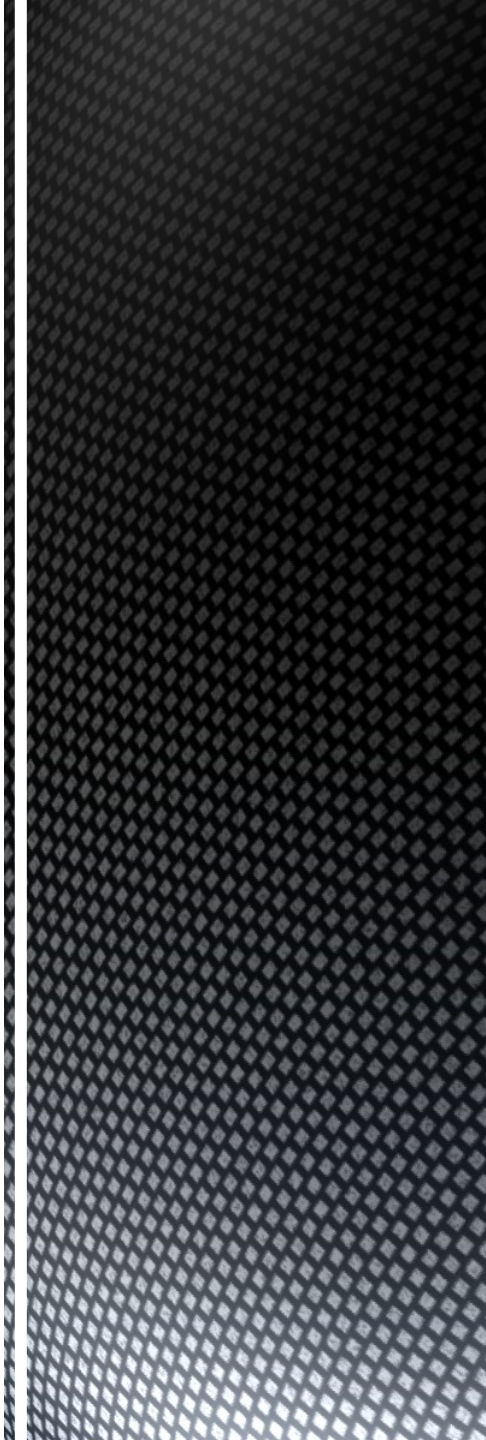
OPERATING SYSTEMS

Chapter 2: Operating-System Structures



- The Slide does not contain all the information and cannot be treated as a study material for Operating System. Please refer the text book for exams.

Chapter 2: Operating-System Structures

- Operating System Services
- User Operating System Interface
- System Calls
- Types of System Calls
- System Programs
- Operating System Structure
- Virtual Machines
- System Boot

Operating System Services

- **User Interface**
  - Command line Interface
  - Batch
  - Graphical Interface
- **Program execution –** load, run and terminate a program
- **I/O Operations –** For efficiency and protection users and not given control to I/O
- **File system manipulation –** read, write, create , delete, search files , permission management

Operating System Services

- **Communications –** Process need to exchange info within the same system or others in network via shared memory or message passing

- **Error Detection –**
  - Errors in memory h/w (memory error, power failure)
  - i/o devices(connection failure)
  - User Program (arithmetic overflow, illegal memory allocation)

- **Resource Allocation –**
  - manage CPU cycles, main memory file storage
  - CPU Scheduling – speed of CPU, jobs, no of registers free

Operating System Services

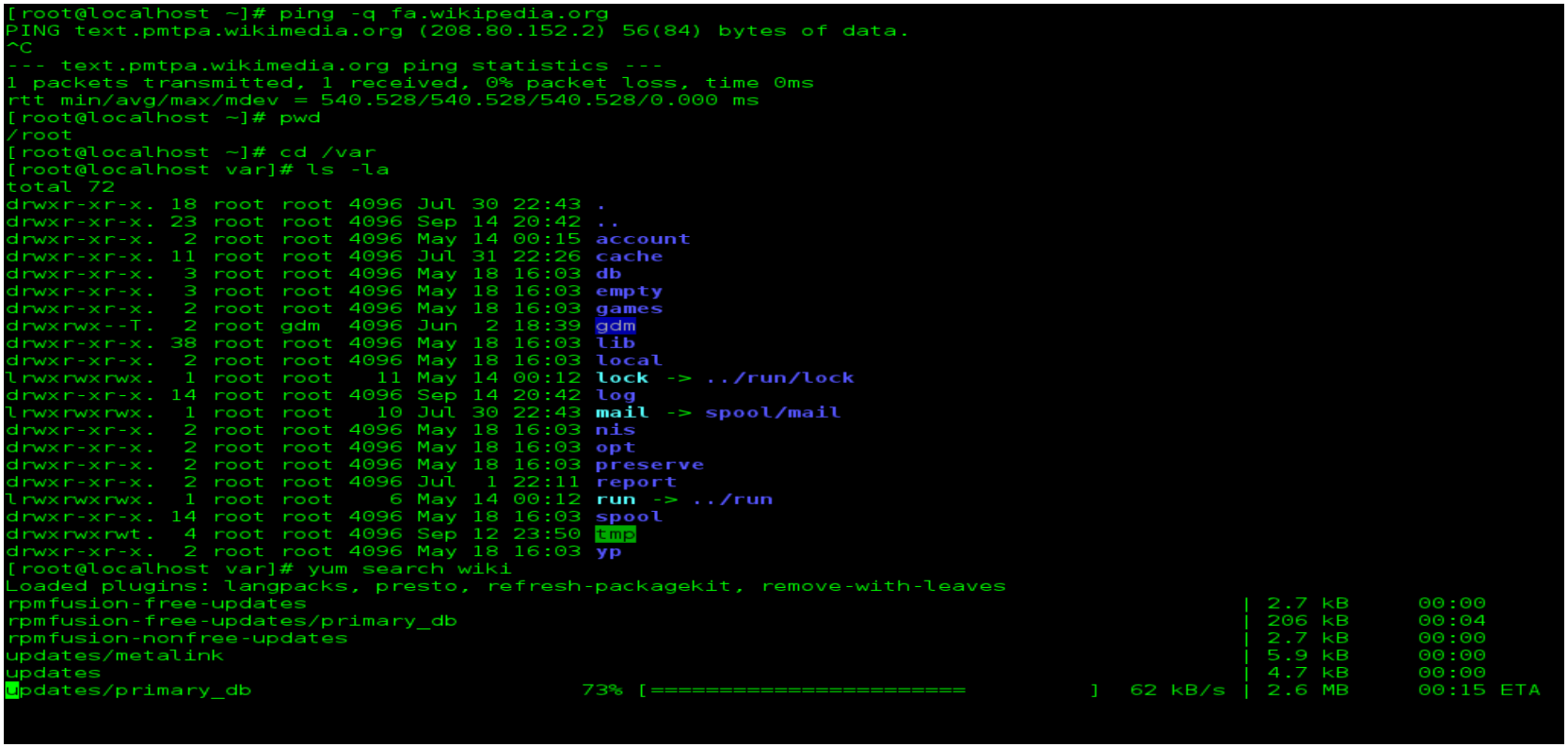
- **Accounting –** Record keeping for billing or collect usage statistics

