Operating System
Session 2

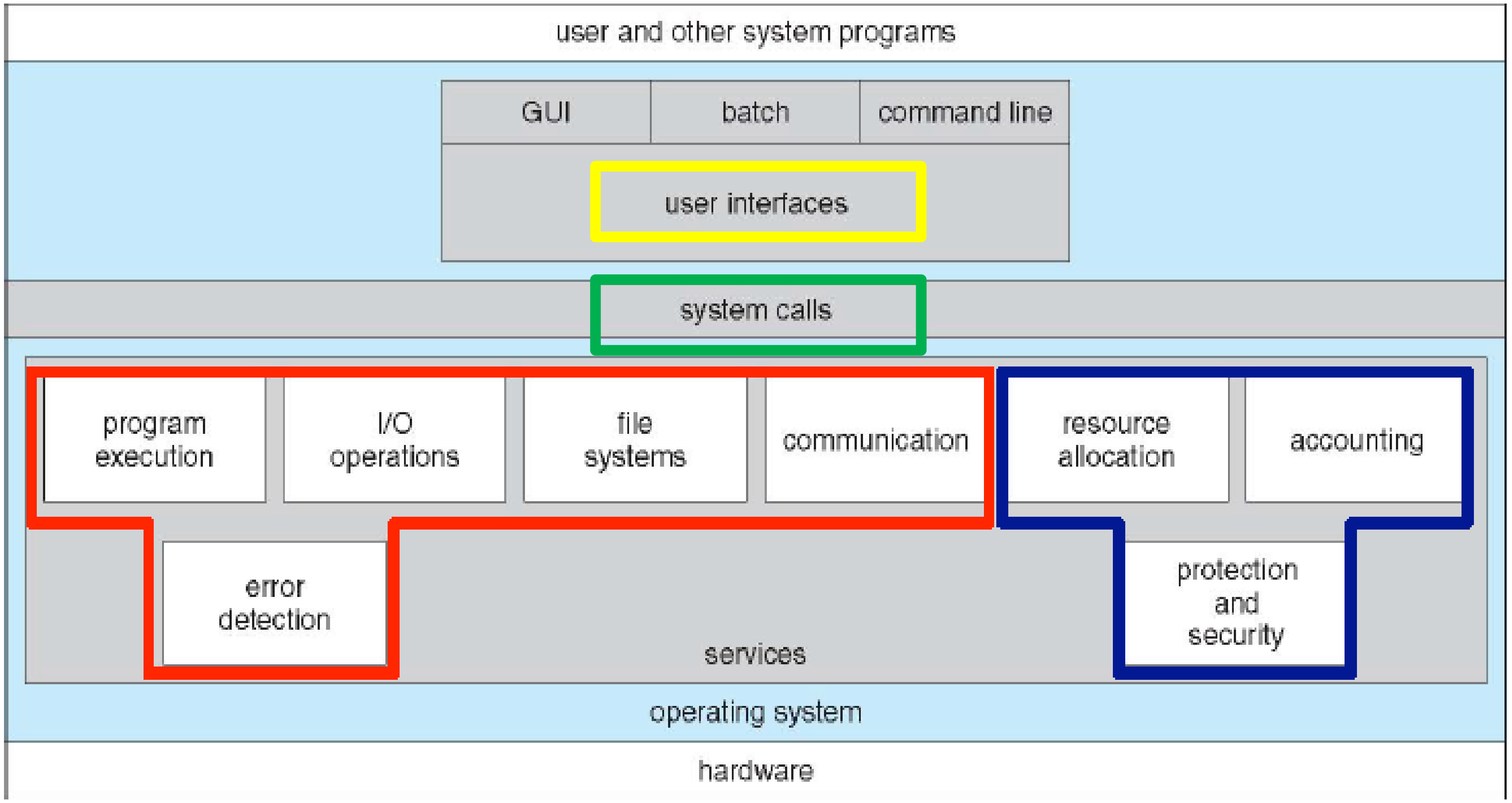
Hakim Sabzevari University

Dr.Malekzadeh

Operating Systems Structure

Operating System Services

- OS Services are shown in the following:



Helpful to users

Better efficiency/operation

Program execution

- Operating system loads a program into memory and assign CPU to it to execute the program.
- The program must be able to end its execution, either normally or abnormally (with error codes).

I/O Operation

- I/O means any I/O files or any I/O devices.
- I/O Operation means using I/O to perform the desired task such as interrupt handling.
- A running program may require any I/O device. For efficiency and protection purposes, the users usually can not have direct access to the I/O devices. So it is the duty of the operating system to provide the I/O operation since users cannot execute them directly.

File management system

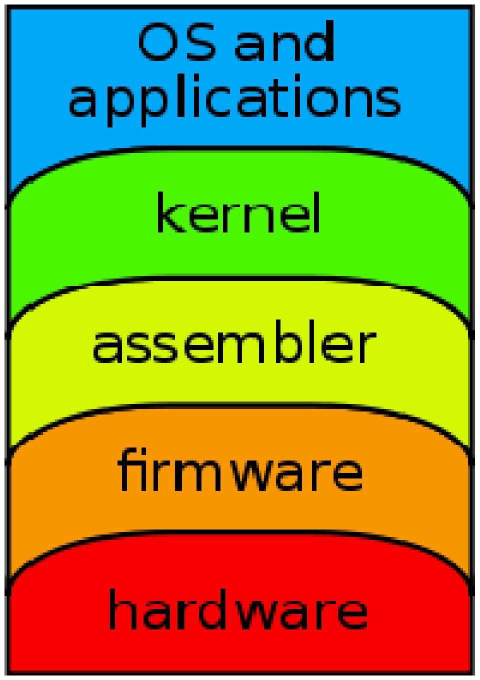
- OS keeps track of the files for their location and status. OS decides who gets use of the files to enforce protection requirements.

- File management system calls include:
  - Operations such as create file, delete file, open, close, read, write, reposition, get file attributes, and set file attributes.
  - These operations may also be supported for directories as well as ordinary files.
  - The support of primitives for manipulating files and directories.

Primitive means callable function or instruction like interfaces for system calls. For example, when a program calls fopen() it invokes the OS's C library implementation.

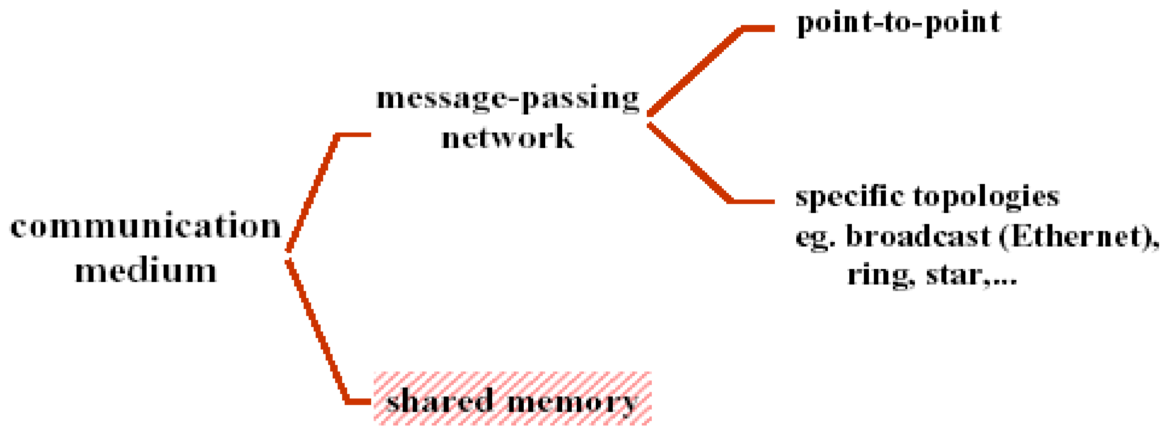
Abstraction

- Abstraction means hiding the underlying mechanisms and implementation details; an abstraction is software that hides lower layer details and provides a set of higher-layer functions.



