Tishk International University
Applied Science Faculty
IT Department

Operating Systems - IT 327

Lecture 1: Introduction to OS

3rd Grade - Fall Semester

Instructor: Alaa Ghazi

OPERATING SYSTEMS

Information Technology Department
Alaa Ghazi
E-mail: alaa.ghazi@tiu.edu.iq
Room no.: 315

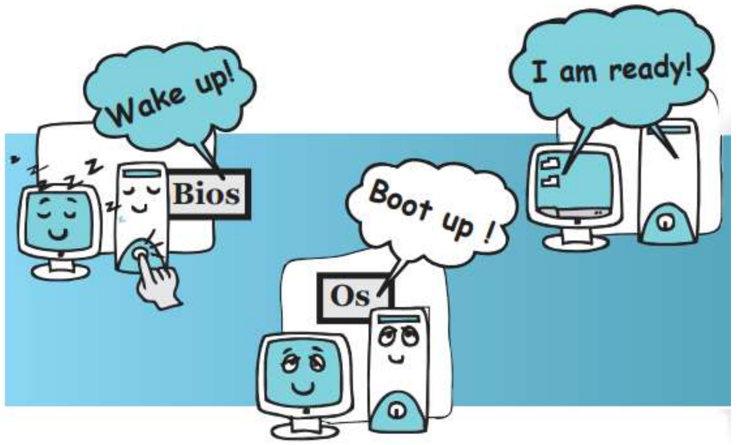
Text Books:

1. Operating System Concepts (9th Edition)
 Abraham Silberschatz (Yale University),
Peter B. Galvin (Pluribus Networks),
Greg Gagne (Westminster College),
Wiley 2012.
2. Introduction to Operating Systems
Prof. Chester Rebeiro
Indian Institute of Technology, Madras.

Lecture 1: Introduction to OS

Agenda:

- What is an Operating System?
- Computer System Components
- Operating System Services
- Functions of Operating Systems
- User Interface Types
- System Call Implementation
- Operating Systems Types
- Computer-Startup Operation



What is an Operating System?

An **operating system (definition)** is a program that manages the computer hardware and provides the base for application programs.

- It acts as an intermediary between the user application and hardware
- It provides **an environment** within which Application programs can do work
- It acts as the **resource manager** of the following resources:
  1. CPU time
  2. Memory
  3. File Storage
  4. I/O devices
- It acts as a **control program** that manages the execution of application programs and prevent errors and improper use of the computer.

Kernel :The operating system core component that is running at all times on the computer.

Computer System Components

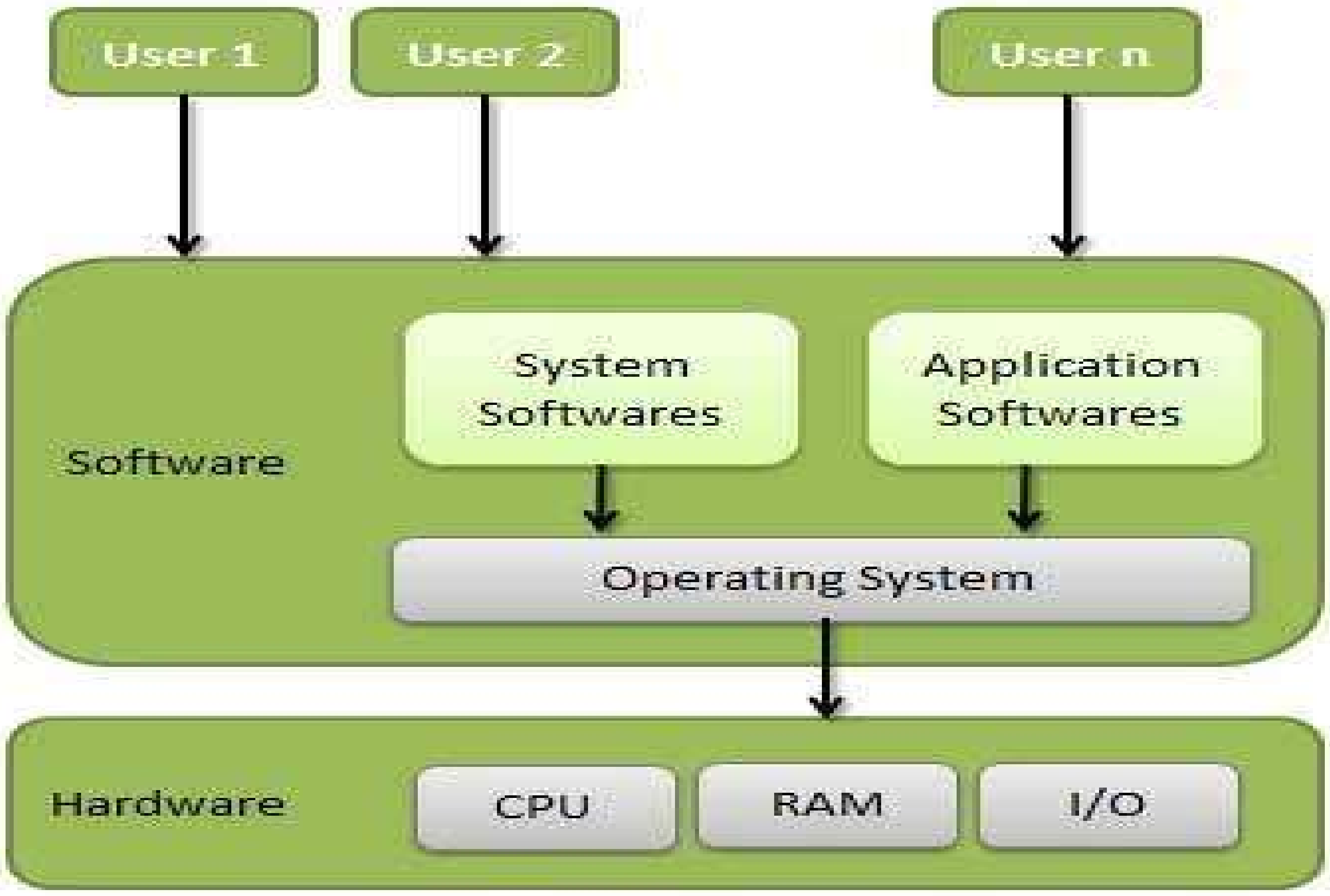
1. Hardware: provides basic computing resources: CPU, memory, secondary storage, and I/O devices

