

# MODULE I

## LINUX BASICS

by: **Dr. Ram Paul Hathwal**  
Dept of CSE, ASET, AUUP

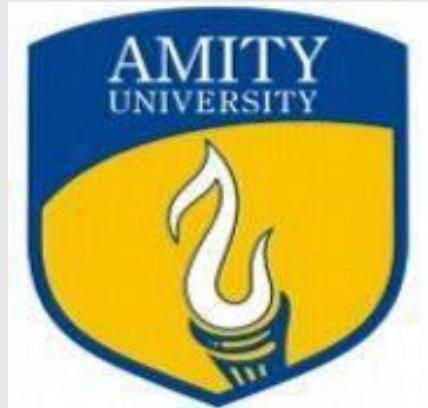


AMITY  
UNIVERSITY

# Linux for Devices

## CSE438

Department of Computer  
Science and Engineering



Dr. Ram Paul Hathwal  
Department of Computer Science and Engineering  
Amity University Uttar Pradesh



- Syllabus\_Linux for Devices\_CSE438.docx

## Course Goals

- ✓ Introduce you to the basics of Linux.
- ✓ Explores the basic characteristics of Linux Networking.
- ✓ Teach you to learn Linux Shell, File Structure, and Network Administration Services.
- ✓ Makes you understand various Linux security techniques.

# Learning Objective

- To understand how Linux was developed and which are the features that make it popular as compared to other operating systems
- To give an abstract description of the overall architecture of Linux
- To present various designs of the kernel and among them which design is adopted for Linux kernel development
- To present the design and importance of each directory/file in the Linux file system
- To present the idea of how a communicator takes responsibility to send only correct instructions to the master component of O/S so that the master can do more productive work
- To impart the most useful Linux commands
- Understand the basics of Linux
- Application of Linux in Industry Usage

# Learning Objective

- Can understand the popularity of Linux and its various distributions that can be used for personal and commercial purposes without any cost
- Understand the essential components of Linux system architecture
- Able to distinguish the design of Linux kernel with non-Linux O/S kernel
- Able to understand the importance of file system
- Able to understand that to increase the throughput of the system, the correctness of the user instruction must be checked by the mediator before execution
- Comfortable to use commands in any organization where Linux o/s is primarily used.
- Analyze the category of different types of users.
- Able to assign appropriate access permission to users working in the same system.
- Perform the basic operations for Linux.
- Analyze the potential of Linux in Industry.

# Topics

- ✓ Introduction to Linux
- ✓ File System of the Linux
- ✓ General usage of Linux kernel & basic commands
- ✓ Linux users and group
- ✓ Permissions for file
- ✓ Directory and users
- ✓ Searching a file & directory
- ✓ Zipping and Unzipping concepts
- ✓ Linux for the Industry 4.0 Era
  - OPENIL and its advantages
  - Features of OPENIL

# Why Linux

- A Linux Distribution has thousands of dollars worth of software for no cost.
- Linux is a complete operating system:
  - **Stable** - the crash of an application is much less likely to bring down the OS under Linux.
  - **Reliable** - Linux servers are often up for hundreds of days compared with the regular reboots required with a Windows system.
  - *Extremely powerful*

# Why Linux (continued)

- Excellent networking facilities.
- Ideal environment to run servers such as a web server, *or* an ftp server
- A wide variety of commercial software is available if not satisfied by the free software
- Easily upgradeable.
- Supports multiple processors.
- True multi-tasking, multi-user OS.
- An excellent window system called X, the equivalent of Windows but much more flexible.

# Before Linux

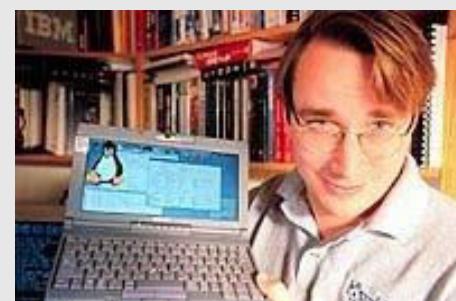
- In 80's, Microsoft's DOS was the dominant OS for PC
  - single-user, single-process system
- Apple MAC is better, but expensive
- UNIX is much better but much more expensive.
- Only for minicomputers for commercial applications
- People were looking for a UNIX-based system, which is cheaper and can run on a PC
- Both DOS, MAC, and UNIX are **proprietary**, i.e., the source code of their kernel is protected.
  - No modification is possible without paying high license fees

# GNU Project

- Established in 1984 by **Richard Stallman**, who believes that software should be free from restrictions against copying or modification to make better and more efficient computer programs
- **GNU** is a recursive acronym for “**GNU's Not Unix**”
- Aim at developing a complete Unix-like operating system which is free for copying and modification.
- Companies make their money by maintaining and distributing the software, e.g. optimally packaging the software with different tools (Redhat, Slackware, Mandrake, SuSE, *etc*)
- Stallman built the first free GNU C Compiler in 1991.
- But still, an OS was yet to be developed

# Beginning of Linux

- A famous professor Andrew Tanenbaum developed **Minix**, a simplified version of UNIX that runs on PC
- Minix is for class teaching only. No intention for commercial use
- In Sept 1991, **Linus Torvalds**, a second-year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1
- It was put to the Internet and received enormous response from worldwide software developers
- By December came version 0.10. Still Linux was little more than in skeletal form.



# Confrontation and Development

- Message from Professor Andrew Tanenbaum
  - "I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-)" (Andrew Tanenbaum to Linus Torvalds)
  - "Linux is obsolete".
  - (Remark made by Andrew Tanenbaum)
- But work went on. Soon more than a hundred people joined the Linux camp. Then thousands. Then hundreds of thousands
- It was licensed under **GNU General Public License**, thus ensuring that the source codes will be free for all to copy, study and to change.

# Linux

- Created by Linus Torvalds in 1991
- Includes system utilities & libraries from the GNU Project
- Open Source Hardware:
  - Free
  - Stable
  - Easily fixed if bugs appear
- Why did Linux become popular??
  - Low-cost alternative in sagging economy
  - Fear of Microsoft gaining a stranglehold on corporate
  - Intel loosened its relationship with Microsoft
  - IBM made an effort to be Linux-compatible



- Linux has been used for many computing platforms
  - PC, PDA, Supercomputer,...
- Current kernel version 2.6.13
- Not only character user interface but graphical user interface, thanks to the X-Window technology
- Commercial vendors moved in Linux itself to provide freely distributed code. They make their money by compiling up various software and gathering them in a distributable format
  - Red Hat, Slackware, etc.
- Chinese distribution of Linux also appeared in Taiwan and China - CLE, Red Flag Linux

# Linux Has Many Distributions



## ➤ Advantages over Windows

- It's almost free to relatively inexpensive
- Source code is included
- Bugs are fixed quickly and help is readily available through the vast support in the Internet
- Linux is more stable than Windows
- Linux is truly multi-user and multi-tasking
  - **multiuser**: OS that can simultaneously serve a number of users
  - **multitasking**: OS that can simultaneously execute a number of programs
- Linux runs on equipment that other operating systems consider too underpowered, e.g. 386 systems, PDA, etc.

# Linux Pros and Cons

## ➤ Advantages over Windows

- Isn't as popular as Windows
- No one commercial company is responsible for Linux
- Linux is relatively hard to install, learn and use Linux runs on equipment that other operating systems consider too underpowered, e.g. 386 systems, PDA, etc.