Communication

- Since even a single user request may result in multiple processes running in the operating system on the user's behalf, the processes need to communicate with each other.
- Every process needs to exchange information with another process. The communications can occur within processes executing on the same computer or between processes executing on different computers linked by a network.
- Thus, communication between process can be implemented by two methods:

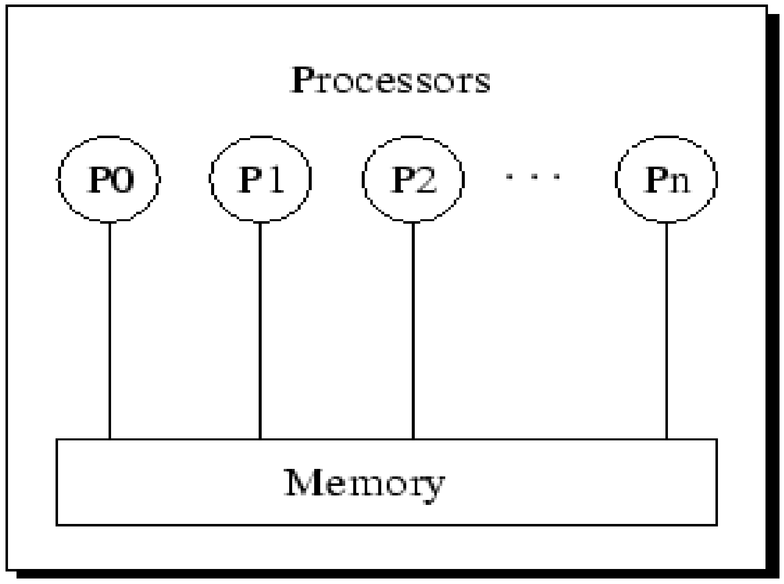


Communication cont…

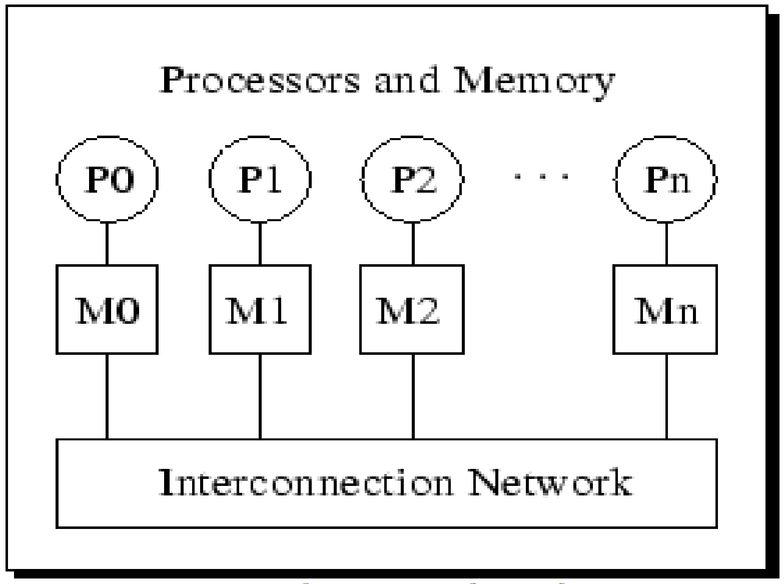
- Inter Process Communication (IPC) provides **abstractions** for exchanging information between processes.
- Example: Using the Clipboard for IPC

Communication cont…

1. Shared memory: processes communicate by reading/writing to shared memory blocks. This is simple but data can rewrite behind your back.
2. Distributed memory: processes communication is done by message passing. Less error can happen because memory is not shared the problem is overhead caused by many message exchanging.



shared memory



message passing non-shared memory

Error detection

- The OS needs:
  - To be aware of all errors
  - To take action for those errors

- Errors can happen any time and anywhere in the system. Error may occur in:
  - CPU
  - I/O devices
  - Memory hardware
  - ...

Accounting

- The operating system keeps track of which users use how much and what kinds of resources. Logs of each user must be kept for accounting purposes. These statistics are also useful for the researchers who wish to re-configure the system to improve computing services.

- The accounting data can be used for:
- Gather the statistics
- Bill
- Impose limit
- …

Protection/Security

- Security is an important issue in multi-user computer systems. Protection involves ensuring that all access to system resources is controlled.

- **Research**: Encrypting File System (EFS): NTFS encryption in Windows

User interfaces

- A **shell (sh)** is a user interface (UI) to access (OS) services.
- The shell acts as an interface between the user and the kernel.
- Shell as an interpreter takes users programs (called commands) and gives them to the OS to perform.
- Therefore, a shell is a program for starting other programs (wraps around another program, hence its name) such as applications, scripts, and user programs.

Types of shell

- The shell starts when you log in or open a terminal. In general, operating system shells use either:
  - Command line interface (CLI)
  - Graphical user interface (GUI)

Command line interface (CLI) shell

- The CLI Shell is responsible for displaying the prompt that you see in terminal. Terminal/console is a text-only input/output window to input users commands.

- CLI shells allow to perform some operations faster in some situations, especially when there is no a proper GUI. However, they require the user to memorize commands and their syntax.

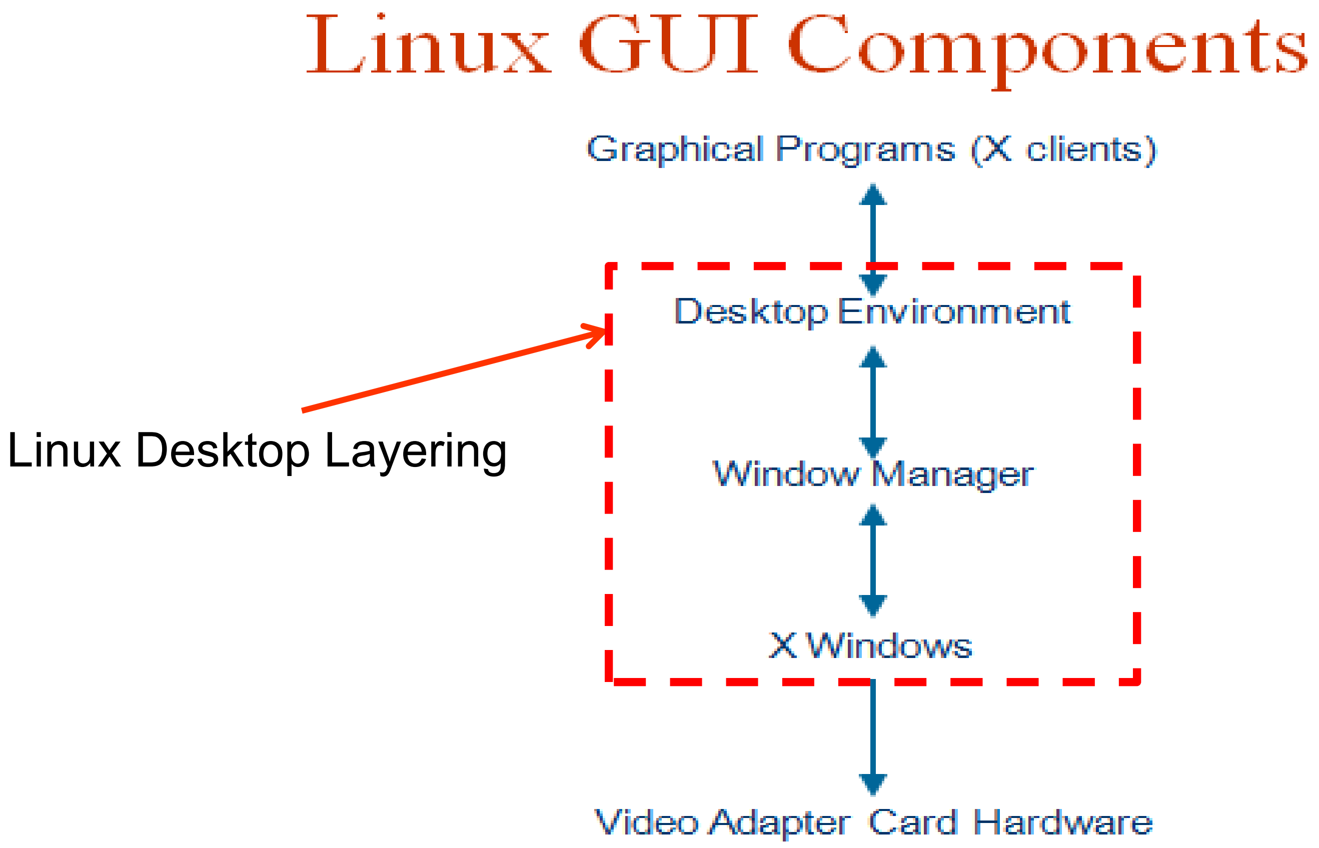
Graphical user interface (GUI) shell

- The GUI shell allows the user to interface with the OS with mouse, keyboard, and monitor, thus are simple and easy to use.