- **Protections and Security**
  - No interference between two process or OS
  - Security from outsiders

User Operating System Interface

- **Command Line Interface**
- **Graphical User Inteface**

Command Line interface



Command Line interface

- Shells (Bourne, Bash, Korn, C) execute commands
- Eg of commands – ls, rm
- Implemented in two ways
  - Command interpreter contains code
  - System commands(files)

Graphical User Interface

Graphical User Interface

- Employ mouse – based windows and menu
- Desktop, mouse, icons
- Xerox Parc –> Apple Macintosh -> Windows
- Unix – K Desktop Environment(KDE)
  - GNOME Desktop by GNU project
- Users choice – CLI or GUI

System Calls

- System calls provide an interface to the services made available by an OS
- Mostly accessed by programs via a high-level **Application Program Interface (API)** rather than direct system call use
- Three most common APIs
- Win32 API for Windows
- POSIX API for all versions of UNIX, Linux, Mac OS X
- Java API for the Java virtual machine (JVM)

System Calls

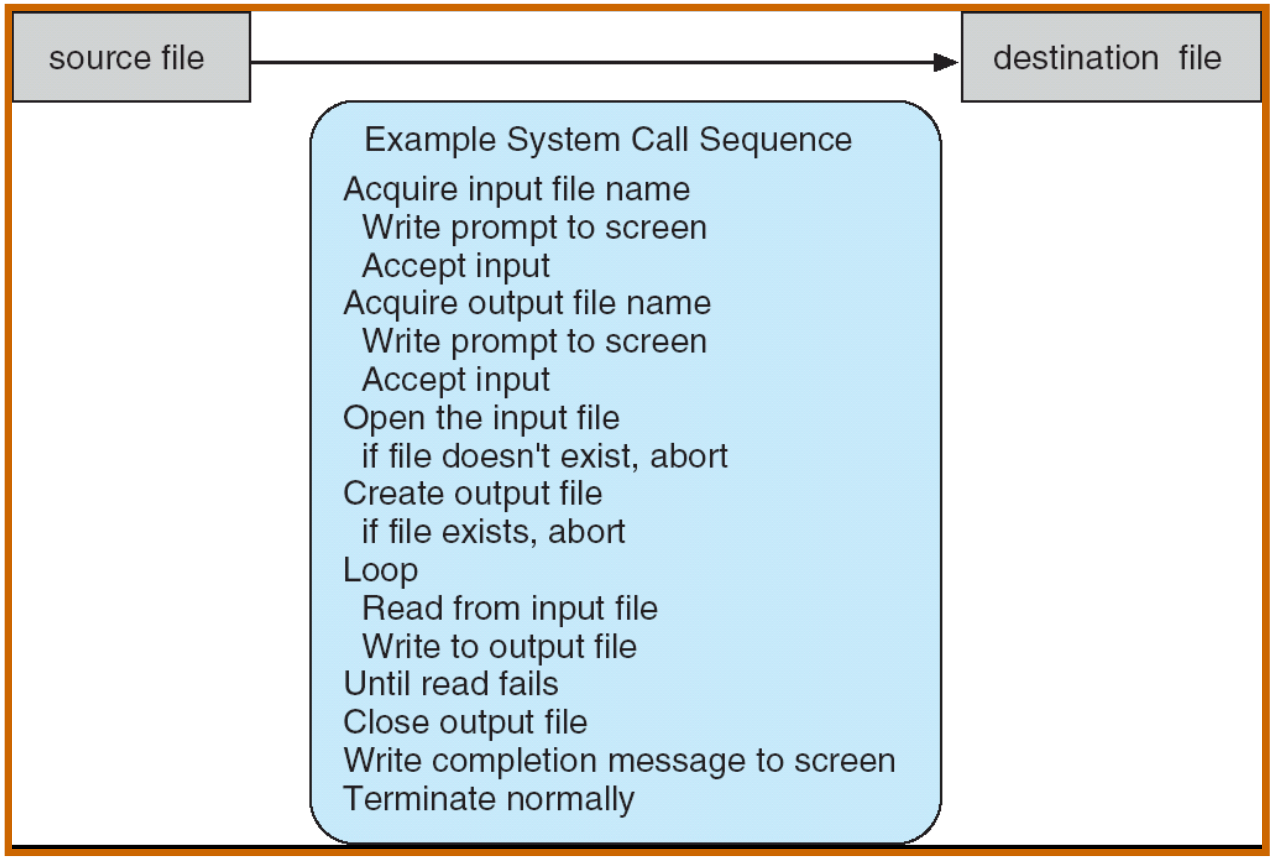
- Why use APIs rather than system calls?

- Expect program to compile and run on any system that supports the same API
- Actual system calls can be more detailed and difficult to work with

