OS CN

Operating System (OS)

Definition

- A collection of programs to control/manage computer systems.
- Acts as an intermediary between the user and computer components.

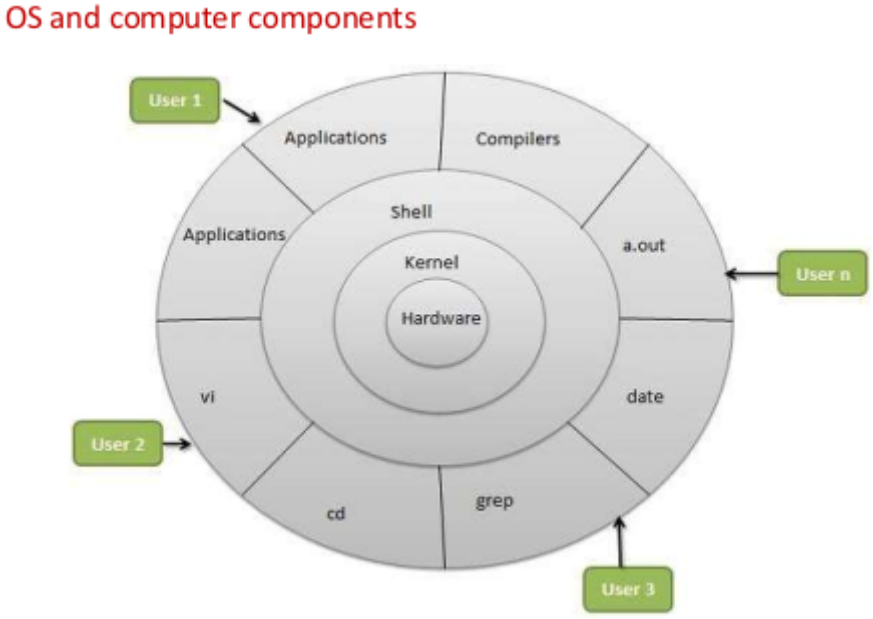
Goals

- Execute user programs and solve user problems.
- Make the computer system convenient to use.
- Use hardware efficiently.

Computer System Components

1. **Hardware** : CPU, memory, I/O devices.
2. **Operating System** : Controls hardware use among applications.
3. **Applications Programs** : Use APIs to interact with the OS.
4. **Users** : People, machines, other computers.

Components of OS



- **Bootloader** : Manages the boot process (e.g., ntldr, BOOTMGR, LILO, GRUB).
- **Kernel** : Core of the OS, manages CPU, memory, and devices.
- **Daemons** : Background services (e.g., printing, sound, network services).
- **Shell** : Command interpreter interface between kernel and user applications.
- **Applications** : Software to perform specific tasks.

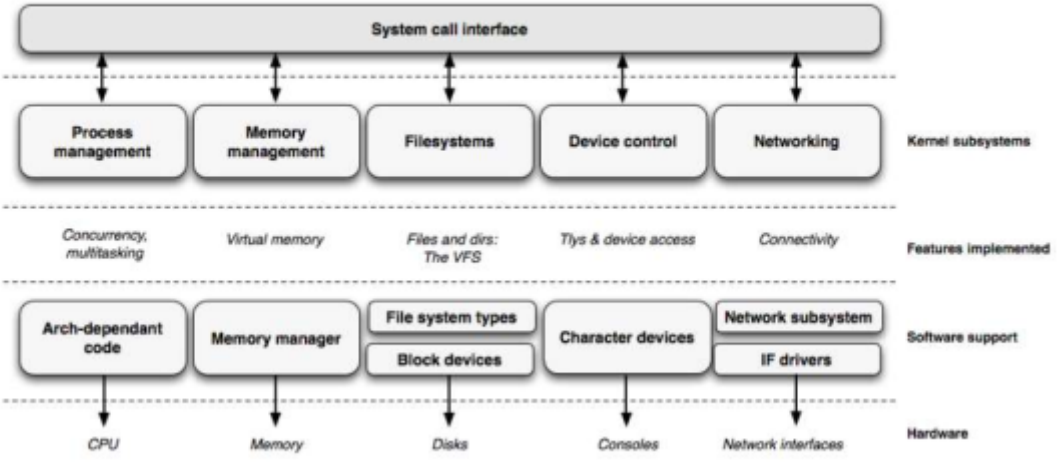
The Kernel

Definition

- Central core of an OS, first program loaded into memory during booting.
- Interacts with the shell, programs, and hardware.

Tasks/Components

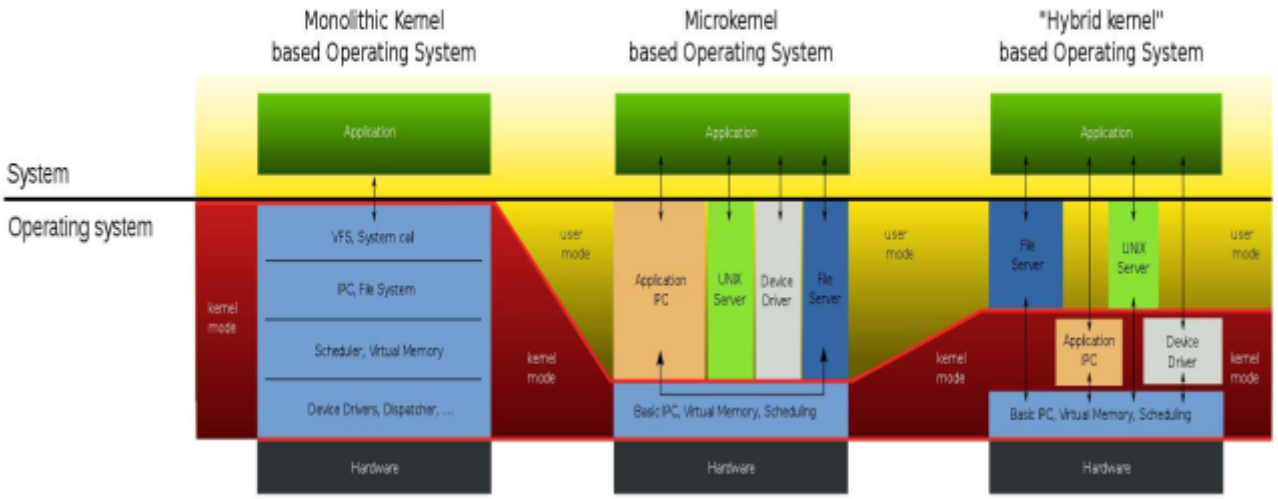
Kernel layout



- **Process Management** : Manages application execution.
- **Device Management** : Controls access to peripherals.
- **Memory Management** : Allocates memory to processes.
- **Interrupt Handling, I/O Communication, File System** : Other critical functions.

Categories of Kernels

Different kernels in comparison



1. **Monolithic Kernels**
    - Entire kernel operates in kernel space.
    - Unix kernels (BSD, Linux, Solaris), DOS, Windows 9x, OpenVMS.
2. **Microkernels**
    - Minimal set of functions, other functions run on top.
    - AmigaOS, MINIX, Symbian.
3. **Hybrid Kernels**
    - Combination of microkernel and monolithic kernel.
    - Windows NT, 2000, XP, Vista, 7, DragonFly BSD.
4. **Exo Kernels**
    - Limited to protection and multiplexing of raw hardware.
    - Allows programmers to choose what level of abstraction they want, such as one for Linux and one for Microsoft Windows, thus making it possible to simultaneously run both Linux and Windows applications
    - MIT's Aegis, XOK.

Batch Processing

- Executes a series of programs without human interaction.
- **Benefits**

1. Shares computer resources among many users.
2. Processes jobs during less busy times.
3. Maximizes utilization of expensive resources.
4. Historically used with mainframe computers.

Multiprogramming Batch System

- Executes one job from memory and switches to another during I/O operations.
- Always keeps CPU and OS busy.
- Uses CPU Scheduling to choose jobs to run.
- Ensures CPU never sits idle.

Time-Sharing Systems

- Extension of multiprogramming systems.
- Shares computing resources among many users simultaneously.
- Focuses on minimizing response time.
- Allocates a small time slice to each user.

Distributed Operating System

- Utilizes powerful microprocessors and advanced communication technology.
- Comprises many interconnected computers.
- Client/server architecture allows shared database access.
- **Advantages**
    1. Resource sharing.
    2. Fast processing.
    3. Load distribution.
    4. Reliability.
- Requires networking infrastructure (LAN/WAN).

Client-Server Systems

- Server systems satisfy client requests.
- Compute Servers execute actions and return results.
- File Servers manage file operations.

Peer-to-Peer Systems

- Consist of processors with local memory communicating through lines like high-speed buses.
- Loosely coupled systems.

Desktop Systems

- Personal computers dedicated to single users.
- Run various operating systems (Windows, MacOS, UNIX, Linux).

Parallel Systems

- Multiprocessor systems with shared memory and clock.
- Increased throughput, economical, and reliable.
- Symmetric multiprocessing (SMP) and asymmetric multiprocessing.

Clustered Systems

- Multiple systems share storage.
- Provides high reliability.
- Types
    1. *Asymmetric Clustering* : One machine in standby mode.
    2. *Symmetric Clustering* : Multiple hosts run applications and monitor each other.
    3. *Parallel Clustering* : Multiple hosts access shared storage.

Real-Time Systems

- Serve real-time applications with minimal delay.
- Types
    1. Hard Real Time : Strict deadlines.
    2. Firm Real Time : Deadlines with some tolerance.
    3. Soft Real Time : Deadlines with acceptable delays.
- Features
    - Low memory usage.
    - Predictable response times.
- Applications
    - Airline reservations
    - Traffic control
    - Defence systems, etc.

Handheld Systems

- PDAs and cellular phones.

- Limited memory, slow processors, small screens.
- Often lack virtual memory techniques.

Operating System Components

1. Signals and system calls
2. Process management
3. Memory management
4. File subsystem
5. Device drivers

1 Signals and System Calls

- Signals : Event notifications sent to a process.
- System calls : Interfaces provided by programming languages to access OS functions.

Multiprogramming

- Multiple instances of a program can be loaded in memory.
- Concurrent execution : Multiple processes executed over time.

2 Process Management

What is a Process?

- A program in execution.
- Foreground and background processes.
- Process control block (PCB) : Contains descriptive information about a process.
- Task - Job - Thread

Process Creation

- Created by system initialization, user request, batch job, or other processes.
- Uses `fork()` or `clone()` in UNIX/Linux.
- Threads : Sub-processes within a process.

Process Architecture

- Address space divided into : Code(text) , data, heap, and stack segments.
- code - compiled program code (non-volatile)
- data - global & static variables
- heap - dynamic memory allocation
- stack - local variables

3 Memory Management

- Controls and coordinates memory blocks.
- Physical vs. virtual memory.
- Allocation techniques: Single contiguous, partitioned, paged, and segmented memory management.

Memory Allocation

- Low Memory : For the OS.
- High Memory : For user processes.

Partition Allocation

- Schemes : First fit, best fit, worst fit, next fit.

Memory Management sub-system

- **Dynamic Loading** : Loading parts of a program as needed.
- **Dynamic Linking** : Linking dependent programs at runtime.

4 File Subsystem

- Files : Collection of information on non-volatile storage.
- File systems manage file content and metadata.
- Types : Native, virtual (VFS), pseudo (procfs, sysfs).
- **File Attributes**
  - Include name, identifier, location, type, size, protection, time, date, and security.
- **File Types**
  - Character special files, ordinary files, directory files, special device files.
- **File Functions**
  - create, write, read, delete, and reposition.

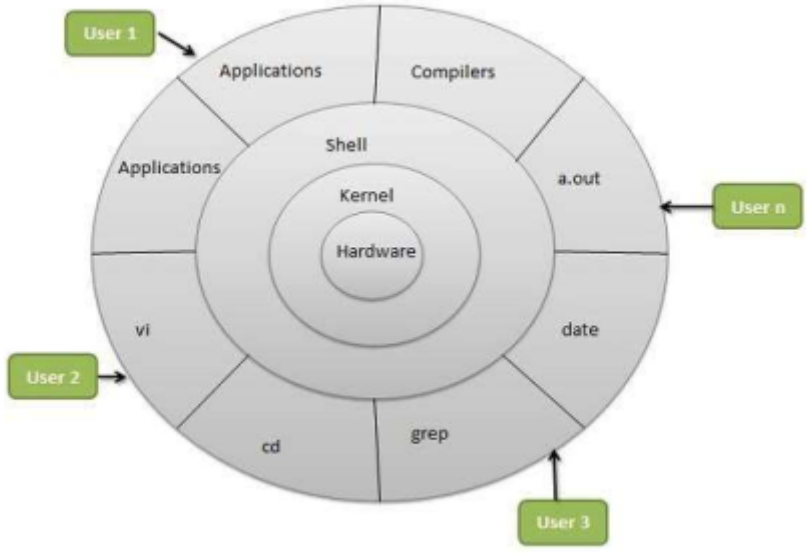
- **File Access Methods**
  - Sequential, direct random, and index sequential access.

5 Device Drivers

- Control peripheral devices.
- Types : Character, block, network device drivers.