## ➤ Hence currently, Linux is mainly used in commercial applications, server implementation

## ➤ More than 75% current network servers are developed based on Linux or Unix systems

✗ – Due to the relatively **high reliability**



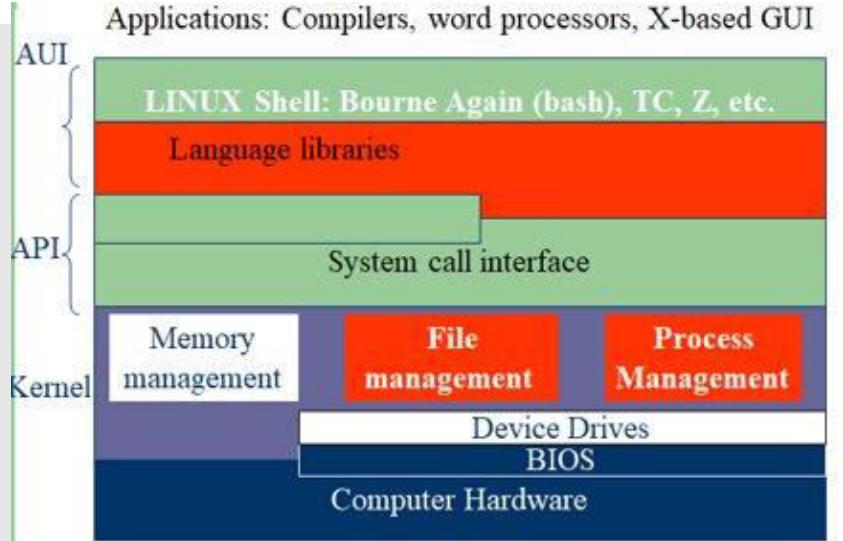
# Linux vs. Unix

- Linux is free, but Unix is not
- Unix is compatible with Linux at the system call level, meaning most programs written for either Unix or Linux can be recompiled to run on the other system with a minimum of work



- Both offer
  - Graphics capabilities
  - Networking capabilities
  - But Linux networking is excellent

- **Kernel**
  - The part of an OS where the real work is done
- **System call interface**
  - Comprise a set of functions (often known as **Application Programmer's Interface (API)**) that can be used by the applications and library routines to use the services provided by the kernel
- **Application User's Interface**
  - Interface between the kernel and user
  - Allow user to make commands to the system
  - Divided into text-based and graphical-based
- **File Management**
  - Control the creation, and removal of files and provide directory maintenance
  - For a **multiuser system**, every user should have its own right to access files and directories



- **Process Management**
- For a **multitask system**, multiple programs can be executed simultaneously in the system
  - When a program starts to execute, it becomes a **process**
  - The same program executing at two different times will become two different processes
  - Kernel manages processes in terms of creating, suspending, and terminating them
  - A process is protected from other processes and can communicate with others

## ➤ Memory management

- Memory in a computer is divided into **main memory** (RAM) and **secondary storage** (usually referred to as hard disk)
- Memory is small in capacity but fast in speed, and hard disk is vice versa
- Data that are not currently used should be saved to a hard disk first, while data that are urgently needed should be retrieved and stored in RAM
- The mechanism is referred to as **memory management**

## ➤ Device drivers

- Interfaces between the kernel and the BIOS
- Different device has different driver

## ➤ Kernel

- A core component of the Operating System without which OS can't work
- Kernel is the nervous system of the OS
- It controls everything in OS including I/O Management, Process Management and so on
- It is a bridge between applications and the actual data processing done at the hardware level
- It acts as an interface between the user applications and the hardware

Kernel Version	Name (Reason)
1.2.0	Linux '95 <sup>[2]</sup>
1.3.51	Greased Weasel <sup>[3]</sup>
2.2.1	Brown Paper Bag <sup>[4]</sup>
2.4.15	Greased Turkey <sup>[5]</sup>
<b>2.6.2–2.6.3–2.6.4–</b>	Feisty Dunnart <sup>[6]</sup>
<b>2.6.5–2.6.6–2.6.7–2.6.8–2.6.9</b>	Zonked Quokka <sup>[7]</sup>
<b>2.6.10-rc1–2.6.10–2.6.11–2.6.12–2.6.13–</b>	Woozy Numbat <sup>[8][9]</sup>
<b>2.6.14-rc1–2.6.14–</b>	Affluent Albatross <sup>[10]</sup>
<b>2.6.15-rc6–2.6.15–2.6.16–</b>	Sliding Snow Leopard <sup>[11]</sup>
stable: 2.6.16.28-rc2–	Stable Penguin
2.6.17-rc5	Lordi Rules <sup>[12]</sup> (Eurovision 2006 winners) <sup>[13]</sup>
<b>2.6.17-rc6–2.6.17–</b>	Crazed Snow-Weasel <sup>[14]</sup>
<b>2.6.18–2.6.19–</b>	Avast! A bilge rat! (TLAPD 2006) <sup>[15]</sup>
<b>2.6.20-rc2–2.6.20–</b>	Homicidal Dwarf Hamster <sup>[16][17]</sup>
<b>2.6.21-rc4–2.6.21–</b>	Nocturnal Monster Puppy <sup>[18]</sup>
2.6.22-rc3–2.6.22-rc4	Jeff Thinks I Should Change This, But To What?
<b>2.6.22-rc5–2.6.22–</b>	Holy Dancing Manatees, Batman! <sup>[19]</sup>
<b>2.6.23-rc4–2.6.23-rc6</b>	Pink Farting Weasel <sup>[20]</sup>

3.6-rc7–3.6–3.7–	Terrified Chipmunk <sup>[43][44]</sup>
3.8-rc6–3.8–3.9–3.10–	Unicycling Gorilla <sup>[45][46]</sup>
stable: 3.8.5–	Displaced Humerus Anterior <sup>[47]</sup>
stable: 3.9.6–	Black Squirrel Wakeup Call <sup>[48]</sup>
stable: 3.10.6–	TOSSUG Baby Fish <sup>[49][50][51][52]</sup>
3.11-rc1–3.11	Linux for Workgroups (20 years of Windows 3.11) <sup>[53]</sup>
3.12-rc1–	Suicidal Squirrel <sup>[54]</sup>
3.13-rc1	One Giant Leap for Frogkind <sup>[55]</sup> (NASA LADEE launch photo) <sup>[56]</sup>
3.14-rc1	Shuffling Zombie Juror <sup>[57]</sup>
3.18-rc3	Diseased Newt <sup>[58]</sup>
4.0	Hurr durr I'ma sheep <sup>[59]</sup> (Internet poll)
4.1.1	Series 4800 <sup>[60]</sup>
4.3-rc5	Blurry Fish Butt <sup>[61][62]</sup>
4.6-rc6	Charred Weasel <sup>[63]</sup>
4.7-rc1	Psychotic Stoned Sheep <sup>[64]</sup>
4.9	Roaring Lionus <sup>[65][66]</sup>
4.10-rc5	Anniversary Edition <sup>[67]</sup>
4.10-rc6	Fearless Coyote <sup>[68]</sup>
4.17-rc4	Merciless Moray <sup>[69]</sup>
4.19	"People's Front" <sup>[70]</sup>
4.20-rc4–5.0	Shy Crocodile <sup>[71]</sup>
5.2-rc2	Golden Lions <sup>[72][73]</sup>
5.2	Bobtail Squid <sup>[74]</sup>
5.4-rc2	Nesting Opossum <sup>[75]</sup>
5.4-rc5	Kleptomaniac Octopus <sup>[76]</sup>

# Tasks of kernel

## 1. Central Processing Unit

The Kernel takes responsibility for deciding at any time which of the many running programs should be allocated to the processors

## 2. Random Access Memory

It is used to store both program instructions and data. Often multiple programs will want memory access, frequently demanding more memory than the computer has available. The Kernel is responsible for deciding which memory each process can use and determining what to do when not enough memory is available.