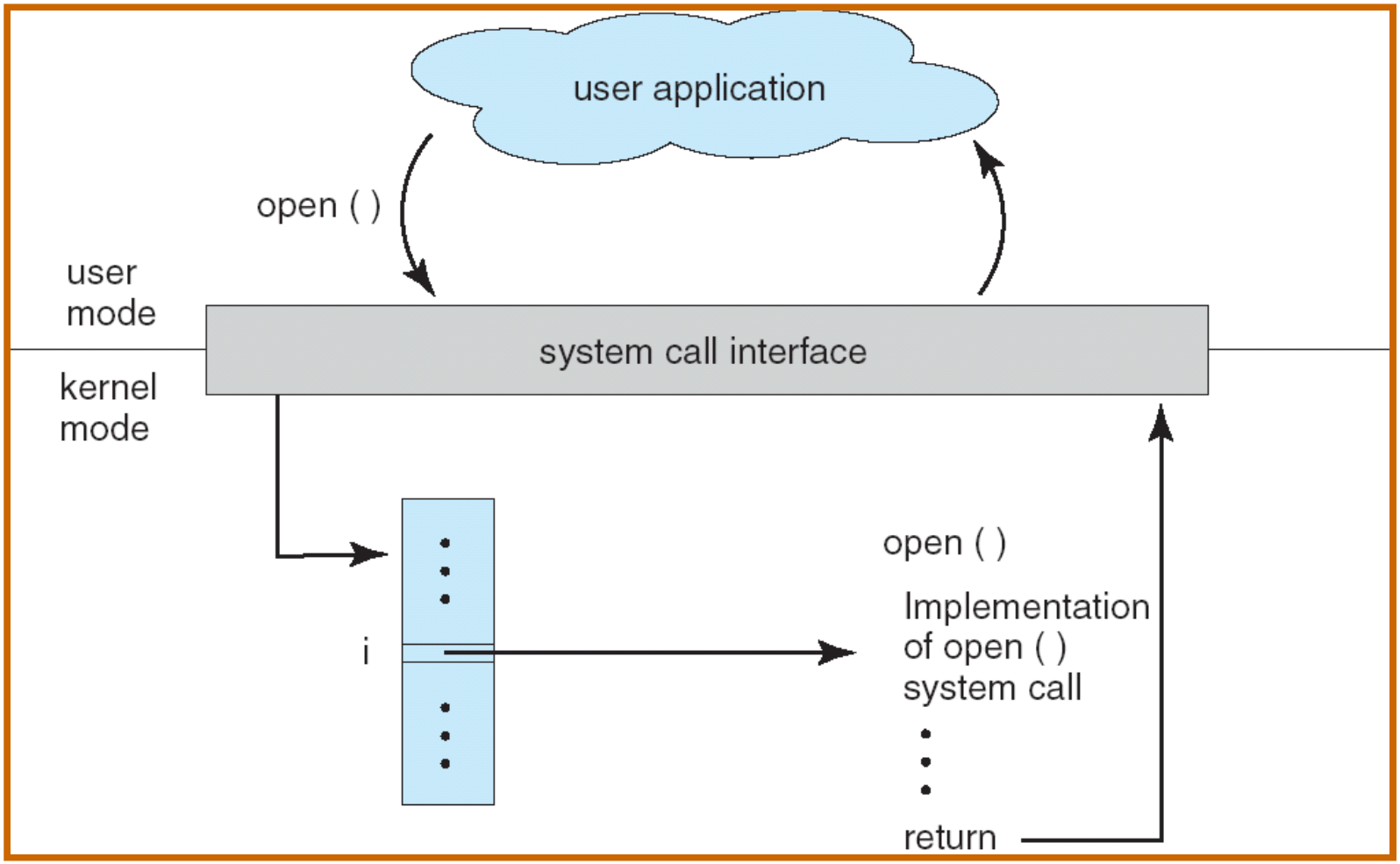
- **System call implementation**
- System call interface maintains a table indexed with number associated with each system call