1. Character
   - perform I/O in a byte stream
   - keyboard, mouse, printer
2. Block
   - supports FS
   - Tape drive, HDD, CDROM, ROM, RAM
3. Network
   - receive and transmit data packets on h/w interface
   - HUB, Switches , Bridges

OS and computer components



What is Linux?

- Best-known open source OS.
- "GNU Linux" includes Linux kernel + programs, tools, services.
- Examples : Android, Ubuntu, Fedora, openSUSE, Debian, Mandriva.

History of Linux

- Started in 1991 by Linus Torvalds.
- Frustration with MINIX licensing led to Linux kernel development.
- Initially called "Freax"; renamed to "Linux" by Ari Lemmke.
- Linux kernel first published under its own license, later GNU GPL.

Differences from Other OS

- Open source software.
- Many customizable distributions.
- Core components can be swapped out.

Unix vs. Linux

- Unix : Developed in the 1970s at Bell Labs.
- Linux : Created to be similar to Unix but is open source and more popular.

Ownership of Linux

- "Linux" refers to the kernel.
- Commonly called GNU/Linux.
- Examples: Android uses Linux kernel, few GNU tools.

Why Linux?

- Multi-user/multitasking/timesharing.
- Portability
- Modularity
- File structure
- Security.
- Strong networking support
- Advanced graphics.

Uses of Linux

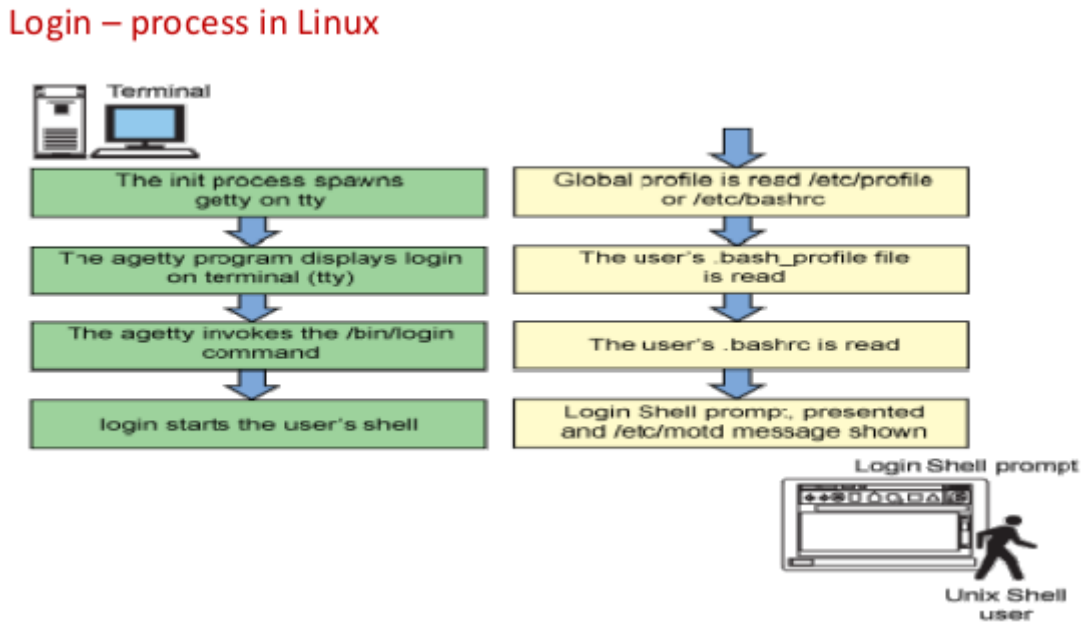
- Companies : Google, Twitter, Facebook, Amazon, IBM.
- Devices : Watches, Mobile, Space, Desktops, NASA, Nuclear Projects, Bullet Trains.

Best Linux Distributions

- Desktop : Ubuntu.
- Laptop : openSUSE.
- Enterprise Desktop : SUSE Linux Enterprise Desktop.
- Enterprise Server : SUSE Linux Enterprise Server, Red Hat Enterprise Linux.
- LiveCD : KNOPPIX.
- Security : SELinux.
- Multimedia : Ubuntu Studio.

Login and Logout Process

- Login : Start using the shell.



- Logout : Use `exit` or `logout` command to stop processes.

Man Pages

`• $man [section-num] [command/tool name]`

Explanation of different sections in the manual pages

- Section 1 : Executable programs or shell commands.
- Section 2 : System calls provided by the kernel.
- Section 3 : Library calls within program libraries.
- Section 4 : Special files typically found in `/dev`.
- Section 5 : File formats and conventions (e.g., `/etc/passwd`).

- Section 6 : Games.
- Section 7 : Miscellaneous information, including macro packages and conventions.
- Section 8 : System administration commands (usually for root).
- Section 9 : Kernel routines (non-standard).
- a : display all manual pages for a given input
- f : print short descriptions
- k : make man search considering input as regular expression
- w : display location of man files

Booting Linux

---

1. **BIOS POST**
   - Initial hardware checks and locating boot sectors.
   - boot sector is first stage of boot loader (GRUB, GRUB2, and LILO)
2. **Kernel**
   - Bootloader loads the kernel from `/boot` to RAM
   - Kernal loads systemd
3. **systemd**
   - Initializes the system.
   - 1st process to run on Linux
   - systemd mounts the filesystems as defined by `/etc/fstab`
   - Uses `/etc/systemd/system/default.target` to determine boot target.
4. **Targets and Runlevels**
   - Runlevels mapped to systemd targets.
   - `runlevel3.target` -> `multi-user.target`
   - `runlevel5.target` -> `graphical.target`

SystemV Runlevels	systemd target	systemd target aliases	Description
	halt.target		Halts the system without powering it down.
0	poweroff.target	runlevel0.target	Halts the system and turns the power off.
S	emergency.target		Single user mode. No services are running; filesystems are not mounted. This is the most basic level of operation with only an emergency shell running on the main console for the user to interact with the system.
1	rescue.target	runlevel1.target	A base system including mounting the filesystems with only the most basic services running and a rescue shell on the main console.
2		runlevel2.target	Multiuser, without NFS but all other non-GUI services running.
3	multi-user.target	runlevel3.target	All services running but command line interface (CLI) only.
4		runlevel4.target	Unused.
5	graphical.target	runlevel5.target	multi-user with a GUI.
6	reboot.target	runlevel6.target	Reboot
	default.target		This target is always aliased with a symbolic link to either multi-user.target or graphical.target. systemd always uses the default.target to start the system. The default.target should never be aliased to halt.target, poweroff.target, or reboot.target.

Shell Commands

---

1. **Login/Logout** : `login` , `logout` , or `exit` .
2. **Changing Password** : `passwd`
3. **Terminal Type** : Set appropriate terminal type (e.g., `vt100` , `xterms`).
4. **Using Online Help** : `man`
5. **Types of Shells**
   - Bourne (sys V)
   - Korn (Solaris)
   - 'C' shell (BSD)
   - Bash shell (Linux)
6. **Bash Shell**
   - Command line editing
   - Unlimited command history
   - Job control
   - Shell functions and aliases
   - Indexed array of unlimited size
7. **Startup Files**
   - System-wide: `/etc/profile` , `/etc/bash_profile`
   - User-specific: `~/.bash_profile` , `~/.bash_login` , `~/.profile`
8. **Built-in Commands** - Examples: `cd` , `echo` , `exit` , `break` , `pwd` , `declare` , `help`
9. **Variables**
   - Global - exported variables
   - Local - user defined

- Creating &export variables :
  - `var_name = value`
  - `export var_name`
- Environment variables : `env`
  - DISPLAY
  - EDITOR
  - GROUP
  - HOME
  - IFS - Internal Field Seperators
  - LOGNAME
  - PATH

Basic Commands

---

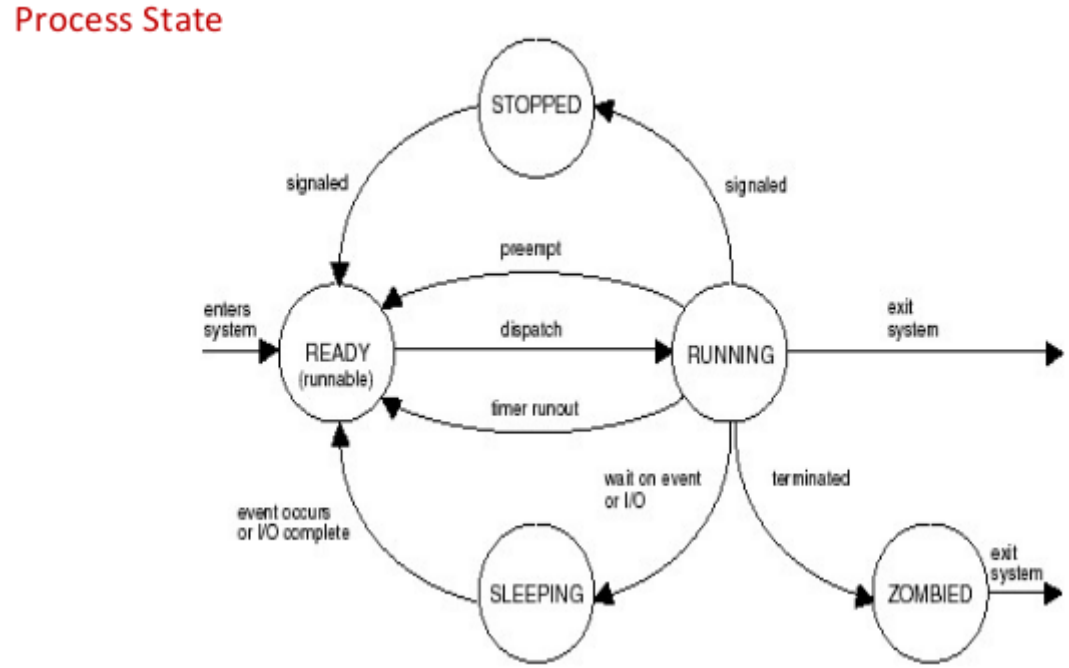
1. **who**
   - Displays logged-in users.
   - `who -q`
   - `who -H`
   - `who -a`
2. **w**
   - Displays user activity.
   - `w`, `w -u`, `w username`
   - USER – User name.
   - TTY – Terminal type such as pts/0 or console.
   - FROM – The remote host name or IP address.
   - LOGIN@ – Login time.
   - IDLE – Idel time.
   - JCPU – The JCPU time is the time used by all processes attached to the tty.
   - PCPU – The PCPU time is the time used by the current process displayed in WHAT field.
   - WHAT – The command line of USER’s current process.
3. **whoami** : Displays the current user.
4. **date** : Displays/sets system date and time.
5. **cal** : Displays calendar.
6. **whatis** : Displays brief information about commands.

Process Creation

- **System Calls :** `fork`, `spawn`
- **Parent and Child**
  - Parent process creates child process
  - Resource sharing
  - Execution
  - Address space
- **Identifiers :** PID, PPID
- **Init Process :** PID 1, parent of all processes

Process State

- **States :** Ready, Running, Wait/Sleep, Stopped, Zombie



Process Scheduling

- **Scheduling :**
  - Determines process execution order
  - CPU Scheduler
- **Dispatcher**
  - Gives control of the CPU to the selected process
  - context switching
  - switching to user mode
  - jump in user program
- **Categories**
  - Non-preemptive

    - Preemptive
- **Criteria**
    - CPU utilization : keep CPU busy
    - Throughput : no of process completed per time
    - Turnaround time : time to execute a process
    - Waiting time : waiting in ready queue
    - Response time : time from a request submission to 1st response

Scheduling Algorithms

- **FCFS**
    - First-Come, First-Served
    - Poor performance
    - High avg-wait time
- **SJF**
    - Shortest-Job-First (Non-preemptive, Preemptive)
    - Optimal - min avg-wait time
- **Priority Scheduling**
    - Based on priority number, with aging to prevent starvation
    - starvation - low priority ps may never execute
    - aging - as time progresses increase the priority of ps
- **RR :** Round Robin with fixed time slices

Commands

- **ps :** Show current processes
- **pstree :** Display process tree
- **bg :** Move suspended process to background
- **fg :** Bring background job to foreground
- **jobs :** List background/foreground jobs
- **top :** Real-time system view
- **kill :** Terminate a process

What is a Thread?

- An execution unit with its own program counter, stack, and set of registers.
- Also known as lightweight processes.
- Used to improve application performance through parallelism.

- CPU switches rapidly among threads, creating an illusion of parallel execution.

Single Threading

- Processing one command at a time.
- Opposite of multithreading.
- In functional programming, can mean "backtracking within a single thread."

Multithreading

- Found in multitasking operating systems.
- Allows multiple threads within a single process to share resources but execute independently.
- Provides concurrent execution abstraction.
- Enables parallel execution on multiprocessor systems.

Advantages of Multithreaded Applications

- **Responsiveness**
  - Keeps applications responsive by offloading long tasks to worker threads.
- **Faster Execution**
  - Operates faster on multi-core processors or clusters.
- **Lower Resource Consumption**
  - Uses fewer resources compared to multiple process copies.
- **Better System Utilization**
  - Higher throughput and lower latency.
- **Simplified Sharing and Communication**
  - Threads share data, code, and files.
- **Parallelization**
  - Utilizes multicore systems for parallel task execution.

