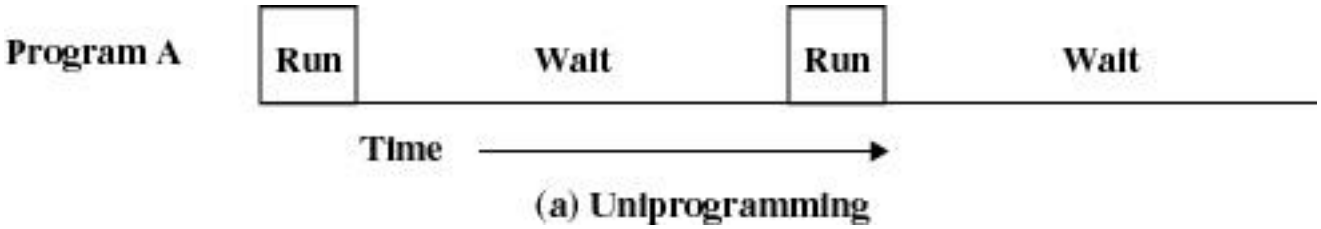
- Monitors
  - Software that controls the running programs
  - Batch jobs together
  - Program branches back to monitor when finished
  - Resident monitor is in main memory and available for execution
- Job Control Language (JCL)
  - Special type of programming language
  - Provides instruction to the monitor
    - what compiler to use
    - what data to use

Simple Batch Systems

- Hardware Features
  - Memory protection
    - do not allow the memory area containing the monitor to be altered
  - Timer
    - prevents a job from monopolizing the system

Uniprogramming

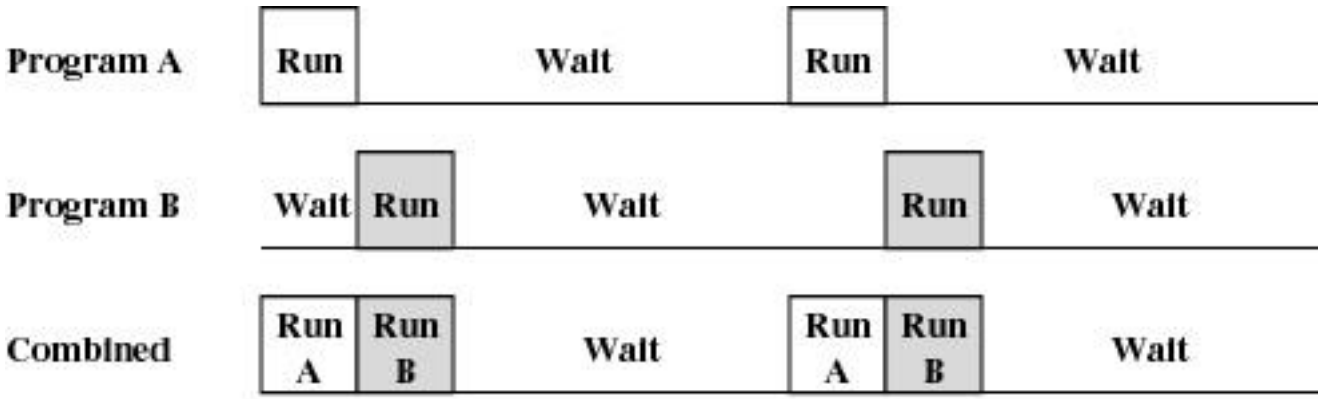
- Processor must wait for I/O instruction to complete before preceding



(a) Uniprogramming

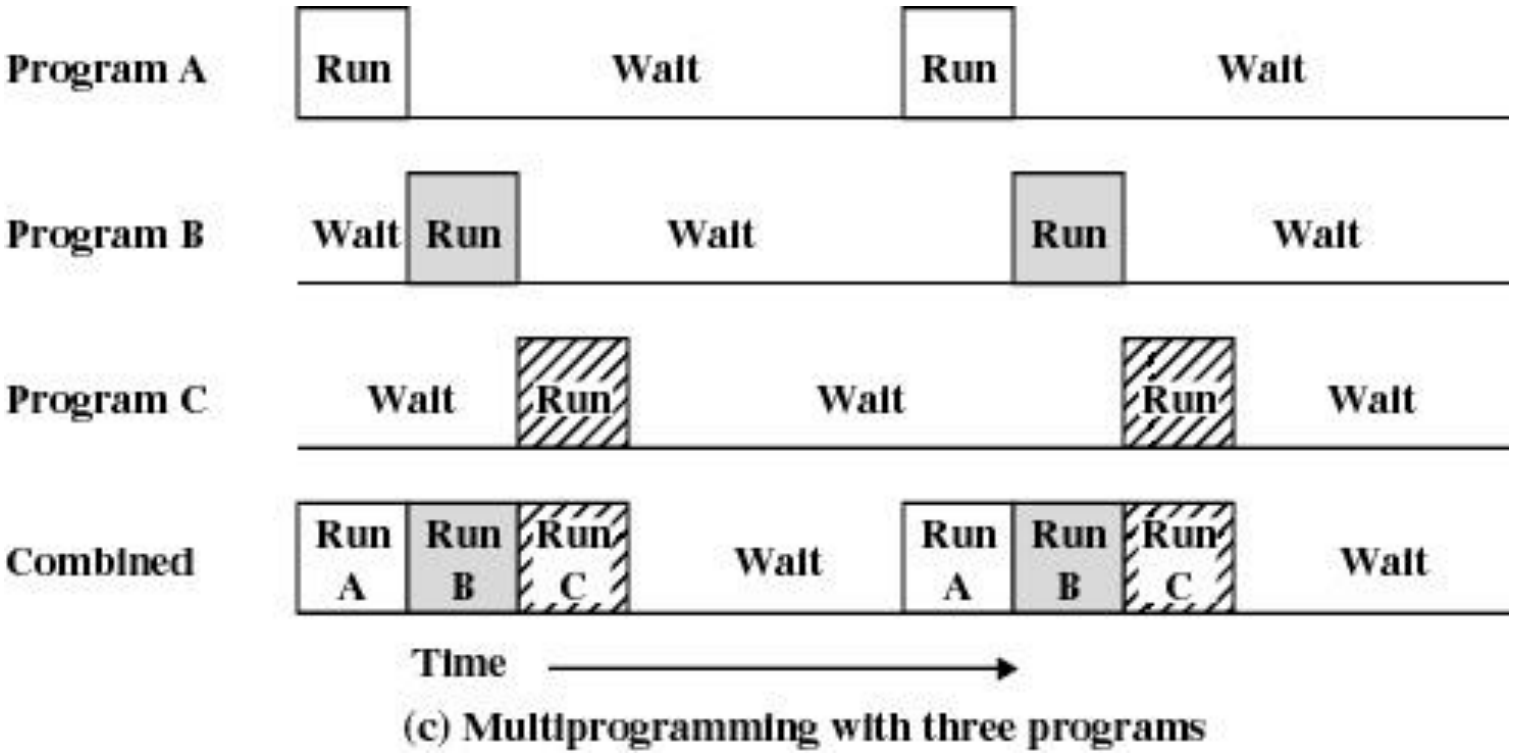
Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

Multiprogramming



(c) Multiprogramming with three programs

Time Sharing

- Using multiprogramming to handle multiple interactive jobs
- Processor's time is shared among multiple users
- Multiple users simultaneously access the system through terminals

OS Architecture

Operating Systems Architectures (Kernel Architectures)

- Operating systems tend to be complex
  - Provide many services
  - Support variety of hardware and software
  - Operating system architectures help manage this complexity
    - Organize operating system components

      (e.g. Memory Management, Process Scheduler, IPC, File System , I/O Controllers, Network Managers …)
    - Specify privilege with which each component executes

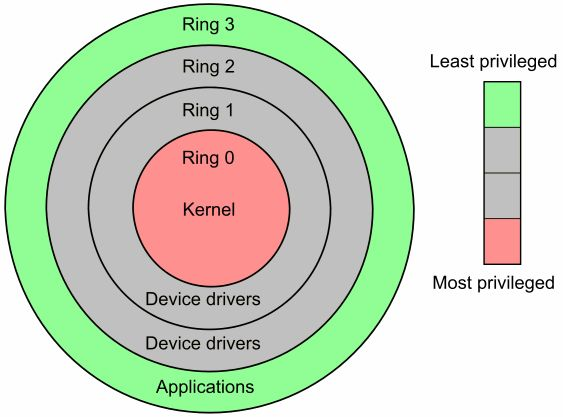
Kernel

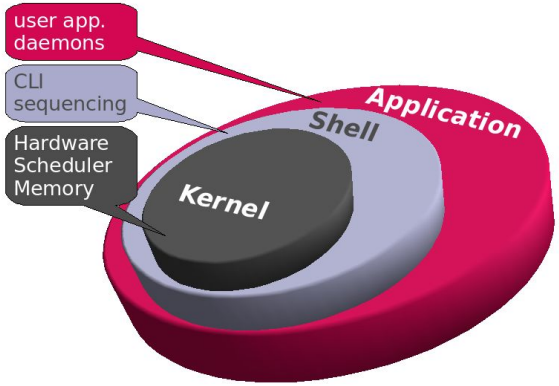
- Also called the nucleus
- Portion of operating system that is in main memory
- Functionalities
  - Primary function is to mediate access to the computer's resources, including
    - The Central Processing Unit (CPU)
    - Random Access Memory
    - Input/output (I/O) devices
  - Kernels also usually provide methods for synchronization and communication between processes called inter-process communication (IPC).
  - Memory Management.
    - Virtual Addressing (paging and/or segmentation)

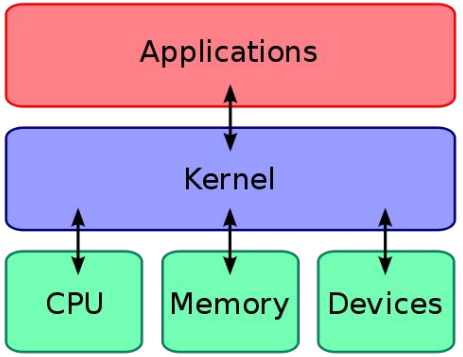
Kernel

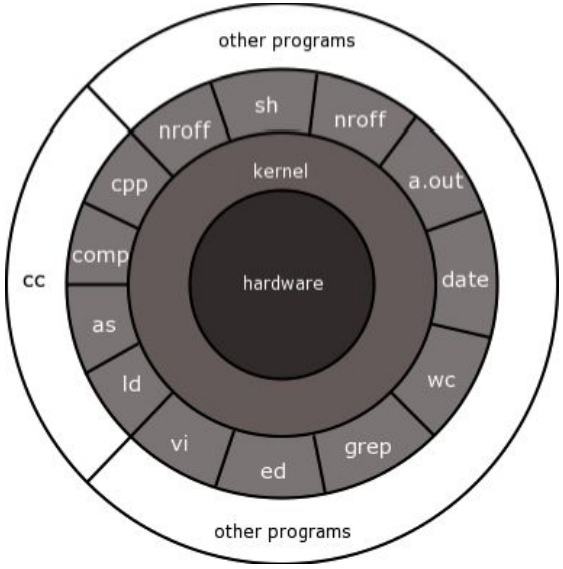
- Kernel is the core of Operating System
- All the functionalities of an OS may be included in this core.
  - This leads to different types of Kernels or Kernel Architecture.
- (In modern Operating Systems) User processes don't have direct control over system resources.
  - To accomplish a task the processes make requests to the kernel.
  - These requests are called **System Calls**.

