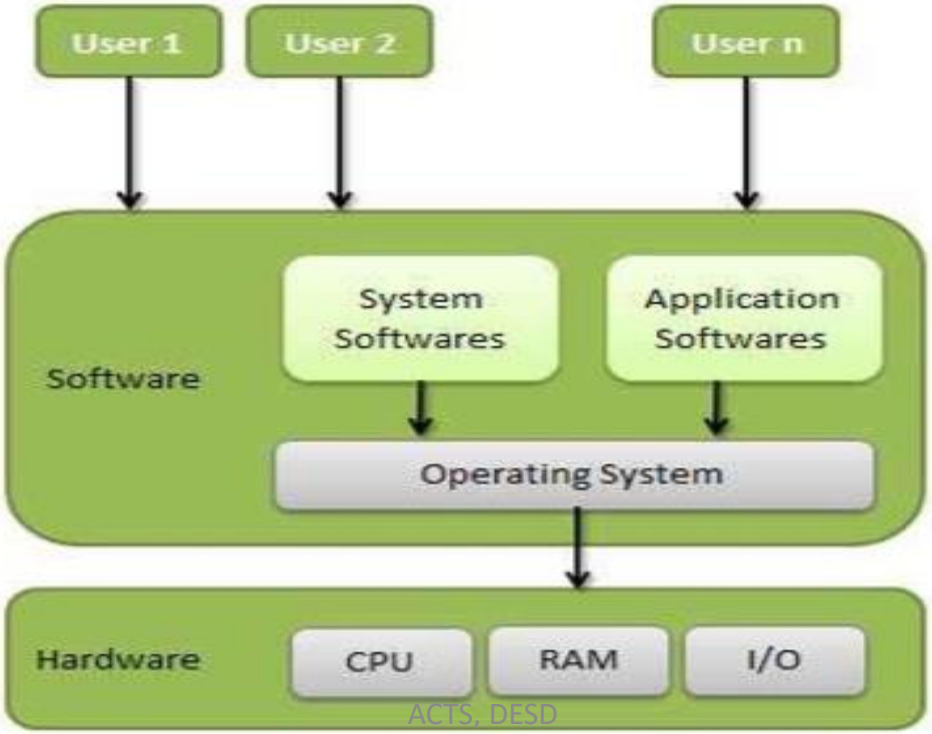
Operating System Overview

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently.

In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.