2. Operating System: as in the definition above

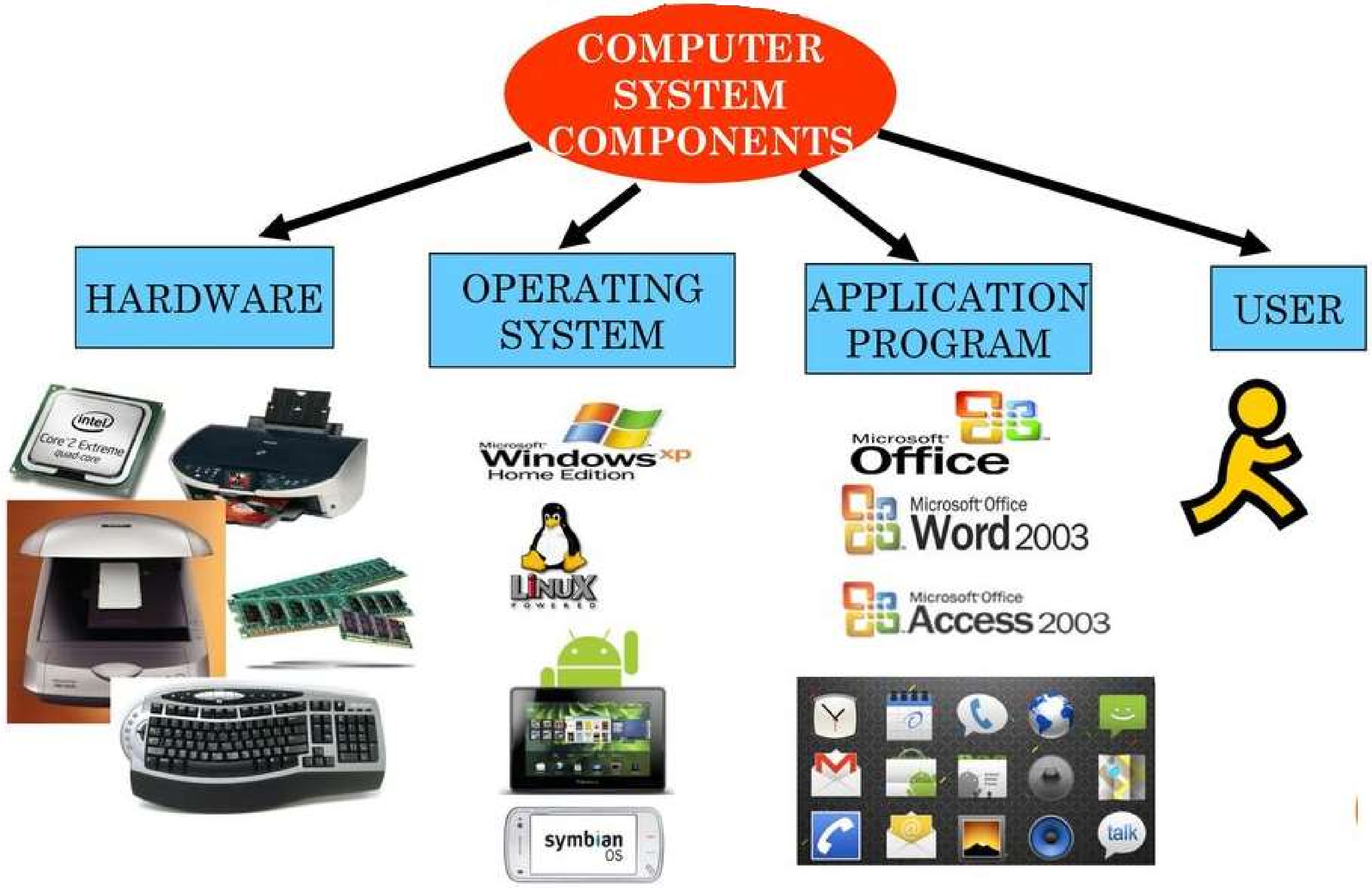
3. System Software: Utilities that are associated with the operating system but not part of the kernel.

4. Application Software – The programs that helps end-users to perform specific tasks such as: Word Editor, web browsers, database systems, video games …etc.

4. **Users:** People, or other machines.