### 3. Input/Output Devices

The Kernel allocates requests from applications to perform I/O to an appropriate device and provides convenient methods for using the device

### 4. Memory Management

The Kernel has full access to the system's memory and must allow processes to safely access this memory as they require it

### 5. Device Management

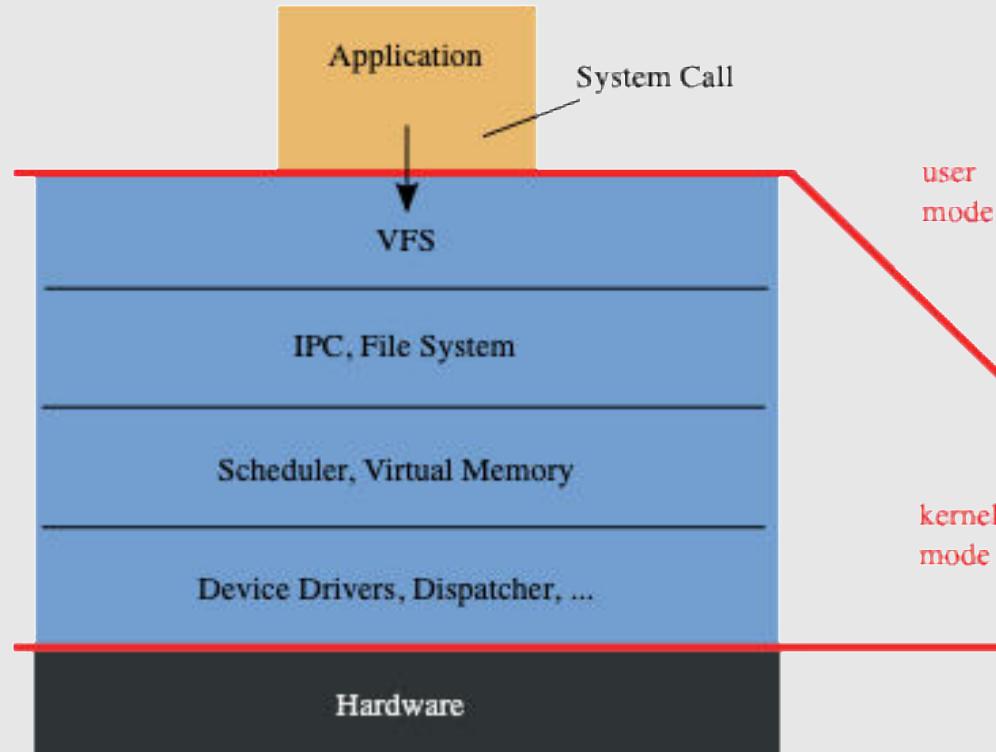
A Kernel must maintain a list of available devices. This list may be known in advance configured by the user or detected by the operating system at run time (normally called plug and play)

# Continued...

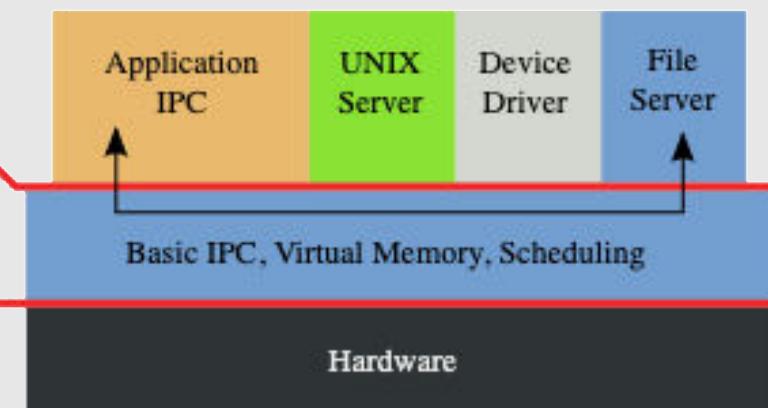
- Scheduling of Process (Dispatching)
- Interprocess Communication
- Process Synchronization
- Context Switching
- Manipulation of Process Control Blocks
- Interrupt Handling
- Process Creation and Destruction
- Process Suspension and Resumption

# Types of Kernel

Monolithic Kernel  
based Operating System



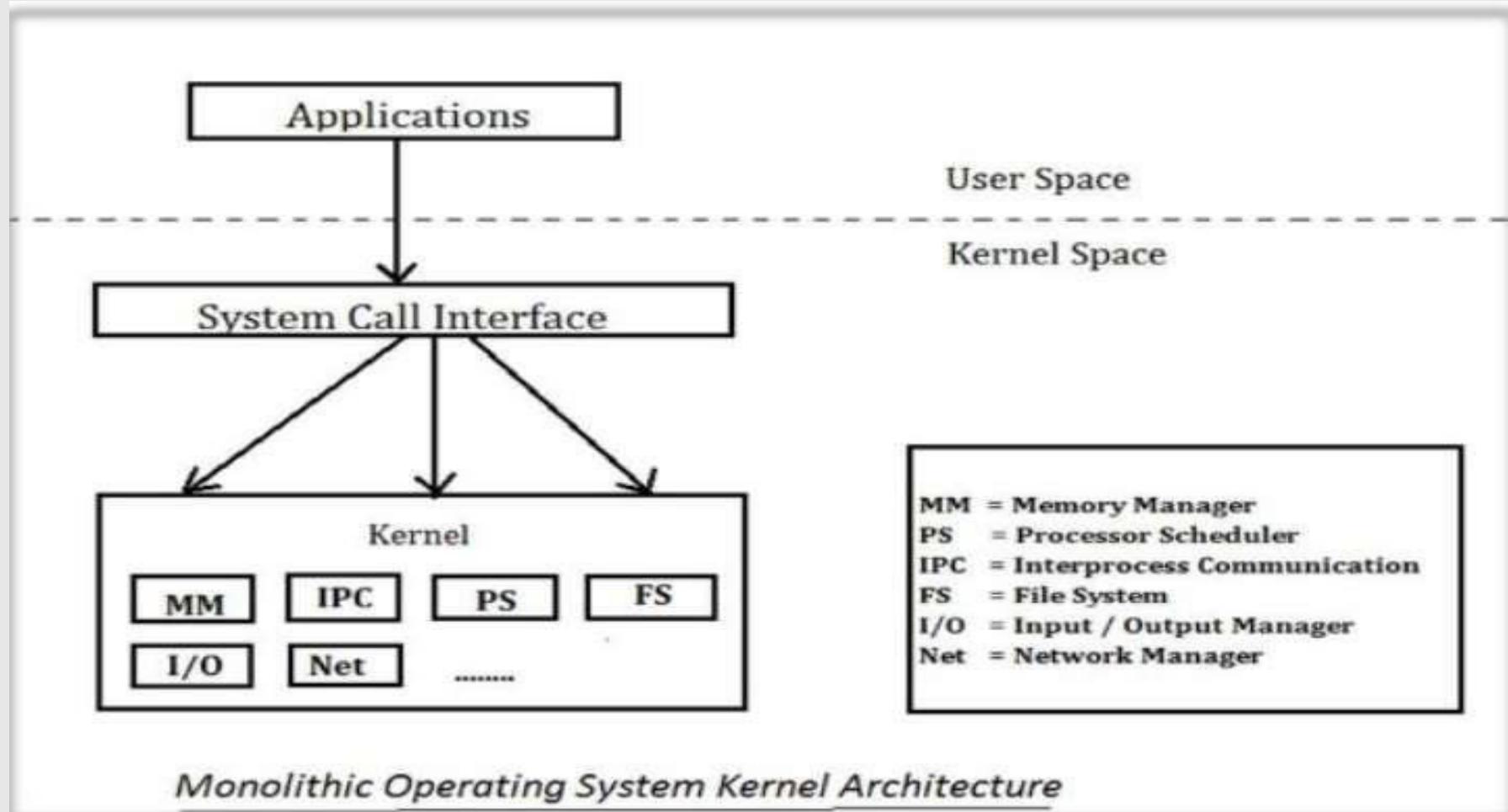
Microkernel  
based Operating System



# Monolithic Kernel

- Run in Kernel space every basic system service like
  - process and memory management,
  - interrupt handling and I/O Communication,
  - file system *etc.* in Kernel space
- Used by Unix, Linux like Operating systems
  - Since there is less software involved, it is faster
- Every component of the Operating system is contained in the kernel and can directly communicate with each other simply by using function calls.
- The Kernel typically executes with unrestricted access to the computer system
- This approach provides rich and powerful hardware access

# Monolithic Kernel



# Pros of Monolithic Kernel

- Smaller in source and compiled forms
- Less code generally means fewer bugs and security problems is also less
- System calls are used to do operations in a monolithic kernel
- Execution is fast
- It has all the things in the kernel itself so we don't need any extra mechanism for handling of I/O and process at the time of application making.