Definition

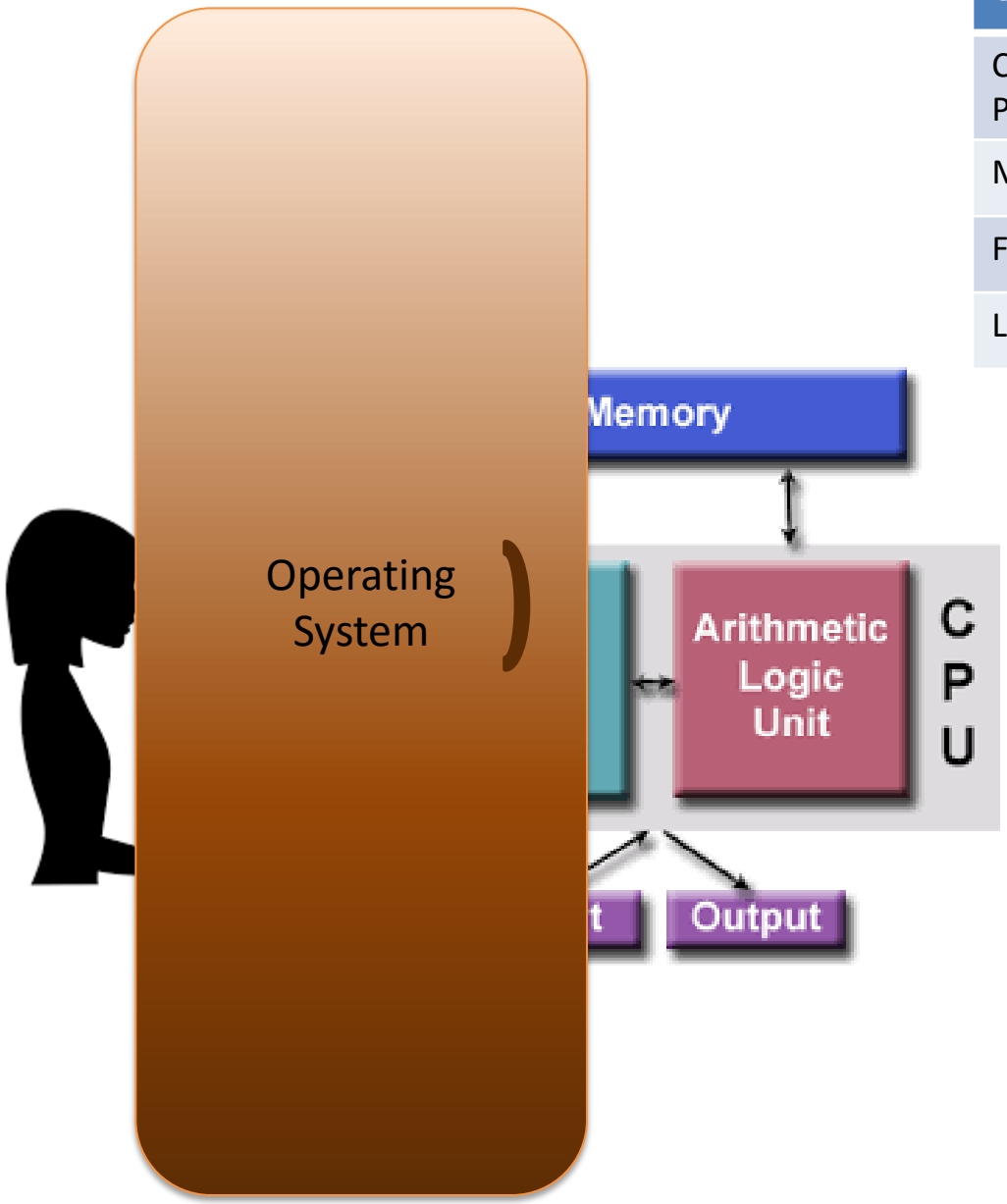
An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



Operating System

- A program or a software that governs the functioning of other programs
- Interface between User and the Hardware
- Hides details of different hardware configurations
  - Applications do not need to be tailored for each possible device present on the system
- Manages access to shared hardware resources
  - Enables multiple applications to share the same hardware simultaneously

Operating System Concepts
CPU Management/Scheduling, Process Management
Memory Management
File System Management
Linux Operating System