System Calls

Sequence of System calls to copy data from one file to

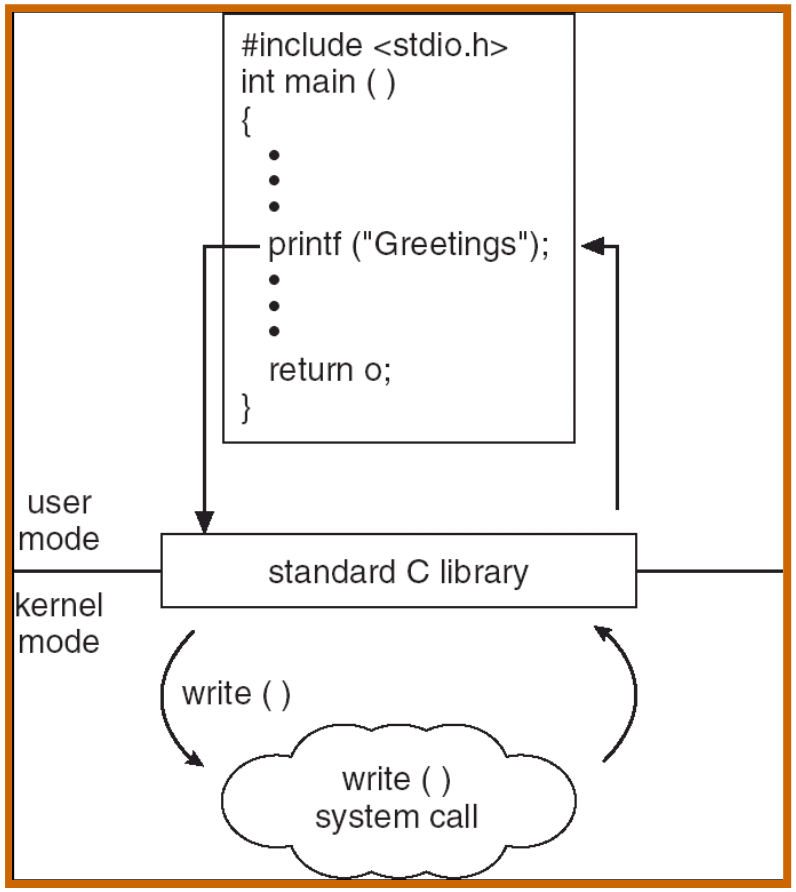


API – System Call – OS Relationship



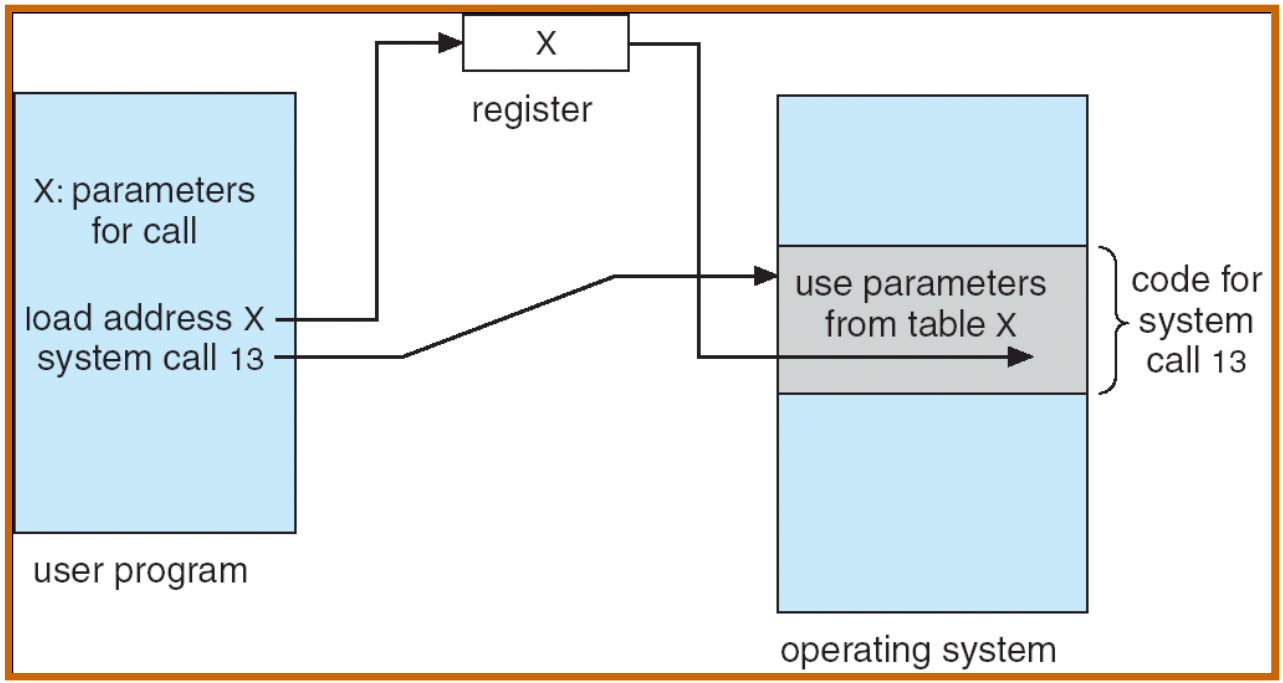
Standard C Library Example

C program invoking printf() library call, which calls write() system call



System call Parameter passing

- Three ways of passing parameters to OS
- Registers, parameters in block, push to stack



Types of System calls

- Process control
- File management
- Device management
- Information maintenance
- Communications

Process control

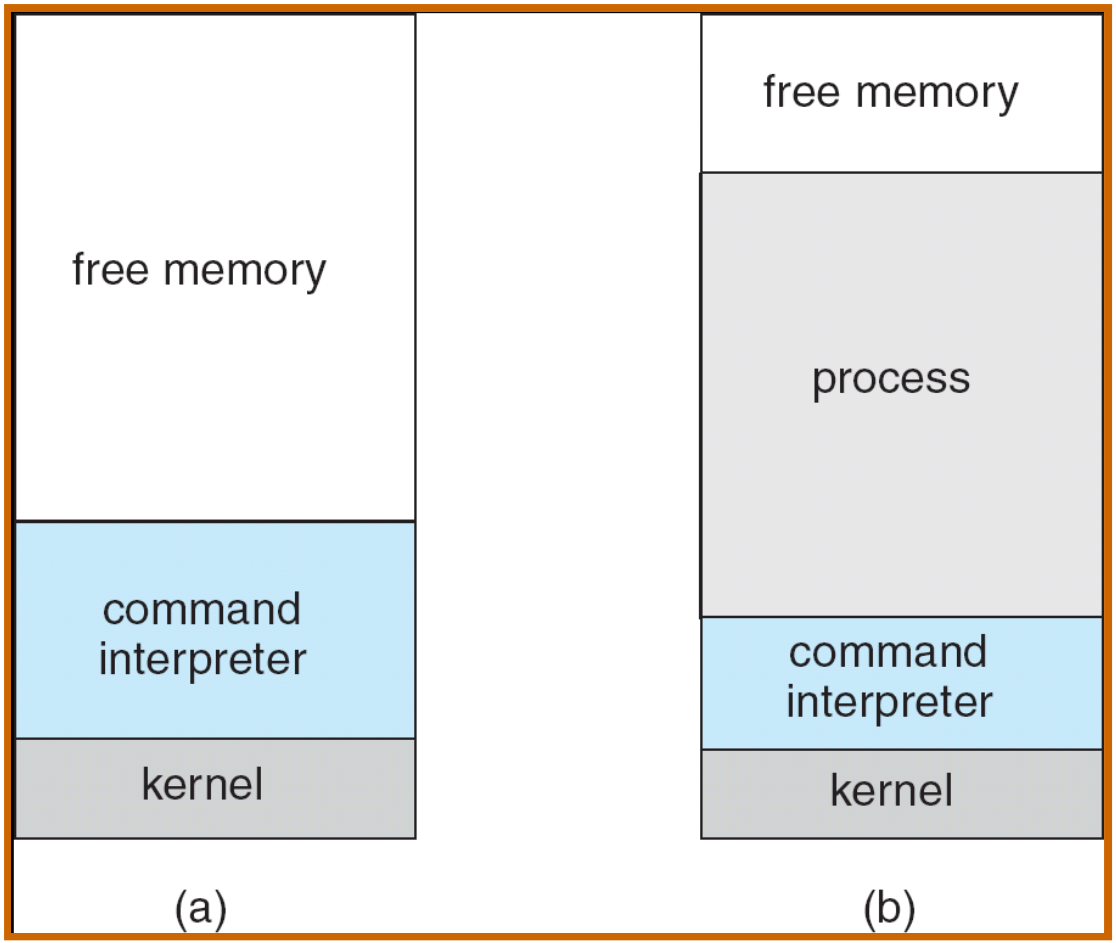
- A running program halts its execution normally or abnormally (abort) in CLI.
- The error could be dumped to memory
- In GUI- Pop up window alert the user to the error and ask for guidance
- **Control card** is a batch system concept
- Error level code – higher the level more severe the error

Types of System calls

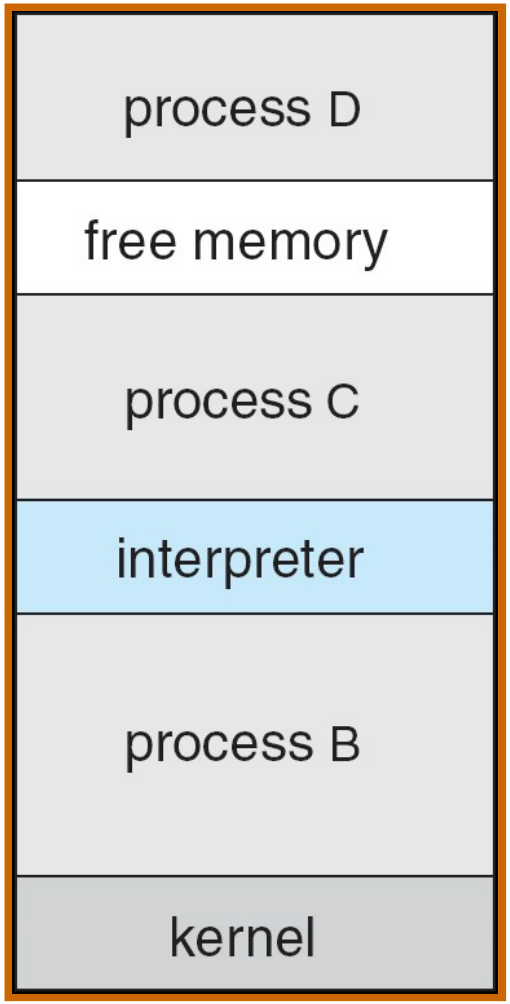
- **Process control**
  - End, abort
  - Load, execute
  - Create process, terminate process
  - Get process attributes, set process attributes
  - Wait for time
  - Wait event, signal event
  - Allocate and free memory