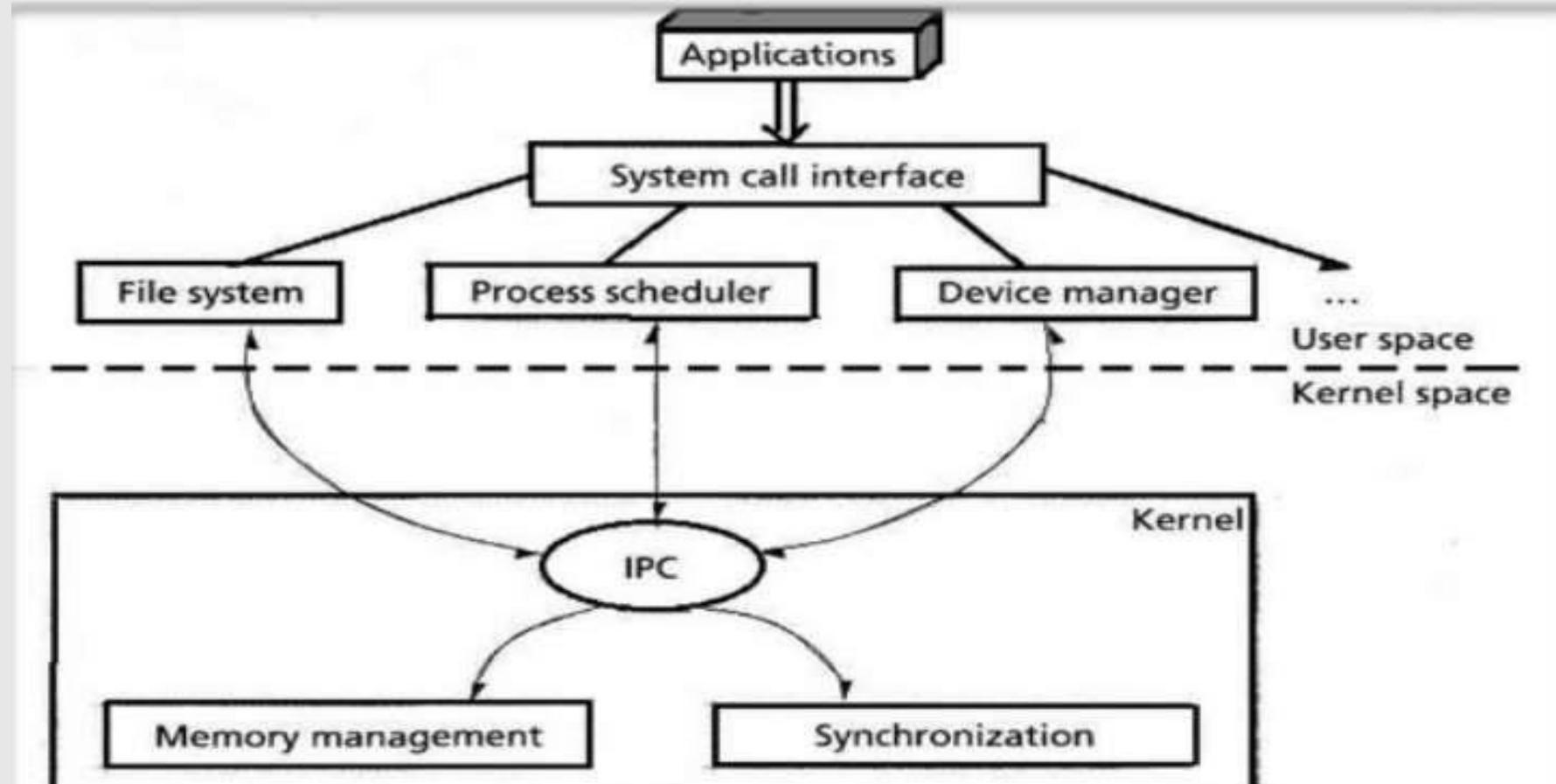
# Cons of Monolithic Kernel

- Coding in kernel space is hard since you cannot use common libraries
- Debugging is harder, rebooting the computer is often needed
- Bugs in one part of the kernel produce strong side effects
- Kernels often become very huge and difficult to maintain
- Not portable one. Monolithic kernels must be rewritten for each new architecture that the OS is to be used on.

# Micro Kernel

- The kernel provides basic functionality that allows the execution of servers and separate programs
- The Kernel is broken into separate processes known as server
  - Some of the servers run in user space and some in kernel space
  - All servers are kept separate and run in different address spaces
- The communication in microkernels is done via message passing
  - The servers communicate through Interprocess Communication(IPC)
  - Servers invoke "services" from each other by sending messages
- It is also possible to dynamically switch among operating systems and to have more than one active simultaneously

# Micro Kernel



Microkernel Operating System

# Advantages of Micro Kernel

- Easier to maintain than Monolithic Kernel
- Crash resistant (If one server fails, other servers can still work efficiently)
- Portable
- Smaller in size
- Contains smaller amount of code. It increases stability and security.

# Key Comparisons

Basis for Comparison	Micro Kernel	Monolithic Kernel
Size	Smaller in size	Larger than microkernel
Execution	Slow	Fast
Extendible	Easy to extend	Hard to extend
Security	If a service crashes, it does effects on working on the microkernel	If a service crashes, the whole system crashes in monolithic kernel
Code	To write a microkernel more code is required	To write a monolithic kernel less code is required
Example	QNX, Symbian, L4Linux etc.	Linux, BSDs(FreeBSD, OpenBSD, NetBSD)etc.



**Thanks!...**

**Any question?**

# MODULE I

## MORE ABOUT LINUX

by: **Dr. Ram Paul Hathwal**

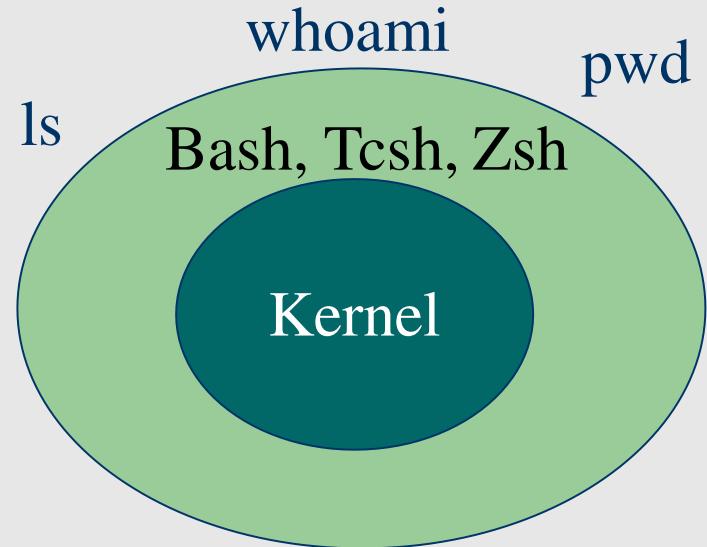
Dept of CSE, ASET, AUUP

# Topics

- Introduction to Linux
- File System of the Linux
- General usage of Linux kernel & basic commands
- Linux users and group
- Permissions for file
- Directory and users
- Searching a file & directory
- Zipping and Unzipping concepts
- Linux for the Industry 4.0 Era
  - ✓ OPENIL and its advantages
  - ✓ Features of OPENIL

# Linux Shell

- Shell interprets the command and request service from kernel
- Similar to DOS but DOS has only one set of interface while Linux can select different shell
  - Bourne Again shell (Bash), TC shell (Tcsh), Z shell (Zsh)
- Different shell has similar but different functionality
- Bash is the default for Linux
- Graphical user interface of Linux is in fact an application program work on the shell



# Example: Linux Shell Commands

- Frequently used commands available in most shells:
  - ls** : to show (**list**) the names of the file in the current directory
  - cd** : **c**hange **d**irectory,
    - e.g. **cd** change to the root directory
    - /              change to the parent of that
    - cd** directory
  - **cp** : **c**o**p**y one file to another
    - e.g. **cp** abc.txt xyz.txt
    - copy abc.txt to xyz.txt
  - **rm** : **r**emove a file