Computer System Components

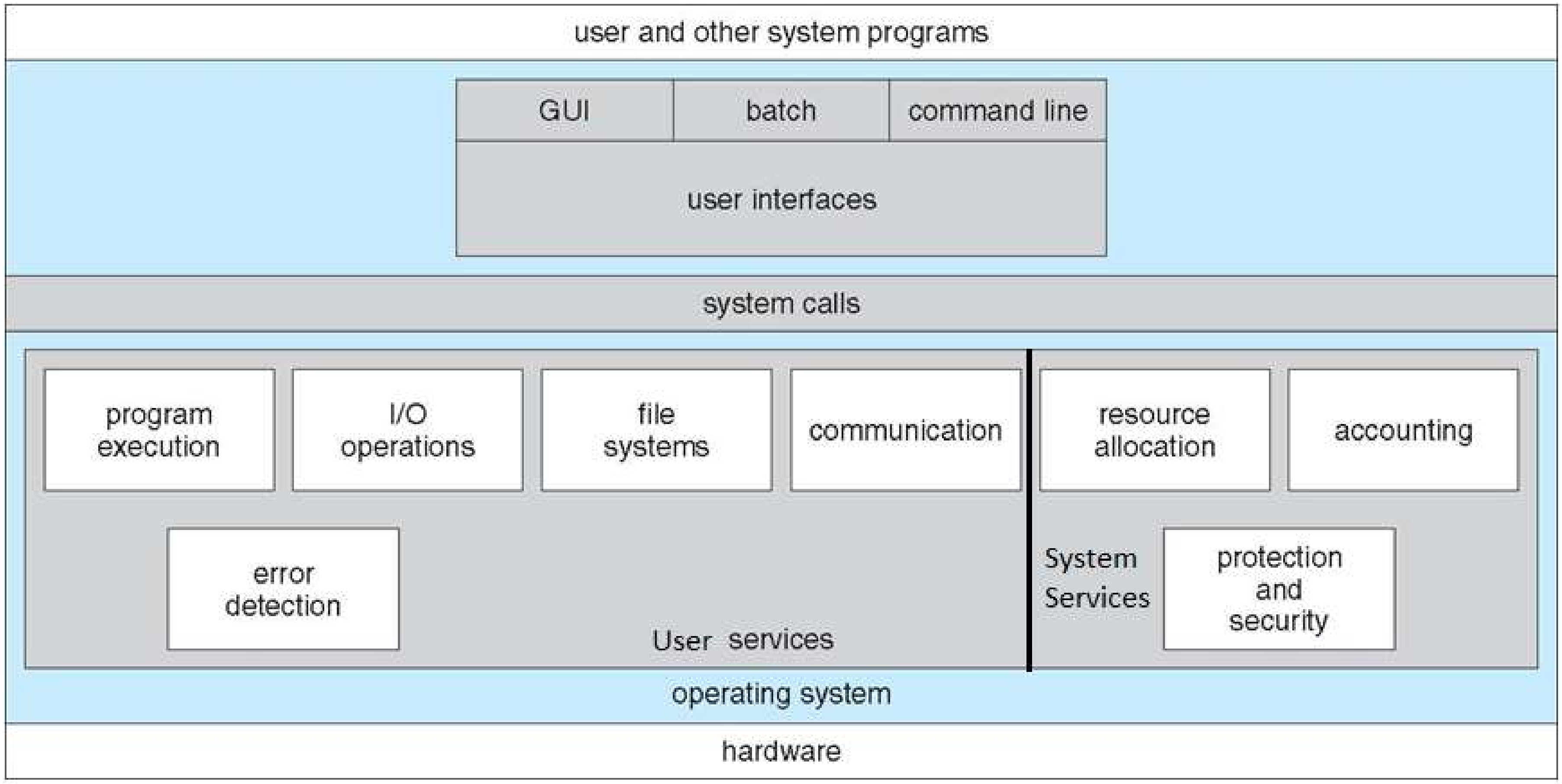

COMPUTER SYSTEM COMPONENTS
HARDWARE
OPERATING SYSTEM
APPLICATION PROGRAM
USER
intel Core 2 Extreme quad-core
Microsoft Windows xp Home Edition
LINUX POWERED
symbian OS
Microsoft Office
Microsoft Office Word 2003
Microsoft Office Access 2003
talk

(not required in the exam)

Operating System Services

1. **User interface** which can be either CLI or GUI
2. **Program execution** - to load and run a program into memory, end execution, either normally or abnormally.
3. **File-system manipulation** – enables running program to:
   - create and delete files and folders
   - read and write files and folders,
   - List and Search files and folders
   - Permission management.
4. **I/O operations** - to offer running program a way of interaction with I/O devices.
5. **Inter-Process Communications** – Enables processes to exchange information.

A View of Operating System Services



Functions of Operating System

Processor Management: An operating system manages the processor's work by allocating various jobs to it and ensuring that each process receives enough time from the processor to function properly.

Memory Management: An operating system manages the allocation and deallocation of the memory to various processes and ensures that the other process does not consume the memory allocated to one process.

Device Management: OS controls the working of these input-output devices. It receives the requests from these devices, performs a specific task, and communicates back to the requesting process.

File Management: An operating system keeps track of information regarding the creation, deletion, transfer, copy, and storage of files in an organized way.

Security: The operating system provides various techniques which assure the integrity and confidentiality of user data.

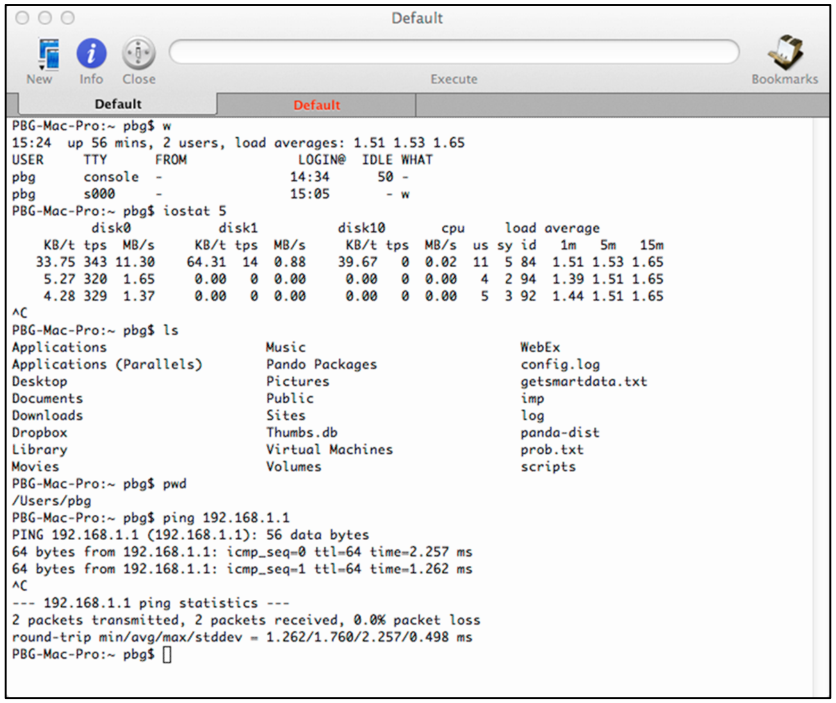
Error Detection: OS checks the hardware for any type of damage, or any software error, then displays a proper alert to the user so that the appropriate action can be taken.

User Interface Types

1- Command Line Interface (CLI)

CLI or **command interpreter** allows direct command entry.

- It is implemented in kernel, or by system program
- It fetches a command from user and executes it
- Commands are either built-in, or other programs

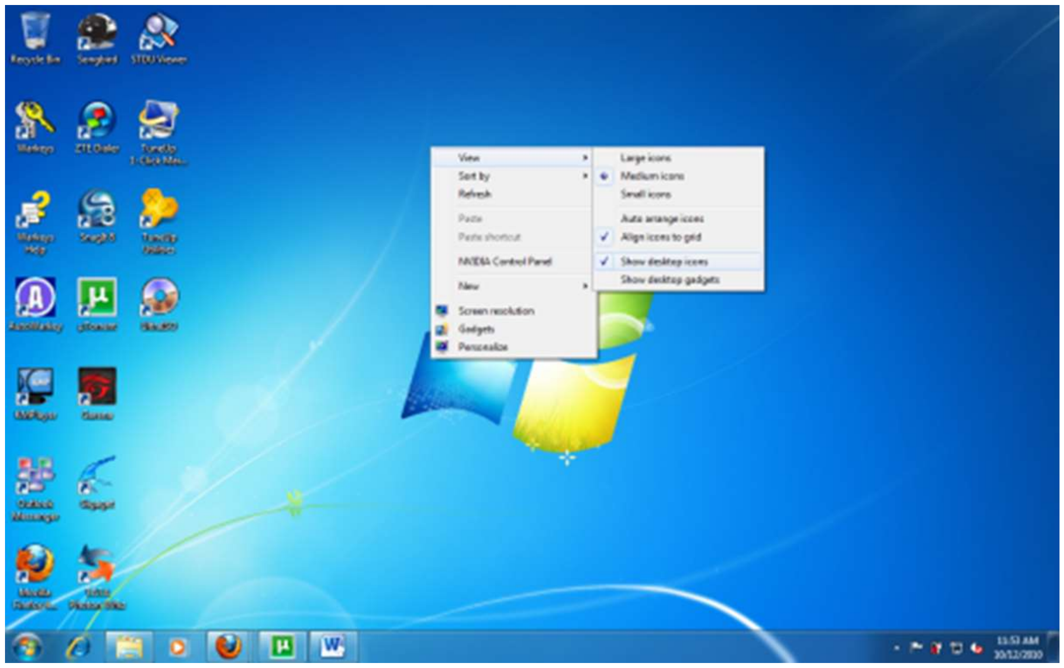


```
PBG-Mac-Pro:~ pbg$ w
15:24  up 56 mins, 2 users, load averages: 1.51 1.53 1.65
USER     TTY      FROM              LOGIN@  IDLE WHAT
pbg      console  -                14:34      50 -
pbg      s000     -                15:05       - w
PBG-Mac-Pro:~ pbg$ iostat 5
          disk0           disk1          disk10       cpu     load average
    KB/t tps  MB/s    KB/t tps  MB/s    KB/t tps  MB/s  us sy id   1m   5m   15m
   33.75 343 11.30   64.31  14  0.88   39.67   0  0.02  11  5 84  1.51 1.53 1.65
    5.27 320  1.65    0.00   0  0.00    0.00   0  0.00   4  2 94  1.39 1.51 1.65
    4.28 329  1.37    0.00   0  0.00    0.00   0  0.00   5  3 92  1.44 1.51 1.65
^C
PBG-Mac-Pro:~ pbg$ ls
Applications                    Music                           WebEx
Applications (Parallels)        Pando Packages                  config.log
Desktop                         Pictures                        getsmartdata.txt
Documents                       Public                          imp
Downloads                       Sites                           log
Dropbox                         Thumbs.db                       panda-dist
Library                         Virtual Machines                prob.txt
Movies                          Volumes                         scripts
PBG-Mac-Pro:~ pbg$ pwd
/Users/pbg
PBG-Mac-Pro:~ pbg$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=2.257 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.262 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.262/1.760/2.257/0.498 ms
PBG-Mac-Pro:~ pbg$ 
```