Process control

MS DOS Execution



Free BSD running multiple programs



Types of System Call

- **File Management**
  - Create file, delete file
  - open,close
  - Read, write, reposition
  - Get file attributes, set file attributes

Types of System Call

- **Device management**
  - Request device, release device
  - Read, write, reposition
  - Get device attributes, set device attributes
  - Logically attach or detach devices

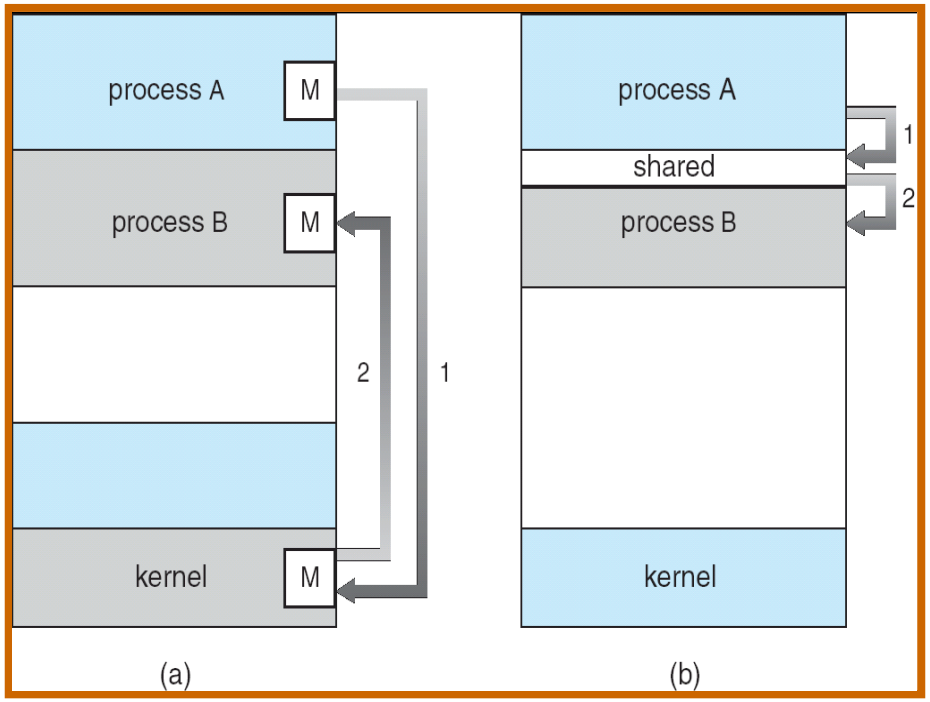
- Physical device – Disk Drives
- Logical Device – Files

Types of System Call

- **Information maintenance**
- Get general data – system time, date
- Trace every system call
- Time profile of a process
  - Get time or date, set time or date
  - Get system data, set system data
  - Get process, file, or device attributes
  - set process, file, or device attributes

Types of System Call

- **Communication**
- **Message Passing Model –**
- Mail Box – connection is opened – pass small messages – within other systems



Types of System Call

- **Communication**
- **Shared Memory Model**
- Process share common data through shared memory.
- The data is maintained by process – no OS control

- **System calls to**
  - Create, delete communication connection
  - Send, receive message
  - Transfer status information
  - Attach or detach remote devices

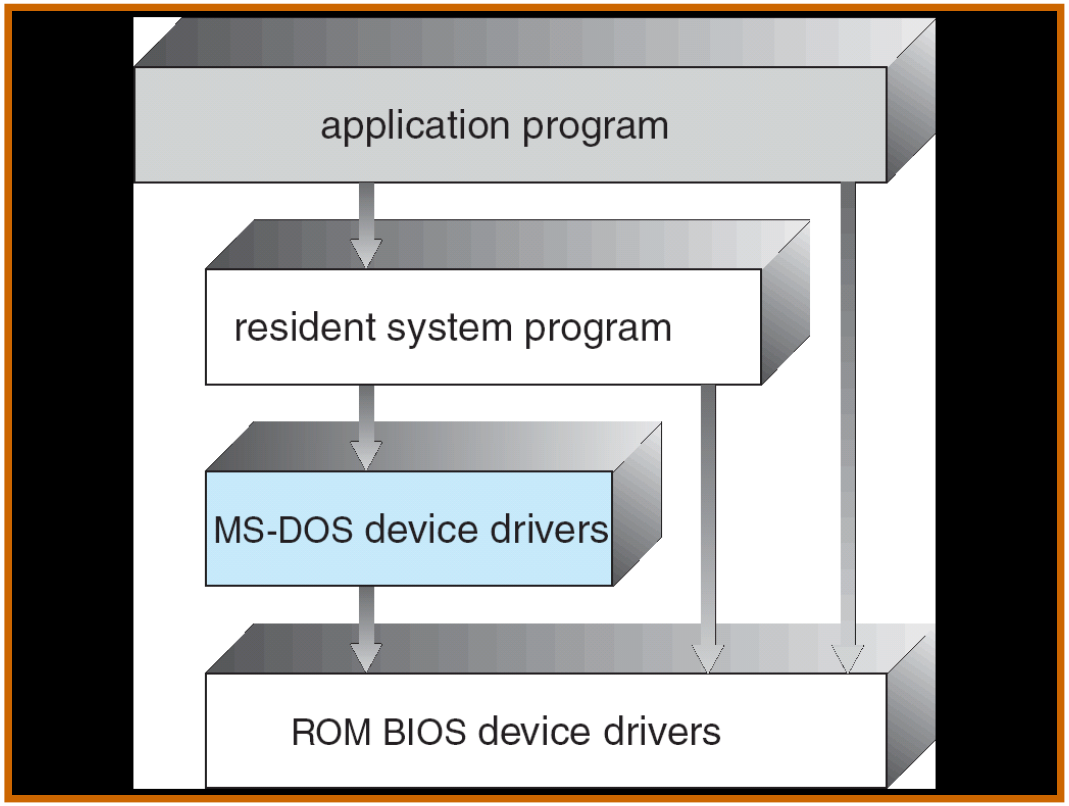
System Programs

- File manipulation
- Status information
- File modification
- Programming language support
- Program loading and execution
- Communications
- Application programs