Drawbacks of Multithreading

- **Synchronization Issues**
  - Risk of race conditions and deadlocks.
- **Process Crashes**
  - A misbehaving thread can crash the entire process.

Multiple Threads

- Multiple processes can execute in parallel by increasing the number of threads.

Thread Credentials

- **Operations**
  - creation
  - termination
  - synchronization
  - scheduling
  - data management
  - process interaction
- **Shared Resources**
  - instructions
  - open files
  - signals
  - working directory
  - user and group ID
- **Unique Attributes**
  - Thread ID
  - registers
  - stack pointer
  - local variables
  - signal mask
  - priority, and scheduling policy.

Threads vs. Processes

- **Address Space**
  - Threads share the same address space
  - Processes do not share same address space
- **Synchronization**
  - Process synchronization - Kernal
  - Thread synchronization - Process
- **Context Switching**
  - Faster between threads than between processes
- **Communication**
  - Easier among threads than processes.

Types of Threads

- **User Threads**
  - Managed by user-level thread libraries without kernel support.
  - Examples: POSIX P threads, Mach C-threads, Solaris threads.
- **Kernel Threads**
  - Managed by the kernel
  - Supported by all modern OSs.

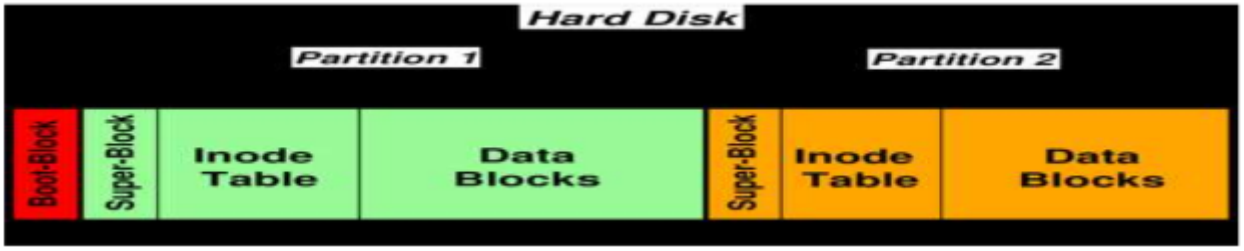
What are filesystems?

- File system is the way files are organized on the disk.
- A filesystem is the methods and data structures that an OS uses to keep track of files on a disk or partition.
- Different kinds of file systems vary in structure, logic, speed, flexibility, security, size, etc.
- Common Linux file systems: ext2, ext3, ext4, XFS, JFS, btrfs, VFS, ReiserFS

Ext, Ext2, Ext3, Ext4, JFS, XFS, btrfs and swap

- **Ext** : Old and no longer used due to limitations.
- **Ext2** : First Linux FS allowing 2TB data.
- **Ext3** : Ext2 with upgrades, lacks file recovery or snapshots.
- **Ext4** : Faster, supports larger files.
- **JFS** : Made by IBM, works with small and big files, prone to corruption over time.
- **XFS** : Slow with small files, great with large files.
- **Btrfs** : Made by Oracle, good performance, still under development.

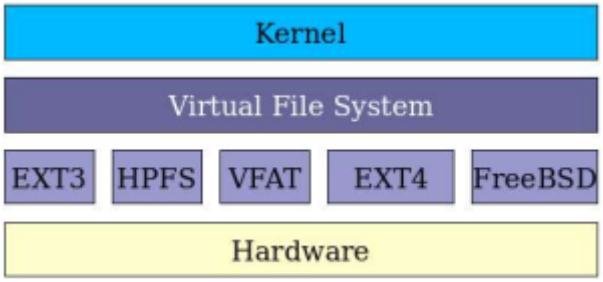
The file system layout



- **Boot block** : Contains bootstrap code for booting.
- **Super block** : Describes the state of the FS (size, max files, free space).
- **Inode list** : Contains the inode table, used by the kernel to get file info.

- **Data block** : Stores user files, administrative files at the start.

Virtual File System (VFS)



- Provides a single set of commands for the kernel to access all types of filesystems.
- VFS software calls specific device drivers to interface with various filesystems.

Ext2 / ext3 / ext4 file system

- **Boot sector** : Includes a small boot record and partition table.
- **Superblock** : Contains metadata defining filesystem structures.
  - Total inode count
  - Total block count
  - Free block count
  - Free inode count
  - Blocks per group
  - Inodes per group
  - Mount time
  - Number of mounts since last fsck

i-node block

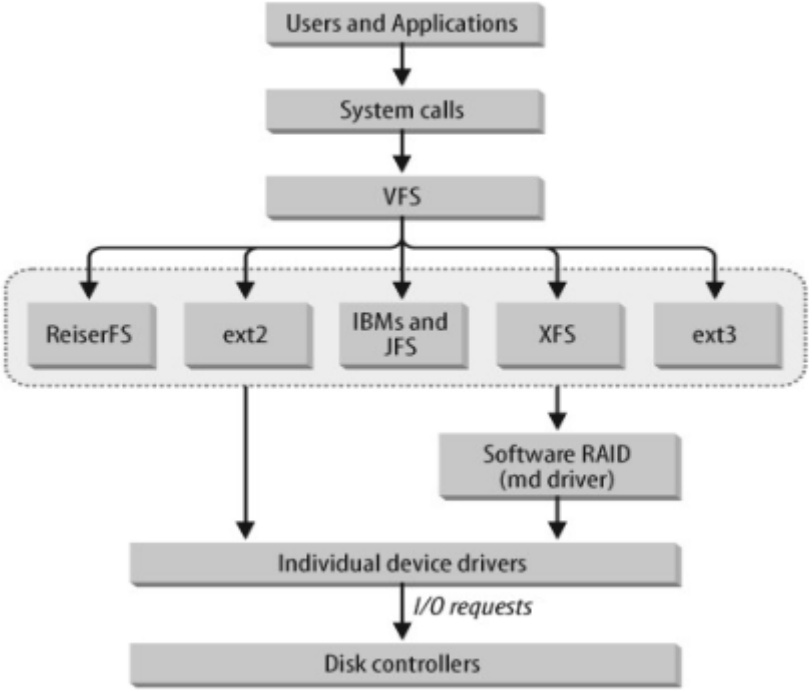
- Contains numbers of data blocks for storing file data.
- **File mode** : Determines file type and access rights.
- **Link count** : Number of hard links pointing to the inode.
- **User ID and Group ID** : Owner identifiers.
- **Device ID** : If the file is a device file.

- **File size** : In bytes.
- **Timestamps** : ctime, mtime, atime.
- **Preferred I/O block size**.
- **Number of blocks allocated**.

Accessing data block

- Inode does not contain the file name.
- Access to a file is via the directory entry, which contains the file name and a pointer to the inode.

VFS



- Manages all mounted file systems.
- Keeps data structures describing the virtual file system and real mounted file systems.
- Each filesystem registers itself with the VFS during OS initialization.
- VFS inodes describe files and directories.
- Frequently accessed inodes are kept in the inode cache for quicker access.
- VFS superblock contains device identifier, inode pointers, block size, and superblock operations.

Mounting a File System

- The superuser validates arguments passed in the system call.

- VFS searches through known filesystems by looking at the `file_system_type` data structures.
- Example mount command: `mount -t iso9660 -o ro /dev/cdrom /mnt/cdrom`

Data structures

- When a process opens a file the following informations are maintained by the kernel.
- These structures are maintained in the kernel tree `/usrc/src/linux/include/linux`
- **Task_struct** (sched.h) points to **Files_struct** (fdtable.h).
- **Files_struct** points to **struct file** (fs.h).
- **struct file** points to **struct path** (path.h).
- **struct path** points to **struct dentry** (dcache.h).
- **struct dentry** points to **struct inode** (fs.h).

File System Basics

- **File System :** Controls how data is stored and retrieved.
- **Aspects**
  - space management
  - filenames
  - directories
  - metadata
  - abstract user interface

File Naming Conventions

- Avoid special characters (@#$%^&*).
- Case sensitive.
- Avoid spaces use underscores
- **Paths:**
  - Absolute: `/home/user/file.txt`
  - Relative: `file.txt`

Important Directories

- **/home :** User home directories.
- **/bin :** Executable files.
- **/sbin :** System executables.
- **/lib :** Essential libraries.

- **/etc :** System configuration files.
- **/dev :** Device files.
- **/mnt :** Mount points for external file systems.
- **/tmp :** Temporary files.
- **/var :** Variable data files.
- **/root**
- **/usr**
- **/usr/bin**
- **/sys**
- **/proc :** information about kernel components

Common Commands

---

pwd - Print Working Directory

- Displays current directory.
  - Example: `$ pwd` -> `/home/user`

cd - Change Directory

- Changes the current directory.
  - Examples:
  - `cd /` (root directory)
  - `cd ..` (one level up)
  - `cd ~/folder` (user's folder)

ls - List Directory Contents

- Lists files and directories.
  - Options:
  - `-l` : Long format
  - `-a` : All files (including hidden)
  - `-i` : list inode no of file in first column
  - `-s` : reports disk blocks occupied by file
  - `-R` : Recursive
  - `-F` : Classify files by type
  - `-C` : display files in columns

cat - Concatenate and Display Files

- Displays file content.
  - Examples:
  - `cat file1` (display file1)
  - `cat > file1` (create/overwrite file1)
  - `cat >> file1` (append to file1)

cp - Copy Files

- Copies files or directories.
  - Examples:
  - `cp file1 file2` (copy file1 to file2)
  - `cp -r dir1 dir2` (copy directory)

mv - Move/Rename Files

- Moves or renames files.
  - Example: `mv file1 file2` (rename file1 to file2)

rm - Remove Files

- Deletes files or directories.
  - Options:
  - `-r`: Recursive (directories)
  - `-i`: Interactive (confirm deletion)

mkdir - Create Directory

- Creates directories.
  - Examples:
  - `mkdir folder1`
  - `mkdir -p parent/child` (create parent and child directories)

rmdir - Remove Directory

- Deletes empty directories.
  - Example: `rmdir folder1`

more & less - View File Content

- View content page by page.
  - Controls: `space` (next page), `q` (quit)

wc - Word Count

- Counts lines, words, and characters.
  - Options:
  - `-l`: Lines
  - `-w`: Words
  - `-c`: Characters

ln - Link Files

- Creates hard or soft links.
- Hard link
  - Same inode
  - Can't cross file systems
  - Survives original deletion
  - There is link count maintained by the file system
  - Only one data copy
  - `ln file1 file2`
- Soft link
  - Different inode
  - Can cross file systems
  - Can't survive original deletion
  - There is no link count maintained by the file system
  - Only one data copy
  - `ln -s file1 file2`

Key Commands and Utilities

---

cmp

- Compares two files byte by byte
- If a difference is found, it reports the byte and line number where the first difference is found

- `cmp file1.txt file2.txt`

```sh
cmp file1.txt file2.txt
```

```
### diff
- Shows differences between files
- `-b` ignore trailing blanks
- `-i` ignore the case of letters
- `-w` ignore space and tab characters
- `-r` apply diff recursively through common sub-directories

- `diff [options] file1 file2`

  ```sh
  diff -i file1.txt file2.txt
  ```

file
- Determines the file type

- `file [options] filename`

  ```sh
  file -i file1.txt
  ```

cut
- Removes sections from each line of files
- `-b` select only these bytes
- `-c` select only these characters
- `-d` use DELIM instead of TAB for field delimiter
- `-f` select only these fields

- `cut [OPTIONS] [FILE]`

  ```sh
  cut -d ":" -f1,2 /etc/passwd
  ```

paste
- Merges files line by line
```

- `paste [options] file1 file2`

  ```sh
  paste file1.txt file2.txt
  ```

touch
- Creates an empty file or updates the timestamp of an existing file

- `touch [options] [date_time] file`

  ```sh
  touch newfile.txt
  ```

sort
- Sorts contents of text files based on 1st char of each line

- `sort [options] [file]`

  ```sh
  sort -k2 file.txt
  ```

uniq
- Filters out or reports repeated lines in a file \
- `-c` prefix lines by the number of occurrences
- `-d` only print duplicate lines, one for each group
- `-D` print all duplicate lines
- `-f` avoid comparing the first N fields

- `uniq [options] [file]`

  ```sh
  uniq -c file.txt
  ```

tr
- Translates or deletes characters

- `tr [options] SET1 SET2`

  ```sh
  echo "hello" | tr 'a-z' 'A-Z'
  ```

File Compression & Archiving
---
compress
- Compresses files, creating a `.Z` extension

- `compress [options] file`

  ```sh
  compress file.txt
  ```

uncompress
- Decompresses files compressed by `compress`

- `uncompress [options] file.Z`

  ```sh
  uncompress file.txt.Z
  ```

gzip
- Compresses files, creating a `.gz` extension

- `gzip [options] filename`

  ```sh
  gzip file.txt
  ```

gunzip
- Decompresses `.gz` files

- `gunzip [options] filename.gz`

  ```sh
  gunzip file.txt.gz
  ```

```
### tar
- tape archive
- Archives files into a single file

- **Options**
  - `-c` create a new archive
  - `-t` list the contents of an archive
  - `-x` extract the contents of an archive

 - - `tar [options] [file(s)]`