- Since graphical shells come with certain disadvantages (for example, lack of support for easy automation of operation sequences called bash scripts), most GUI-enabled operating systems also provide additional CLI shells.

GUI shell cont…

- Linux has multiple software packages available for each layer. There are basically three layers that can be included in the Linux desktop. These Linux GUI components are shown as follow:



X Window/X server

- X Window/X server is the most popular windowing system. It is the foundation that give the required instructions to the Window Manager to draw graphic elements (such as icons and menus).

- The X Window builds the primitive framework that allows:
  - draws windows
  - moving of windows
  - interactions with keyboard and mouse

- These are required for any graphical desktop.

Window Manager

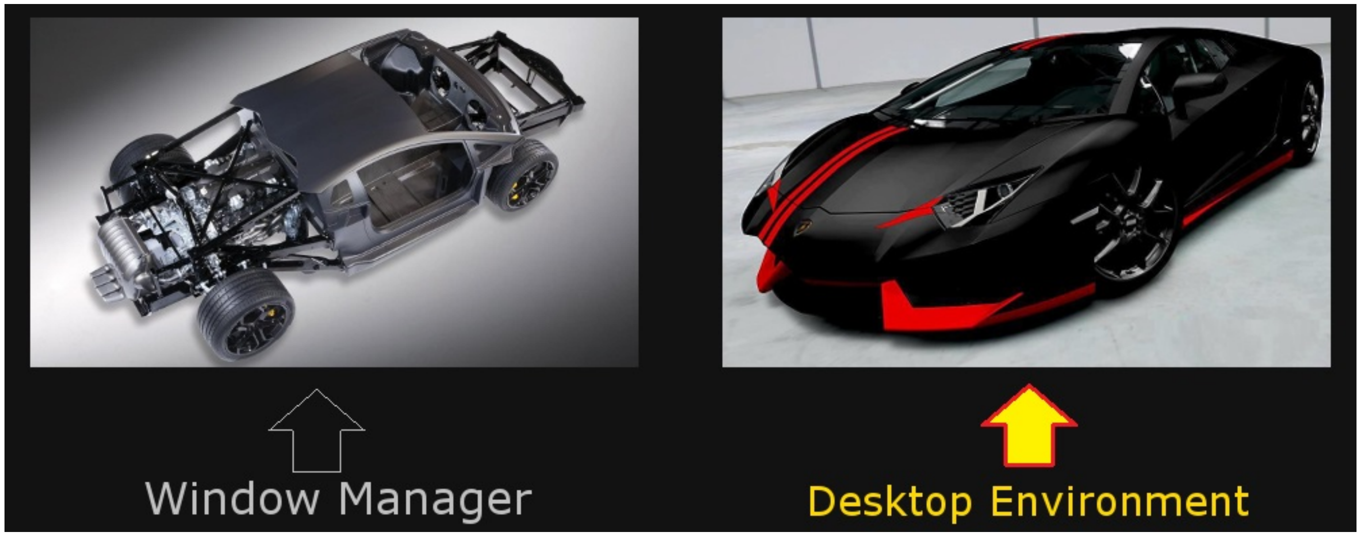
- Window Manager is a software that manages the windows:
  - It controls the placement and appearance of windows for user personalization.
  - It puts the window decoration around the contents including the buttons to minimize or close.
  - It allows resizing and moving the windows around, decides which window is on top.
  - It does **not** provide taskbars, system trays, login managers, icons, screensavers and other common features expected of a modern GUI. Only the windows and the ability to interact with them.

- There are many popular Window managers, including AfterStep, Enlightenment, KWM (K Windows Manager), Windowmaker, iceWM, fvwm2, blackbox, and sawmill.

Desktop environment (میز کار)

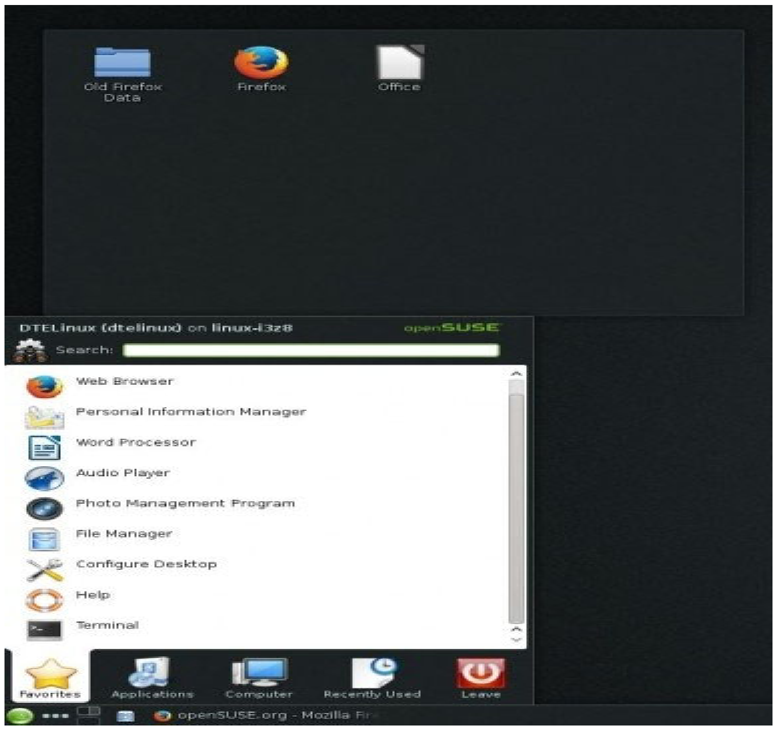
- Desktop environment rides on top of a Window Manager and adds many features including login screen, panels, status bars, drag-and-drop capabilities, and many integrated applications and tools.

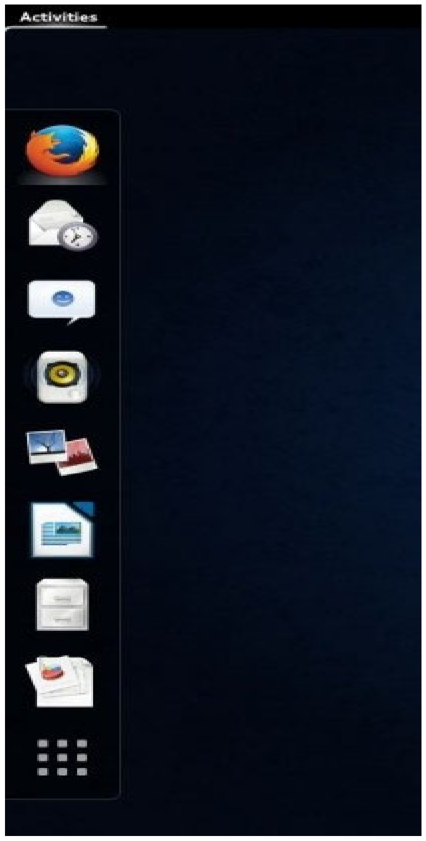
- A desktop environment provides a complete GUI by using both X Window and Window Manager. Desktop environments are responsible for adding visual appear, customization options, and lots of polish to the window manager.