# Example: Linux Shell Commands

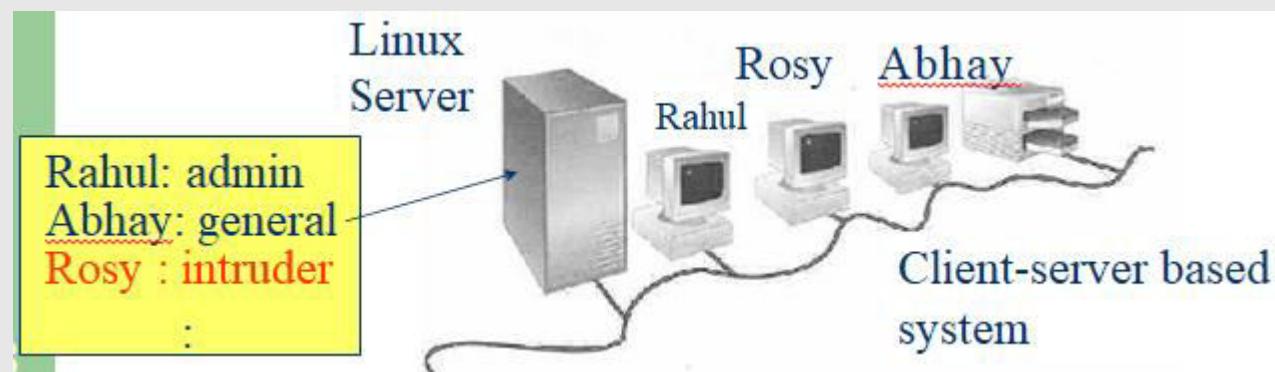
- **man** : ask for the **manual** (or help) of a command
  - e.g. man cdask for the manual of the command cd
- **pwd** : show the name of the **present working directory**
- **cat** : to show the content of a text file
  - e.g. cat abc.txt show the content of abc.txt
- **whoami** : to show the **username** of the current user

[More Linux/Unix commands](#)

# Topics

- Introduction to Linux
- File System of the Linux
- General usage of Linux kernel & basic commands
- **Linux users and group**
- Permissions for file
- Directory and users
- Searching a file & directory
- Zipping and Unzipping concepts
- Linux for the Industry 4.0 Era
  - OPENIL and its advantages
  - Features of OPENIL

- Linux is a **multiuser OS**
- Allow multiple users to use the resource of a computer at the same time
- Every user needs to **login** the system with the **password** provided to identify their right in using the resource
- Require for both client-server based system or desktop



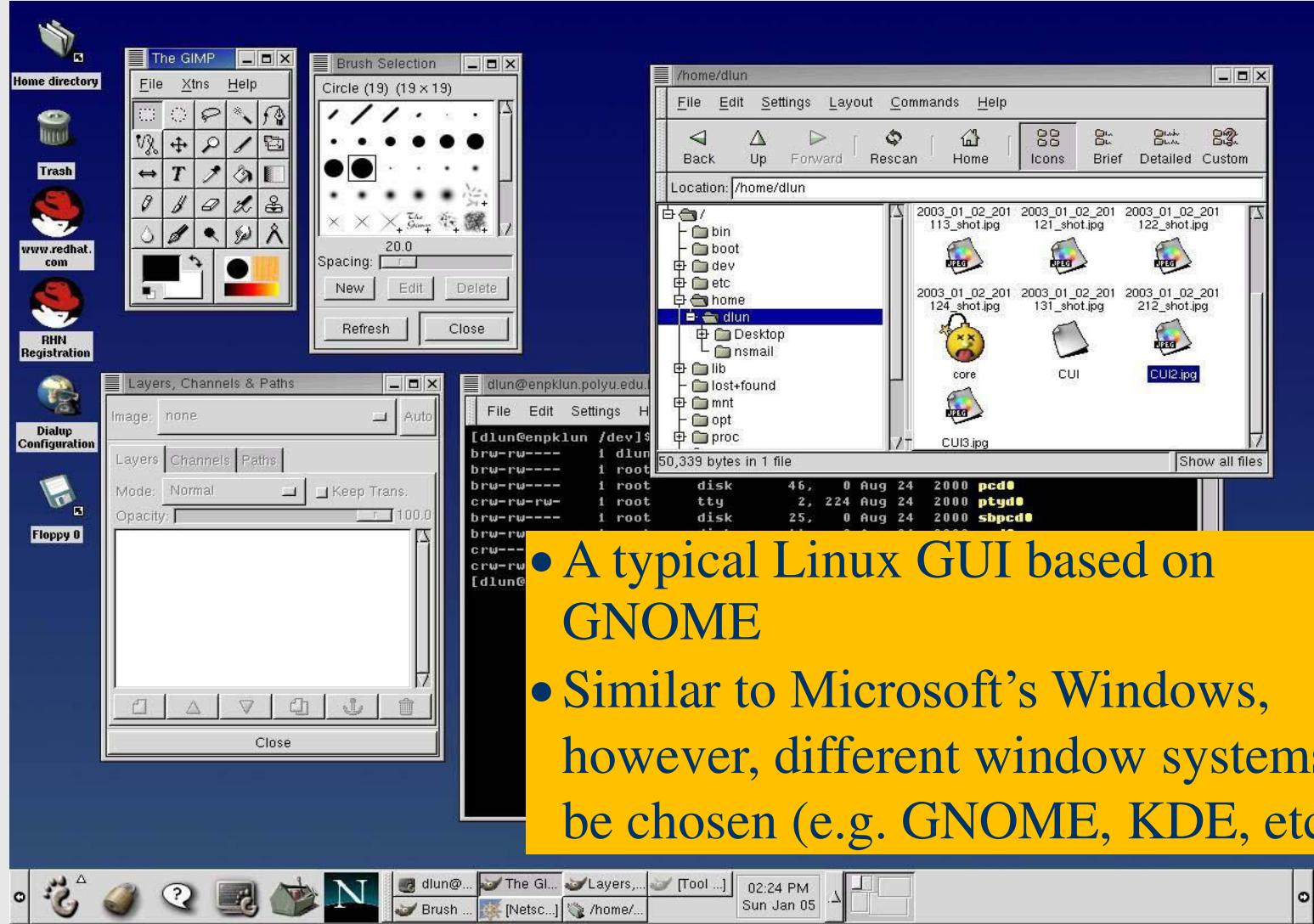
# Linux User Interface

- Traditional Linux (Unix also) uses command-driven interface (or text-based interface)
  - User need to type lines of command to instruct the computer to work, similar to DOS
  - **Advantage:** fast in speed. Very few resource is required for its implementation
  - **Disadvantages:** user needs to type, hence can easily make error. Besides, user needs to memorize all commands
  - Suitable for expert users and for the systems that interaction with user is not frequent, such as servers

# Linux User Interface

- By adopting the **X-Window technology**, graphical user interface (**GUI**) is available for Linux:
  - Uses pointing devices (e.g. mouse) to control the system, similar to Microsoft's Windows
  - Provide menu-driven and/or icon-driven interfaces
    - **menu-driven**: user is provided with a menu of choices. Each choice refers to a particular task
    - **icon-driven**: tasks are represented by pictures (icon) and shown to user. Click on an icon invokes one task
  - **Advantages**: No need to memorize commands.
    - Always select task from menus or icons
  - **Disadvantages**: Slow and require certain resource for its implementation
    - Suitable for general users and systems, such as PC

# Linux User Interface



- A typical Linux GUI based on GNOME
- Similar to Microsoft's Windows, however, different window systems can be chosen (e.g. GNOME, KDE, etc)

# Topics

- Introduction to Linux
- File System of the Linux
- General usage of Linux kernel & basic commands
- Linux users and group
- Permissions for file
- Directory and users
- Searching a file & directory
- **Zipping and Unzipping concepts**
- Linux for the Industry 4.0 Era
  - OPENIL and its advantages
  - Features of OPENIL

# ZIP command

- ZIP is a compression and file packaging utility for Unix.
- Each file is stored in single .zip {.zip-filename} file with the extension .zip.
- The zip program puts one or more compressed files into a single zip archive, along with information about the files (name, path, date, time of last modification, protection, and check information to verify file integrity). An entire directory structure can be packed into a zip archive with a single command.
- Compression ratios of 2:1 to 3:1 are common for text files.
  - zip has one compression method (deflation) and can also store files without compression.
  - zip automatically chooses the better of the two for each file to be compressed.