2- Graphical User Interface (GUI)

User-friendly **desktop** interface that interacts with mouse, keyboard, and monitor.

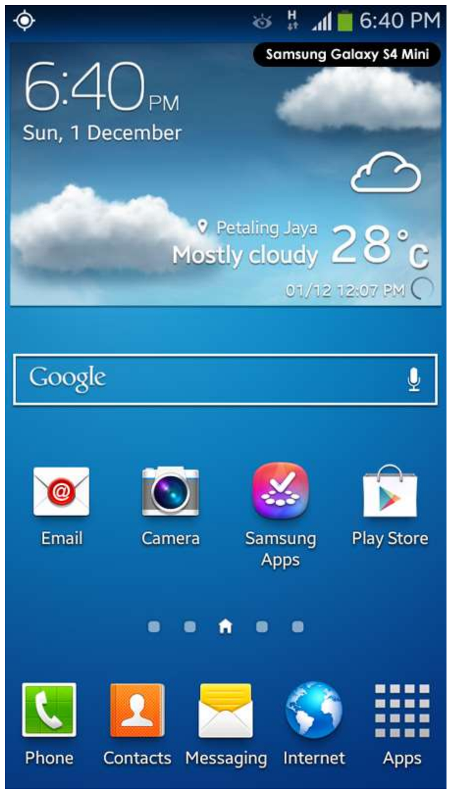
- **Icons** represent files, programs, actions, etc
- Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open a **folder**)
- Many systems now include both CLI and GUI interfaces



3- Touch Screen Interface

A touchscreen interface is a combined display/input device; where the screen displays a graphical interface, and a user's physical touching of the screen is interpreted as an input

- Mouse is not desired
- Actions and selection are based on gestures
- It uses virtual keyboard for text entry



System Call Implementation

System Calls are Programming interface to the services provided by the OS.

An Application Programming Interface (API) is a set of routines, data structures, object classes and/or protocols provided by libraries and/or operating system services in order to support the building of applications.

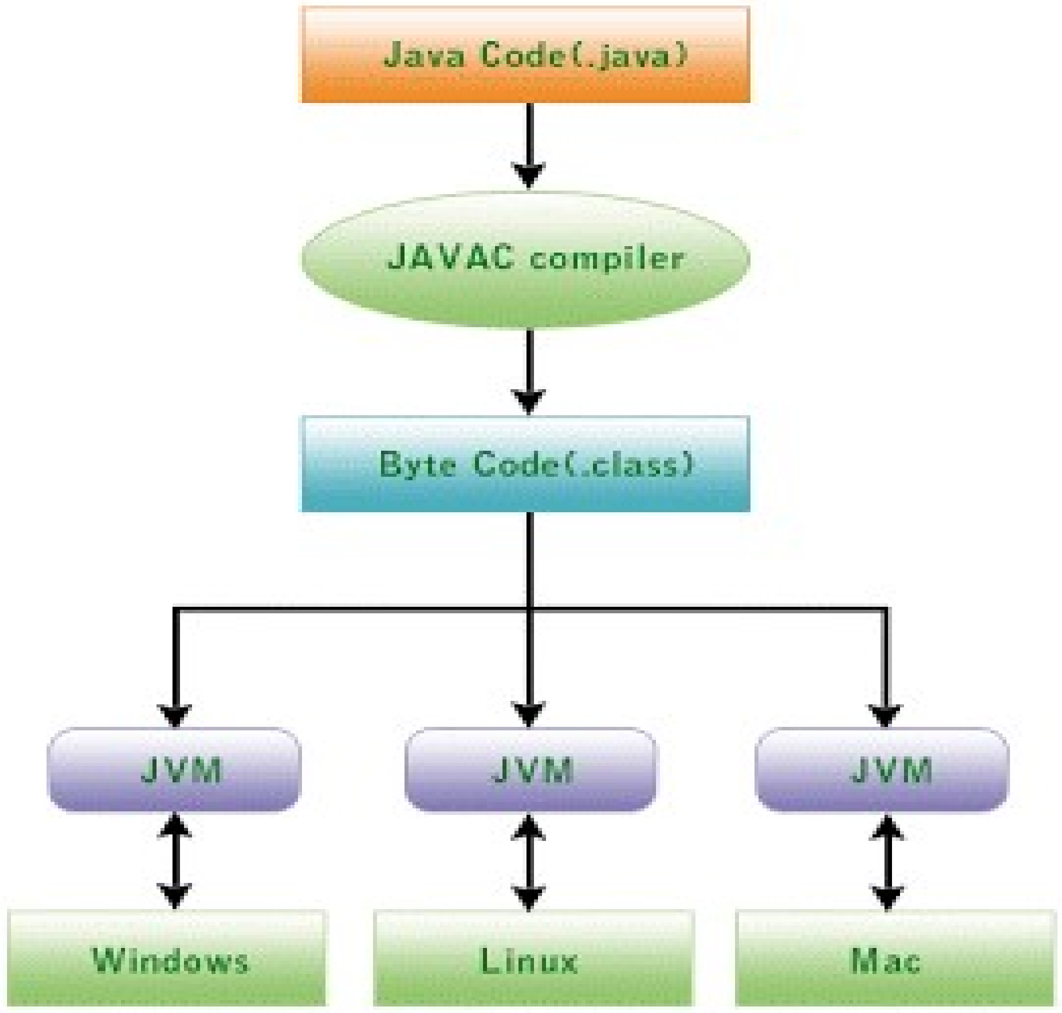
DLLs are used in Windows OS which hold data structures, object classes and APIs

Three most common APIs are:

- ❖.NET API for Windows,
- ❖POSIX API for POSIX-based systems (mainly UNIX versions)
- ❖Java API for the Java virtual machine (JVM)

- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values
- The caller Just needs to obey API and understand what OS will do as a result call.
- Most details of OS interface hidden from programmer by API

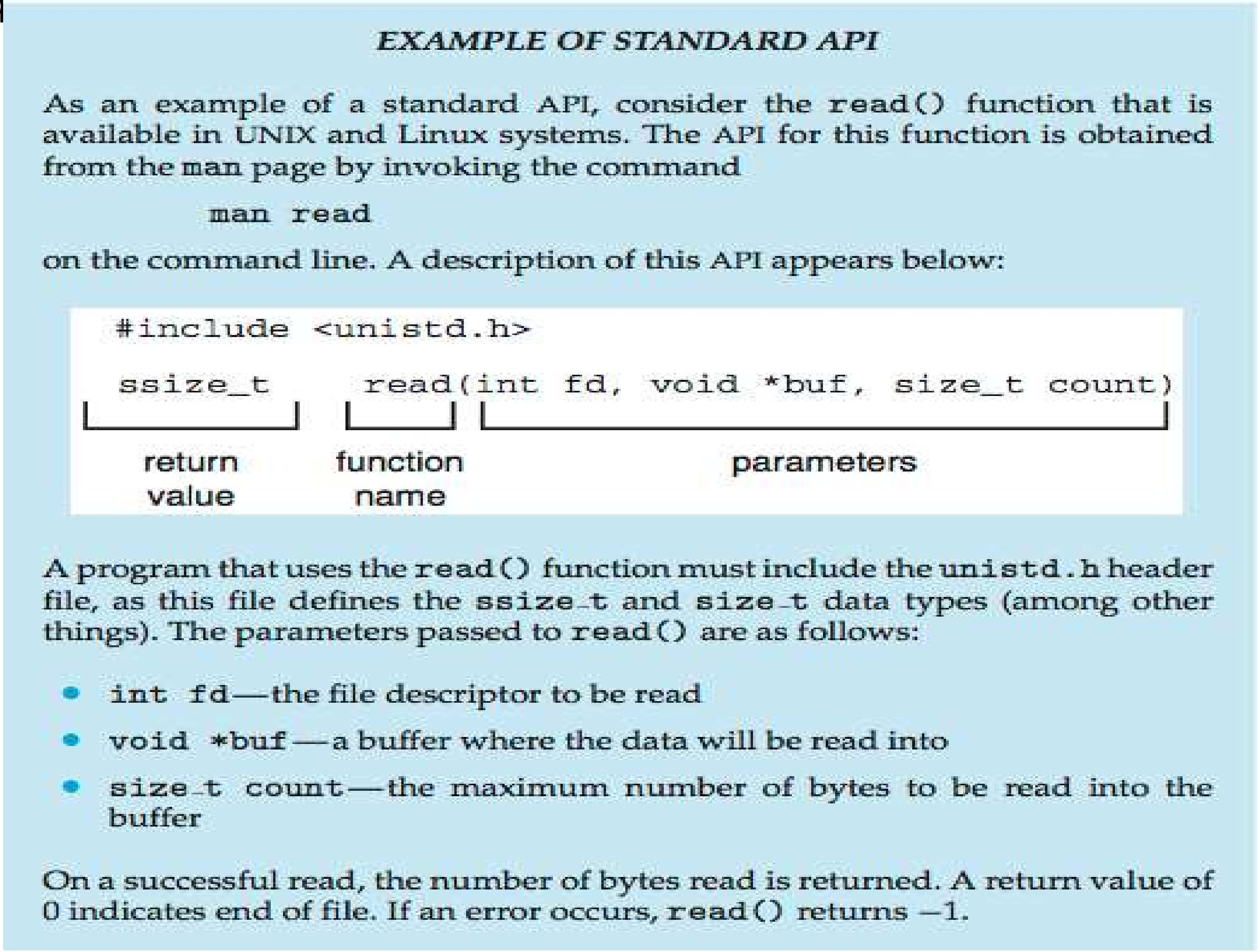
Java Virtual Machine (JVM)



(not required in the exam)

Java Virtual Machine (JVM)

Example of Standard API
(not req



EXAMPLE OF STANDARD API

As an example of a standard API, consider the `read()` function that is available in UNIX and Linux systems. The API for this function is obtained from the `man` page by invoking the command

```
man read
```

on the command line. A description of this API appears below:

```
#include <unistd.h>
```

$$\underbrace{\texttt{ssize\_t}}_{\text{return value}}\quad \underbrace{\texttt{read}}_{\text{function name}}\underbrace{\texttt{(int fd, void *buf, size\_t count)}}_{\text{parameters}}$$

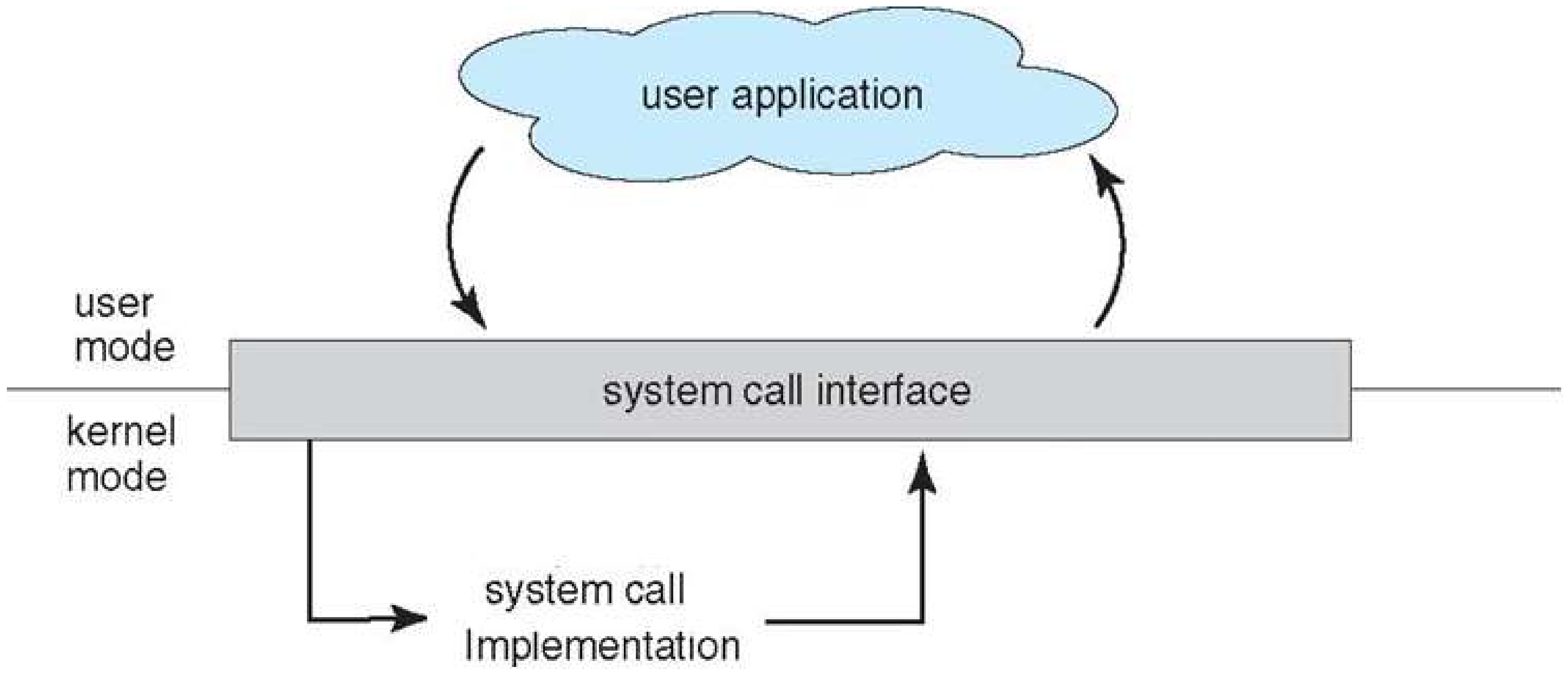
A program that uses the `read()` function must include the `unistd.h` header file, as this file defines the `ssize_t` and `size_t` data types (among other things). The parameters passed to `read()` are as follows:

- `int fd`—the file descriptor to be read
- `void *buf`—a buffer where the data will be read into
- `size_t count`—the maximum number of bytes to be read into the buffer

On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, `read()` returns −1.

(not required in the exam)

System Call – OS Relationship



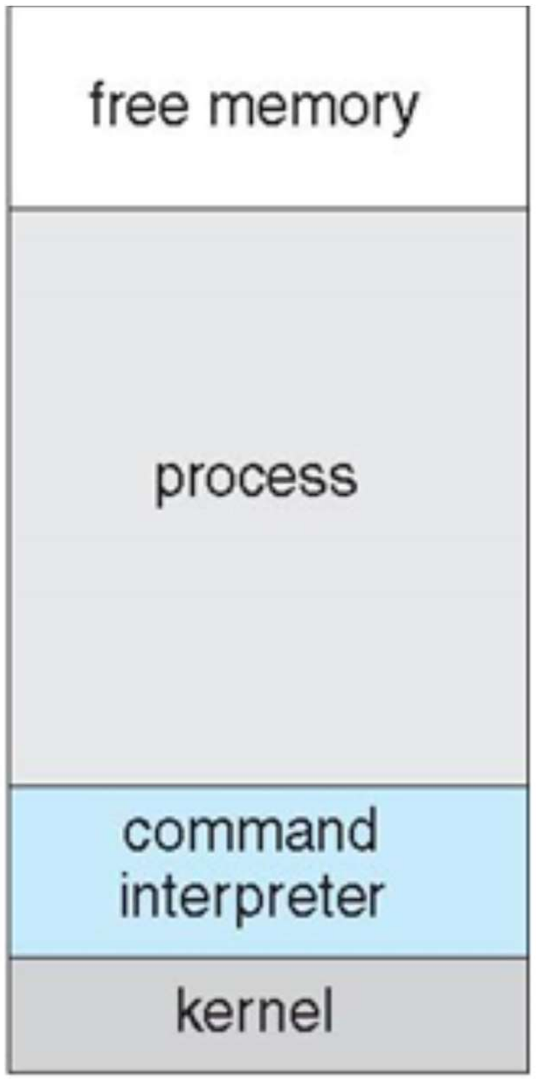
System Call- Parameter Passing Methods

1. **Registers**: pass the parameters in registers
   NOTE: We have limited no. of registers
2. **Memory Block**: Parameters are stored in memory block, and address of block is passed as a parameter in a register
3. **Stack**: Parameters **pushed**, onto the **stack** by the program and **popped** off the stack by the operating system

NOTE: Block and stack methods do not limit the number or length of parameters being passed