  ```sh
  tar -cvzf archive.tar.gz *.txt
  ```

df
command to summarize disk block and file usage
size
du
command to summarize disk block and file usage

Commands and Examples

File Access Permissions

Read (r)
- List the contents of directory
- Remove the directory
Write (w)
- copy files to the directory
- remove files from the directory
- rename files in the directory
- make a subdirectory
- remove a subdirectory from the directory
- move files to and from the directory
Execute (x)
- display the contents of a directory file from within the directory
- change to the directory
- display a file in the directory
- copy a file to or from the directory

chmod
Set file access permissions:
```

```bash
chmod nnn [file]
chmod [who]op[perm] [file]
```

Example:

```bash
chmod 755 file1
chmod u=rwx,go=rx file1
```

umask

Set default file permissions : `/etc/profile`

```bash
umask 022
```

chown

Change file ownership

```bash
chown [options] user[.group] file
```

Example:

```bash
chown new_owner file
```

chgrp

Change group ownership

```bash
chgrp [options] group file
```

Example:

```bash
chgrp new_group file
```

File Searching Commands

---

find

Search for files

```
find directory [search options] [actions]
```

Examples:

```
find / -name passwd -print
find . -newer library -print
find . -name "*ar*" -ls
```

grep

- Global Regular Expression Printer
- Search within files using regular expressions

```
grep [options] pattern [file(s)]
egrep [options] pattern [file(s)]
fgrep [options] pattern [file(s)]
```

Examples:

- `-i` : ignore case
- `-c` : outs the count of no of lines containing matches
- `-v` : invert the search, display lines that do not match
- `-n` : display the line number along with the line on which a match was found

```
grep '15' num.list
grep -c '15' num.list
grep '^ ' num.list
grep '^[^ ]' num.list
grep '^[1-9]' num.list
```

whatis

Get brief information about commands

```
whatis [options] command
```

Example:

```
whatis write
whatis -s "2" open
whatis -w 'ab*'
```

whereis

Locate binary `-b`, source `-s`, and man `-m` pages of commands

```
whereis [options] command
```

Example:

```
whereis open
whereis -b whereis
```

which

Locate executables in the system

```
which [options] command
```

Example:

```
which ls gdb open grep
```

Basic Linux Commands & Examples

`su`

- Substitute User / Switch User
- Switch user or become superuser during a login session

```
su - sanjay
```

```
### `sudo`
- Super User DO
```

```
- Provides administrative access without sharing the root password
- Example : Precede an admin command with `sudo`

### Configuring `sudo` Access
1. Log in as root
2. Create a user with `useradd`
3. Set a password with `passwd`
4. Edit `/etc/sudoers` with `visudo`
5. Enable sudo for the wheel group by uncommenting the line in the file
6. Save changes and exit

### Mounting File Systems
- Mount syntax : `mount -t type device mount-point`
- View mounted partitions : `mount`

```bash
mount -t vfat /dev/hda1 /mnt
```

`dmesg`

- Viewing Kernel Logs
- List loaded drivers : `dmesg`
- Clear buffer logs : `dmesg -c`
- Use with text-manipulation tools : `dmesg | grep sda`

Linux Network Configuration and Troubleshooting Commands

`ifconfig`

- Interface Configurator
- Initialize interface, assign IP, enable/disable interface

```
ifconfig eth0 192.168.50.5 netmask 255.255.255.0
```

`ping`

- Packet INternet Groper
- Test connectivity using ICMP (Internet Control Message Protocol)
```

- `-c` to limit requests

```
ping www.google.com
ping -c 5 www.google.com
```

`netstat`

- Network Statistic
- Display connection info, routing table

```
netstat -r
```

`hostname`

- Identify the system in a network
- Set permanently in `/etc/sysconfig/network`

`host`

- Find name to IP or IP to name in IPv4 or IPv6 and also query DNS records
- Query DNS records with `-t` option to find out DNS Resource Records like CNAME, NS, MX, SOA

```
host wwww.google.com
host -t CNAME www.redhat.com
```

`uname`

- Print system information
- `-a` print all information
- `-s` kernel-name
- `-n` network node hostname
- `-r` kernel-release
- `-v` kernel-version
- `-m` machine hardware name
- `-p` processor : print the processor type or "unknown"
- `-i` hardware-platform

`telnet`

- Connect to a remote Linux computer

```
telnet hostname
telnet ip-address
```

`ssh`

- Secured Shell
- Securely connect to a remote computer

```
ssh user@hostname
```

`scp`

- Secure Copy
- Copy files between hosts securely on a n/w
- Use ssh for data transfer

```
scp -r SourceDirectory/ user@hostname:/path/
```

Tools

`putty`

- PuTTY is a terminal emulator application which can act as a client for the SSH, Telnet, rlogin, and raw TCP
- Stores hosts and preferences
- Control over the SSH encryption key and protocol version
- IPv6 support

`winscp`

- Windows Secure Copy
- Open-source SecureFTP client for Windows
- Secure file transfers between local and remote server

- Uses SSH for security

Visual Editor

- Visual editor for entering and editing text files
- Screen-oriented text editor included with most UNIX system distributions
- Command driven

Categories of Commands

- General administration
- Cursor movement
- Insert text
- Delete text
- Paste text
- Modify text

Editing Commands

Invoking Vi Editor

- To edit a file : `vi [filename]`
- To recover an editing session : `vi -r [filename]`

Text Insertion / Replacement

- `i` - inserts text to the left of the cursor
- `a` - inserts text to the right of the cursor
- `I` - inserts text at the beginning of the line
- `A` - appends text at end of the line
- `o` - opens line below
- `O` - opens line above
- `R` - replaces text from cursor to right
- `s` - replaces a single character with any number of characters
- `S` - replaces entire line

Vi Commands

Exiting Vi Editor

- `:x<Return>` - quit vi, writing modified file
- `:wq<Return>` - quit vi, writing modified file
- `:q<Return>` - quit vi
- `:q!<Return>` - quit vi without saving changes

Moving the Cursor

- `j` or `<Return>` (down-arrow) - move cursor down one line
- `k` (up-arrow) - move cursor up one line
- `h` or `<Backspace>` (left-arrow) - move cursor left one character
- `l` or `<Space>` (right-arrow) - move cursor right one character
- `0` (zero) - move cursor to start of current line
- `$` - move cursor to end of current line

Deletion Commands

- `x` - delete character at cursor position
- `3x` - delete 3 characters at cursor position
- `dw` - delete word
- `2dw` - delete 2 words
- `dd` - delete a line
- `2dd` - delete 2 lines

Yanking Commands

- `Y` - copy line into buffer
- `3Y` - copy 3 lines into buffer
- `p` - paste buffer below cursor
- `P` - paste buffer above cursor

Save and Quit

- `:w` - save
- `:w!` - save as `:w! filename`
- `:x` - save and quit
- `:q` - cancel changes
- `:q!` - cancel and quit

Screen Manipulation Commands

Saving Files

- `^f` - move forward one screen
- `^b` - move backward one screen
- `^d` - move down (forward) one half screen
- `^u` - move up (back) one half screen
- `^l` - redraw screen
- `^r` - redraw screen, removing deleted lines

Write Commands

- `:w<Return>` - write current contents to file
- `:w newfile<Return>` - write current contents to a new file named newfile
- `:12,35w smallfile<Return>` - write contents of lines 12-35 to a new file named smallfile

Search & Replace Commands

Searching Through Text

- `?pattern` : Backward for pattern
- `/pattern` : Forward for pattern
- `n` : Repeat previous search
- `N` : Reverse direction of previous search
- `:beg,end g/pattern/p` : Show all lines containing pattern
- `:1,$g/compiler/p` : prints all lines with the pattern "compiler"

Substitute Patterns

- `:%s/notfound/found/g` - change all occurrences of "notfound" to "found"

Useful ex Commands

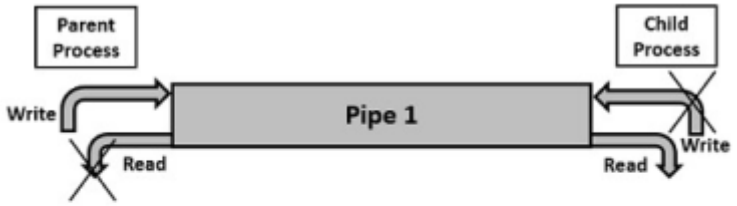
- `:set all` - Display all Set options
- `:set autoindent` - Automatically indent following lines
- `:set ignorecase` - Ignore case during pattern matching
- `:set list` - Show special characters in the file
- `:set number` - Display line numbers
- `:set shiftwidth=n` - Set width for shifting operators `<<` and `>>`
- `:set showmode` - Display mode when in Insert, Append, or Replace mode
- `:set wrapmargin=n` - Set right margin for auto-wrapping lines (0 turns it off)

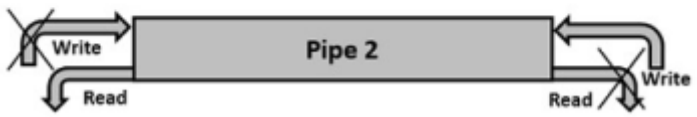
IPC Mechanisms

IPC is essential for exchanging data between processes in a multitasking OS

1 Pipe (Unnamed Pipe)

- Communication between related processes (siblings of a process)
- **Half-duplex** : Unidirectional data flow
- PIPEs are created by the pipe()
- Local to the system
- Read & Write done at same time
- No control over ownership & permissions
- Vanishes as soon as it is closed or one of the processes completes execution





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

int main() {
    int fd[2];
    pipe(fd);
    // fd[0] for reading, fd[1] for writing
    return 0;
}
```

2 FIFO (Named Pipe)

- Communication between unrelated processes
- Exists in the filesystem
- Bidirectional data flow
- mknod() to create FIFO in C
- Capable of communicating across different computers & n/ws
- Read & Write doesn't require to be occur at same time
- Ownership & permissions control
- FIFO remain till system reboots

```c
/* reader Process */
if(mknod(FIFO_FILE, S_IFIFO|0666, 0) != 0)
        perror("FIFO creation error in server\n");
if((fp = fopen(FIFO_FILE, "r")) == NULL)
        perror("FIFO openning error\n");
fgets(readbuf, 80, fp);
printf("Server program Received string: %s\n", readbuf);
fclose(fp);
unlink(FIFO_FILE); // to remove fifo

/* Writer Process */
FILE *fp;
if((fp = fopen(FIFO_FILE, "w")) == NULL)
        perror("FIFO openning error\n");
fputs(argv[1], fp);
fclose(fp);
```

3 Message Queue

- Sending and receiving messages
- Messages read by type

- Not restricted to FIFO order

```
#include <sys/msg.h>
int msqid = msgget(key, 0666 | IPC_CREAT);
msgsnd(msqid, &msg, sizeof(msg), 0);
msgrcv(msqid, &msg, sizeof(msg), 0, 0);
```

4 Semaphore

- Mutex and Semaphore
- Resource that contains an int value
- Allows ps to synchronize by testing and setting the value
- Coordinate access to resources
- Counting and signaling mechanism
- **Operations**
  - P (wait) : sem_wait() - decrement : lock
  - V (signal) : sem_post() - increment : unlock

```
#include <semaphore.h>
int main() {
        sem_t *mySemaphore;
        int count = 0;
        sem_init(mySemaphore, 0, 1);
        sem_wait(mySemaphore);
        count++;
        sem_post(mySemaphore);
        sem_destroy(mySemaphore);
        return 0;
}
```

5 Shared Memory

- Share data between multiple processes
- **Steps**
  1. Key generation (`ftok`)
  2. Memory segment creation (`shmget`)

```
int shmget(key_t,size_tsize,intshmflg)
```

```
  3. Attachment (`shmat`)
  ```bash
  void *shmat(int shmid ,void *shmaddr ,int shmflg)
```

4. Detachment (`shmdt`)

```
int shmdt(void *shmaddr)
```

5. Destroy (`shmctl`)

```
shmctl(int shmid,IPC_RMID,NULL)
```

```c
#include <sys/ipc.h>
#include <sys/shm.h>
int shmid = shmget(key, size, 0666 | IPC_CREAT);
void *shared_memory = shmat(shmid, NULL, 0);
shmdt(shared_memory);
shmctl(shmid, IPC_RMID, NULL);
```

System V IPC

- Derived from UNIX System V release 4
- This mechanism includes : Message Queues, Semaphores, Shared Memory
- `-q` : Show only message queuesipcs
- `-s` : Show only semaphoresipcs
- `-m` : Show only shared memoryipcs

RPM (Red Hat Package Manager)

- Manages software packages on Linux systems
- **Installation**

```
rpm -i package.rpm
```

- **Upgrade**

```
rpm -Uvh package.rpm
```
```

- **Querying Packages**
  - To find out information about an RPM package
  - Listing Files in an RPM

```
rpm -ql package
rpm -qlp package.rpm
rpm -qf /usr/bin/file
```

- **List Installed Packages**