OS Structure

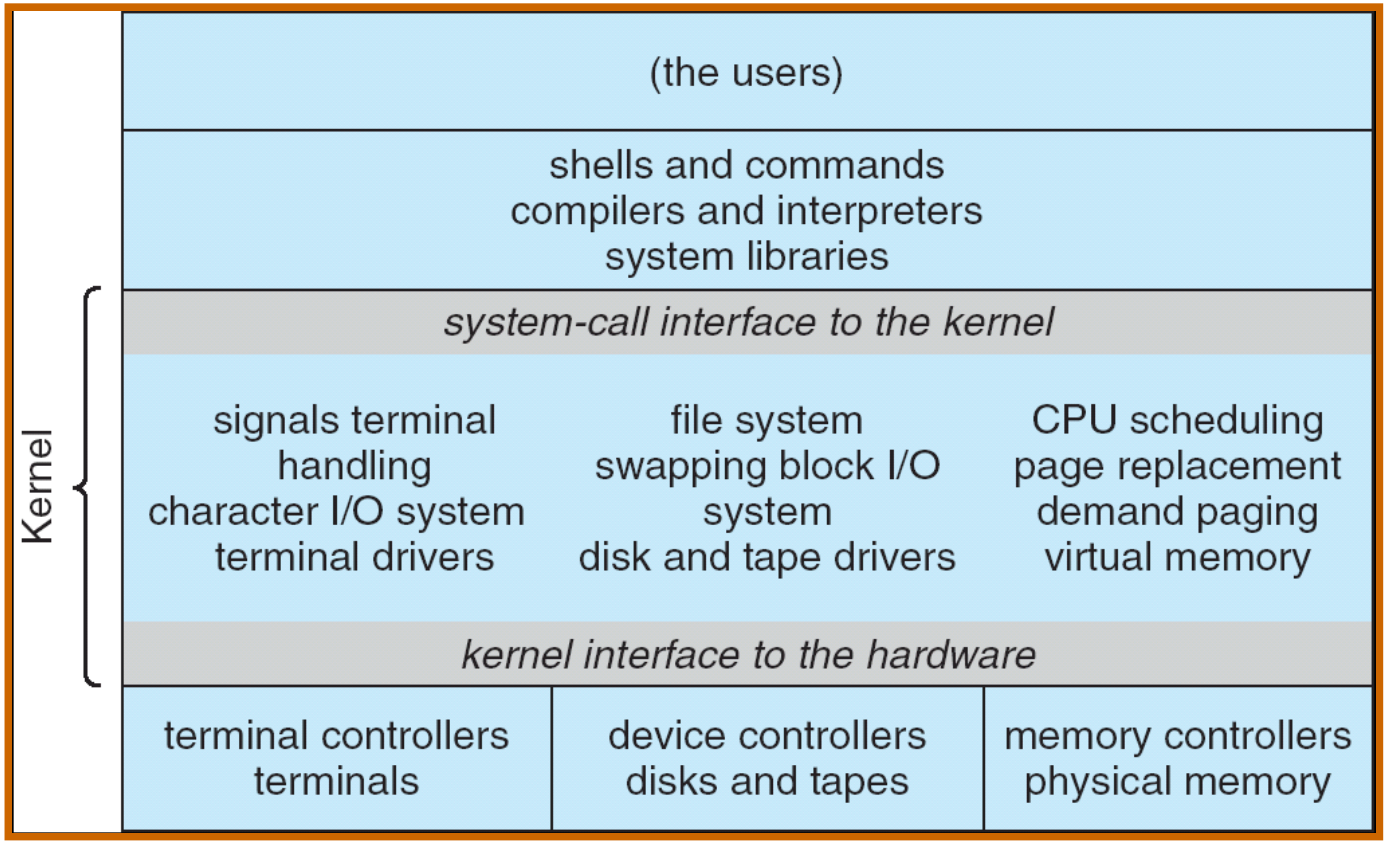
- Four types of structure
  - Simple structure
  - Layered approach
  - Microkernels
  - Modules

MS DOS Layered Style(Simple Structure)