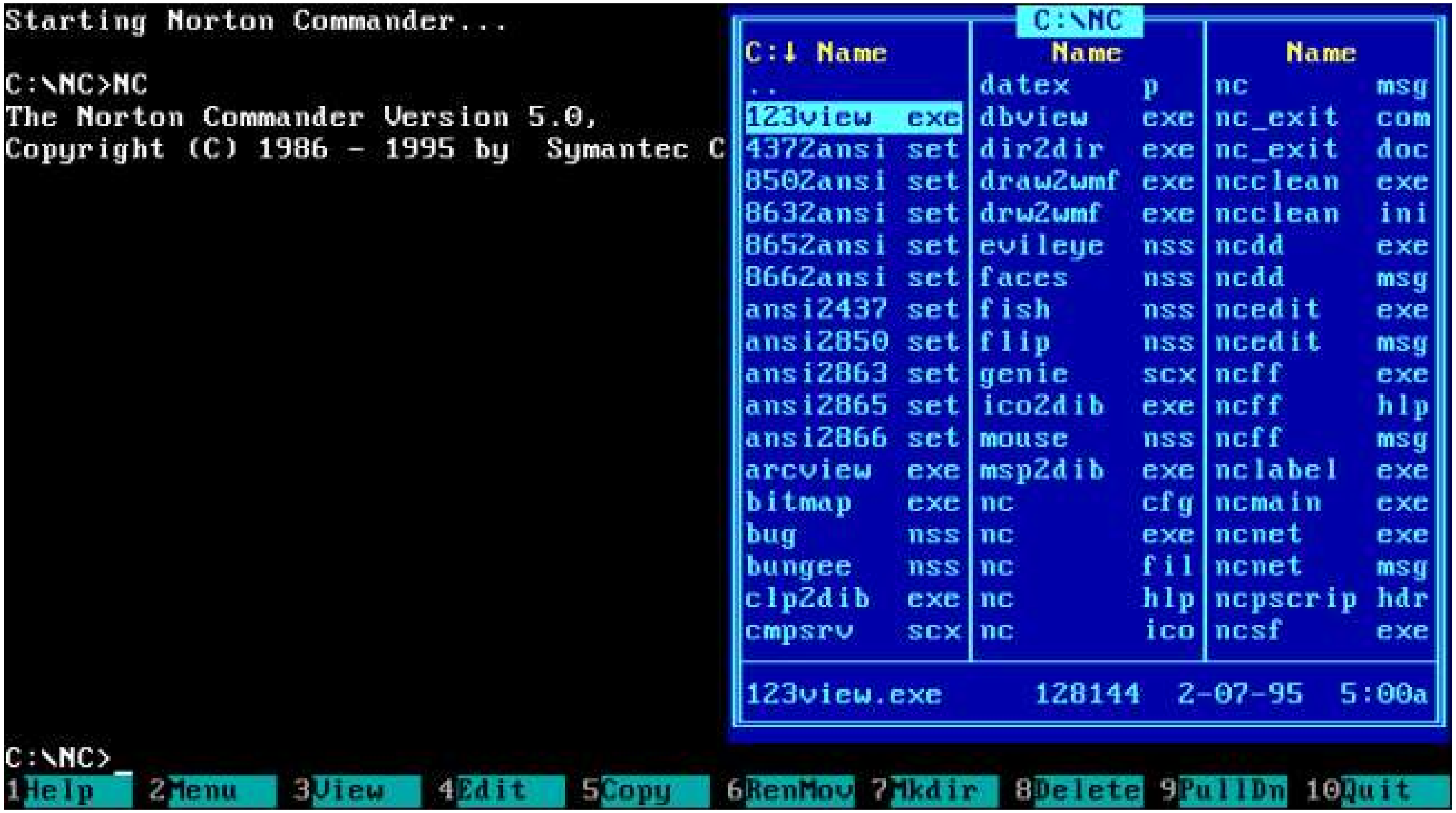
Common Types of Operating Systems

1- Single-Task OS
Single-task and Single memory space
CLI Loads program into memory, overwriting all but the kernel but the application program takes full control of the computer
After Program exit -> CLI is reloaded
Example: MS-DOS,



Single-Task OS Memory Map

MS-DOS Screen Example
(not required in the exam)



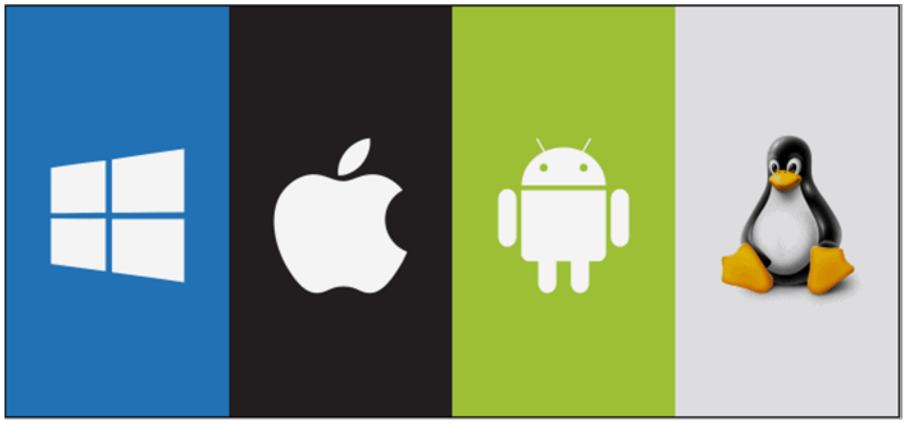
```
Starting Norton Commander...

C:\NC>NC
The Norton Commander Version 5.0,
Copyright (C) 1986 - 1995 by  Symantec C
```