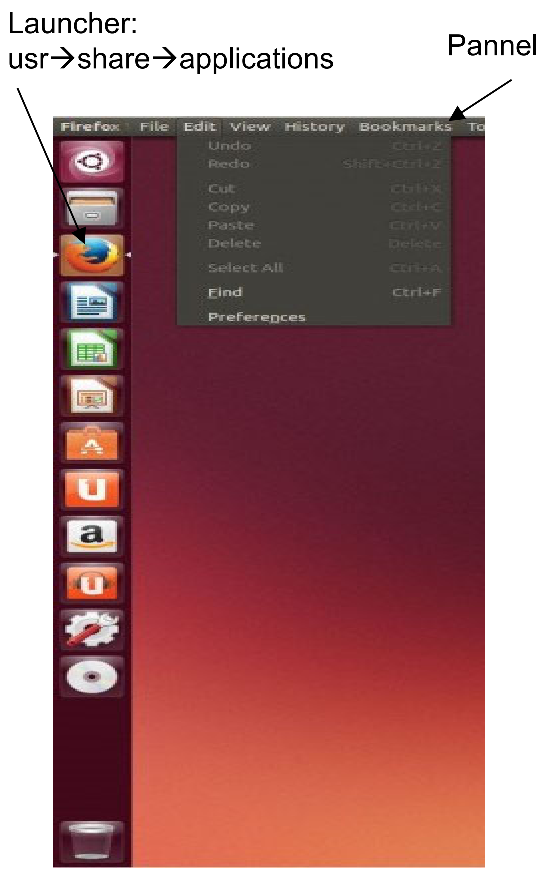


GUI shell cont…

- The three most popular **desktop** environment for Linux are:
  - K Desktop Environment (KDE)
  - GNU Network Object Model Environment (GNOME; developed by the GNOME Project, part of the GNU Project)
  - Unity
  - XFCE







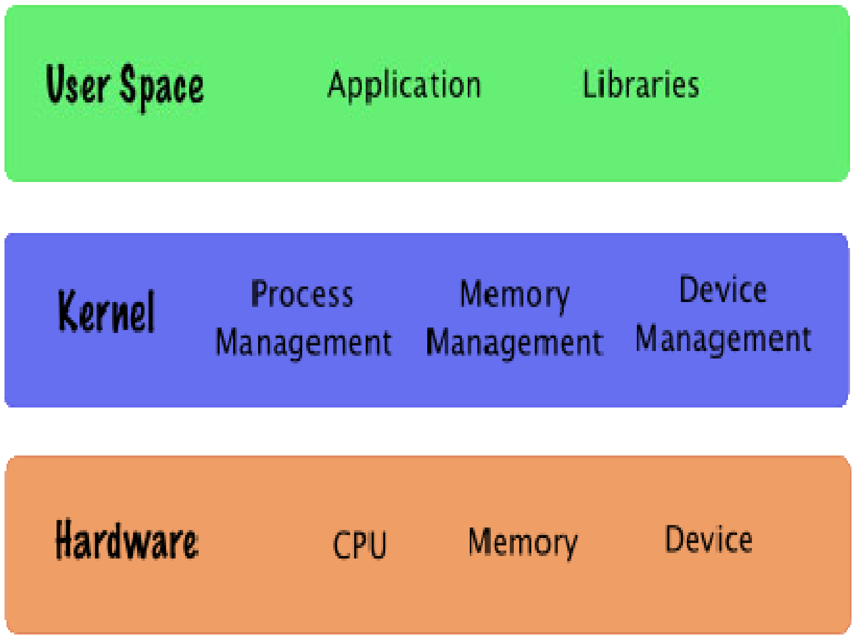
Modes of Execution of the OS

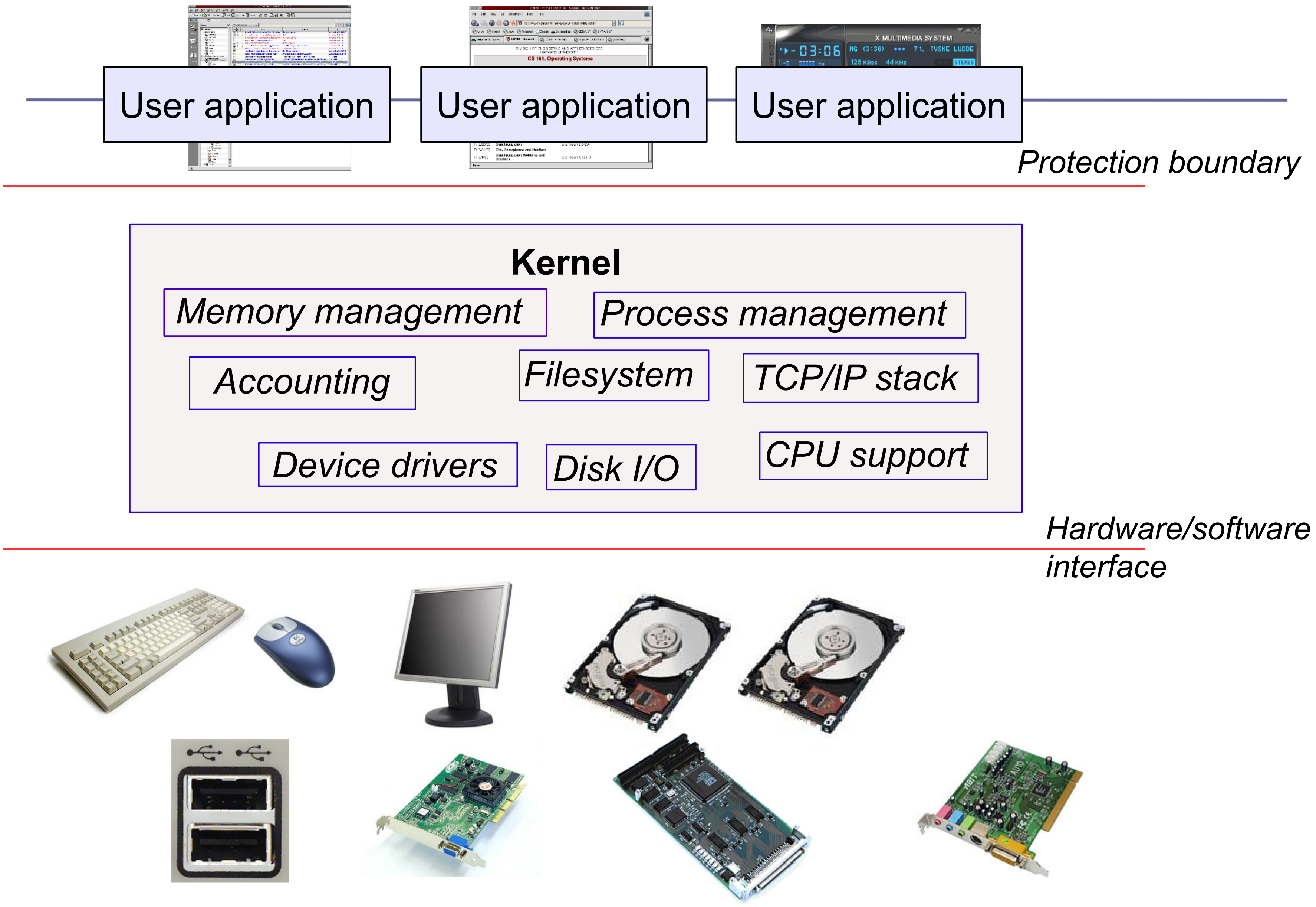
- The OS provides two modes of execution (for protection) which differ in the privilege (امتياز access level):
  - User mode: is lower-privilege and is limited to certain address spaces and operations. By default an operating system runs in user mode.
  - Kernel/Supervisor mode: can do anything the CPU is capable of and has unrestricted access to everything .

Kernel

- The kernel is the core code and heart of an operating system which as the lowest-level code provides the basic abstractions that all other codes require. The kernel is present between user applications and hardware.

- The kernel is **brain** of the operating system, which controls everything from access to the hard disk to memory management. Whenever you want to do anything, it goes though the kernel.