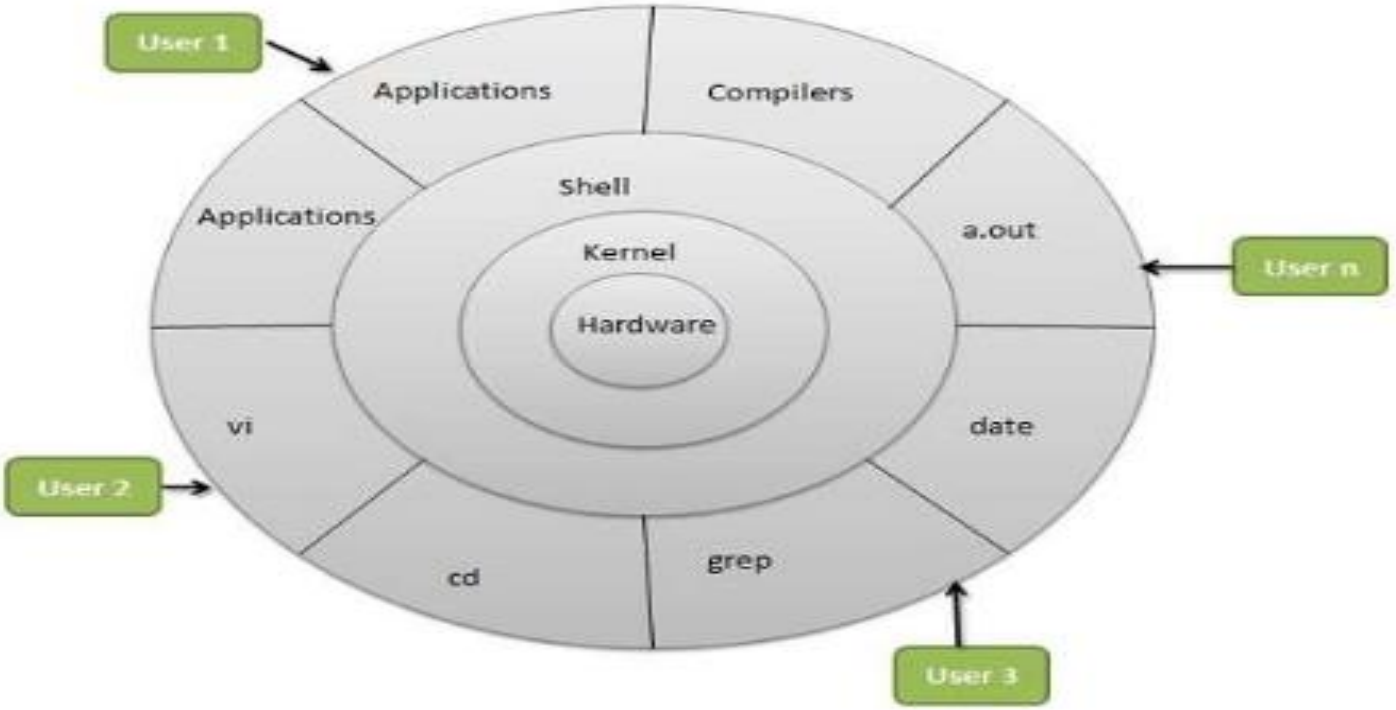


OS Functions

- Memory Management
- Process Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Architecture

- **Hardware layer** - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc.).
- **Kernel** - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** - An interface to kernel, hiding complexity of kernel's functions from users. Takes commands from user and executes kernel's functions.
- **Utilities** - Utility programs giving user most of the functionalities of an operating systems.



Shell

- Interface between the user and the kernel
- When a user logs in, the login program checks the username and password, and then starts another program called the shell
- Command Line Interpreter
- Unix Shells – Korn shell, bash shell, C shell, Bourne shell
- Filename Completion
- History