```
rpm -qa
```

- **Remove Packages**

```
rpm -e package
```

yum (Yellowdog Updater Modified)

- Interactive, RPM-based package manager\
- It can automatically perform system updates, including dependency analysis and obsolete processing
- **Install a Package**

```
yum install package
```

- **Remove a Package**

```
yum remove package
```

- **Update a Package**

```
yum update package
```

- **List a Package**

```
yum list package
```

- **Search for a Package**

```
yum search package
```

- **Get Information on a Package**

```
yum info package
```

APT (Advanced Package Tool)

- Manages Debian-based distributions like Ubuntu
- `apt-get` installing, upgrading, and cleaning packages
- `apt-cache` is used for finding new packages
- **Install a Package**

```
sudo apt-get install package
```

- **Update Package Database**
  - `hit` no change in the package version
  - `ign` the package is being ignored
  - `get` new version of the package available

```
sudo apt-get update
```

- **Upgrade Packages**

```
sudo apt-get upgrade
```

- **Search for a Package**

```
apt-cache search package
```

- **Show Package Info**

```
apt-cache show package
```

- **Remove a Package**

```
sudo apt-get remove package
sudo apt-get purge package
sudo apt-get clean
sudo apt-get autoclean
```

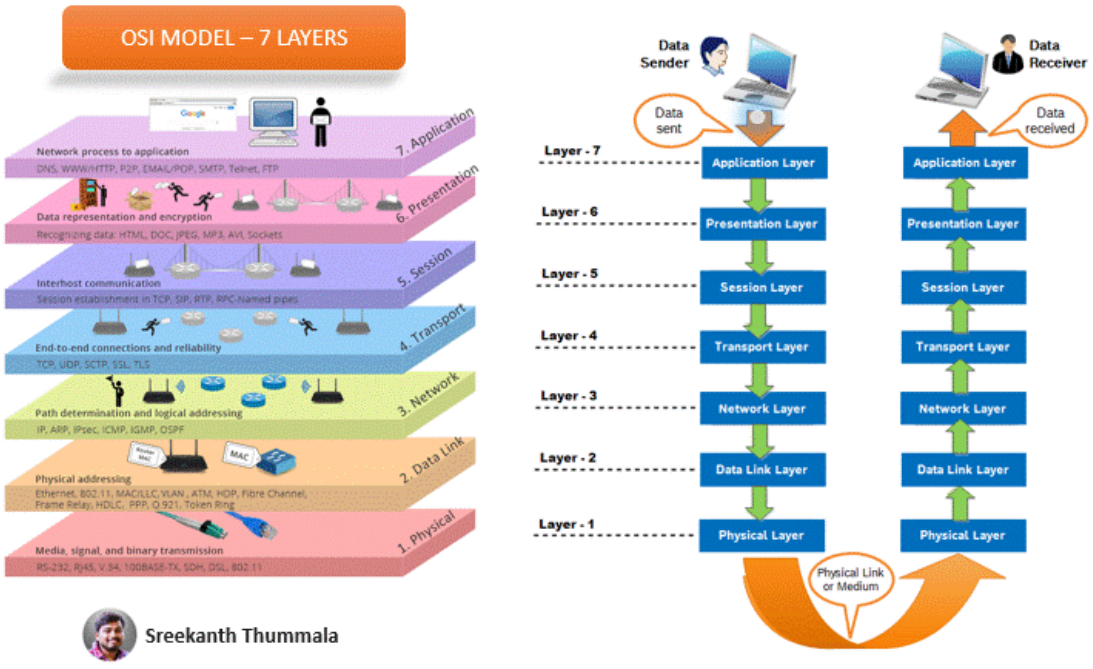
apt-get vs apt

- `apt` is more structured
- Provides necessary options needed to manage packages