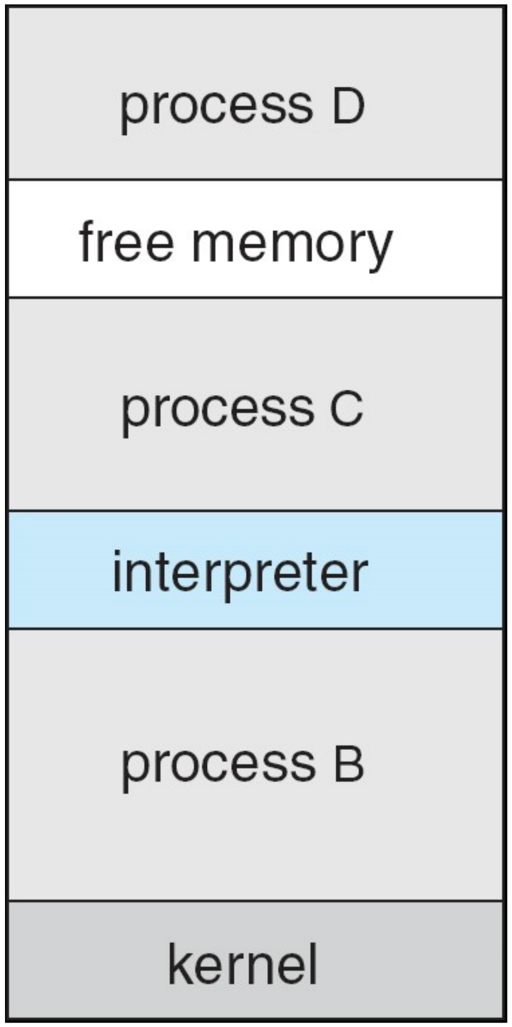
```
                         C:\NC
C:↓ Name        |     Name          |     Name
..              | datex        p    | nc           msg
123view    exe  | dbview       exe  | nc_exit      com
4372ansi   set  | dir2dir      exe  | nc_exit      doc
8502ansi   set  | draw2wmf     exe  | ncclean      exe
8632ansi   set  | drw2wmf      exe  | ncclean      ini
8652ansi   set  | evileye      nss  | ncdd         exe
8662ansi   set  | faces        nss  | ncdd         msg
ansi2437   set  | fish         nss  | ncedit       exe
ansi2850   set  | flip         nss  | ncedit       msg
ansi2863   set  | genie        scx  | ncff         exe
ansi2865   set  | ico2dib      exe  | ncff         hlp
ansi2866   set  | mouse        nss  | ncff         msg
arcview    exe  | msp2dib      exe  | nclabel      exe
bitmap     exe  | nc           cfg  | ncmain       exe
bug        nss  | nc           exe  | ncnet        exe
bungee     nss  | nc           fil  | ncnet        msg
clp2dib    exe  | nc           hlp  | ncpscrip     hdr
cmpsrv     scx  | nc           ico  | ncsf         exe

123view.exe       128144   2-07-95   5:00a
```

```
C:\NC>_
1Help   2Menu   3View   4Edit   5Copy   6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

2- Multi-Task OS

Multiple processes can run at the same time, each with its own memory space. Example: Ubuntu, Windows 10, Android, IOS.





Multi-Task OS Memory Map

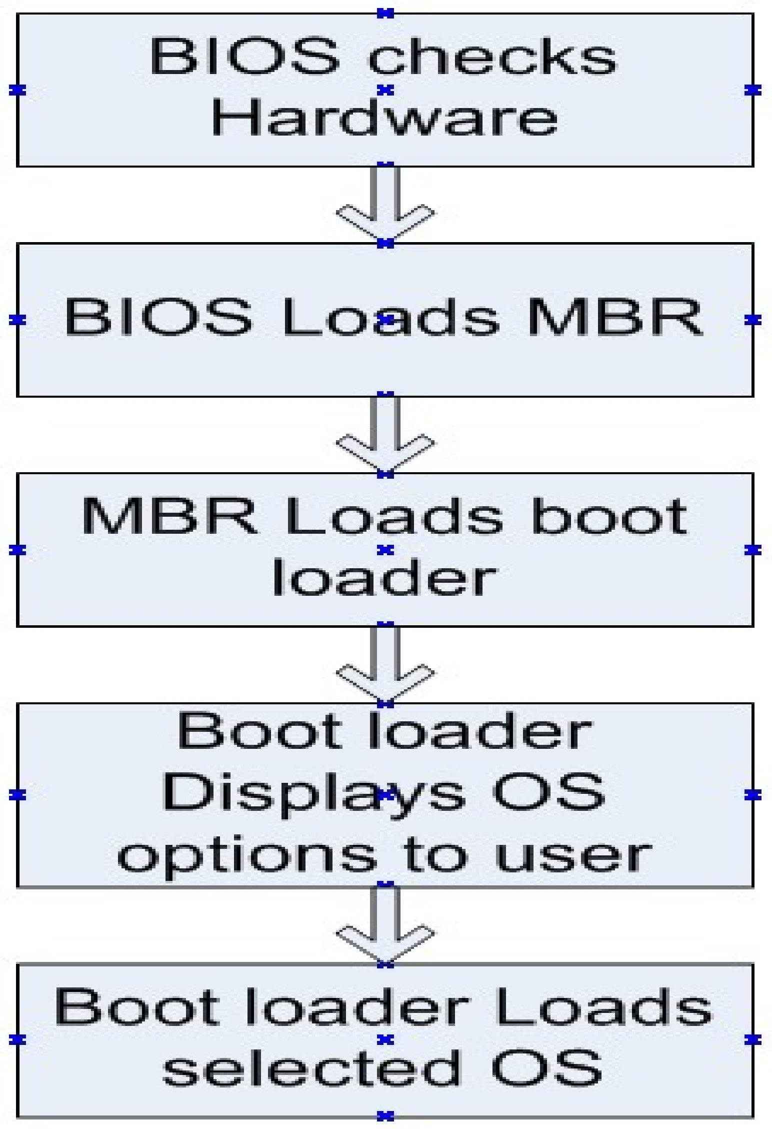
Computer Startup Operation

The computer start-up has the following stages:

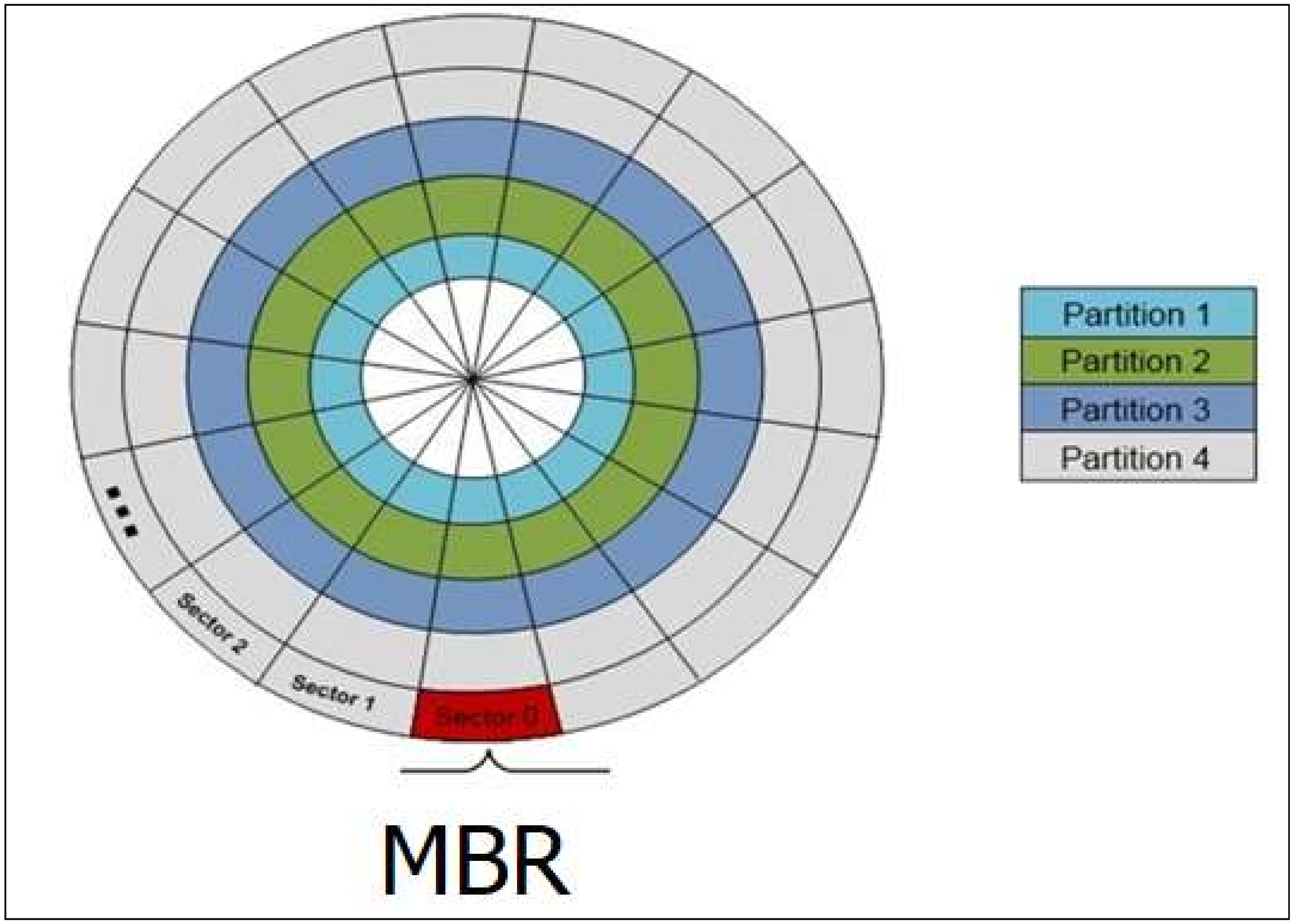
- **BIOS** is a simple initial program that runs when the computer is powered up, and transfers control to the code in MBR.
- **Master Boot Record (MBR):** it is the first sector on the disk, which contains the small code that starts the boot process. When executed it will transfer the control to boot loader.
- **A boot loader** (boot manager), is a small program that loads selected OS into memory.
- After the **Operating System** is loaded in to memory, it takes control of the computer, the computer system will sit idle waiting for an event.

Modern OSs are interrupt driven: If there is nothing for the OS to do, it will wait for something to happen by interrupts. There are two types of them:

- Hardware Interrupts usually occur by sending a signal to the CPU from specified device. (like keyboard click or a message from remote computer).
- Software Interrupts usually occur by executing a system call from the application program. (like email program checking for new emails).


BIOS checks Hardware
BIOS Loads MBR
MBR Loads boot loader
Boot loader Displays OS options to user
Boot loader Loads selected OS

Computer Startup Operation



MBR location
(not required in the exam)