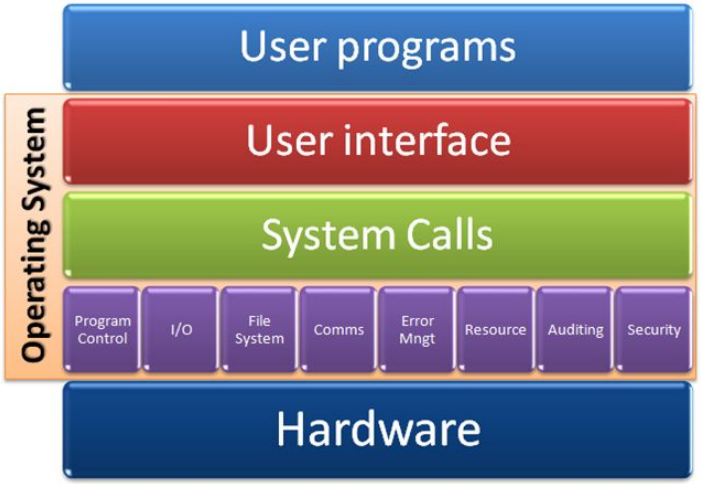
Kernel (Different Representations)

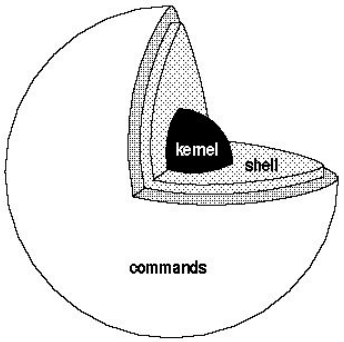












Source : Google Images

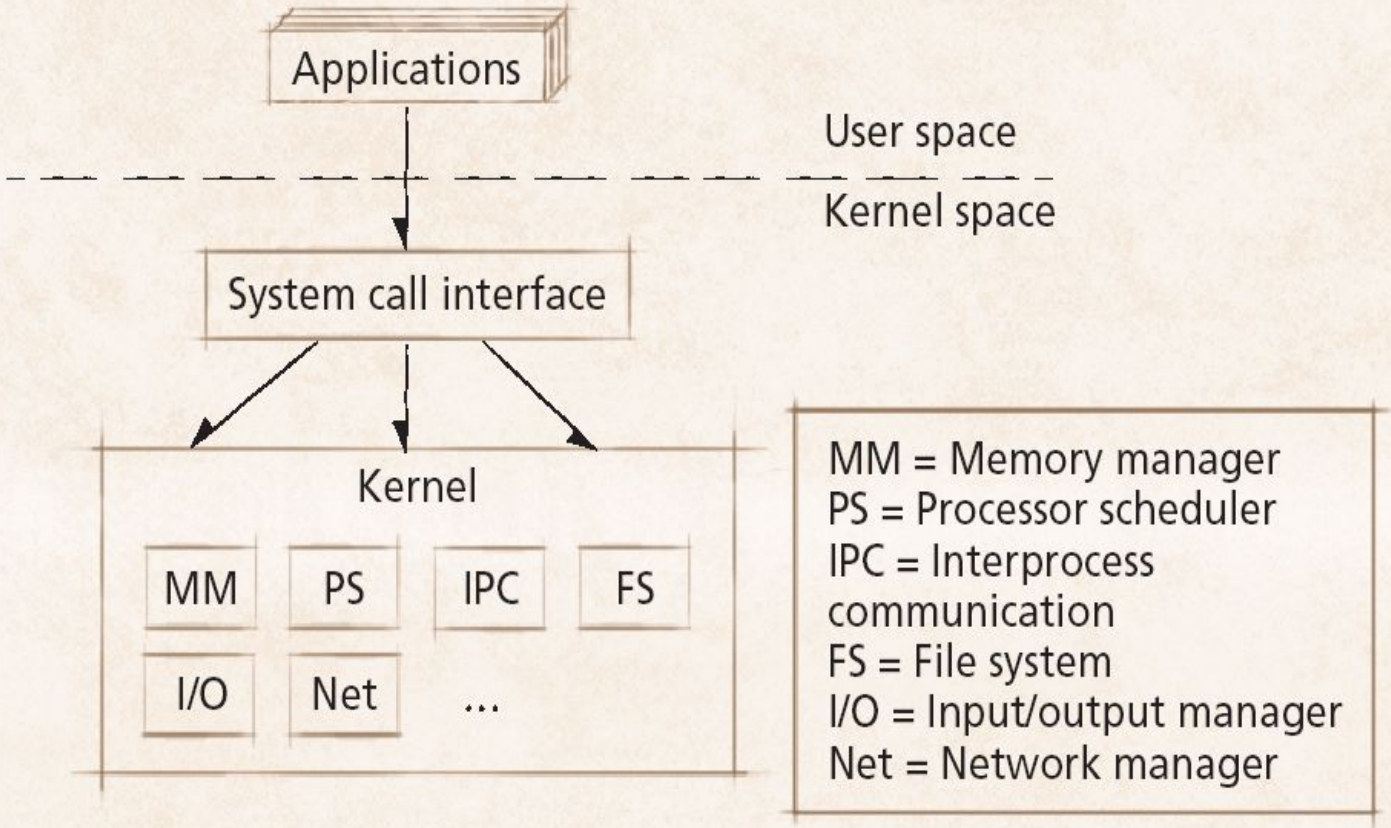
Kernel Architectures

- Monolithic
- Layered
- Micro-kernel
- Distributed

Monolithic Architecture

- Monolithic operating system
  - Every component contained in kernel
    - Direct communication among all elements
    - Highly efficient
  - Problems:
    - Complexity
    - New devices, emerging technologies
  - E.g. DOS

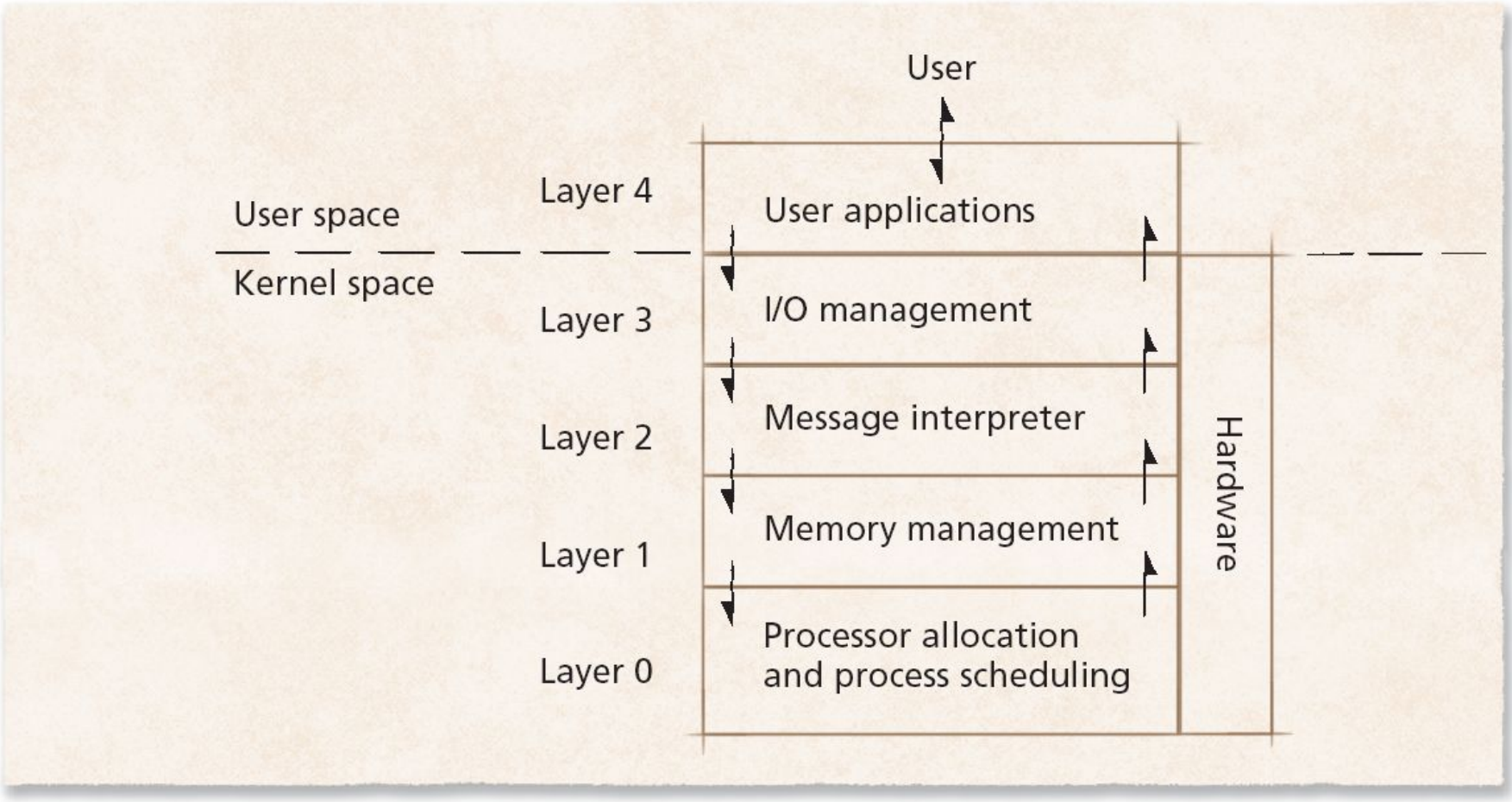
Monolithic Architecture



Layered Architecture

- Groups components that perform similar functions into layers
- Each layer communicates only with adjacent layer
- System calls might pass through many layers before completion
- Drawback : Low application performance.
- E.g. Unix