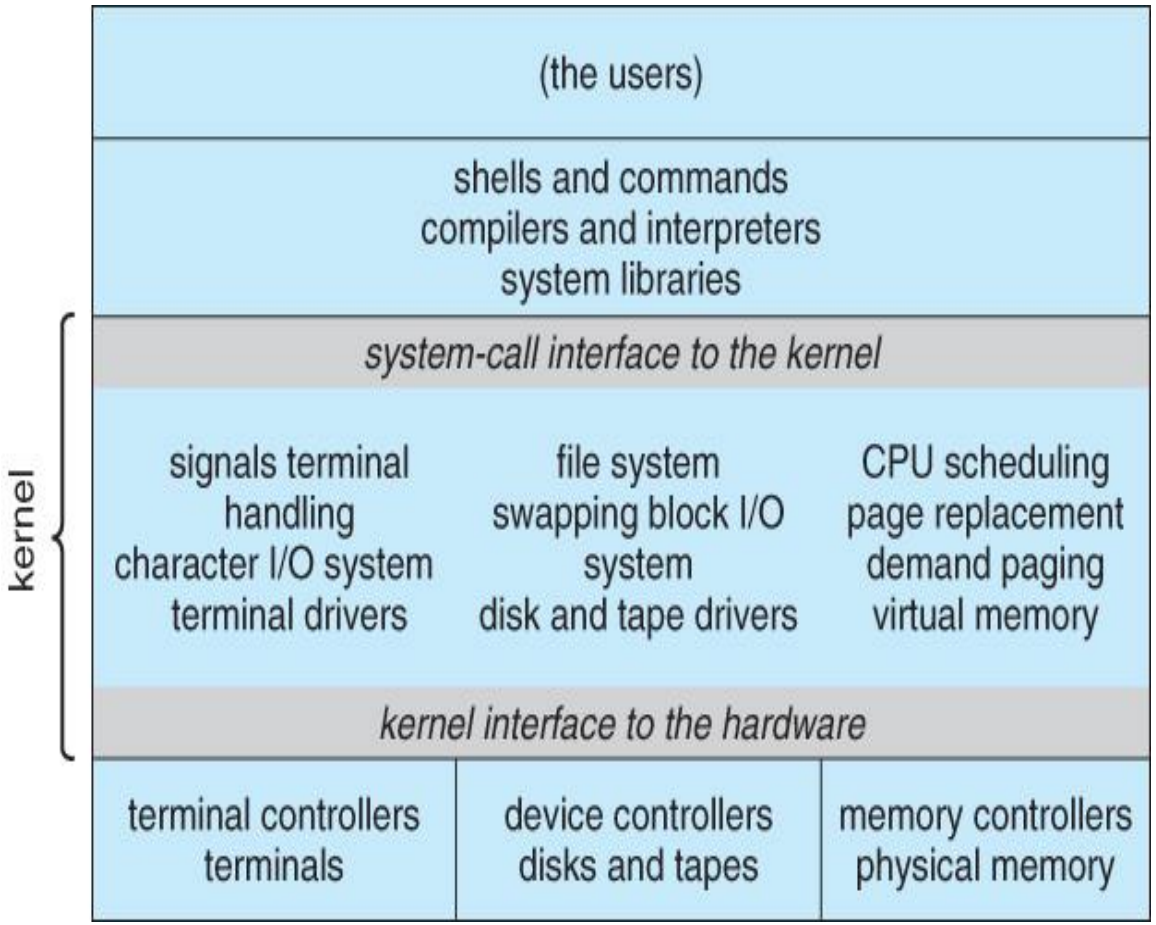
Kernel

- Core or nucleus of an operating system
- Portion of operating system that is in main memory
- Interacts with the hardware
- It allocates time and memory to programs and handles the filestore and communications in response to system calls
- Shell requests services of kernel through system calls
  - Suppose a user types rm myfile. The shell searches the filestore for the file containing the program rm, and then requests the kernel, through system calls, to execute the program rm on myfile. When the process rm myfile has finished running, the shell then returns the UNIX prompt % to the user, indicating that it is waiting for further commands

Kernel

- Unix has a clear distinction between ``**kernel space**'' and ``**user space**''.
- The kernel performs its tasks, such as executing processes and handling interrupts, in ***kernel space*** (higher privilege), whereas everything a user normally does, such as writing text in a text editor or running programs in a GUI (graphical user interface), is done in ***user space***. This separation prevents user data and kernel data from interfering with each other.
- **System calls** have been the means through which user space programs can access kernel services.
- When a process makes requests of the kernel, the request is called a system call.
- Kernel component code executes in a special **privileged mode** called **kernel mode** with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch.