# ZIP Command

- **Syntax**
  - **zip [options] zipfile files\_list**
- **Extracting from .zip**
  - Unzip <filename.zip>
- **Possibilities:**
  1. **-d** removes files from a zip archive  
*Zip -d <filename.zip> <file.txt>*
  1. **-u** updates a file in zip archive  
*Zip -u <filename.zip> <file.txt>*
  3. **-m** deletes original files after zipping  
*Zip -m <filename.zip> file.txt*
  4. **-r** recursively zips files in a directory  
*Zip -r <filename.zip> <directory\_name>*
  5. **-v** verbose mode or prints diagnostic information  
*Zip -v <filename.zip> <file1.txt>*

# Gzip/gunzip Command

- Another popular compression algorithm that allows to reduce the size of a file and keep original file mode, ownership and timestamp.
- *.gz file format and uses gzip utility to compress and decompress files*
- By default, *gzip* keeps the original file name and timestamp in the compressed file
- *Syntax:*
  - **gzip <test.txt>**
  - Shall create a <test.txt.gz> and delete the original file
- For compress multiple files or directory into one file, first you need to create a Tar archive and then compress the .tar file with Gzip. A file that ends in .tar.gz or .tgz is a Tar archive compressed with Gzip.



## ➤ Gunzip synopsis

- **gunzip** [ **-acfhlLnNrtvV** ] [ **-S suffix** ] [ *name ...* ]
- *gunzip* also recognizes the special extensions **.tgz** and **.taz** as shorthands for **.tar.gz** and **.tar.Z** respectively.

## ➤ Possibilities:

1. **-k** allows to keep original file

*Gzip -k <filename>*

Or **-c**

*Gzip -c <filename> > filename.gz*

1. **-v** prints the percentage reduction and name of files processed

*Gzip -v <filename>*

3. Compress multiple files

*Gzip <file1> <file2> <file3>*

# gzip OPTIONS

- **-f –force**
  - Force compression or decompression even if the file has multiple links or the corresponding file already exists

# gzip OPTIONS

➤ *-l -list*

- For each compressed file, list the following fields:
  - compressed size: size of the compressed file
  - uncompressed size: size of the uncompressed file
  - ratio: compression ratio (0.0% if unknown)
  - uncompressed\_name: name of the uncompressed file

➤ *-n --no-name*

- When compressing, do not save the original file name and time stamp by default.
- When decompressing, do not restore the original file name if present (remove only the *gzip* suffix from the compressed file name) and do not restore the original time stamp if present.

# gzip OPTIONS

- **change the compression level**
- **-# --fast --best**
  - Regulate the speed of compression using the specified digit #
    - **-1 or --fast**
      - indicates the fastest compression method, less compression
    - **-9 or --best**
      - indicates the slowest compression method
      - best compression
    - The default compression level is **-6**
    - That is, biased towards high compression
  - **Gzip -9 <filename>**

# gzip decompression

- Compressed files can be restored to their original form using:
  - *gzip -d* or
  - *Gunzip* or
  - *Zcat*
    - `Gzip -d <filename.gz>`
- *gunzip* can currently decompress files created by *gzip*, *zip*, *compress*, *compress -H* or *pack*.
  - The detection of the input format is automatic.
- *gzip* and *gunzip* can also compress or decompress data from standard input and output
  - `ls -laR $HOME | gzip > filelist.gz`
  - Or by using *-c* option

# Linux Command bzip2

- bzip2, bunzip2
  - a block-sorting file compressor, v1.0.2
- bzcat
  - decompresses files to stdout
- bzip2recover
  - recovers data from damaged bzip2 files
- on the average about 10-20% better than *gzip*
  - at the expense of longer compression times
- Output with *.bz2* filename extension

# bzip2: Synopsis

- **bzip2** [ **-cdfkqstvzVL123456789** ] [ *filenames ...* ]
- **bunzip2** [ **-fkvsVL** ] [ *filenames ...* ]
- **bzcat** [ **-s** ] [ *filenames ...* ]
- **bzip2recover** *filename*
- Each file is replaced by a compressed version of itself, with the name "original\_name.bz2"

<https://www.geeksforgeeks.org/bzip2-command-in-linux-with-examples/>

<https://www.javatpoint.com/linux-bzip2-bunzip2>

<https://www.javatpoint.com/linux-bzcat-bzmore>

<https://www.geeksforgeeks.org/tar-command-linux-examples/>

# bzip2 OPTIONS

- **-c –stdout**
  - (as in gzip) Compress or decompress to standard output.
- **-t –test**
  - Check integrity of the specified file(s), but don't decompress them.
    - This really performs a trial decompression and throws away the result.
- **-k –keep**
  - Keep (don't delete) input files during compression or decompression.

# bzip2 OPTIONS

- **-s --small**
  - Reduce memory usage, for compression, decompression and testing.
- **-1 (or --fast) to -9 (or --best)**
  - Set the block size to 100 k, 200 k .. 900 k when compressing.
  - Has no effect when decompressing.

# bzip2 decompression

- **bzcat**
  - decompresses files to stdout
- *bunzip2*
- Cannot use bunzip2 to uncompress files compressed with gzip and vice versa
- **bzip2 -d**
  - decompresses all specified files.
  - Files which were not created by *bzip2* will be detected and ignored
    - warning issued.

# bzip2 decompression

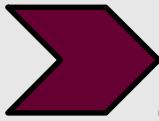
filename.bz2

filename.bz

filename.tbz2

filename.tbz

anyothername



filename filename

filename.tar

filename.tar

anyothername.out

# Linux Command: tar

- tar is a general-purpose archiving utility
  - Stands for Tape Archive
  - capable of packing many files into a single archive file,
  - while retaining information needed to restore the files fully, such as file permissions and ownership.

# tar options

- **-c**
  - Create a new archive
- **-x**
  - Extract files from an archive
- **-t**
  - List table of contents to an archive
- **-r**
  - append

# tar options

- **-u**
  - Update files
- **-d**
  - Compare files in the archive to those in the file system
- **-v**
  - – print verbose information
- **-k**
  - To keep any existing files when extracting
- **B *filename***
  - To specify that the tar file to be read or written is *filename*.

# Other commands

- Synopsis:

- compress [ -f ] [ -v ] [ -c ] [ -V ] [ -r ] [ -b bits ] [ name ... ]
- uncompress [ -f ] [ -v ] [ -c ] [ -V ] [ name ... ]
- zip [-aABcdDeEfFghjkllMoqrRSTuvVwXyz!@\$] [- b path] [-n suffixes] [-t mmddyyyy] [-tt mmddyyyy] [ zipfile [ file1 file2 ...]] [-xi list]
- unzip [-Z] [-cflptuvwxyz[abjnoqsCLMVX\$/:]] file[.zip] [file(s) ...] [-x xfile(s) ...] [-d exdir]

# Topics

- Introduction to Linux
- File System of the Linux
- General usage of Linux kernel & basic commands
- Linux users and group
- Permissions for file
- Directory and users
- Searching a file & directory
- Zipping and Unzipping concepts
- Linux for the Industry 4.0 Era
  - OPENIL and its advantages
  - Features of OPENIL

# OPENIL

- It is an open source software platform for industrial automation
- It provides following features:
  - Secure:
    - Industrial-grade security
    - Trusted computing
    - Hardened software
    - Cryptographic operations
    - End to end security
  - Real-Time
    - OpenIL provides a true real-time software platform
    - Real-time Linux kernel, professionally developed, tested and validated
  - Deterministic
    - OpenIL supports the latest Time-Sensitive Networking technology.
    - OpenTSN architecture runs on various hardware.