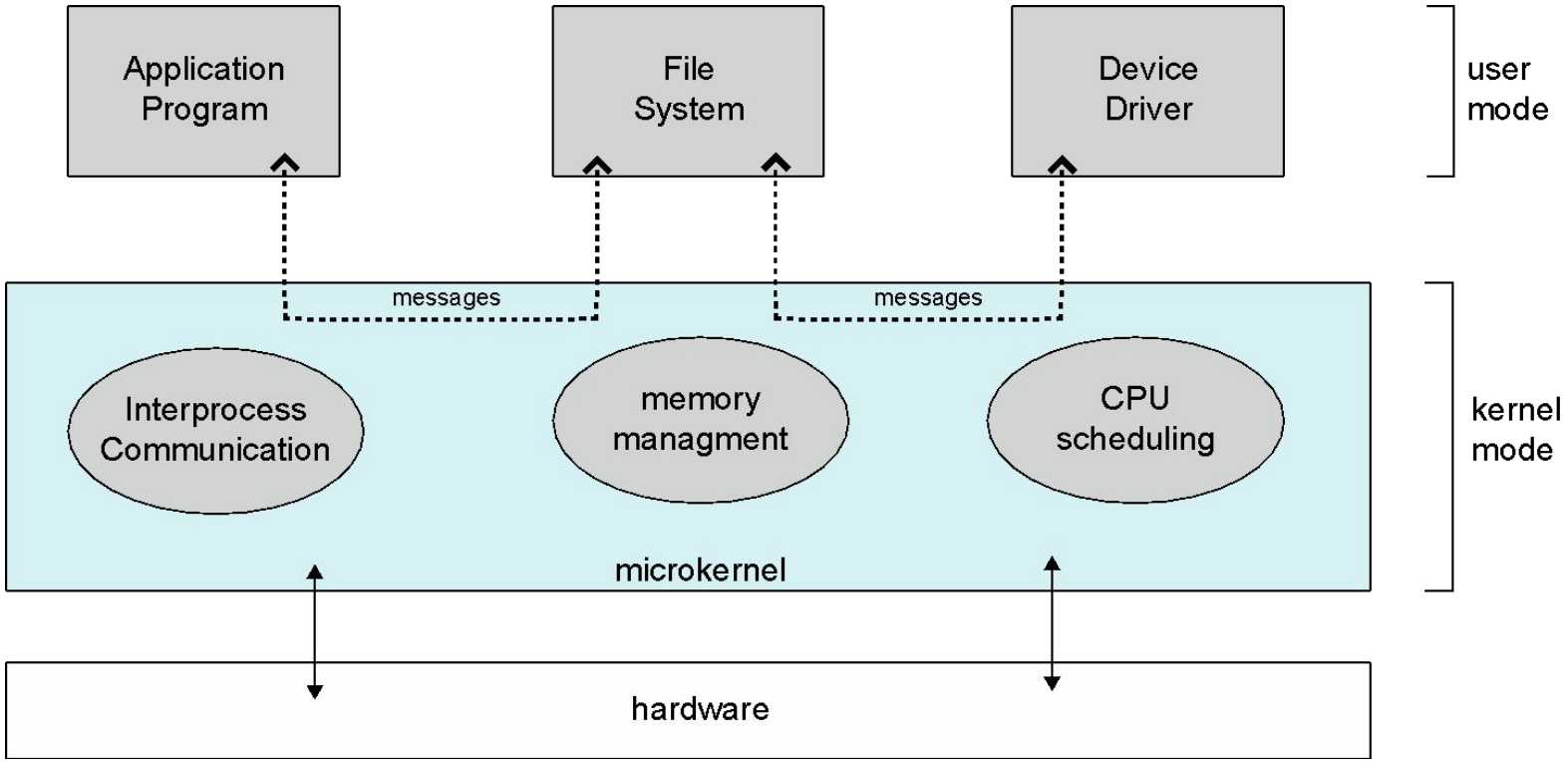
Layered Architecture



Microkernel System Structure

- Microkernel
  - provides only small number of services
  - attempt to keep kernel small and scalable
- Moves as much from the kernel into user space
- Mach is example of microkernel
  - Mac OS X kernel (Darwin) partly based on Mach
- Communication takes place between user modules using message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user space to kernel space communication

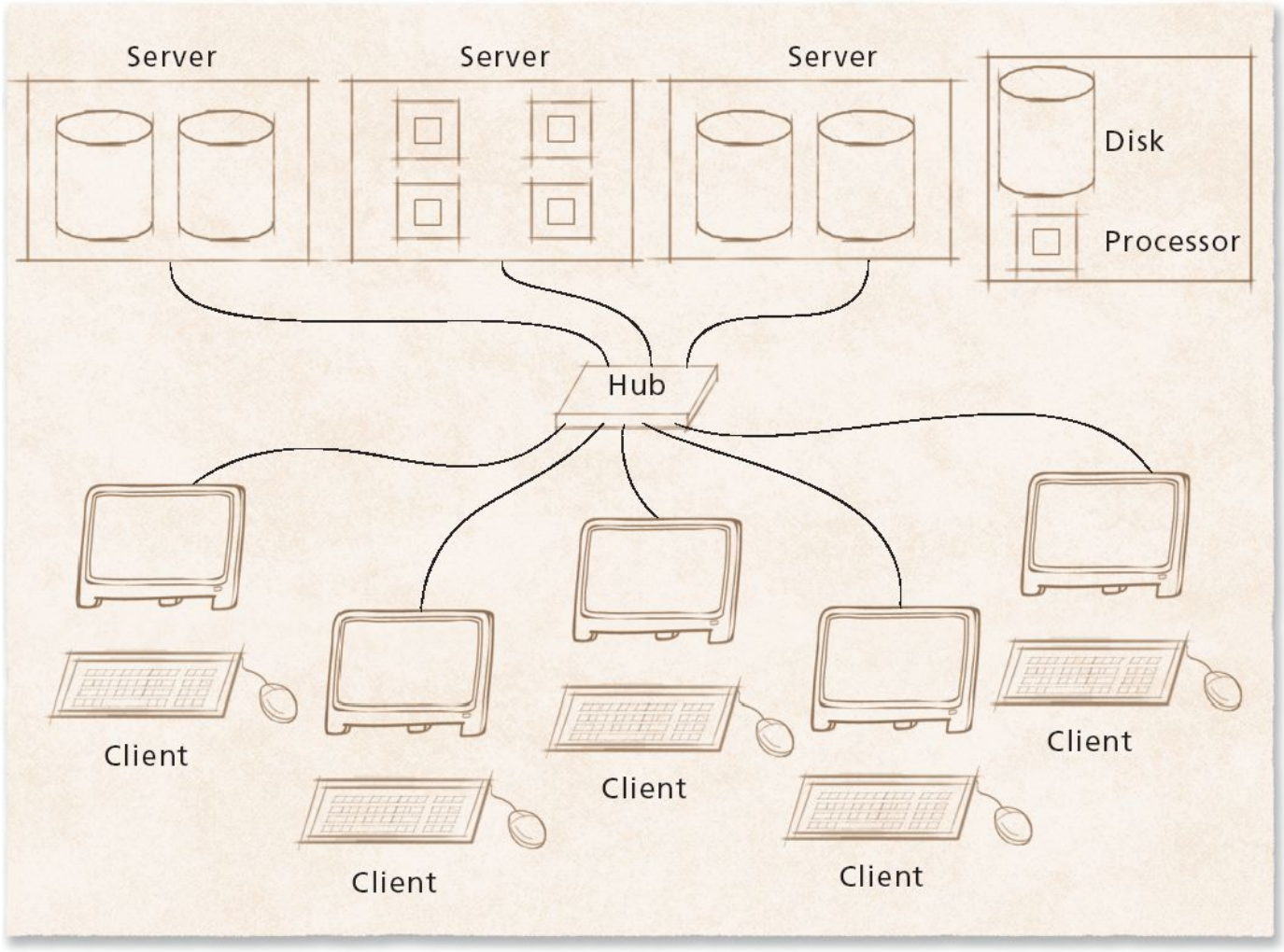
Microkernel System Structure



Distributed Operating Systems

- Network operating system
  - Runs on one computer but allows its processes to access remote resources
- Distributed operating system
  - Single OS manages resources on more than one computer

Distributed Operating Systems



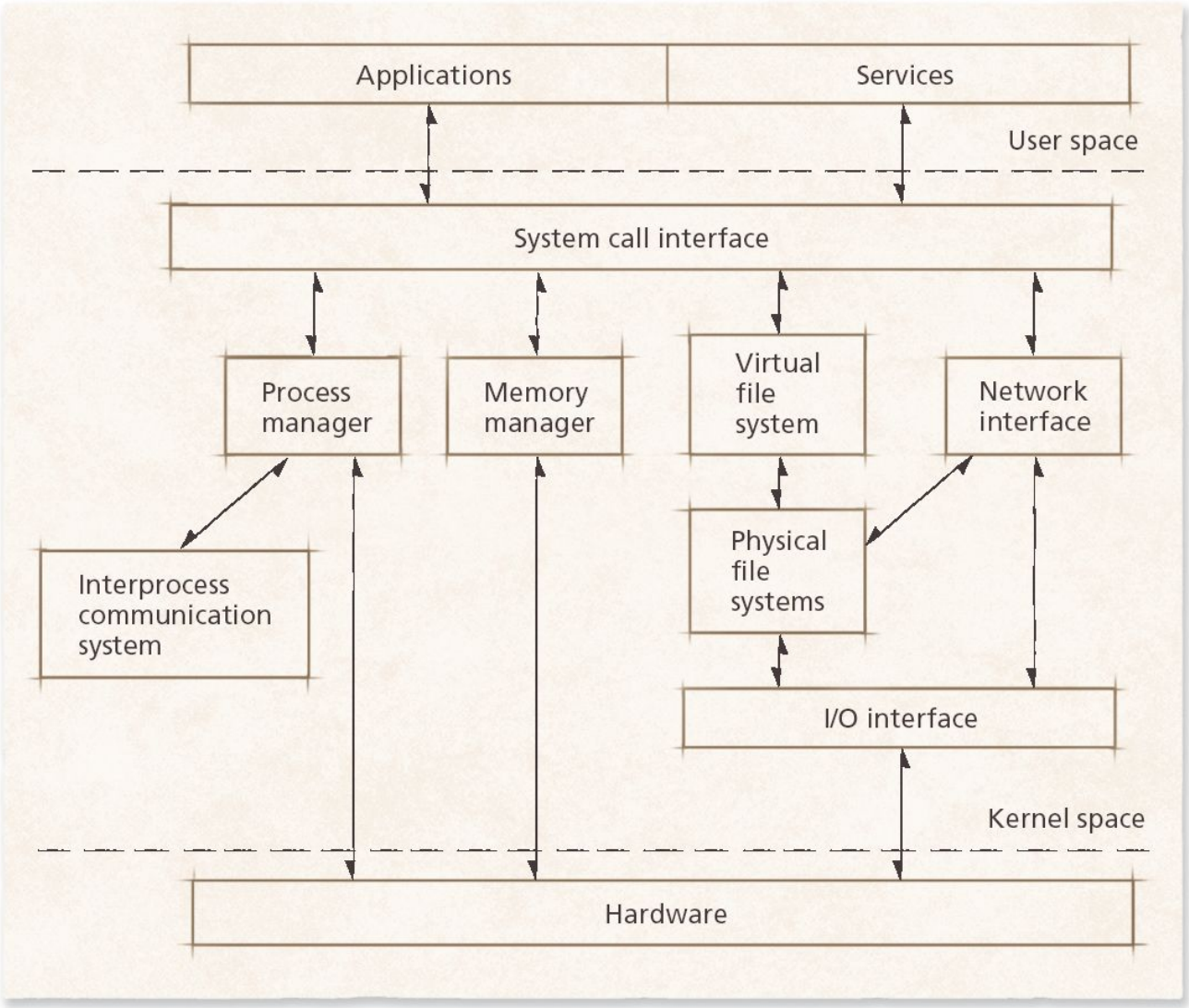
Linux Kernel Architecture

- Monolithic kernel:

  Contains modular components

  - Process management
  - Interprocess communication
  - Memory management
  - File system management
    - VFS: provides a single file system interface
  - I/O management
  - Networking

Linux Kernel Architecture



Characteristics of Modern OS

Characteristics of Modern Operating Systems

- Microkernel architecture
  - assigns only a few essential functions to the kernel
    - address space
    - interprocess communication (IPC)
    - basic scheduling

Characteristics of Modern Operating Systems

- Multithreading
  - process is divided into threads that can run simultaneously
- Thread
  - dispatchable unit of work
  - executes sequentially and is interruptable
- Process is a collection of one or more threads

Characteristics of Modern Operating Systems

- Symmetric multiprocessing
  - there are multiple processors
  - these processors share same main memory and I/O facilities
  - All processors can perform the same functions

Characteristics of Modern Operating Systems

- Distributed operating systems
  - provides the illusion of a single main memory and single secondary memory space
  - used for distributed file system

Characteristics of Modern Operating Systems

- Object-oriented design
  - used for adding modular extensions to a small kernel
  - enables programmers to customize an operating system without disrupting system integrity

Operation Modes

Kernel Mode

User Mode

Kernel Mode of Operation

- Also referred as System Mode.
- In Kernel mode, the executing code has complete and unrestricted access to the underlying hardware.
- It can execute any CPU instruction and reference any memory address.
- Kernel mode is generally reserved for the lowest-level, most trusted functions of the operating system.
- Supported by Hardware Designs
- Crashes in kernel mode are catastrophic.