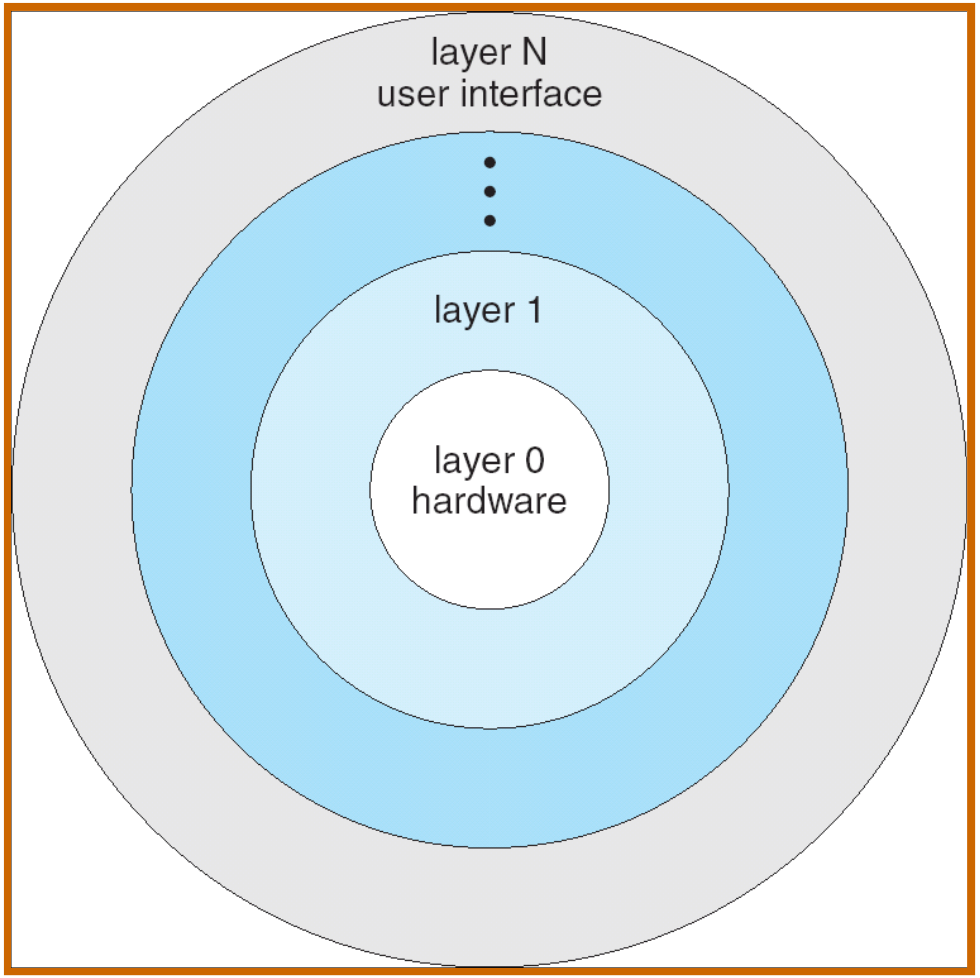


- Start as small, simple limited system and then grow
- Limited by H/W
- Application access the basic I/O routines and write

Unix System Structure(Simple Structure)



Layered Operating System



- Adv – simplicity of construction and debugging
- Dis adv - Difficult to define each layer
- Less efficient – overhead of system calls

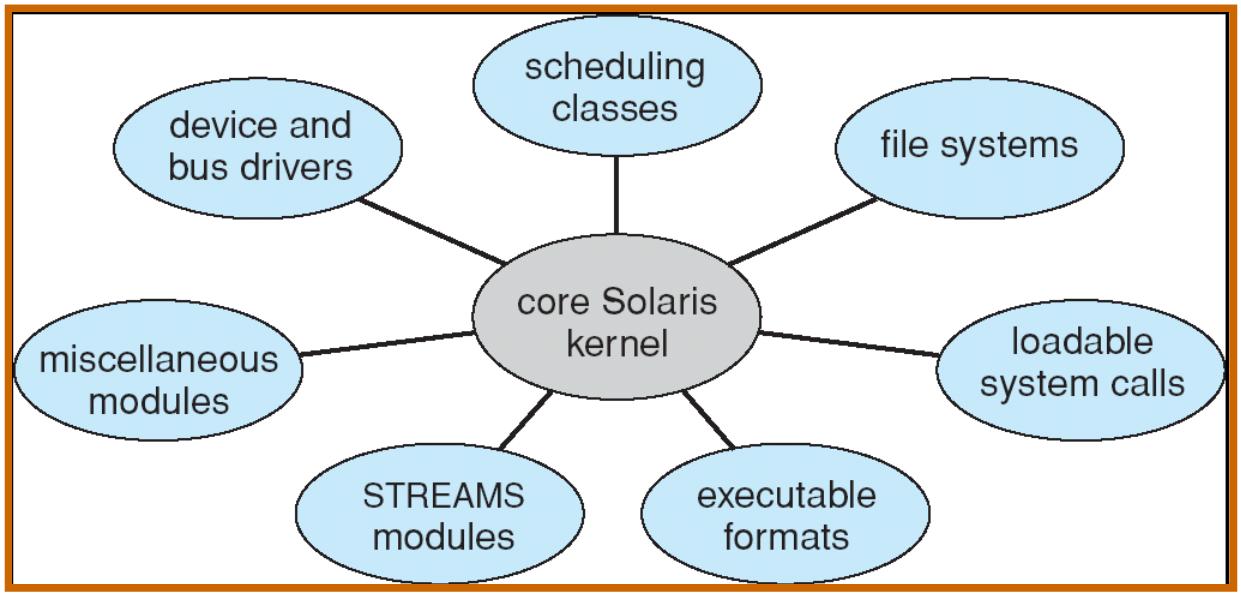
Microkernels

- Move functionality from kernel to user space
- Pass message between process using message passing
- **Benefit**
- Ease of extending OS
- More Security and reliability

- **Disadvantage**
- Poor performance due to system overhead

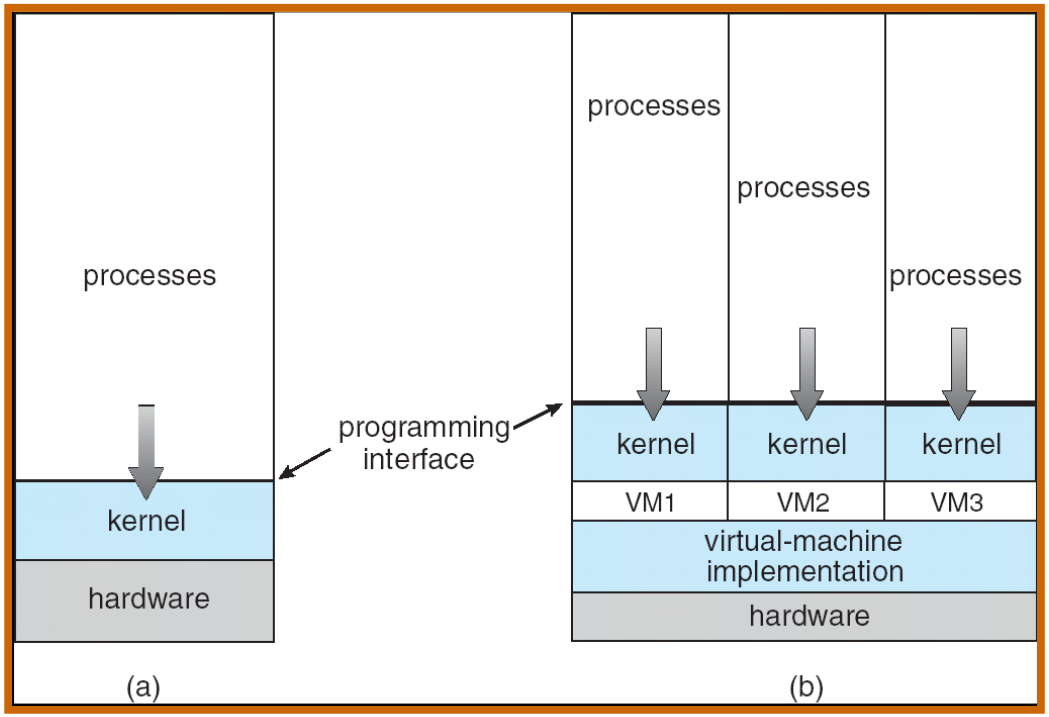
Modules

- Object oriented approach
- Talk to each module through interface that dynamically link
- Any module can call any other module
- SOLARIS MODULAR APPROCH