- **Unix**


(the users)
shells and commands
compilers and interpreters
system libraries
system-call interface to the kernel
signals terminal handling
character I/O system
terminal drivers
file system
swapping block I/O system
disk and tape drivers
CPU scheduling
page replacement
demand paging
virtual memory
kernel interface to the hardware
terminal controllers
terminals
device controllers
disks and tapes
memory controllers
physical memory
kernel

User space vs Kernel Space

Various layers within Linux, also showing separation between the userland and kernel space

User mode	**User applications**	For example, bash, LibreOffice, Apache OpenOffice Blender, 0 A.D., Mozilla Firefox, etc.				
	Low-level system components:	**System daemons**: *systemd, runit, ConsoleKit, logind, networkd, soundd...*	**Windowing system**: *X11, Wayland, Mir, SurfaceFlinger (Android)*	**Other libraries:** *GLib, GTK+, Qt, EFL, SDL, SFML, FLTK, GNUstep*, etc.		**Graphics**: *Mesa 3D, AMD Catalyst, ...*
	C standard library	`open()`, `exec()`, `sbrk()`, `socket()`, `fopen()`, `calloc()`, ... (up to 2000 subroutines) *glibc* aims to be POSIX/SUS-compatible, *uClibc* targets embedded systems, *bionic* written for Android, etc.				
Kernel mode	**Linux kernel**	`stat`, `splice`, `dup`, `read`, `open`, `ioctl`, `write`, `mmap`, `close`, `exit`, etc. (about 380 system calls) The Linux kernel System Call Interface (SCI, aims to be POSIX/SUS-compatible)				
		Process scheduling subsystem	IPC subsystem	Memory management subsystem	Virtual files subsystem	Network subsystem
		Other components: ALSA, DRI, evdev, LVM, device mapper, Linux Network Scheduler, Netfilter Linux Security Modules: *SELinux, TOMOYO, AppArmor, Smack*				
Hardware (CPU, main memory, data storage devices, etc.)						

Courtesy : wikipedia

Common Commands

- **pwd**
- **cd *<dir>***
- **ls**
- **cat**
- **Echo**
- **man**
- **cp *<fromfile> <tofile>***
- **mv *<fromfile> <tofile>***
- **rm *<file>***
- **mkdir *<newdir>***
- **rmdir *<dir>***

Shell Script

```
# This script displays the date, time, username and
# current directory.
echo "Date and time is:"
date
echo
echo "Your username is: `whoami` \\n"
echo "Your current directory is: \\c"
pwd
```

alias rm='rm -i'

Shell Initialization Files

```
# /etc/profile
# System wide environment and startup programs, for login setup

PATH=$PATH:/usr/X11R6/bin

USER="`id -un`"
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"

HOSTNAME=`/bin/hostname`
HISTSIZE=1000

# Keyboard, bell, display style: the readline config file:
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
   INPUTRC=/etc/inputrc
fi

PS1="\u@\h \W"

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC PS1
```

Shell Initialization Files

```
#####################################################################
# # # .bash_profile file
# # # # Executed from the bash shell when you log in. # # #
#####################################################################
source ~/.bashrc
source ~/.bash_login
case "$OS" in
        IRIX) stty sane dec
        stty erase ;;
# SunOS)
#       stty erase
#       ;;
   *)
        stty sane
        ;;
esac
alias ll='ls -l'
alias rm='rm -i'
alias c='clear'
alias mroe='more'
```

OS Functions

- Memory Management
- Process Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management

Management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be access directly by the CPU. So for a program to be executed, it must in the main memory.

OS MM Functions:

- Keeps tracks of primary memory i.e. what part of it are in use by whom, what part are not in use.
- In multiprogramming, OS decides which process will get memory when and how much.
- Allocates the memory when the process requests it to do so.
- De-allocates the memory when the process no longer needs it or has been terminated

Process Management

In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling.

Operating System process management functions

- Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.
- Allocates the processor (CPU) to a process.
- De-allocates processor when processor is no longer required.

Device Management

OS manages device communication via their respective drivers.