User Mode of Operation

- The executing code has no ability to directly access hardware or reference memory.
- Code running in user mode must delegate to system APIs to access hardware or memory.
- Due to the protection afforded by this sort of isolation, crashes in user mode are always recoverable.
- Most of the user applications run in this mode.

Extra

Major Achievements of Modern OS

- Processes
- Memory Management
- Information protection and security
- Scheduling and resource management
- System structure

Processes

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources

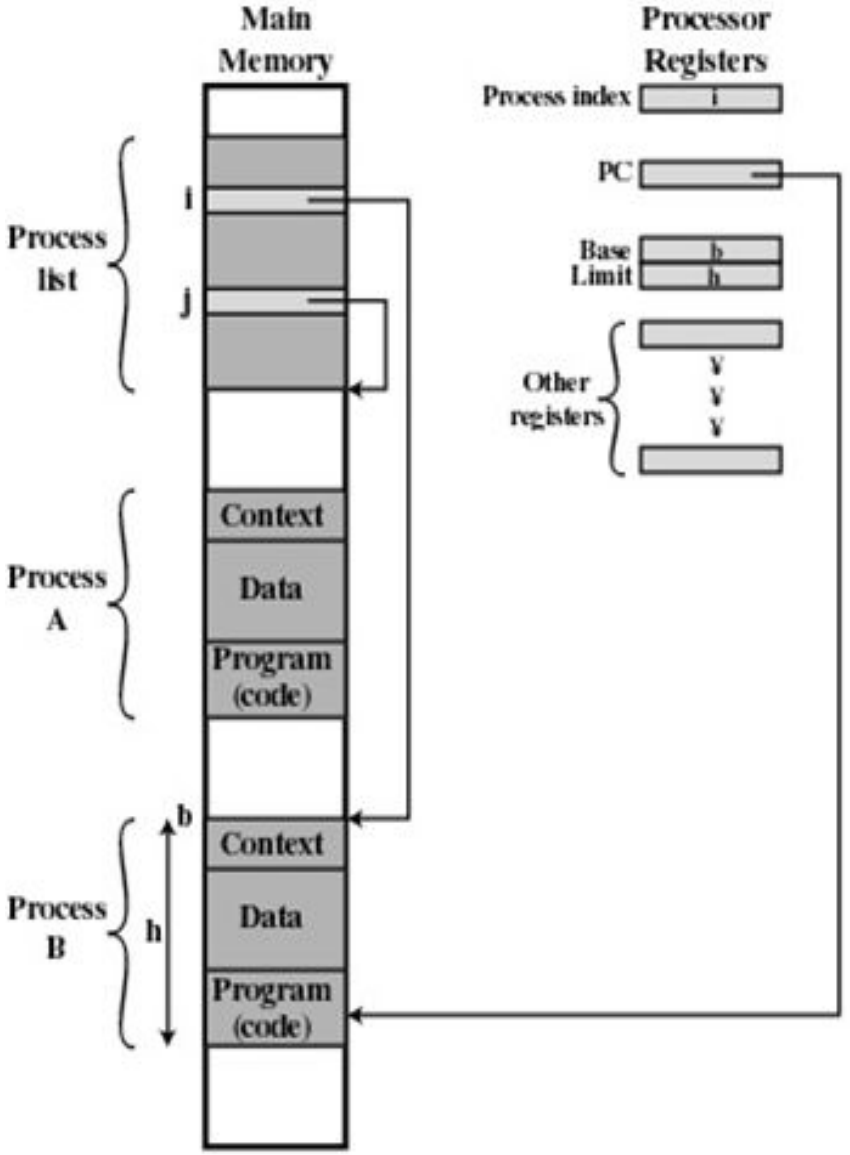
Difficulties with Designing System Software

- Improper synchronization
  - ensure a process waiting for an I/O device receives the signal
- Failed mutual exclusion
- Nondeterminate program operation
  - program should only depend on input to it, not relying on common memory areas
- Deadlocks

Process

- Consists of three components
  - An executable program
  - Associated data needed by the program
  - Execution context of the program
    - All information the operating system needs to manage the process

Process


Main Memory
Processor Registers
Process index
i
PC
Base
Limit
b
h
Other registers
Process list
i
j
Process A
Context
Data
Program (code)
Process B
b
h
Context
Data
Program (code)

Memory Management

- Process isolation
  - Deny interference
- Automatic allocation and management
  - Dynamic and efficient memory allocation
  - Transparent to programmer
- Support for modular programming
- Protection and access control
  - OS must allow access of portions of memory, to various users in various ways.
- Long-term storage
  - Keeping processes in memory even when the system is down.

Virtual Memory

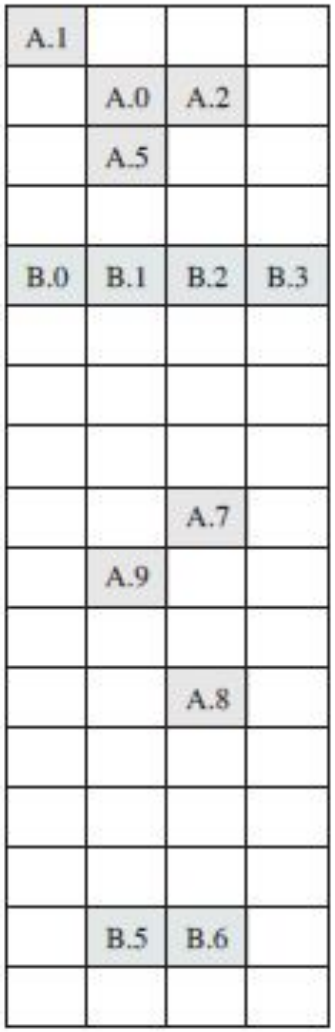
- Allows programmers to address memory from a logical point of view
- While one process is written out to secondary store and the successor process read in there in no hiatus

File System

- Implements long-term store
- Information stored in named objects called files

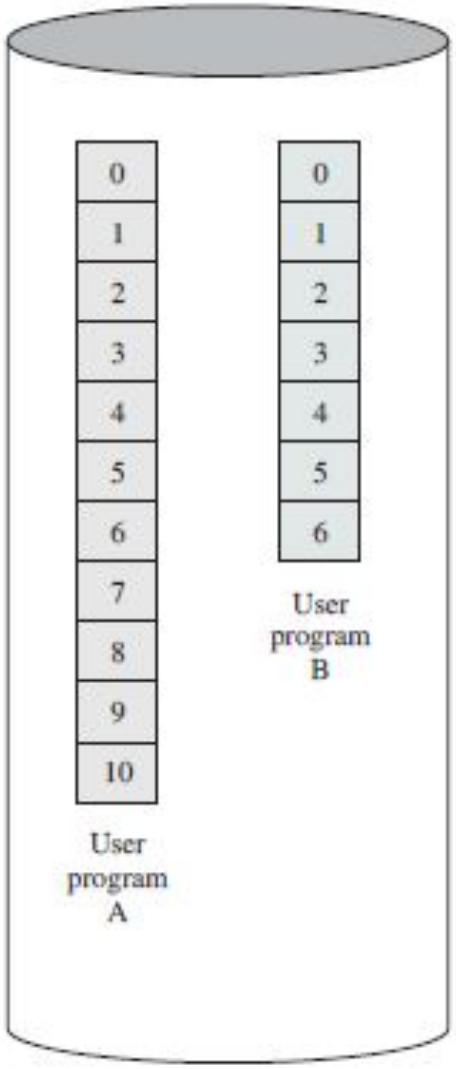
Paging

- Allows process to be comprised of a number of fixed-size blocks, called pages
- Virtual address is a page number and an offset within the page
- Each page may be located any where in main memory
- OS provides mapping between virtual address and real address (or physical address) in main memory



Main memory

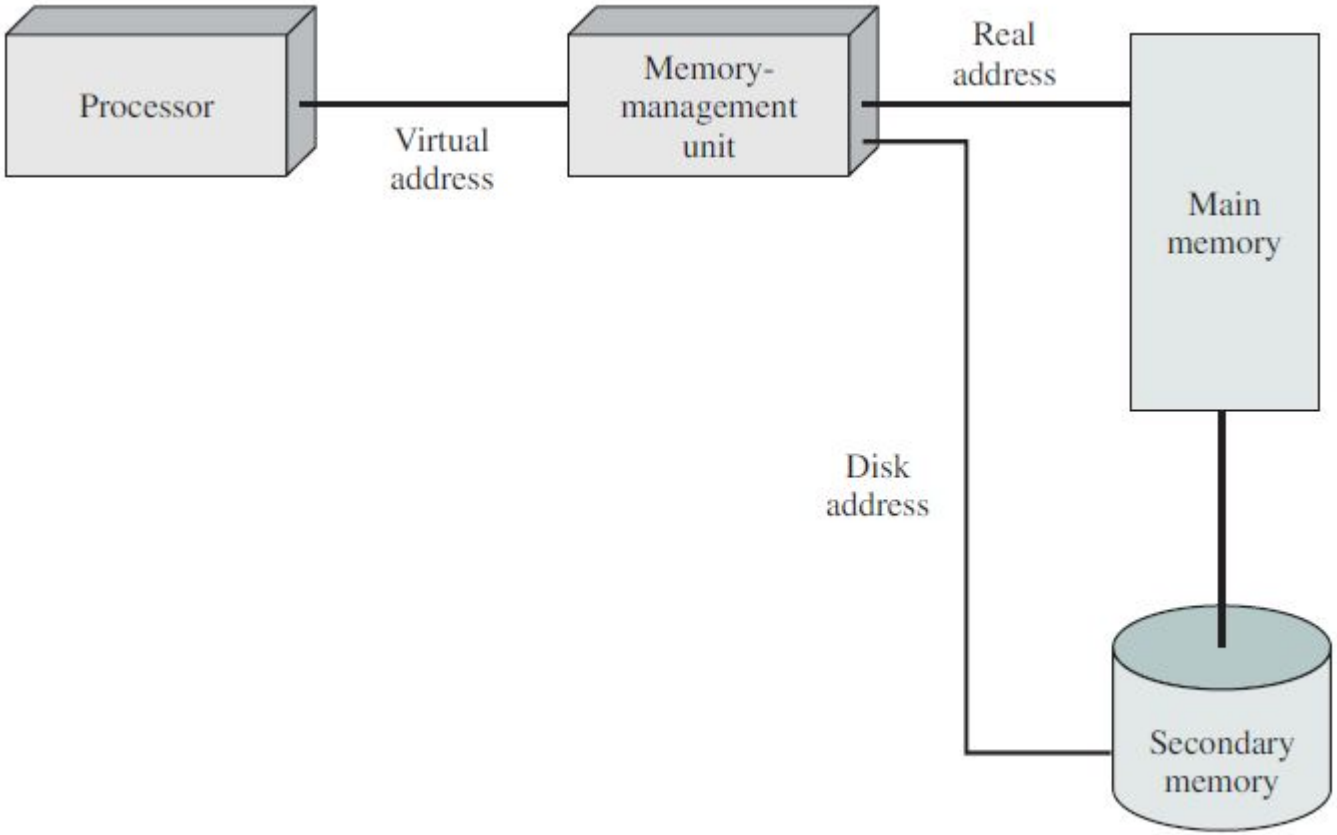
Main memory consists of a number of fixed-length frames, each equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.