Virtual Machines

- Abstract H/W of single computer - CPU, Memory, Disk Drive, N/W interface card



Virtual Machines

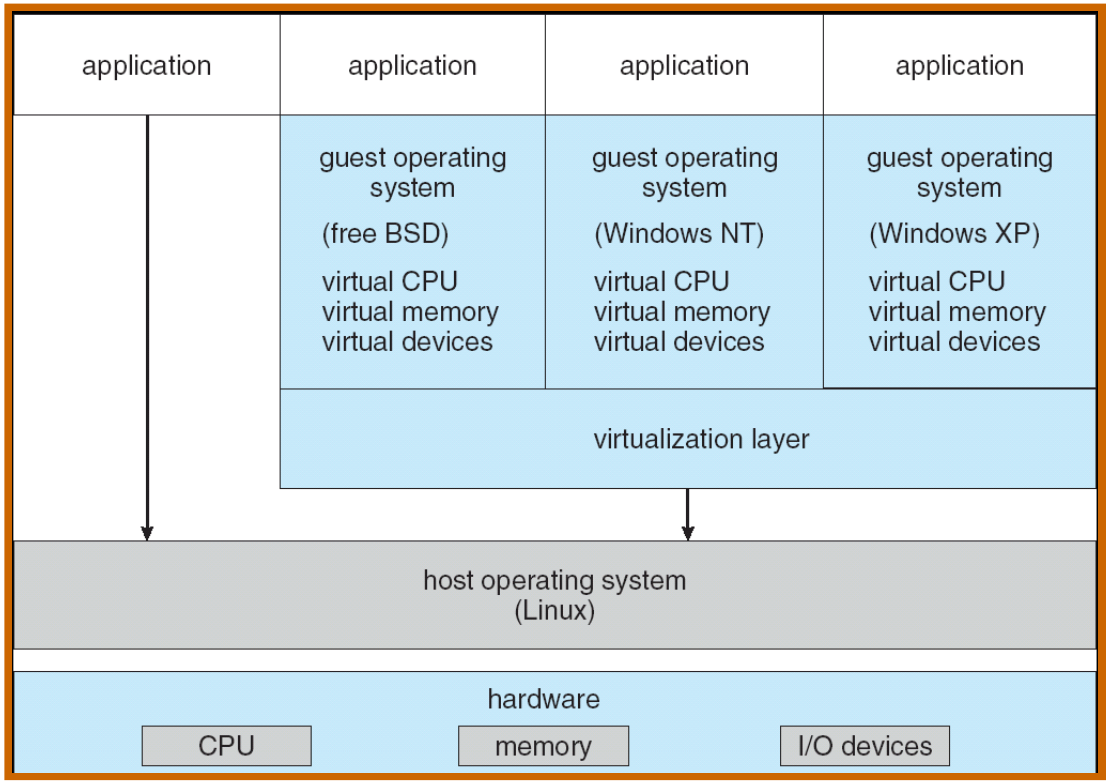
- Abstract H/W of single computer - CPU, Memory, Disk Drive, N/W interface card
- **Benefits**
- Host system is protected from VM and VM's from each other
- **Disadvantage**
- No direct sharing of resource
- Solution
- Share file system volume – share files
- Network of Virtual Machines – send info over the network

Virtual Machines

- **Why VM??**
- Make and test changes in OS without modifying the host
- Rapid porting and testing of programs in various environment

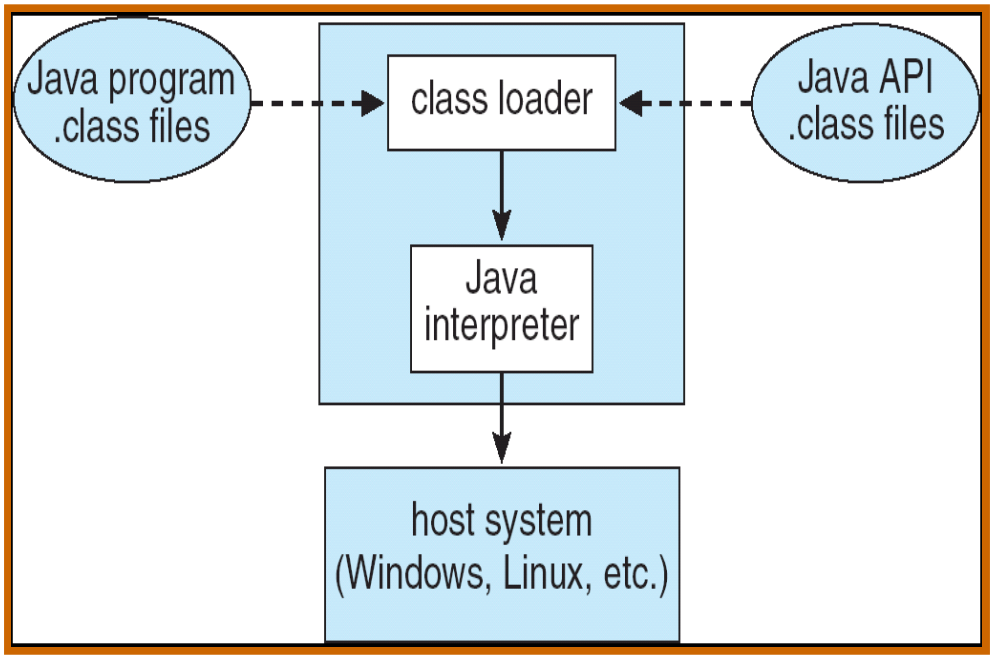
- **Simulation**
- Emulator translates outdated s/m instruction into native instruction.
- Slower and Writing correct emulator is writing CPU again.

Vmware Architecture



- Abstracts intel X86 into isolated VM
- The guest owns and manages a file within the file system of host OS

The Java Virtual Machine



- Byte Code
- The class loader and Java Interpreter
- Check for overflow, pointer
- Garbage collection
- Can be implemented in H/W or S/W
- Just in Time Compiler

System Boot

- **Booting -** The procedure of starting the computer by loading the kernel
- **Bootstrap program** – locates the kernel, loads to main memory & starts execution
- Tasks of Bootstrap program – state of machine, initialize aspects of system
- ROM vs EPROM
- Firmware – slower and expensive
- **Boot Block** – Large OS stored in a single location from disk.

REFERENCES

http://www.tech-punch.com/wp-content/uploads/2013/01/windows-8-start-screen.jpg

http://lifeinopensource.files.wordpress.com/2010/11/screenshot-1.png

http://upload.wikimedia.org/wikipedia/commons/2/29/Linux_command-line._Bash._GNOME_Terminal._screenshot.png