User application
User application
User application
Protection boundary
Kernel
Memory management
Process management
Accounting
Filesystem
TCP/IP stack
Device drivers
Disk I/O
CPU support
Hardware/software interface

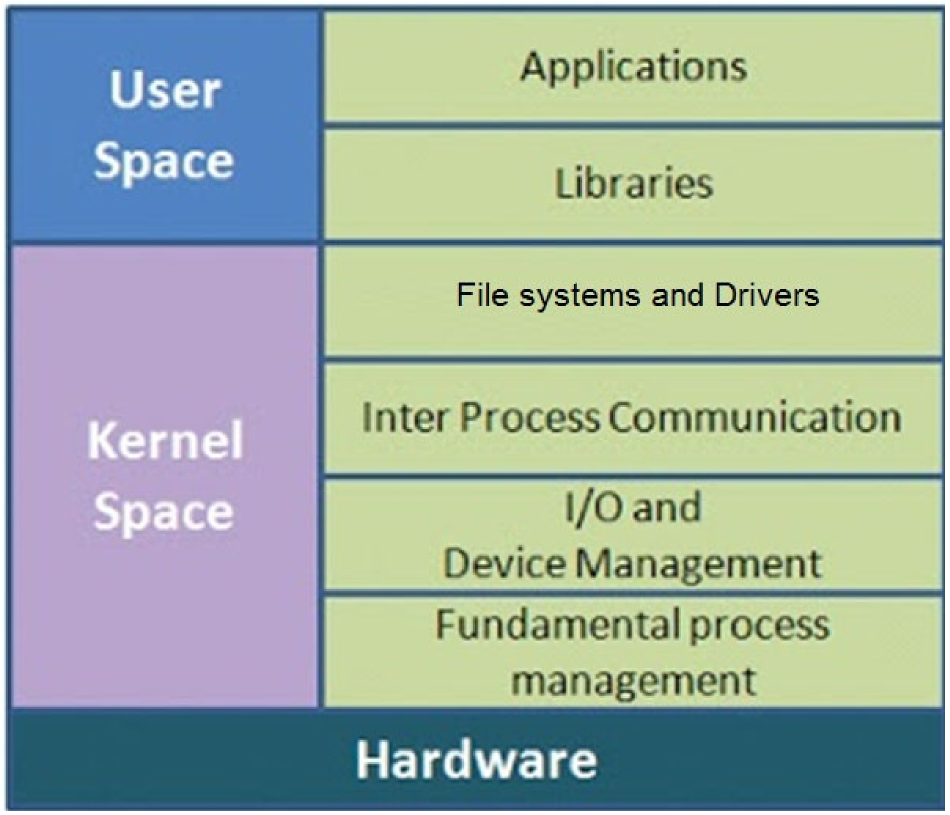
Kernel cont…

- When system starts up, the kernel is loaded into the main memory, and remains there till the system shutdown. The kernel sits in the main memory in a protected space called **kernel space** which is used for many functions it performs.

- The remaining space is called the **user space** which is used by various programs that users run on OS.

Kernel cont…

- Applications do not have direct access to the computer hardware. Applications have to request hardware access from kernel as a third-party that provides all access to computer resources.


User Space
Applications
Libraries
Kernel Space
File systems and Drivers
Inter Process Communication
I/O and Device Management
Fundamental process management
Hardware

Kernel cont…

- A device driver is a computer program that enables the operating system to interact with a hardware device. It provides the operating system with information of how to control and communicate with that hardware.

- The kernel’s job is to talk to the hardware and software, and to manage the system’s resources as best as possible. Kernel talks to the hardware via the drivers that are included in the kernel (or additionally installed later on in the form of a kernel module). This way, when an application wants to do something (say change the volume setting of the speakers), it can just submit that request to the kernel, and the kernel can use the driver it has for the speakers to actually change the volume.

Kernel functions

- Some of kernel operations include:
  - Process Management: Process creation, destruction, and inter-process communication; scheduling and dispatching; process switching; management of process control blocks
  - Resource Management: Memory (allocation of address space; swapping; page and segment management), secondary storage, I/O devices, and files
  - Input/Output: Transfer of data between memory and I/O devices; buffer management; allocation of I/O channels and devices to processes
  - Interrupt handling: Process termination, I/O completion, service requests, software errors, hardware malfunction

Interrupt

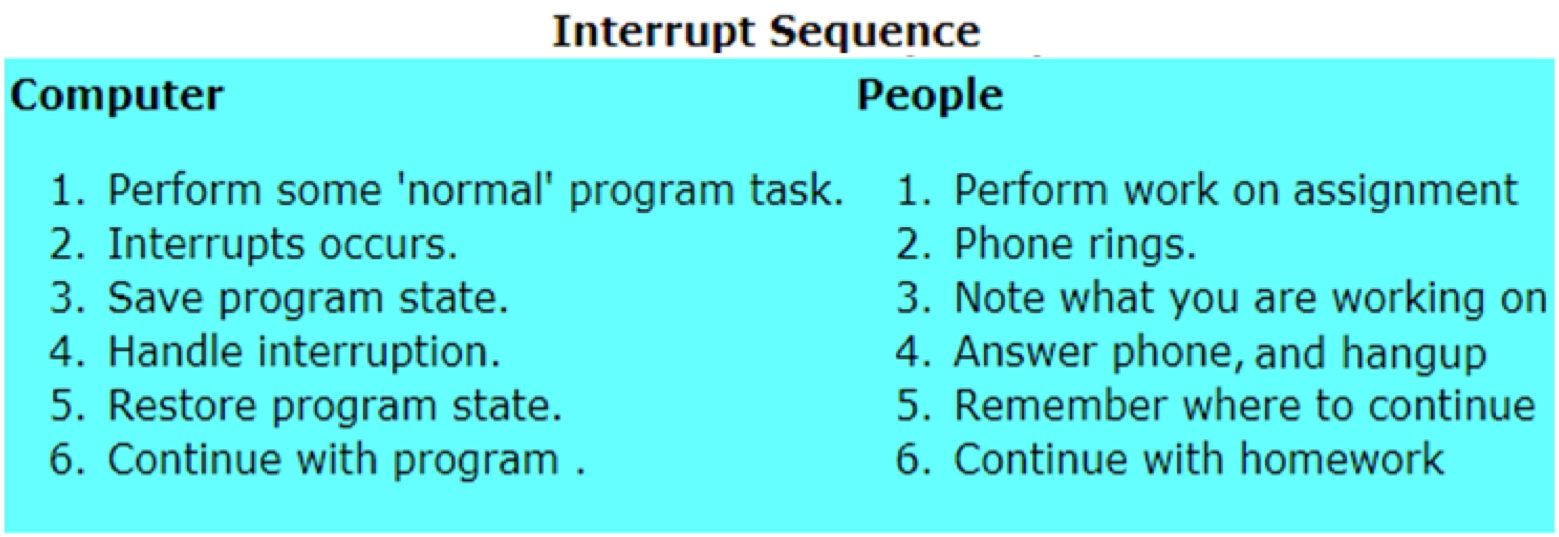
- A computer interrupt can be compared to interruptions in everyday life. For example, the ringing of your telephone or a knock on your door are interrupt-type events.

- Your phone or doorbell can ring at any time during your day, and when they ring you will typically stop what you are doing, deal with the reason behind the phone call or front door visit, and then go back to what you were doing.

- Likewise, just as you can ignore a phone call or doorbell if what you are doing requires your immediate and undivided attention, so it is possible for CPU to defer interrupts during certain critical operations, or even to ignore them entirely.

Interrupt cont…

- Interrupts are signals sent to the CPU to stop its current activities and execute the appropriate part of the operating system.
- Once the interrupt execution path is completed, execution returns to the normal path that was interrupted.