# OPENIL Advantages

- **100% OPEN SOURCE**
  - Linux distribution specifically developed, tested, and packaged for industrial systems
- **CROSS PLATFORM**
  - True cross-platform software platform, not limited to specific device or hardware architecture
- **INDUSTRY LEADING EXPERTS**
  - Technology experts and developers from industry-leading companies are actively building OpenIL software and solutions



# OPENIL Advantages

- WELL DOCUMENTED AND SUPPORTED
  - Development and user manuals are well documented with active community support.
- FIELD PROVEN
  - Every release is rigorously tested in industrial environments on production systems.
- CONTINUOUSLY EVOLVING
  - OpenIL targets broad industrial applications, and is always evolving.
  - [Go on to the OPENIL website](#)

# Summary

- Linux is an operating system that is built by a community of software developers around the world and led by its creator, Linus Torvalds
- It is derived originally from the UNIX operating system, but has grown beyond UNIX in popularity and power over the years
- Most Linux software projects are protected by one of a set of licenses that fall under the Open Source Initiative umbrella. The most prominent of these is the GNU Public License (GPL)
- The GNOME desktop environment has become the default desktop environment for many Linux systems, including Fedora and RHEL

# Summary

- To become an expert Linux user, you must be able to use the shell to type commands
- Commands for moving around the filesystem, copying files, moving files, and removing files are among the most basic commands you need to work from the shell
- Use the locate and find commands for finding files and grep for searching files



- Internet
  - Major players like Amazon, Google and Netflix all rely on Linux for delivering services
  - Facebook and Twitter each use a Linux variant that has been tweaked in-house
- Finance
  - A number of markets trade using solutions built on a Linux stack, including the Wall Street, NYSE\_Euronext, London Stock Exchange and Chicago Mercantile Exchange



- Insurance
  - AIG uses Red Hat and SUSE variants in its architecture
  - Health insurer, Cigna, uses Red Hat to build operating environments on Linux
- Healthcare
  - Red Hat skills come in handy for Linux system administration jobs at healthcare organizations such as Beth Israel Deaconess Medical Center
  - Many Debian “pure blends” — subsets of the Debian GNU/Linux distribution — address the specific needs of healthcare providers and researchers by packaging the kernel with drug databases or electronic medical record systems



- Other Industries
  - Government, education and military are other sectors increasing their reliance on Linux system administration
  - Innovative companies using Linux systems to support advances in autonomous vehicles, wearable devices, and other “smart” combinations of hardware and software
  - Startups and smaller businesses often choose the open-source OS for their operating servers because so many distributions are freely available



# Job Positions

- DevOps Engineer
- Java Developer
- Software Engineer
- Systems Administrator
- Systems Engineer
- Senior Software Engineer
- Python Developer
- Network Engineer
- Software Developer
- Linux Engineering Admin
- Software Developer
- Linux Engineer
- TechOps Engineer
- Senior Java Developer
- Build release management Engineer



# Linux Certifications

- Most of the hiring manager are looking to recruit Linux professionals.
- The emergence of open cloud platforms is creating increasing demand for Linux professionals who have the right expertise
- Linux-certified professionals always be a better position in the job market
- Employers are looking for more Linux talent.
- Better salary increments for Linux certified professionals



# Linux Certifications

- Some of the best Linux certification are given below:
  - Linux Foundation
  - Red Hat Linux SuSE Linux
  - Oracle
  - Linux Professional Institute (LPIC)
  - CompTIA

[Check it](#)

# References

- Linux Bible, The comprehensive Tutorial Resource, 8th Edition
- Linux: The Complete Reference, Sixth Edition
- Linux Programming and Scripting, Video Course NPTEL
- <https://www.openil.org/>



**Thanks!...**

**Any question?**

# MODULE I

## FILE SYSTEM OF LINUX

by: **Dr. Ram Paul Hathwal**

Dept of CSE, ASET, AUUP

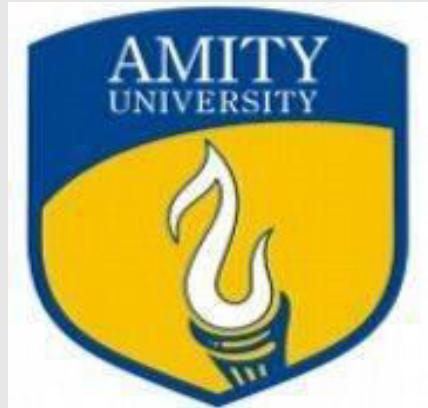


AMITY  
UNIVERSITY

# Linux for Devices

## CSE438

Department of Computer  
Science and Engineering



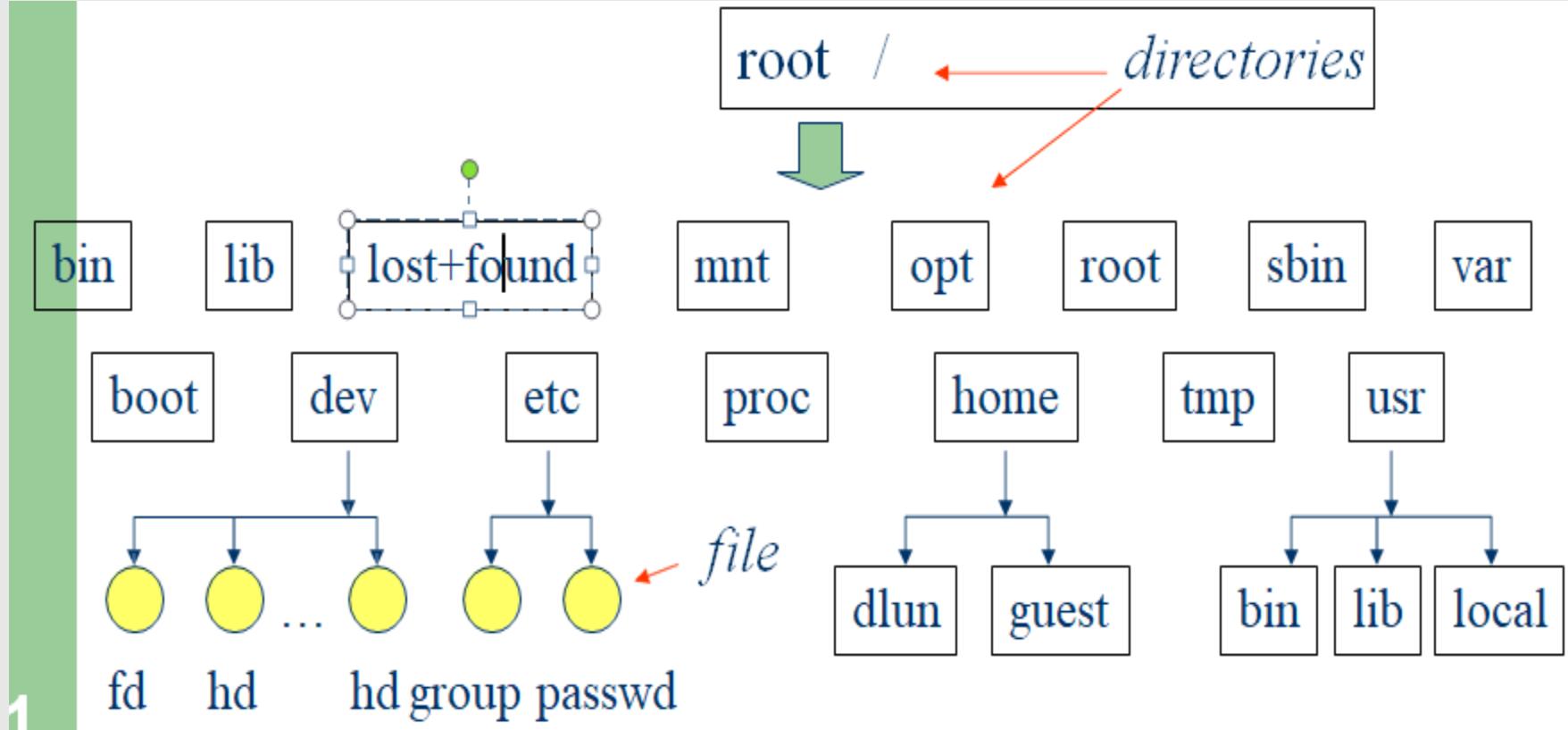
Dr. Ram Paul Hathwal  
Department of Computer Science and Engineering  
Amity University Uttar Pradesh

# Topics

- ✓ Introduction to Linux
  - **File System of the Linux**
  - General usage of Linux kernel & basic commands
  - Linux users and group
  - Permissions for file
  - Directory and users
  - Searching a file & directory
  - Zipping and Unzipping concepts
  - Linux for the Industry 4.0 Era
    - OPENIL and its advantages
    - Features of OPENIL

# Linux File System Structure

- According to the **File System Standard (FSSTND)** proposed in 1994, every LINUX system should contain a set of standard files and directories



- Root Directory ( / )
  - Top of the file system. Similar to \ in DOS
- /bin
  - Contain the binary (executable code) of most essential Linux commands, e.g. bash, cat, cp, ln, ls, etc.
- /boot
  - Contain all the files needed to boot the Linux system, including the binary of the Linux kernel. E.g., on Red Hat Linux 6.1, the kernel is in /boot/vmlinu-2.2.5-15 file
- /dev
  - Contain the special files for devices, e.g. fd0, hd0, etc.