Disk

Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages for all programs plus the operating system are on disk, as are files.

Virtual Memory Addressing



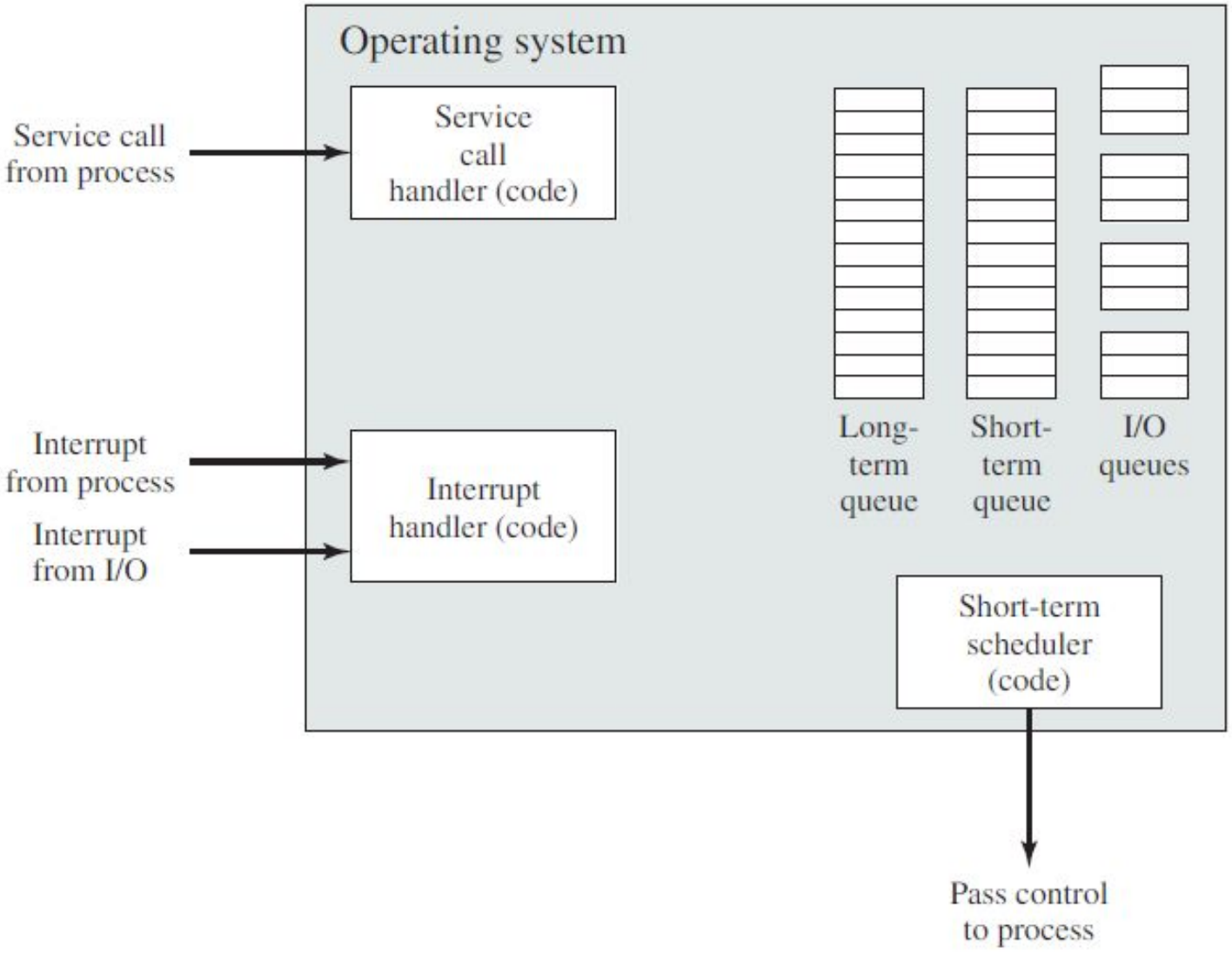
Information Protection and Security

- Access control
  - regulate user access to the system
- Information flow control
  - regulate flow of data within the system and its delivery to users
- Certification
  - proving that access and flow control perform according to specifications

Scheduling and Resource Management

- Fairness
  - give equal and fair access to all processes
- Differential responsiveness
  - discriminate between different classes of jobs
- Efficiency
  - maximize throughput, minimize response time, and accommodate as many uses as possible

Major Elements of Operating System



System Structure

- View the system as a series of levels
- Each level performs a related subset of functions
- Each level relies on the next lower level to perform more primitive functions
- This decomposes a problem into a number of more manageable subproblems

Operating System Design Hierarchy

Level	Name	Objects	Example Operations
13	Shell	User programming environment	Statements in shell language
12	User processes	User processes	Quit, kill, suspend, resume
11	Directories	Directories	Create, destroy, attach, detach, search, list
10	Devices	External devices, such as printer, displays and keyboards	Open, close, read, write
9	File system	Files	Create, destroy, open, close read, write
8	Communications	Pipes	Create, destroy, open. close, read, write

Operating System Design Hierarchy

Level	Name	Objects	Example Operations
7	Virtual Memory	Segments, pages	Read, write, fetch
6	Local secondary storechannels	Blocks of data, device	Read, write, allocate, free
5	Primitive processes	Primitive process, semaphores, ready list	Suspend, resume, wait, signal

Operating System Design Hierarchy

Level	Name	Objects	Example Operations
4	Interrupts	Interrupt-handling programs	Invoke, mask, unmask, retry
3	Procedures	Procedures, call stack, display	Mark stack, call, return
2	Instruction Set	Evaluation stack, micro-program interpreter, scalar and array data	Load, store, add, subtract branch
1	Electronic circuits	Registers, gates, buses, etc.	Clear, transfer, activate, complement

Fumdamental Of Operating System

)?What is an Operating System (2

- An Operating System is a program that acts as an intermediary/interface between a user of a computer and the computer hardware.
- OS goals:
  - Control/execute user/application programs.
  - Make the computer system convenient to use.
  - Ease the solving of user problems.
  - Use the computer hardware in an efficient manner.

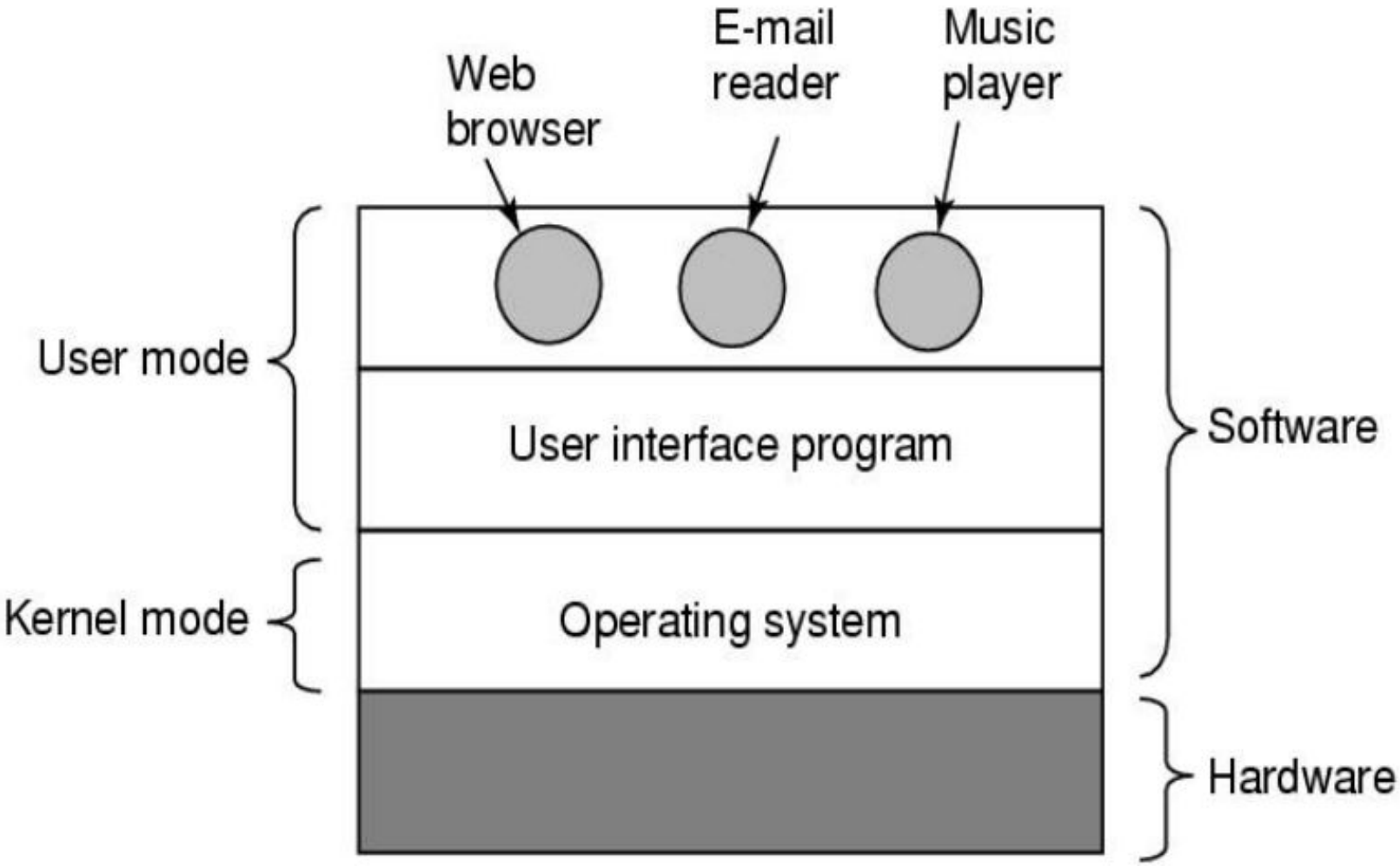
Components of an Operating System

- Operating Systems (OS) are software programs that control thousands of operations, provide an interface between the user and the computer, and run applications.
- An OS is designed to control the operations of programs such as web browsers, word processors, and e-mail programs.

What is OS?