```
apt help
sudo apt install glances
sudo apt content glances
sudo apt depends glances
sudo apt search apache2
sudo apt show firefox
sudo apt check firefox
sudo apt version firefox
sudo apt update
sudo apt upgrade
sudo apt autoremove
sudo apt autoclean
sudo apt clean
sudo apt purge glances
```

OSI Model

- The OSI (Open Systems Interconnection) model is a conceptual framework used to understand and implement network protocols in seven distinct layers
- Each layer serves a specific function and interacts with the layers directly above and below it
- **OSI Model**
  - 1 Physical - data cables, cat6
  - 2 Datalink - switching, MAC addresses
  - 3 N/W - IP, routing
  - 4 Transport - TCP/UDP
  - 5 Session - session management
  - 6 Presentation - WMV, JPEG, MOV
  - 7 Application - HTTP, SMTP



1. Physical Layer (Layer 1)

- **Function:** Responsible for the transmission and reception of raw bitstreams over a physical medium (e.g., cables, switches).
- **Data Unit:** Bits
- **Devices:** Hubs, cables, and repeaters
- **Protocols:** Ethernet (at the physical level), RS-232

2. Data Link Layer (Layer 2)

- **Function:** Provides node-to-node data transfer—a link between two directly connected nodes. It also handles error detection and correction from the physical layer.
- **Data Unit:** Frames
- **Devices:** Switches, bridges
- **Sub-layers:**
  - **MAC (Media Access Control):** Controls how devices in a network gain access to a medium and permission to transmit data.
  - **LLC (Logical Link Control):** Handles error checking and frame synchronization.
- **Protocols:** Ethernet, PPP (Point-to-Point Protocol), HDLC (High-Level Data Link Control)

3. Network Layer (Layer 3)

- **Function:** Manages the routing of data between devices across different networks and handles packet forwarding including routing through different routers.
- **Data Unit:** Packets
- **Devices:** Routers
- **Protocols:** IP (Internet Protocol), ICMP (Internet Control Message Protocol), ARP (Address Resolution Protocol)

4. Transport Layer (Layer 4)

- **Function:** Provides reliable data transfer services to the upper layers. It ensures complete data transfer with error checking and recovery. It also manages flow control and data segmentation.
- **Data Unit:** Segments (TCP), Datagrams (UDP)
- **Devices:** Gateways, firewalls (at a basic level)
- **Protocols:** TCP (Transmission Control Protocol), UDP (User Datagram Protocol)

5. Session Layer (Layer 5)

- **Function:** Manages sessions or connections between two applications. It establishes, maintains, and terminates connections. It also handles session checkpointing and recovery.
- **Data Unit:** Data
- **Protocols:** NetBIOS, PPTP (Point-to-Point Tunneling Protocol)

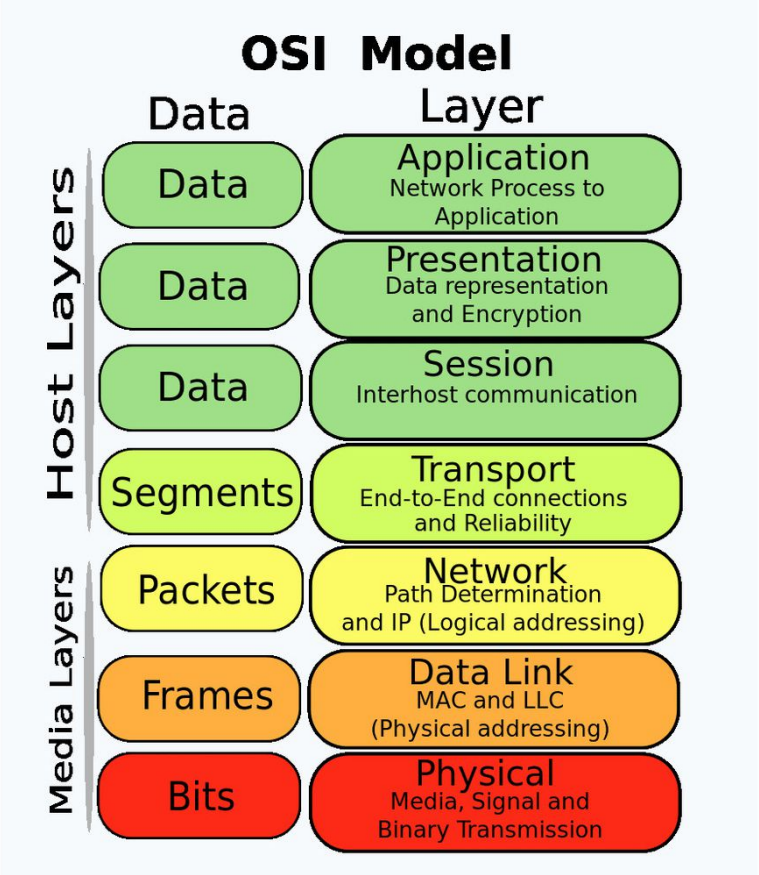
6. Presentation Layer (Layer 6)

- **Function:** Ensures that data is in a usable format and is where data encryption occurs. It translates data between the application layer and the network format.
- **Data Unit:** Data
- **Functions:** Data translation, encryption/decryption, data compression
- **Protocols:** SSL/TLS (Secure Sockets Layer/Transport Layer Security), JPEG, MPEG

7. Application Layer (Layer 7)

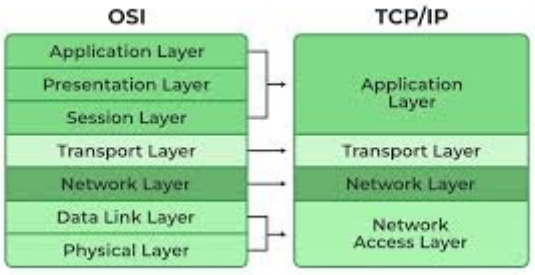
- **Function:** The closest layer to the end-user, it interacts with software applications that implement a communicating component. It provides services such as email, file transfer, and network resource sharing.
- **Data Unit:** Data

- **Protocols:** HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), DNS (Domain Name System)



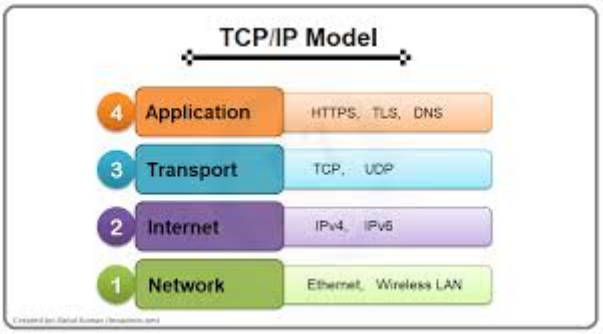
- Other network architectures

1. TCP/IP Model

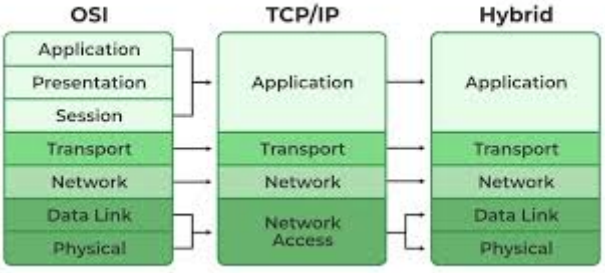


- **Description:** The TCP/IP (Transmission Control Protocol/Internet Protocol) model is a more practical and widely used framework compared to the OSI model. It was developed to standardize the way data is transmitted over the internet and is the foundation of the modern internet.
- **Layers:**
  1. **Network Interface Layer:** Corresponds to the OSI's Physical and Data Link layers. Handles the physical network hardware and protocols for data transmission.
  2. **Internet Layer:** Maps to the OSI's Network layer. Manages packet routing and addressing (e.g., IP).

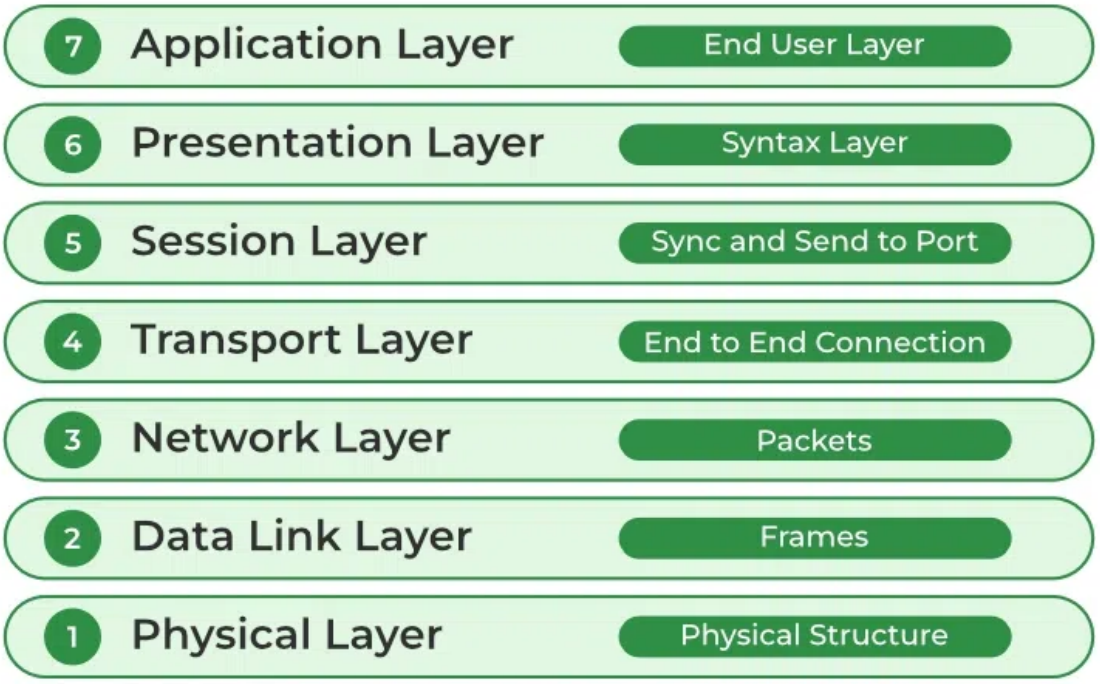
3. **Transport Layer:** Maps to the OSI's Transport layer. Provides end-to-end communication services (e.g., TCP and UDP).
4. **Application Layer:** Covers OSI's Session, Presentation, and Application layers. Handles application-level protocols and data handling (e.g., HTTP, FTP, SMTP).



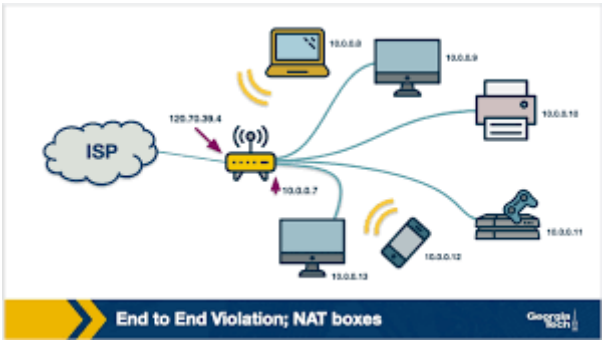
2. Hybrid Model



- **Description:** A combination of the OSI and TCP/IP models, often used to leverage the strengths of both. It applies the OSI model’s detailed layer structure while incorporating the practical aspects of the TCP/IP model.
- **Example:** The TCP/IP model’s Application layer can be mapped to OSI’s Application, Presentation, and Session layers, while TCP/IP’s Network Interface layer maps to OSI’s Physical and Data Link layers.

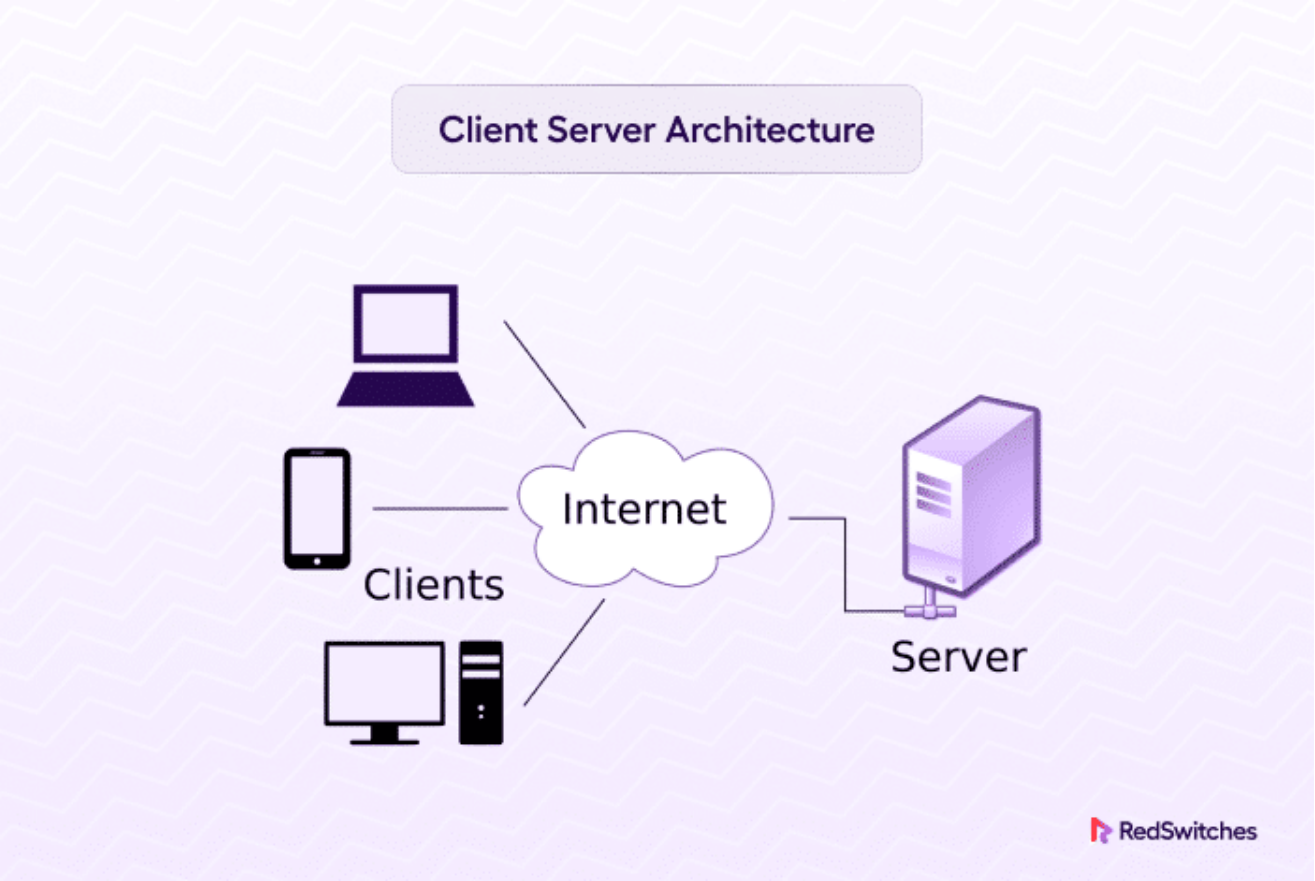


3. Internet Architecture



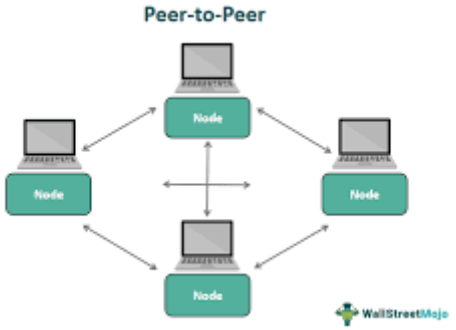
- **Description:** Refers to the actual structure of the internet, focusing on how various components like routers, switches, and servers interact with one another to facilitate communication.
- **Components:**
  - **Core Network:** High-capacity backbone that connects large networks and data centers.
  - **Edge Network:** Connects end-user devices to the core network, involving ISPs and local networks.
  - **Access Networks:** Local networks that connect end devices to the internet (e.g., Wi-Fi, Ethernet).

4. Client-Server Model



- **Description:** A network architecture where one machine (the client) requests services or resources from another machine (the server).
- **Characteristics:**
  - **Clients:** End-user devices or applications requesting resources.
  - **Servers:** Provide resources or services to clients (e.g., web servers, database servers).

5. Peer-to-Peer (P2P) Model



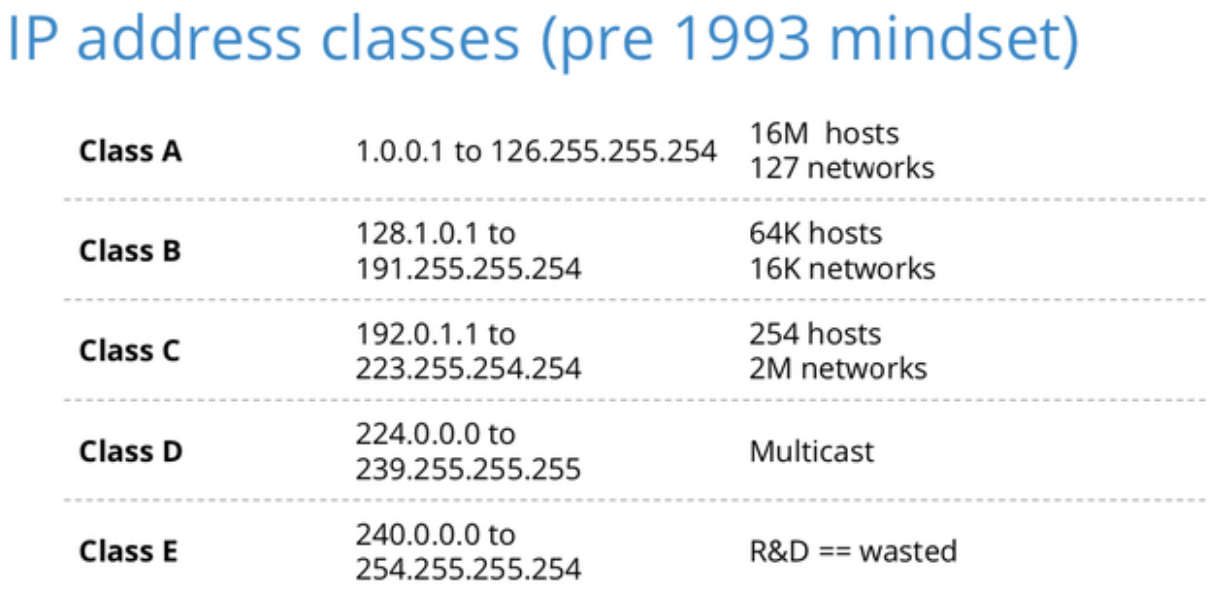
- **Description:** A decentralized network architecture where each device (peer) can act as both a client and a server, sharing resources directly with other peers without requiring a central server.
- **Characteristics:**
  - **Decentralized:** No central authority, with peers communicating directly.

    - **Scalable:** Suitable for file-sharing networks and collaborative applications (e.g., BitTorrent).
- An IP address (Internet Protocol address) is a unique identifier assigned to each device connected to a network that uses the Internet Protocol for communication
- It serves two main purposes : identifying the host or network interface and providing the location of the host in the network

Types of IP Addresses

IPv4 (Internet Protocol version 4)

- **Format:** IPv4 addresses are 32-bit numbers, typically represented in decimal format as four octets separated by periods (e.g., `192.168.1.1`)
- **Range:** 0.0.0.0 to 255.255.255.255
- **Example:** `192.168.0.1`



IPv4 Address Classes and Ranges						
Address Class	**Type**	**Range**	**Default Subnet Mask**	**Number of Networks**	**No of Hosts Per Network**	**Use**
A	Public	1.0.0.0 to 127.0.0.0	255.0.0.0	126	16,777,214	Governments and Large Number of Hosts
	Private	10.0.0.0 to 10.255.255.255				
B	Public	128.0.0.0 to 191.255.255.255	255.255.0.0	16,382	65,534	Medium Companies
	Private	172.16.0.0 to 172.31.255.255				
C	Public	192.0.0.0 to 223.255.255.255	255.255.255.0	2,097,150	254	Small Companies and LANs
	Private	192.168.0.0 to 192.168.255.255				
D	N/A	224.0.0.0 to 239.255.255.255	Not Applicable	N/A	N/A	Reserved for Multicasting
E	N/A	240.0.0.0 to 254.255.255.255	Not Applicable	N/A	N/A	Expermential
Special	Special	127.0.0.1 to 127.255.255.255	N/A	N/A	N/A	Loopback Testing

Note:
- Addesses 127.0.0.1 to 127.255.255.255 cannot be used and are reserved for loopback testing

- APIPA address range is 169.254.0.1 to 169.254.255.254 and has 65, 534 usable IP addresses, with the subnet mask of 255.255.0.0.

IPv6 (Internet Protocol version 6)

- **Format:** IPv6 addresses are 128-bit numbers, represented in hexadecimal format and separated by colons (e.g., `2001:0db8:85a3:0000:0000:8a2e:0370:7334`)
- **Range:** 340 undecillion addresses, allowing for virtually limitless devices
- **Example:** `2001:0db8:85a3:0000:0000:8a2e:0370:7334`

IP Address Components

1. **Network Portion :** Identifies the specific network. The size of this portion depends on the subnet mask.
2. **Host Portion :** Identifies the specific device (or host) within the network.

Subnet Mask

- A subnet mask defines which portion of an IP address is the network part and which is the host part.
- Example: For an IP address `192.168.1.1` with a subnet mask `255.255.255.0`, the first three octets (`192.168.1`) represent the network, and the last octet (`1`) identifies the specific device on that network.

Public vs. Private IP Addresses

1. **Public IP Address:**
   - Used for devices on the global internet.
   - Must be unique across the entire internet.
   - Example: `8.8.8.8` (Google's public DNS server)
2. **Private IP Address:**
   - Used within private networks (e.g., home or corporate networks).
   - Not routable on the internet but can communicate within the local network.
   - Common ranges:
     - `10.0.0.0` to `10.255.255.255`
     - `172.16.0.0` to `172.31.255.255`
     - `192.168.0.0` to `192.168.255.255`
   - Example: `192.168.1.1`

Dynamic vs. Static IP Addresses

1. **Static IP Address:**
   - Manually assigned and remains constant.
   - Often used for servers and other critical devices that need a consistent address.
2. **Dynamic IP Address:**
   - Automatically assigned by a DHCP (Dynamic Host Configuration Protocol) server.
   - Can change over time, typically used for general-purpose devices like laptops and smartphones.

How IP Addresses Work

- When a device wants to communicate with another device on a network, it uses the destination device's IP address to route the data.
- Routers use IP addresses to determine the best path to forward packets to their destination.

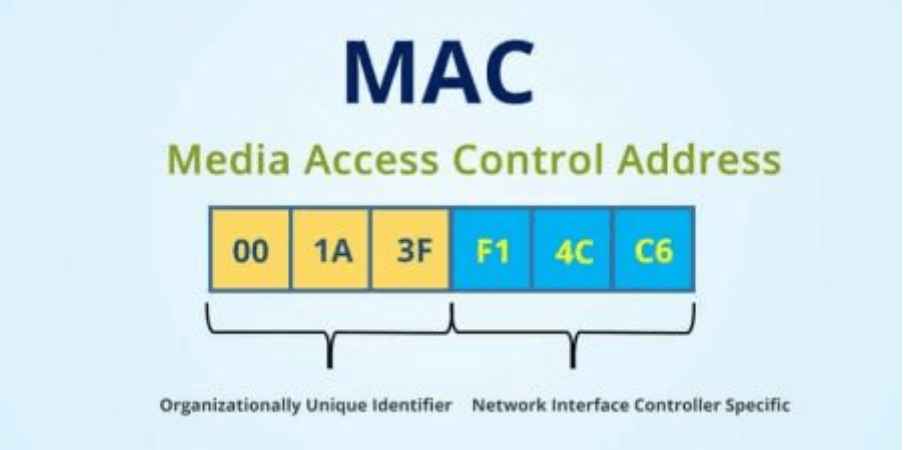
Example Scenario

Let's say you want to visit a website. Here's a simplified process

1. **Your device** sends a request to a DNS server to translate the domain name (like `www.example.com`) into its corresponding IP address.
2. **Your device** then sends a request to the website's IP address.

3. **The website's server** responds to your device, allowing the content to be displayed in your browser.

- A MAC address, or **Media Access Control address**, is a unique identifier assigned to network interfaces for communications on the physical network segment
- It operates at the Data Link Layer (Layer 2) of the OSI model and is used to uniquely identify devices on a network
- Unique identifier assigned to a network interface controller (NIC) for use as a network address in communications within a network segment
- It's also known as a hardware ID number or physical address

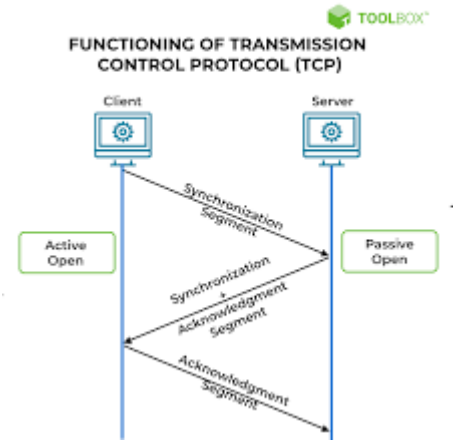


- **Format**
  - A MAC address is typically represented as a 12-digit hexadecimal number, often displayed in six pairs separated by colons (e.g., `00:1A:2B:3C:4D:5E`) or hyphens (e.g., `00-1A-2B-3C-4D-5E`).
- **Uniqueness**
  - Each MAC address is supposed to be unique to the network interface card (NIC) it is assigned to
  - This uniqueness is maintained by the IEEE, which allocates MAC address ranges to hardware manufacturers
- **Purpose**
  - MAC addresses are used in network protocols to ensure that data packets are delivered to the correct device within a local network
  - They are essential for network management and security, helping to direct traffic and filter network access

1. **Device Identification** : MAC addresses help identify devices on a local network. This is crucial for network communication, as it ensures that data is sent to the correct device.
2. **Network Management** : Network administrators use MAC addresses to configure network switches, routers, and other devices. They can create access control lists (ACLs) or apply network policies based on MAC addresses.

3. **Security** : MAC addresses are used in security protocols to filter network access. For example, a router might be configured to allow only devices with specific MAC addresses to connect to the network.

- **Types**
  - **Unicast** : Addresses that identify a single network interface
  - **Broadcast** : A special MAC address (`FF:FF:FF:FF:FF:FF`) used to send packets to all devices on a local network
  - **Multicast** : Addresses used to send packets to a group of devices
- **Static and Dynamic Addresses** : Generally, MAC addresses are fixed (hard-coded into hardware), but some devices allow users to change the MAC address through software, often for privacy reasons
- **TCP (Transmission Control Protocol)** is one of the core protocols of the Internet Protocol Suite, which is used for network communication



Key Features of TCP

1. **Connection-Oriented**
   - TCP establishes a connection between the sender and receiver before transmitting data
   - This is done through a process called the **three-way handshake**
   - The **three-way handshake** involves three steps
     1. **SYN**: The sender sends a synchronization (SYN) packet to the receiver
     2. **SYN-ACK**: The receiver responds with a synchronization acknowledgment (SYN-ACK) packet
     3. **ACK**: The sender acknowledges the receiver's response with an acknowledgment (ACK) packet
2. **Reliable Delivery**
   - TCP ensures that data is delivered accurately and in the correct order
   - It uses acknowledgments and sequence numbers to track and manage the data
   - If a packet is lost or corrupted, TCP will retransmit it

3. **Flow Control**
    - TCP uses flow control mechanisms to prevent a sender from overwhelming a receiver with too much data at once
    - It uses a sliding window technique to manage the amount of data in transit
4. **Congestion Control**
    - TCP adjusts the rate of data transmission based on network congestion
    - It uses algorithms like slow start, congestion avoidance, and congestion recovery to optimize data flow
5. **Error Checking**
    - TCP uses checksums to detect errors in transmitted data
    - If an error is detected, the affected packets are retransmitted
6. **Ordered Data Transfer**
    - TCP ensures that data is received in the same order it was sent
    - It achieves this by numbering packets and reassembling them at the receiver's end

How TCP Works

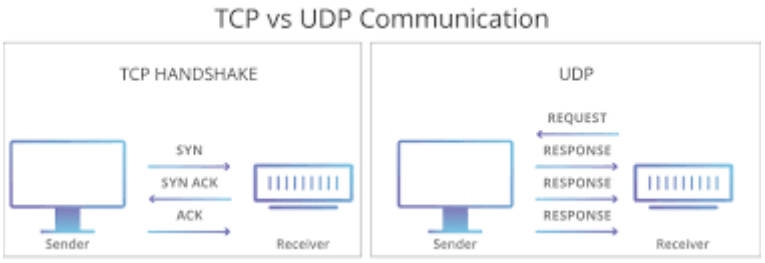
1. **Connection Setup**
    - Before data can be sent, a connection is established using the three-way handshake.
2. **Data Transfer**
    - Data is transmitted in segments. Each segment includes a sequence number, acknowledgment number, and other control information.
    - The receiver sends acknowledgments for successfully received segments. If segments are missing or erroneous, the sender retransmits them.
3. **Connection Termination**
    - When data transmission is complete, the connection is terminated through a four-step process:
        1. **FIN**: One side sends a finish (FIN) packet indicating that it has finished sending data
        2. **ACK**: The other side acknowledges the FIN packet
        3. **FIN**: The second side sends its own FIN packet
        4. **ACK**: The first side acknowledges the second FIN packet

Example Scenario

Imagine you're sending a large file from your computer to a friend's computer over the Internet. TCP would:

1. Establish a connection with your friend's computer
2. Break the file into smaller segments and send them
3. Ensure that each segment is received correctly and in the right order
4. Retransmit any segments that are lost or corrupted
5. Close the connection when the file transfer is complete

- **UDP (User Datagram Protocol)** is one of the core protocols of the Internet Protocol (IP) suite
- It is used for transmitting data over a network and is known for its simplicity and speed



Key Features

- **Connectionless**
  - UDP does not establish a connection before sending data
  - It simply sends packets (called datagrams) to the destination without any handshake process
  - This makes it faster but less reliable compared to connection-oriented protocols like TCP
- **Unreliable**
  - UDP does not guarantee the delivery of packets
  - There is no acknowledgment of receipt, and packets may be lost or arrive out of order
  - It's up to the application to handle any necessary error checking and recovery
- **No Flow Control**
  - UDP does not have mechanisms for flow control or congestion control
  - This means that if the sender is faster than the receiver, packets may be dropped without any mechanism to manage this
- **Low Overhead**
  - Because UDP has minimal overhead (no connection setup, no acknowledgment, and no flow control), it is often used in situations where speed is crucial and the application can tolerate some level of data loss
- **Data is Sent in Packets**
  - Data is divided into small chunks called **datagrams**

- Each datagram is independent of others, and the receiver processes each datagram separately

Use Cases

UDP is often used in applications where speed and efficiency are more important than reliability

- **Streaming Media** : Video and audio streaming where occasional data loss is acceptable
- **Online Gaming** : Where real-time performance is critical, and occasional packet loss can be tolerated
- **DNS Queries** : Domain Name System (DNS) queries use UDP because they are typically short and a few lost queries can be retried

Example