Operating System device management

- Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices

File Management

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

Operating System file management

- Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Functions

- **Security --** By means of password and similar other techniques, preventing unauthorized access to programs and data.
- **Control over system performance --** Recording delays between request for a service and response from the system.
- **Job accounting --** Keeping track of time and resources used by various jobs and users.
- **Error detecting aids --** Production of dumps, traces, error messages and other debugging and error detecting aids.
- **Coordination between other software and users --** Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems

Types of OS

- Batch OS
- Single / Multi tasking
- Single / Multi User
- Time Sharing OS
- Distributed OS
- Network OS
- Real Time OS

Batch OS

- **The users of batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group**
- Lack of interaction between the user and job.
- CPU is often idle, because the speeds of the mechanical I/O devices are slower than CPU.
- Difficult to provide the desired priority

Single/Multi-Task & Single/Multi-User

- **Single Task** - run one program at a time, while a **multi-tasking** operating system allows more than one program to run concurrently by time-sharing
- **Single-user** - have no facilities to distinguish users, but may allow multiple programs to run in tandem
- **Multi-user** - extends the basic concept of multi-tasking with facilities that identify processes and resources, such as disk space, belonging to multiple users, and the system permits multiple users to interact with the system at the same time

Time Sharing OS

- **Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming.**
- **Time-Sharing Systems objective is to minimize response time**
- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication

Distributed OS

- Distributed systems use multiple central processors to serve multiple real time application and multiple users.
- The processors communicate with one another through various communication lines.
- Loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, and computers etc.
- With resource sharing facility user at one site may be able to use the resources available at another.
- If one site fails the remaining sites can continue operating.
- Reduction of the load on the host computer

Network OS

- **Network Operating System runs on a server and provides server the capability to manage data, users, groups, security, applications, and other networking functions.**
- **The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network**
- **E.g. Windows Server 2003/8, UNIX**

Advantages

- **Centralized servers are highly stable.**
- **Security is server managed.**
- **Upgrades are easy.**

Disadvantages

- **High cost of buying and running a server.**
- **Dependency on a central location for most operations.**
- **Regular maintenance and updates are required**

Real Time OS

- guarantees to process events or data within a certain short amount of time
- Rigid time requirements on the operation of a processor or the flow of data- otherwise system will fail
- E.g. Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-appliance controllers, Air traffic control system etc
- uses specialized scheduling algorithms so that a deterministic nature of behavior is achieved (in case of multi tasking)
- Event-driven RT system switches between tasks based on their priorities or external events while time-sharing OS switch tasks based on clock interrupts
- **Hard real-time** systems guarantee that critical tasks complete on time. In hard real-time systems secondary storage is limited or missing with data stored in ROM. In these systems virtual memory is almost never found
- **Soft real time** systems are less restrictive. Critical real-time task gets priority over other tasks. E.g. Multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers etc

Embedded OS

- Embedded operating systems are designed to be used in embedded computer systems.
- Designed to operate on small machines like PDAs.
- They are able to operate with a limited number of resources.
- They are very compact and extremely efficient by design.
- Windows CE and Minix 3 are some examples of embedded operating systems.

OS Functionality

Program Execution

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a passive entity, process is an active entity.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Allocation of resources when process is created or when process is running
  - process also requires initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one program counter specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, and operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- **OS provides uniform, logical view of information storage**
  - **Abstracts physical properties to logical storage unit - file**
  - **Each medium is controlled by device (i.e., disk drive, tape drive)**
    - **Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)**
- **A file is a collection of related information defined by its creator**
  - **represent programs and data**
- **File-System management**
  - **Files usually organized into directories**
  - **Access control on most systems to determine who can access what**
- Creating and deleting files and directories
- Primitives to manipulate files and directories
- Mapping files onto secondary storage
- Backup files onto stable (non-volatile) storage media

I/O Operation

- **I/O subsystem comprised of I/O devices and their corresponding driver software. Drivers hides the peculiarities of specific hardware devices from the user as the device driver knows the peculiarities of the specific device. Operating System manages the communication between user and device drivers.**

- I/O operation means read or write operation with any file or any specific I/O device.
- Program may require any I/O device while running.
- Operating system provides the access to the required I/O device when required.

Thank You