- *Operating System is a software, which makes a computer to actually work.*
- *It is the software the enables all the programs we use.*
- *The OS organizes and controls the hardware.*
- *OS acts as an interface between the application programs and the machine hardware.*
- *Examples: Windows, Linux, Unix and Mac OS, etc.,*

?Where does the OS fit in



1.1 General Definition

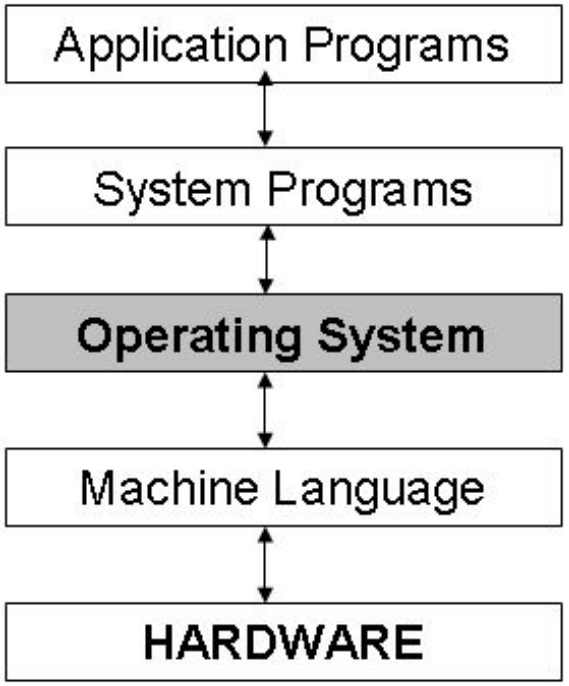
- An OS is a program which acts as an *interface* between computer system users and the computer hardware.
- It provides a user-friendly environment in which a user may easily develop and execute programs.
- Otherwise, hardware knowledge would be mandatory for computer programming.
- So, it can be said that an OS hides the complexity of hardware from uninterested users.

1.1 General Definition

- The OS manages these resources and allocates them to specific programs and users.
- With the management of the OS, a programmer is rid of difficult hardware considerations.
- An OS provides services for
  - Processor Management
  - Memory Management
  - File Management
  - Device Management
  - Concurrency Control

1.1 General Definition

- Another aspect for the usage of OS is that; it is used as a *predefined library* for hardware-software interaction.
- This is why, system programs apply to the installed OS since they cannot reach hardware directly.



Objectives of Operating System

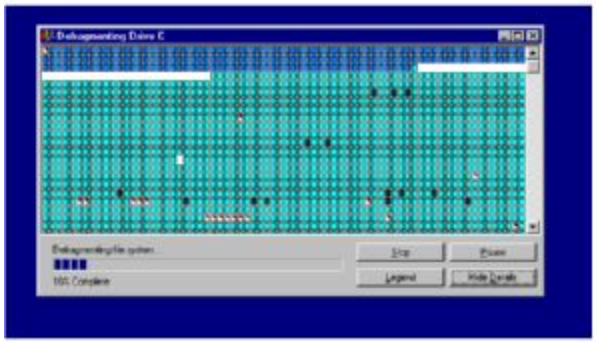
- Convenience
- Efficiency
- Ability to evolve

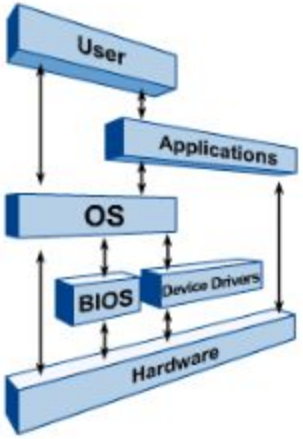
What OS does?

An operating system performs basic tasks such as,

- controlling and allocating memory,
- prioritizing system requests,
- controlling input and output devices,
- facilitating networking and
- managing file systems.

Operating System Functions





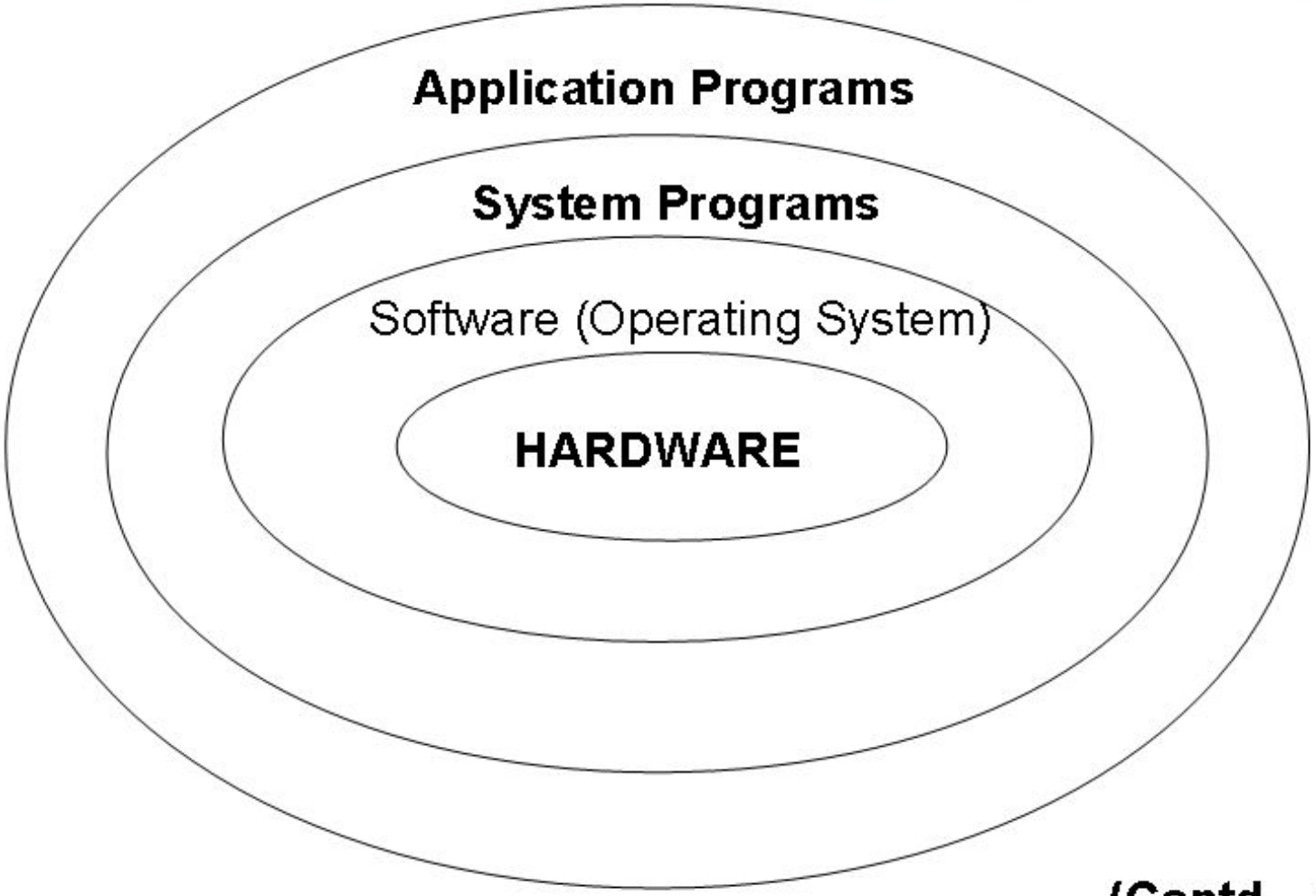
- All operating systems perform the same basic functions:
  - File and folder management
  - Management of applications
  - Support for built-in utility programs
  - Access control to computer hardware (The operating system can either access the hardware through the BIOS or through the device drivers Figure 2)
- Programs written for the UNIX operating system will not work on a Windows-based system, and vice versa.

Operating Systems functions:

- *The main functions of operating systems are:*

1. Program creation
2. Program execution
3. Input/Output operations
4. Error detection
5. Resource allocation
6. Accounting
7. protection

Structure of Operating System:



(Contd…)

Structure of Operating System (Contd…):

- *The structure of OS consists of 4 layers:*
  1. **Hardware**

     Hardware consists of CPU, Main memory, I/O Devices, etc,

  2. **Software (Operating System)**

     Software includes process management routines, memory management routines, I/O control routines, file management routines.

(Contd…)

Structure of Operating System (Contd…):

3. **System programs**

   This layer consists of compilers, Assemblers, linker etc.

4. **Application programs**

   This is dependent on users need. Ex. Railway reservation system, Bank database management etc.,

Services provided by an OS

- Facilities for program creation
  - editors, compilers, linkers, debuggers, etc.
- Program execution
  - loading in memory, I/O and file initialization.
- Access to I/O and files
  - deals with the specifics of I/O and file formats.
- System access
  - resolves conflicts for resource contention.
  - protection in access to resources and data.

Evolution of OS:

- *The evolution of operating systems went through seven major phases.*
- *Six of them significantly changed the ways in which users accessed computers through the open shop, batch processing, multiprogramming, timesharing, personal computing, and distributed systems.*
- *In the seventh phase the foundations of concurrent programming were developed and demonstrated in model operating systems.*

(Contd…)

Evolution of OS (contd..):

Major Phases	Technical Innovations	Operating Systems
Open Shop	The idea of OS	IBM 701 open shop (1954)
Batch Processing	Tape batching, First-in, first-out scheduling.	BKS system (1961)
Multi-programming	Processor multiplexing, Indivisible operations, Demand paging, Input/output spooling, Priority scheduling, Remote job entry	Atlas supervisor (1961), Exec II system (1966)

(Contd…)

Evolution of OS (contd..):

Timesharing	Simultaneous user interaction, On-line file systems	Multics file system (1965), Unix (1974)
Concurrent Programming	Hierarchical systems, Extensible kernels, Parallel programming concepts, Secure parallel languages	RC 4000 system (1969), 13 Venus system (1972), 14 Boss 2 system (1975).
Personal Computing	Graphic user interfaces	OS 6 (1972) Pilot system (1980)
Distributed Systems	Remote servers	WFS file server (1979) Unix United RPC (1982) 24 Amoeba system (1990)

Batch Processing:

- *In Batch processing same type of jobs batch (BATCH- a set of jobs with similar needs) together and execute at a time.*
- *The OS was simple, its major task was to transfer control from one job to the next.*
- *The job was submitted to the computer operator in form of punch cards. At some later time the output appeared.*
- *The OS was always resident in memory. (Ref. Fig. next slide)*
- *Common Input devices were card readers and tape drives.*

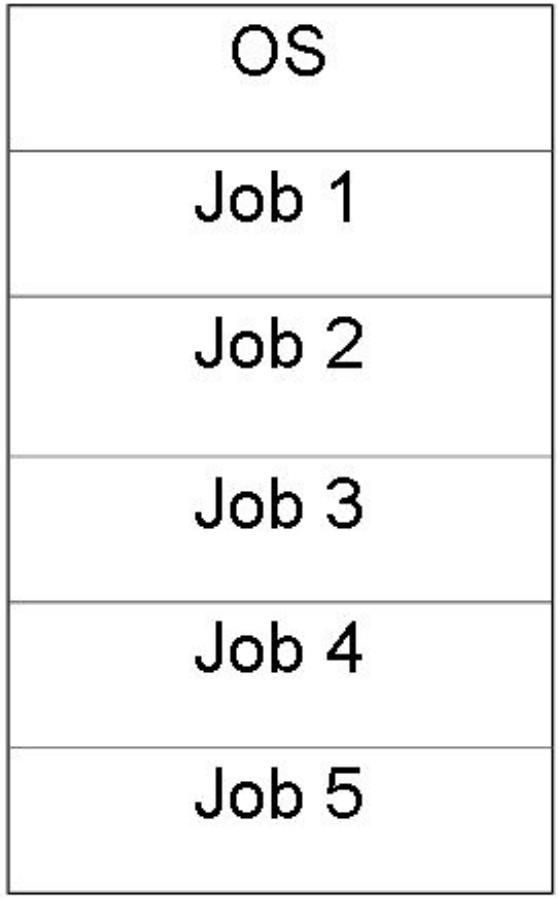
Batch Processing (Contd…):

- *Common output devices were line printers, tape drives, and card punches.*
- *Users did not interact directly with the computer systems, but he prepared a job (comprising of the program, the data, & some control information).*

Multiprogramming:

- *Multiprogramming is a technique to execute number of programs simultaneously by a single processor.*
- *In Multiprogramming, number of processes reside in main memory at a time.*
- *The OS picks and begins to executes one of the jobs in the main memory.*
- *If any I/O wait happened in a process, then CPU switches from that job to another job.*
- *Hence CPU in not idle at any time.*

Multiprogramming (Contd…):



- Figure dipicts the layout of multiprogramming system.
- The main memory consists of 5 jobs at a time, the CPU executes one by one.