Interrupt Sequence

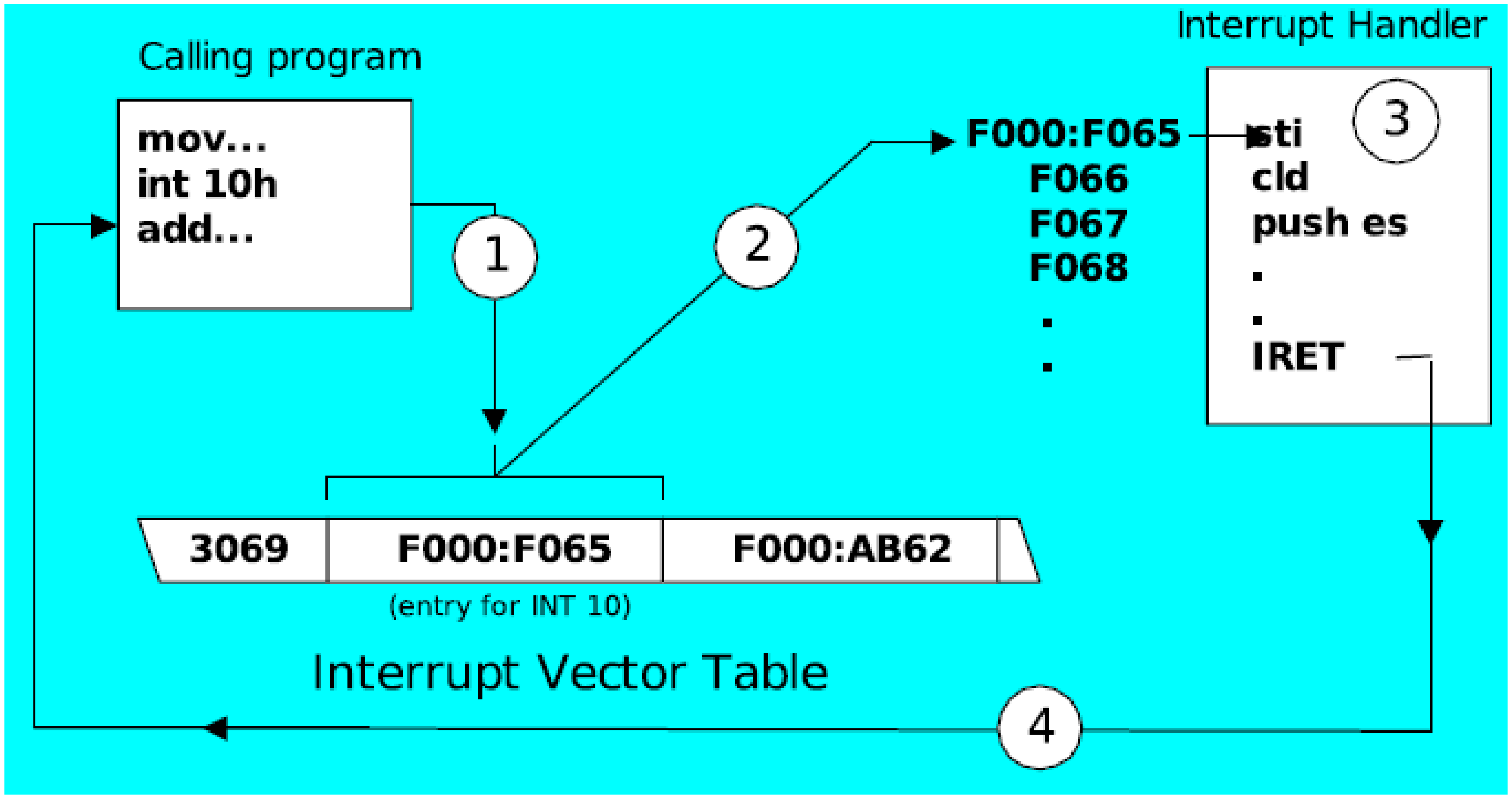
Computer	People
1. Perform some 'normal' program task.	1. Perform work on assignment
2. Interrupts occurs.	2. Phone rings.
3. Save program state.	3. Note what you are working on
4. Handle interruption.	4. Answer phone, and hangup
5. Restore program state.	5. Remember where to continue
6. Continue with program .	6. Continue with homework

Interrupt cont…

- Whenever an interrupt happens, the corresponding interrupt handlers/Interrupt Service Routines (ISR) is called. The ISR is a function in OS or in device driver which contains the corresponding interrupt code.

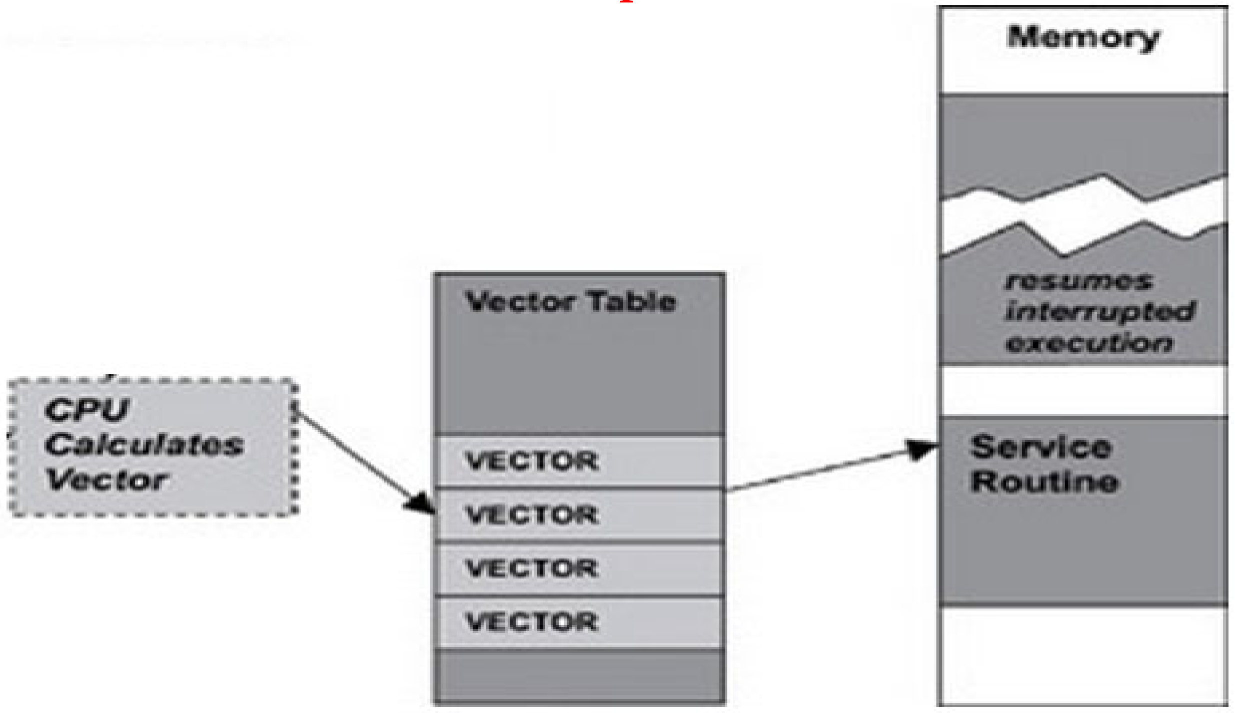
- For example, if you press a key on your keyboard, the keyboard then sets the keyboard interrupt to the CPU. Now the CPU executes the keypressed code (ISR).

Interrupt process



Interrupt cont…

- When an interrupt that is not masked (ignored) occurs, the CPU saves whatever it is currently doing and executes ISR. When ISR finished (with the iret opcode), the saved data gets restored and programs continues where it was paused by interrupt as nothing happened.
- For every interrupt, there is a fixed location in memory that holds the address of its ISR. The group of memory locations set aside to hold the addresses of ISRs is called the interrupt vector table.