```
# UDP Server
import socket

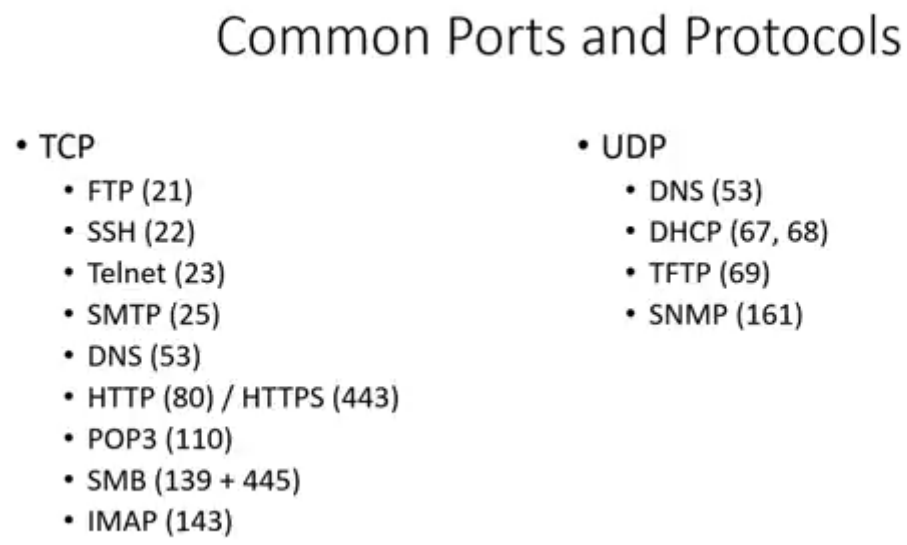
def udp_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind(('localhost', 12345))

    while True:
        data, addr = server_socket.recvfrom(1024)
        print(f"Received message: {data.decode()} from {addr}")

# UDP Client
import socket

def udp_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    message = "Hello, UDP!"
    client_socket.sendto(message.encode(), ('localhost', 12345))
    print("Message sent!")