Advantages:

- Efficient memory utilization
- Throughput increases
- CPU is never idle, so performance increases.

Time Sharing Systems:

- *Time sharing, or multitasking, is a logical extension of multiprogramming.*
- *Multiple jobs are executed by switching the CPU between them.*
- *In this, the CPU time is shared by different processes, so it is called as "Time sharing Systems".*
- *Time slice is defined by the OS, for sharing CPU time between processes.*
- *Examples: Multics, Unix, etc.,*

Types of OS:

Operating System can also be classified as,-

- ***Single User Systems***
- ***Multi User Systems***

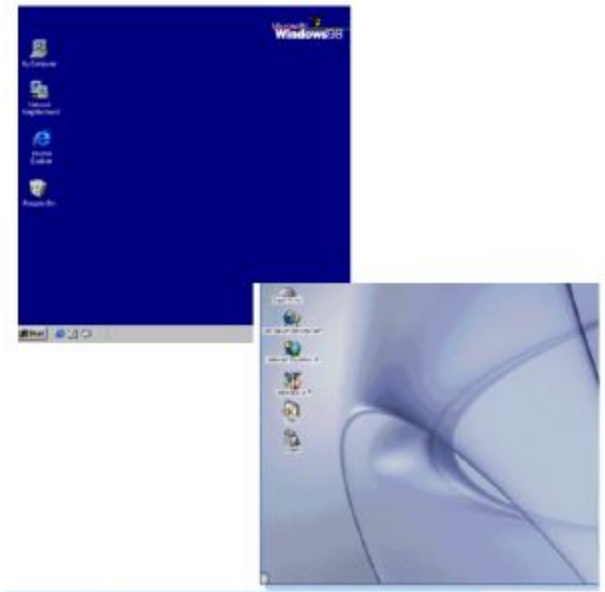
Single User Systems:

- *Provides a platform for only one user at a time.*
- *They are popularly associated with Desk Top operating system which run on standalone systems where no user accounts are required.*
- *Example: DOS*

Multi-User Systems:

- *Provides regulated access for a number of users by maintaining a database of known users.*
- *Refers to computer systems that support two or more simultaneous users.*
- *Another term for multi-user is time sharing.*
- *Ex: All mainframes and are multi-user systems.*
- *Example: Unix*

Operating System Types - Basic Terminology



- The following terms are often used when comparing operating systems:
  - Multi-user
  - Multi-tasking
  - Multi-processing
  - Multi-threading
- A list of some of the most popular operating systems:
  - Microsoft Windows 95, 98, ME
  - Microsoft Windows NT/2000/XP
  - The Macintosh OS
  - UNIX

- Real-time operating system
  - Very fast small OS
  - Built into a device
  - Respond quickly to user input
  - MP3 players, Medical devices

- Single user/Single tasking OS
  - One user works on the system
  - Performs one task at a time
  - MS-DOS and Palm OS
  - Take up little space on disk
  - Run on inexpensive computers

- Single user/Multitasking OS
  - User performs many tasks at once
  - Most common form of OS
  - Windows XP and OS X
  - Require expensive computers
  - Tend to be complex

- Multi user/Multitasking OS
  - Many users connect to one computer
  - Each user has a unique session
  - UNIX, Linux, and VMS
  - Maintenance can be easy
  - Requires a powerful computer

Types Of OS

- Multiuser
- Multiprocessing
- Multitasking
- Multithreading
- Real Time

Different views of OS

- Application view
- Users view
- System view(resource manager)
- Implementation view

Operating system as Resource Manager

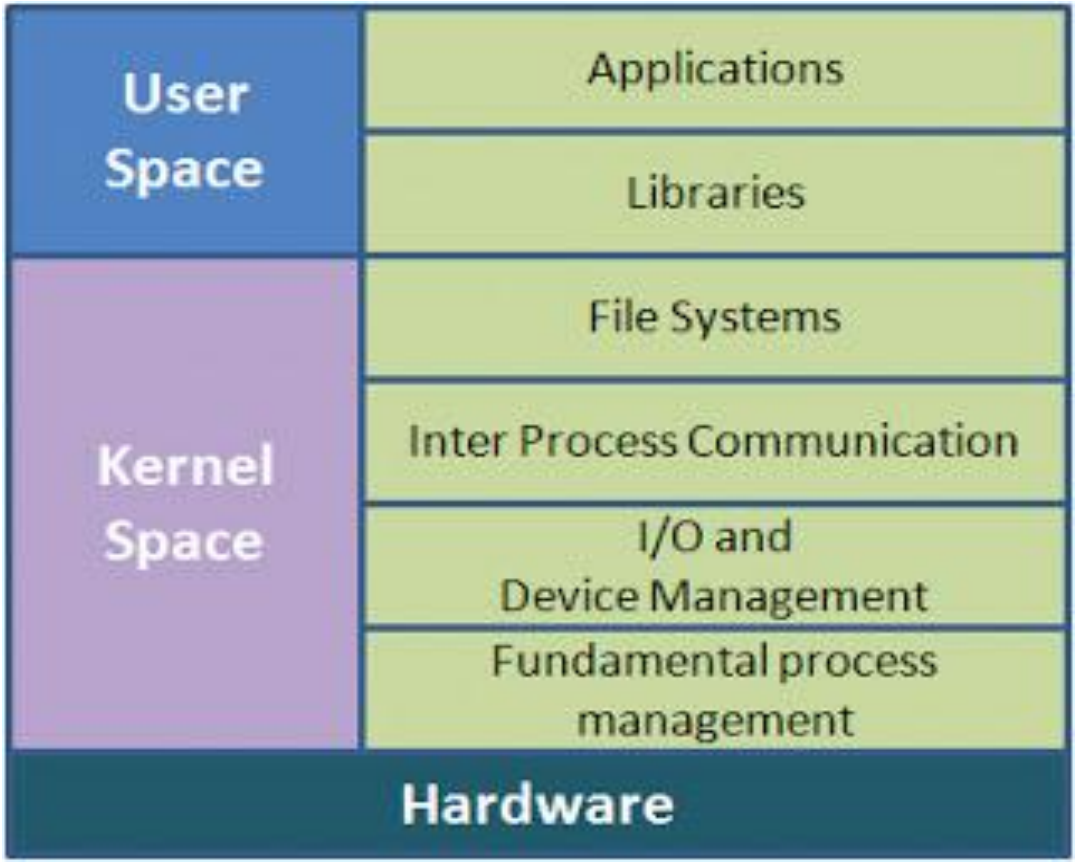
- Resources available in the system : CPU, memory(RAM), file storage space, I/O devices(disk), files, printers
- OS decides which program, for how much time to and after use it reclaims the resource.
- Each program gets time and space with the resource

Multiple users/applications share the resources

- Multiple users/applications share the resources bcoz devices are expensive
- How to allocate them to specific programs (process, jobs)?
- How to protect applications from one another?
- How to provide fair and efficient access to resources?
- How to operate and control the various I/O devices?

Monolithic Kernel/Microkernel

Monolithic kernel



Monolithic kernel

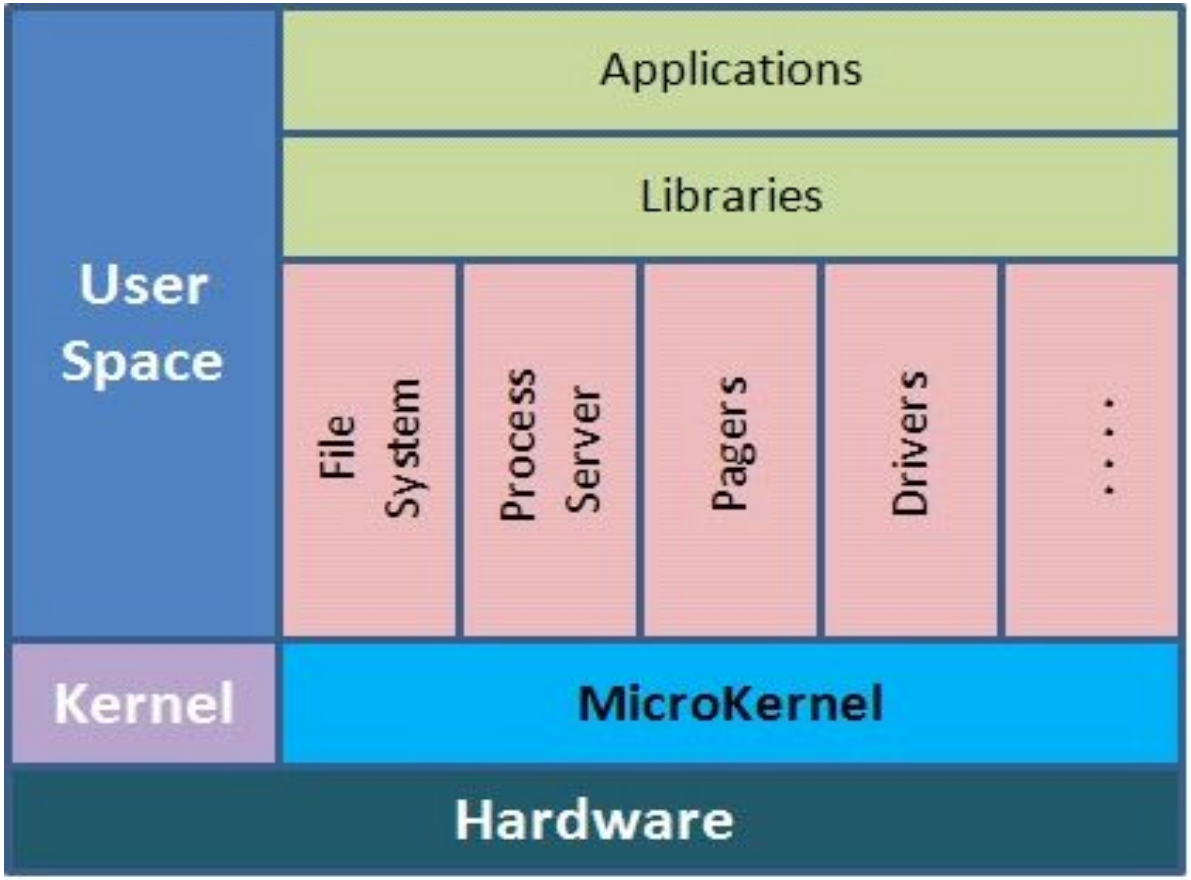
- Older approach(Unix, MS-DOS)
- Runs every basic system service like process and memory management, interrupt handling and I/O communication, file system
- The inclusion of all basic services in kernel space has three big drawbacks.

- The kernel size increase.
- Lack of extensibility.
- The bad maintainability.