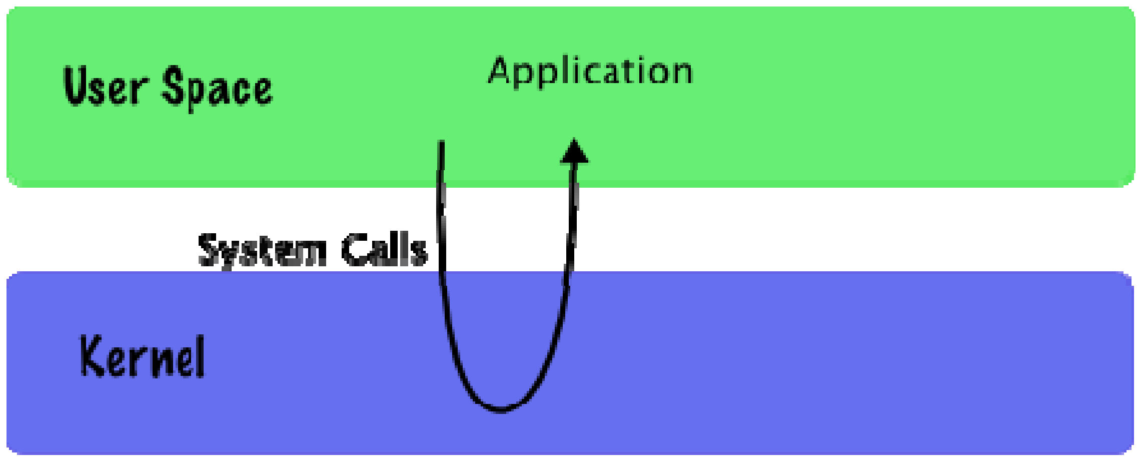
Interrupt cont…

- Interrupt can be emitted either by hardware or software indicating an event that needs immediate CPU attention. There are three types of interrupts:

Hardware	Software	Exception/Trap
Interrupt is sent from device to processor.	A special instruction in the instruction set which causes an interrupt when it is executed.	Exception/Trap is caused by an exceptional condition in the processor itself.

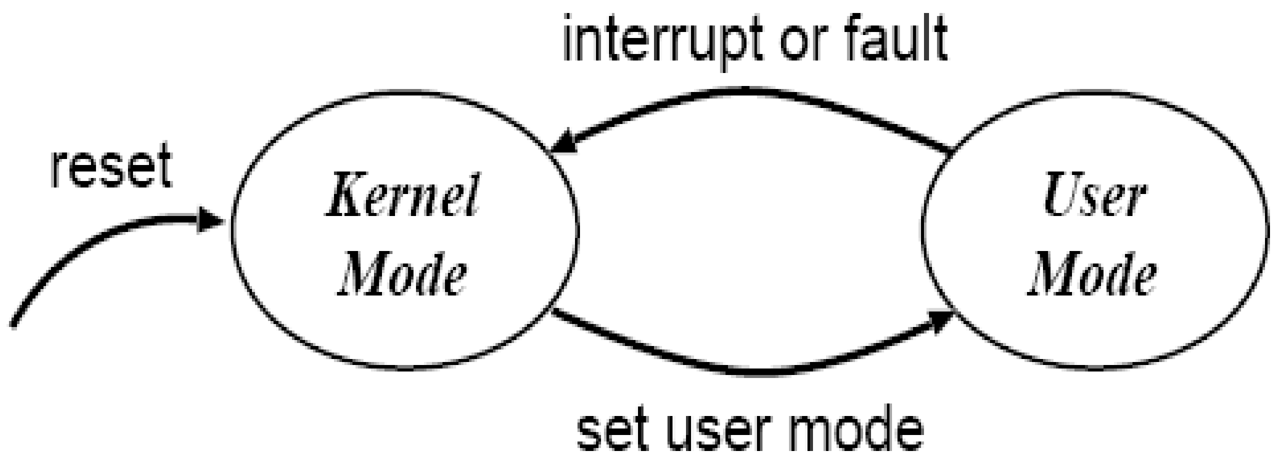
System call

- Software interrupts (Programmed Exceptions) are used to implement system calls.
- When an active process needs to ask for a service (for which it does not have permission itself) from the kernel (such as IO), it uses a system call. By a systems call, the user-space program tells the kernel to do something on behalf of it.



System call cont…

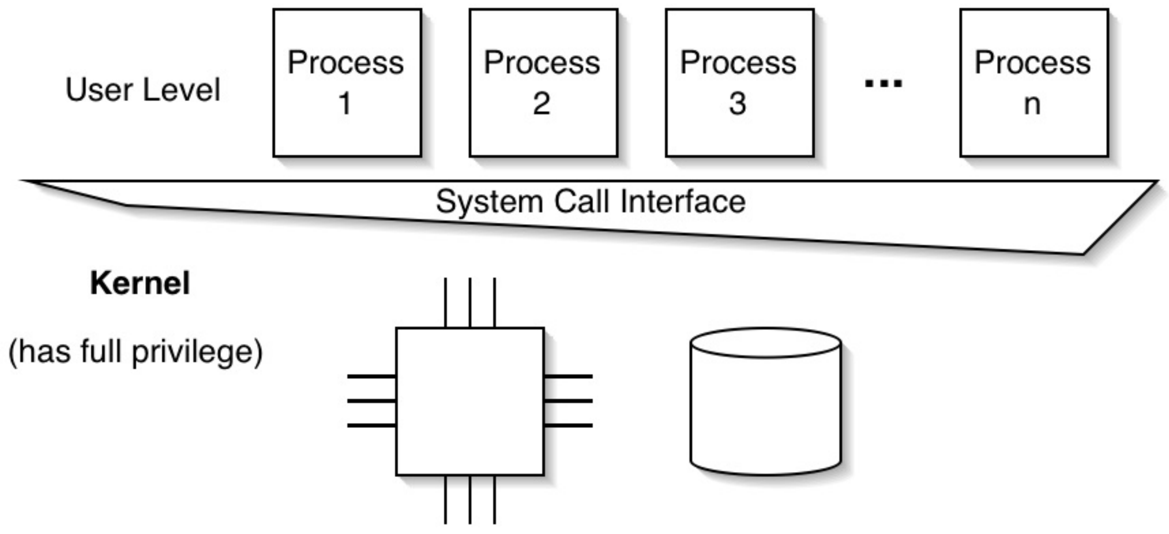
- The only way for an application in User Space to switch to Kernel Mode is to issue a **system call**.
  - When a user-mode program calls a system service, the CPU switches the calling program to kernel mode.
  - After the system service completed, the OS switches the program back to the user mode and allows the caller to continue.



System Call vs. Interrupt

- System call is a special command that a program uses to go from the user mode to kernel mode, while Interrupt is an event, which causes the processor to go to ISR.

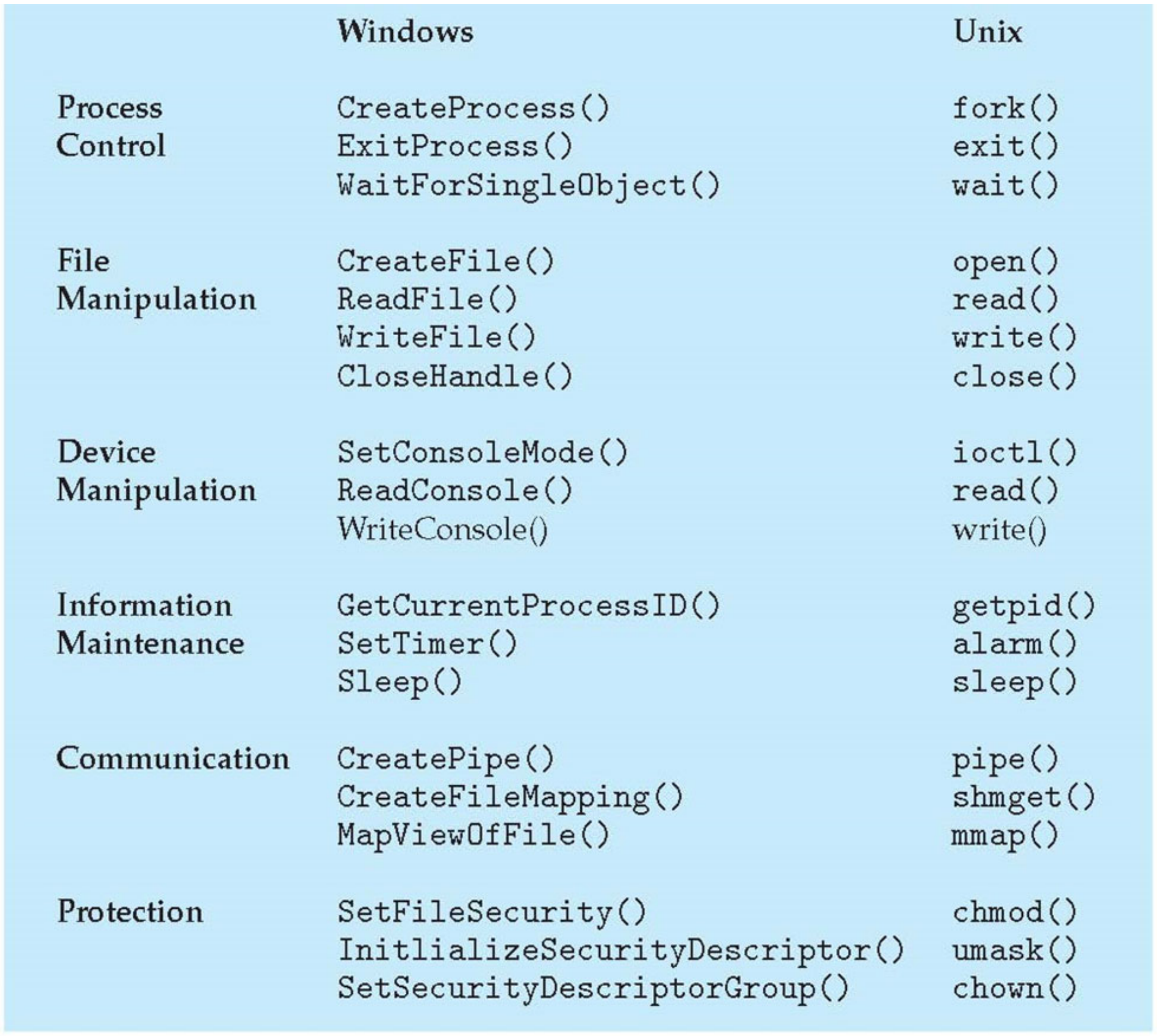
- A system call instruction is an instruction that generates an interrupt that cause the OS to gain control of the CPU.