# Uncomment these lines to run the server and client
# udp_server()
# udp_client()
```



01 TCP Protocols & Ports

1. FTP (File Transfer Protocol)

- **Port**: 21
- FTP is used to transfer files between a client and a server
- It supports both uploading and downloading files
- FTP operates in two modes: active and passive

2. SSH (Secure Shell)

- **Port**: 22
- SSH provides a secure channel over an unsecured network
- It's used for securely accessing and managing remote servers, as well as for secure file transfers using SCP or SFTP

3. Telnet

- **Port**: 23
- Telnet is a protocol used for text-based communication with remote devices over a network
- It provides a command-line interface for managing remote systems but is not secure, as it transmits data in plain text

4. SMTP (Simple Mail Transfer Protocol)

- **Port**: 25
- SMTP is used for sending and routing emails between mail servers
- It's part of the email protocol suite along with IMAP and POP3

5. DNS (Domain Name System)

- **Port**: 53
- DNS translates domain names (like www.example.com) into IP addresses
- It helps in locating services and devices on a network by resolving human-readable domain names to machine-readable IP addresses

6. HTTP (Hypertext Transfer Protocol)

- **Port**: 80
- HTTP is used for transmitting web pages and other content over the Internet
- It's the foundation of data communication on the World Wide Web

7. HTTPS (Hypertext Transfer Protocol Secure)

- **Port**: 443
- **Description**: HTTPS is the secure version of HTTP. It uses encryption (SSL/TLS) to provide a secure connection for transmitting web pages and other data, ensuring privacy and data integrity.

8. POP3 (Post Office Protocol version 3)

- **Port**: 110
- POP3 is used by email clients to retrieve emails from a mail server
- It downloads emails to the local device and typically deletes them from the server, though modern clients often provide options for leaving copies on the server

9. SMB (Server Message Block)

- **Port**: 445
- SMB is used for sharing files, printers, and other resources between computers on a network
- It's commonly used in Windows environments for file and print sharing

10. IMAP (Internet Message Access Protocol)

- **Port**: 143 (or 993 for IMAP over SSL/TLS)
- IMAP is used by email clients to access and manage emails on a mail server
- Unlike POP3, IMAP allows users to view and organize messages on the server, making it suitable for accessing emails from multiple devices

02 UDP Protocols & Ports

1. DNS (Domain Name System)

- **Port**: 53
- DNS translates human-readable domain names (like `www.example.com`) into IP addresses (like `192.0.2.1`)
- This process is crucial for locating and accessing websites and services on the internet
- DNS typically uses UDP for queries because it's usually a short request and response
- However, for larger responses or when additional information is needed, DNS can use TCP

2. DHCP (Dynamic Host Configuration Protocol)

- **Port**: 67 (server) and 68 (client)
- DHCP is used to automatically assign IP addresses and other network configuration settings (like subnet masks and default gateways) to devices on a network
- This simplifies network administration by eliminating the need for manual configuration
- DHCP uses UDP because it involves broadcasting messages to the network, and UDP's connectionless nature is suitable for this purpose

3. TFTP (Trivial File Transfer Protocol)

- **Port**: 69
- TFTP is a simple file transfer protocol used to transfer files between devices on a network
- It is often used for tasks like booting diskless workstations or transferring firmware updates
- TFTP uses UDP because it is designed for simplicity and does not require the complexities of TCP
- TFTP operates in a connectionless manner, which can be advantageous for its use cases

4. SNMP (Simple Network Management Protocol)

- **Port**: 161 (for general SNMP messages) and 162 (for SNMP traps)
- SNMP is used for network management and monitoring
- It allows network devices (like routers, switches, and servers) to be monitored and managed from a central location
- SNMP collects information about network performance and device status
- SNMP uses UDP for sending and receiving network management data
- This is because SNMP typically involves many small messages that can be sent and received quickly without the overhead of establishing and maintaining connections

Subnetting Overview

- Subnetting is the process of dividing a large network into smaller, more manageable sub-networks, or subnets
- This is done by manipulating the subnet mask, which determines how an IP address is split between the network and the host portion

IP Address Structure

- An IP address is a 32-bit number, usually represented in decimal form as four octets separated by periods (e.g., 192.168.1.1)
- Each octet represents 8 bits of the address, and the entire IP address is split into two parts:

1. **Network Portion :** Identifies the network
2. **Host Portion :** Identifies a device within that network

Subnet Mask

- A subnet mask is a 32-bit number that masks an IP address and divides the IP address into the network and host portions
- It also determines how many subnets and hosts per subnet can be created
  Example Subnet Mask
- **255.255.255.0** : This is a common subnet mask for a Class C network. It means that the first three octets (24 bits) are the network portion, and the last octet (8 bits) is the host portion.

Subnetting Process

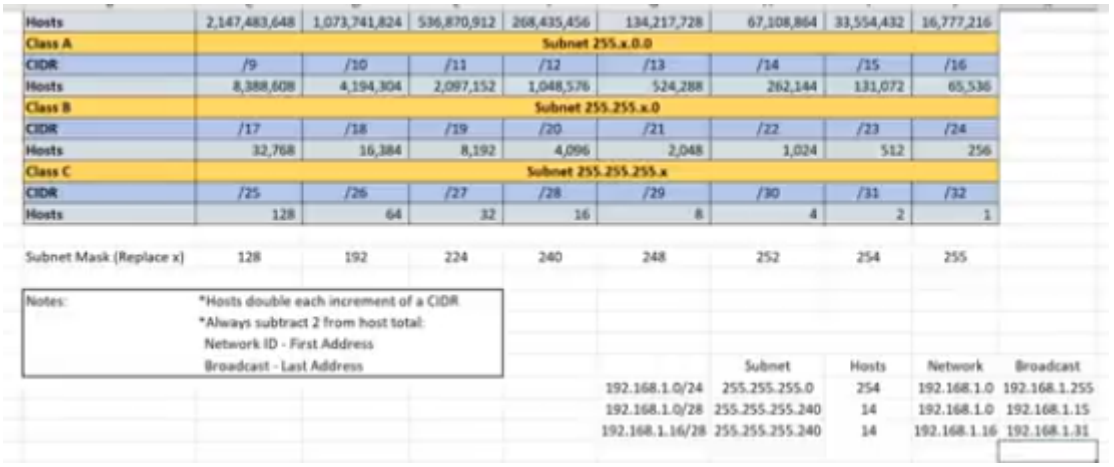
- Let's say you have an IP address **192.168.1.0/24** and you want to divide this into smaller subnets
- The `/24` means that 24 bits are used for the network portion, leaving 8 bits for the host portion

Example: Creating 4 Subnets

1. **Determine the Number of Subnets**
    - You want 4 subnets. To find out how many bits you need to borrow from the host portion, use the formula
      $(2^n \geq \text{Number of Subnets})$
      (2^2 = 4), so you need to borrow 2 bits

- **Subnetting** allows more efficient use of IP addresses by dividing a network into smaller subnets
- **Borrowing Bits** from the host portion creates additional subnets but reduces the number of available hosts per subnet

2. **Calculate the New Subnet Mask**
    - Original subnet mask: **255.255.255.0** (/24)
    - New subnet mask: Borrow 2 bits from the host portion:
        - **11111111.11111111.11111111.11000000** (binary)
        - **255.255.255.192** (/26)
3. **Determine the Subnet IP Ranges**
    - With a /26 subnet mask, each subnet will have 64 IP addresses (2^6 = 64). The subnets would be:
      1. **192.168.1.0/26**: Range: 192.168.1.0 - 192.168.1.63
      2. **192.168.1.64/26**: Range: 192.168.1.64 - 192.168.1.127
      3. **192.168.1.128/26**: Range: 192.168.1.128 - 192.168.1.191
      4. **192.168.1.192/26**: Range: 192.168.1.192 - 192.168.1.255



Hosts	2,147,483,648	1,073,741,824	536,870,912	268,435,456	134,217,728	67,108,864	33,554,432	16,777,216
Class A	Subnet 255.x.0.0							
CIDR	/9	/10	/11	/12	/13	/14	/15	/16
Hosts	8,388,608	4,194,304	2,097,152	1,048,576	524,288	262,144	131,072	65,536
Class B	Subnet 255.255.x.0							
CIDR	/17	/18	/19	/20	/21	/22	/23	/24
Hosts	32,768	16,384	8,192	4,096	2,048	1,024	512	256
Class C	Subnet 255.255.255.x							
CIDR	/25	/26	/27	/28	/29	/30	/31	/32
Hosts	128	64	32	16	8	4	2	1

Subnet Mask (Replace x)	128	192	224	240	248	252	254	255

Notes:	*Hosts double each increment of a CIDR *Always subtract 2 from host total: Network ID - First Address Broadcast - Last Address

	Subnet	Hosts	Network	Broadcast
192.168.1.0/24	255.255.255.0	254	192.168.1.0	192.168.1.255
192.168.1.0/28	255.255.255.240	14	192.168.1.0	192.168.1.15
192.168.1.16/28	255.255.255.240	14	192.168.1.16	192.168.1.31

4. **Calculate Usable IP Addresses**

    - In each subnet, the first IP is the network address, and the last IP is the broadcast address, so the usable IP range is:
        - **192.168.1.1 - 192.168.1.62** (for the first subnet)
        - **192.168.1.65 - 192.168.1.126** (for the second subnet)
        - **192.168.1.129 - 192.168.1.190** (for the third subnet)
        - **192.168.1.193 - 192.168.1.254** (for the fourth subnet)

- This approach helps in managing and utilizing IP address spaces efficiently, especially in large networks

How Subnet Masks Work with IP Addresses

When you apply a subnet mask to an IP address:

1. **AND Operation**
    - The subnet mask is applied to the IP address using a bitwise AND operation
    - This operation helps to extract the network portion of the IP address
    - **Example:**
        - IP Address: `192.168.1.10` → `11000000.10101000.00000001.00001010` (binary)
        - Subnet Mask: `255.255.255.0` → `11111111.11111111.11111111.00000000` (binary)
        - Result (Network Address): `11000000.10101000.00000001.00000000` → `192.168.1.0`
2. **Verification of Network Membership**
    - If you need to check whether two IP addresses are in the same subnet, you would apply the subnet mask to both IP addresses
    - If the resulting network addresses are the same, the IPs are in the same subnet

Kali Linux is a powerful tool for cybersecurity professionals, penetration testers, and ethical hackers. Here are some important Kali Linux commands that are commonly used:

1. Basic File Operations

- `ls`: Lists files and directories in the current directory

```
ls -l
```

- `cd`: Changes the directory

```
cd /path/to/directory
```

- `pwd`: Prints the current working directory

```
pwd
```

- `cp`: Copies files or directories

```
cp source_file destination
```

- `mv`: Moves or renames files or directories

```
mv old_name new_name
```

- `rm`: Removes files or directories

```
rm file_name
```

2. Network Commands

- `ifconfig`: Displays or configures network interfaces

```
ifconfig
```

- `ping`: Sends ICMP echo requests to test network connectivity

```
ping 8.8.8.8
```

- `netstat`: Displays network connections, routing tables, interface statistics, etc

```
netstat -an
```

- `nmap`: Network exploration tool and security/port scanner

```
nmap -sP 192.168.1.0/24
```

3. System Information

- `uname -a`: Displays detailed system information

```
uname -a
```

- `uptime`: Shows how long the system has been running

```
uptime
```

- `df -h`: Displays disk space usage

```
df -h
```

- `top`: Displays real-time system processes and resource usage

```
top
```

4. User Management

- `whoami`: Displays the current logged-in user

```
whoami
```

- `sudo`: Executes a command as the superuser

```
sudo command
```

- `adduser`: Adds a new user to the system

```
sudo adduser username
```

- `passwd`: Changes a user's password

```
passwd
```

5. File Permissions

- `chmod`: Changes file permissions

```
chmod 755 file_name
```

- `chown`: Changes file owner and group

```
chown user:group file_name
```

6. Package Management

- `apt-get update`: Updates the package list

```
sudo apt-get update
```

- `apt-get upgrade`: Upgrades all installed packages

```
sudo apt-get upgrade
```

- `apt-get install`: Installs a new package

```
sudo apt-get install package_name
```

- `apt-get remove`: Removes a package

```
sudo apt-get remove package_name
```

7. Security and Hacking Tools

- `msfconsole`: Opens the Metasploit Framework console

```
msfconsole
```

- `airmon-ng`: Starts monitor mode on a wireless interface

```
sudo airmon-ng start wlan0
```

- `aircrack-ng`: Cracks WEP/WPA-PSK keys

```
aircrack-ng capture_file.cap
```

- `hydra`: Password cracking tool

```
hydra -l user -P wordlist.txt ftp://192.168.1.100
```

- `john`: Password cracker

```
john --wordlist=/path/to/wordlist.txt hashfile.txt
```

- VMWARE - VirtualBox - Kalix64
- **User & Privileges**
- pimpmykali
- cat echo touch nano
- scripting with bash
- grep cut
- ipsweep.sh
- nmap
- python scripting
  - sockets
- 5 stages of ethical hacking
  - COVERING TRACKS
  - RECONNAISSANCE - active and passive
  - SCANNING & ENUMERATION - Nmap, Nessus, Nikto
  - GAINING ACCESS - 'Exploitation'
  - MAINTAINING - access
- Passive Recon
  - Information Gathering
  - Physical/Social - location - job information
  - Web/Host Assessment
    - TARGET VALIDATION
    - FINDING SUBDOMAINS
    - FINGERPRINTING
    - DATA BREACHES
  - Identifying our Target
    - Bugcrowd - progrms - Tesla
    - Discovering email addresses
      - hunter.io
      - phonebook.cz

        - emailhippo
- Breached Credentials
    - hmaberick - breach-parse
    - dehashed
- Hunting Subdomains
    - Web information gathering
        - sudo apt install sublist3r
        - sublist3r -d tesla.com
        - crt.sh %tesla.com
- Identifying website technologies
    - BUILTWIT.com
    - wapalizer
    - whatweb
- Gathering Information with Burpsuite
    - kioptrix
- Scanning with Nmap
    - arg-scan -l
    - netdiscover -r ip/24
    - stuff scanning
    - nmap -T4 -p- -A ip
- Enumerating HTTP/HTTPS
    - nikto -h https://ip
    - dirbuster / gobuster
- Enumerating SMB
    - nsfconsole
    - nsfvenom
    - smbclient -L \\ip\
- Enumerating SSH
    - ssh ip
    - ssh ip -oKexAlgorithms=+hvghv
- Researching potential vulnerabilities
    - searchploit
- Scanning with Nessus
- Reverse shells vs Bind shell
    - nc -nvlp port || nc ip port -e /bin/bash
    - nc ip port || nc -nvlp -e /bin/bash
    - staged vs non-staged payloads || nsfconsole
    - openluck
- Bruteforce Attack

        - hydra -l root -P /usr/share/wordlists/metasploit
    - Credential stuffing and password spraying
- Capstone
    - dhclient - academy setup || ip -a || nmap
    - hashcat to crack md5 hashes
        - locate rockyou.txt
        - ffuf
        - pspy
        - bash reverse shell one liner
    - dev walkthrough
        - boltwire exploit
        - gtfobins
            - sudo exkelations || zip
            - jenkins exploit
            - foxy proxy
            - groovy reverse shell