Monolithic kernel

- Bug-fixing or the addition of new features means a recompilation of the whole kernel
- Time and resource consuming
- Kernel Image = (Kernel Core+Kernel Services). When system boots up entire services are loaded and resides in memory.

Microkernel



Microkernel

- Reduce the kernel to basic process communication and I/O control
- other system services reside in user space in form of normal processes (as so called servers)
- the servers do not run in kernel space anymore, so called ”con-text switches” are needed, to allow user processes to enter privileged mode (and to exit again)
- Kernel Image = Kernel Core. Services are build in to special modules which can be loaded and unloaded as per need.

Difference between Microkernel/Monolithic kernel

- Monolithic kernel is a single large processes
- It runs entirely in a single address space.
- It is a single static binary file.
- All kernel services exist and execute in kernel address space.
- The kernel can invoke functions directly.
- The examples of monolithic kernel based OSs are Linux, Unix

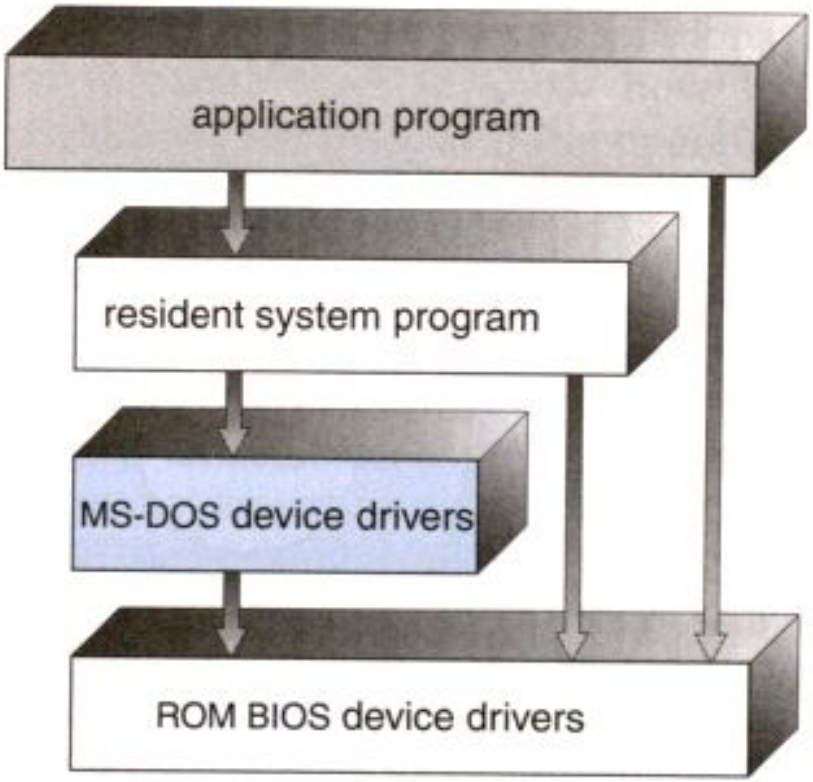
Difference between Microkernel/Monolithic kernel

- In Microkernels, the kernel is broken down into separate processes, known as servers.
- Some of the servers run in kernel space and some run in user-space.
- All servers are kept separate and run in different address spaces.
- The communication in microkernels is done via message passing.
- The servers communicate through IPC
- Servers invoke "services" from each other by sending messages. The separation has advantage that if one server fails other server can still work efficiently. The example of microkernel based OS are Mac OS X and Windows NT

Operating-System Structure

- Simple Structure
- Monolithic Approach
- Layered Approach
- Microkernels

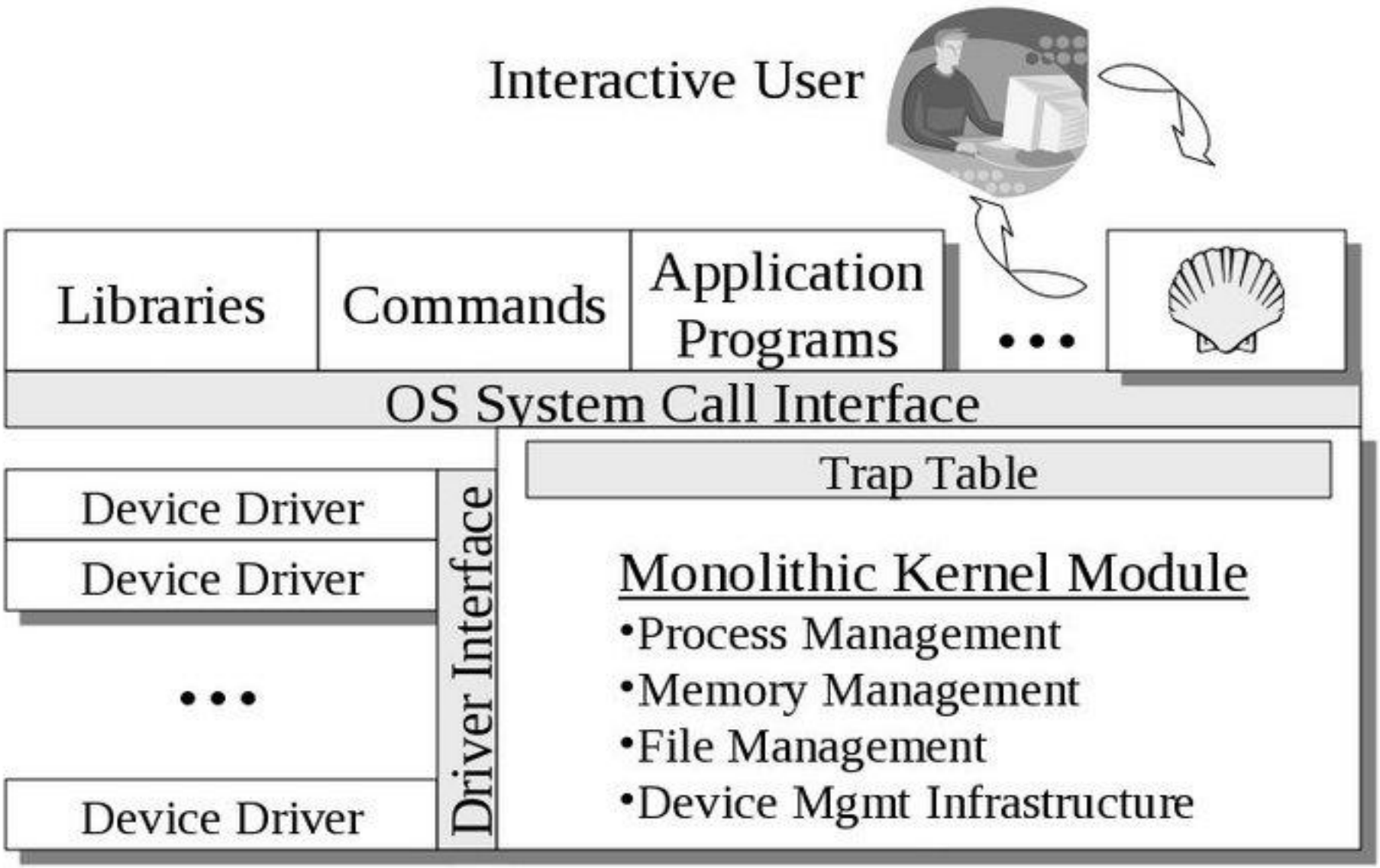
Simple Structure



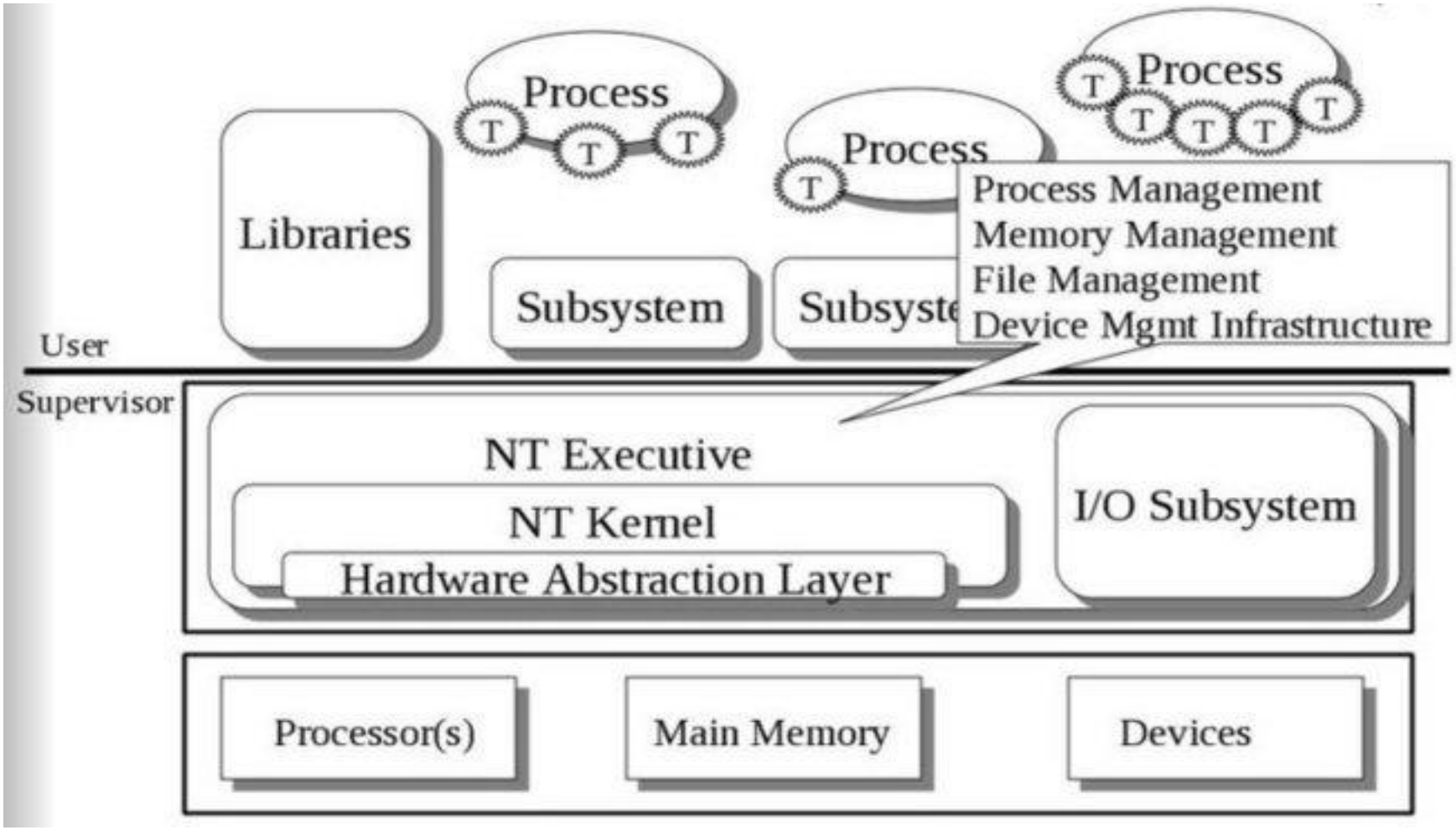
Simple Structure

- Operating systems such as MS-DOS and the original UNIX did not have well-defined structures.
- There is no CPU Execution Mode (user and kernel), and so errors in applications can cause the whole system to crash.

Monolithic Approach



Layered Approach



Microkernels