Picture of Modern Operating System

Examples of Windows and Unix System Calls

- Types of System Calls : There are different categories of system calls



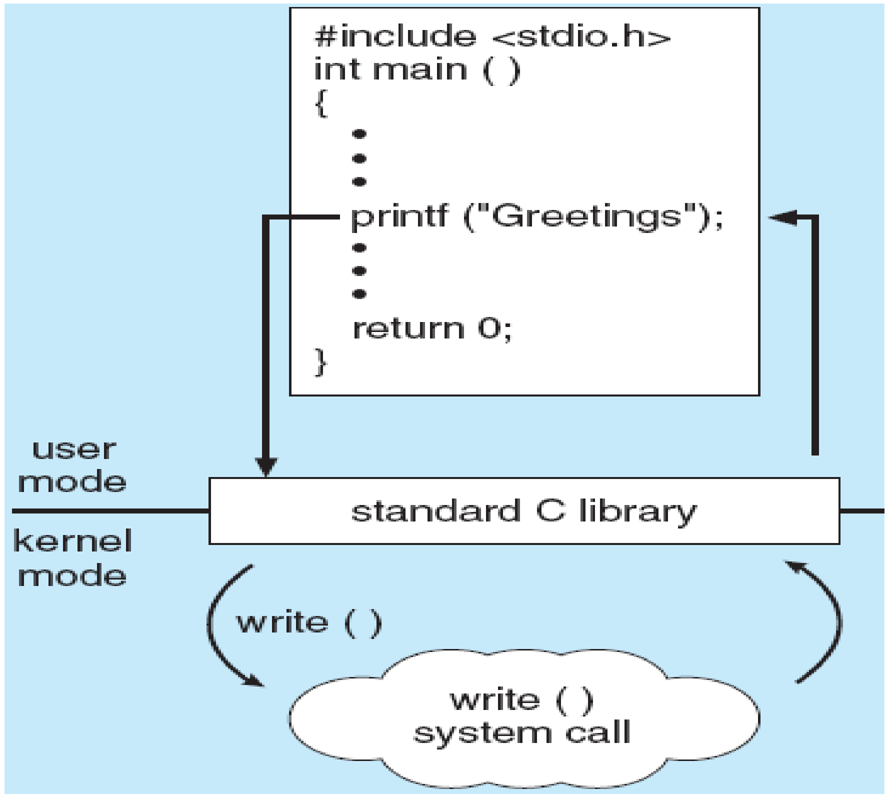
	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Communication	CreatePipe()	pipe()
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitlializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()

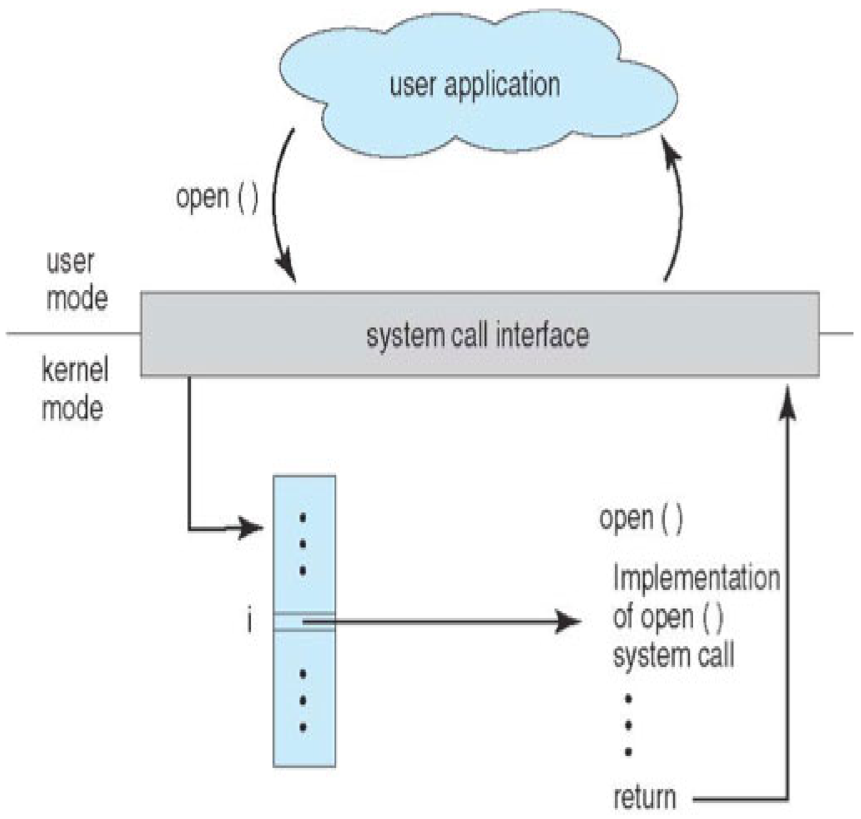
System call cont…

- Instead of directly use system call, usually programmers use a high-level Application Program Interface (API) to invoke system calls. The reasons for doing so:
  - Portability: System calls differ from OS to OS. Therefore when a programmer designs a program that he wants to run on any OS, using system call can cause the program does not work on a different OS. In this case, the programmer uses API which enables the program to compile and run on any system that supports the same API.
  - Complexity: actual system calls can often be more detailed and difficult to work with than the API available to an application programmer. Programmer would need specialized knowledge of I/O registers or order of operations to use them.

System call cont…

- For example a C program invoking printf() library call, which calls write() system call is shown as follow:





Policy vs. Mechanism

- Policies (what to do) and Mechanisms (how to do).
- To make the split between policy and mechanism clearer, let us consider two real-world examples: consider a restaurant.
- It has the mechanism for serving diners, including tables, plates, waiters, a kitchen full of equipment, agreements with credit card companies, and so on.
- The policy is set by the chef, namely, what is on the menu. If the chef decides that rice is out and soup is in, this new policy can be handled by the existing mechanism.

Operating system structure

- The basic idea to design OS is that its kernel codes execute in kernel mode and other codes execute in user mode.
- However, OSs are different in where this user-kernel boundary is drawn: what are the responsibilities of the kernel, and what code to put in it.
- One global principle is separating policies (what to do) and mechanisms (how to do). This separation is important to provide flexibility to a system; can change policy without changing mechanisms

Operating system structure cont…

- Based on the separation of mechanism and policy, there are different types of architecture for operating systems which lead to different kernel designs:
  - Monolithic/Simple systems
  - Layered systems
  - Microkernel/ Client-server
  - Modular kernel
  - Hybrid