- /etc
  - Contain host-specific files and directories, e.g. information about system configuration
  - /etc/passwd
    - This file contains login information of users in the system
    - For every user, one line of record is stored in the following format:

login\_name : dummy\_or\_encrypted\_password : user\_ID :  
group\_ID : user\_info : home\_directory : login\_shell

- E.g. rahul:x:134:105:MohanA Rahul:/home/rahul:/bin/bash
- **rahul** : login name
- **x** : means that it is a dummy password. The encrypted password is stored in /etc/shadow. This field can also be used to store the actual encrypted password. In any case, the original (unencrypted) password cannot be seen by anyone, including the administrator
- **134** : a user id given to that user. Range from 0 to 65535.
  - 0 is assigned to super-user. 1 to 99 are reserved
- **105** : a group id given to that user to indicate which group he belongs to. Range from 0 to 65535. 0 to 99 reserved
- **Mohan A Rahul** : user info, usually user's full name
- **/home/rahul** : home directory of the user
- **/bin/bash** : the location of the shell the user is using

- /home
  - Contain the **home directories of every user** in the system, e.g. dlun, guest, etc
- /lib
  - Store all **essential libraries** for different language compilers
- /lost+found
  - Contain all the files on the system **not connected to any directory**.
  - System administrator should determine the fate of the files in this directory

- /mnt
  - Use by system administrator to mount file systems temporarily by using the mount command
  - Before using any devices, they have to be **mounted** to the system for registration
  - For example, **after mounting a CD-ROM**, the file system in it will be **mapped to /mnt/cdrom directory**
  - User can then read and write files in the CD-ROM by accessing this directory
  - **Similar to mapping a drive letter to a CD-ROM in Windows**
  - Different from the special file in /dev. Special file is only a place where data of the CD-ROM is transferred or stored. No file system concept

- /opt
  - Use to install add-on software packages, e.g. star office, etc.
- /proc
  - Contain process and system information
- /root
  - Home directory of the user root, usually the administrator
- /sbin
  - The directories /sbin, /usr/sbin, and /usr/local/sbin contain **system administration tools, utilities and general root only commands**, such as halt, reboot and shutdown

- /tmp
  - Contain **temporary files**. Usually files in this directory will be deleted from time to time to avoid the system fills with temp files
- /usr
  - One of the largest sections of the Linux file system
  - Contain **read-only data that are shared between various users**, e.g. the manual pages needed for the command man. Stored in /usr/man direcrtry
- /var
  - Contain **data that keeps on changing** as the system is running. E.g. /var/spool/mail directory keeps the mail of user

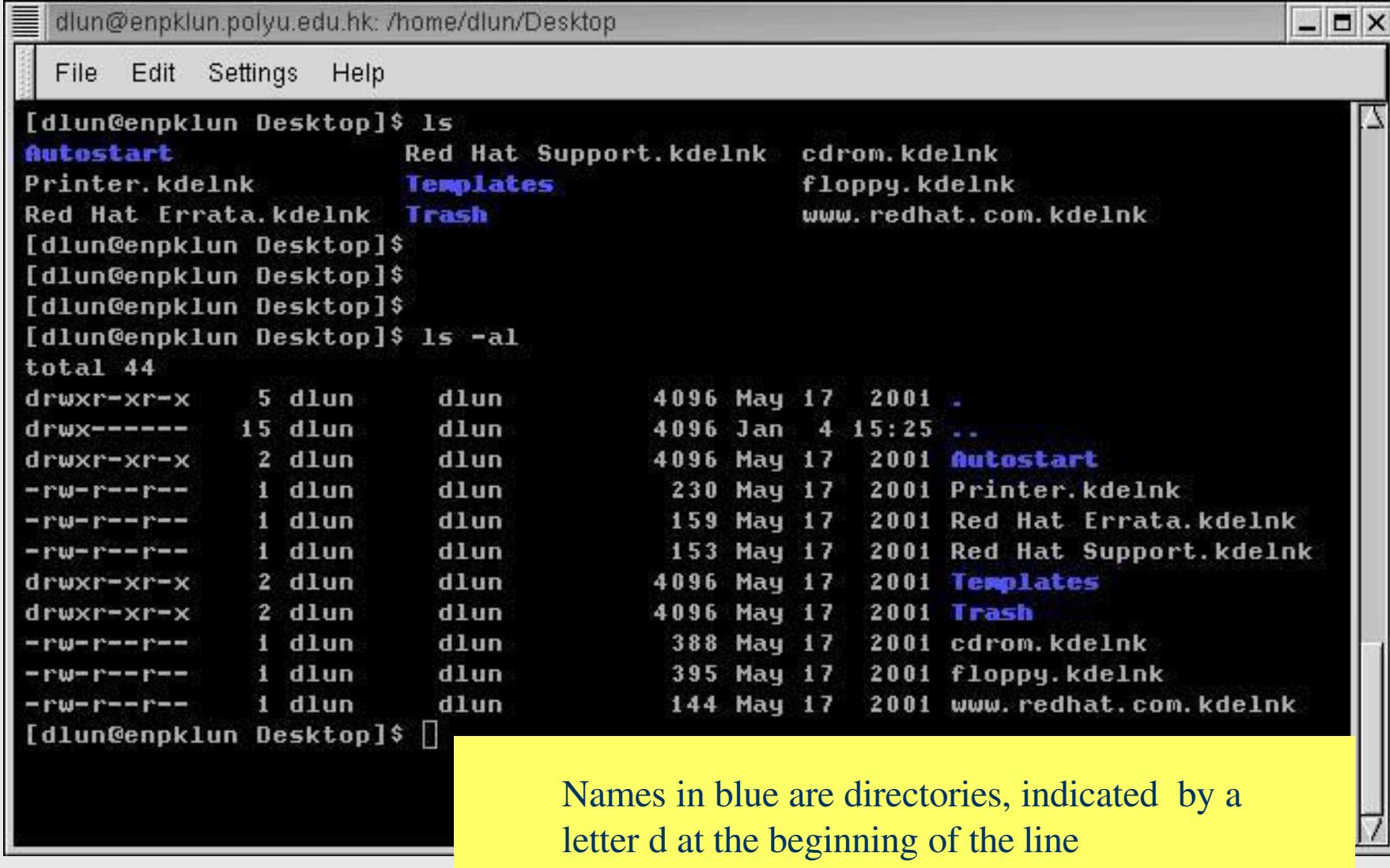
# Topics

- Introduction to Linux
- File System of the Linux
- General usage of Linux kernel & basic commands
- Linux users and group
- **Permissions for file**
- Directory and users
- Searching a file & directory
- Zipping and Unzipping concepts
- Linux for the Industry 4.0 Era
  - OPENIL and its advantages
  - Features of OPENIL

# File Management

- In Linux, file is defined as simply the thing that deals with a sequence of bytes
- Hence **everything are files**
  - An ordinary file is a file; a directory is also a file; a network card, a hard disk, any device are also files since they deal with a sequence of bytes
- Linux supports five types of files
  - simple/ordinary file (text file, c++ file, etc)
  - directory
  - symbolic (soft) link
  - special file (device)
  - named pipe (FIFO)

# Example Linux File



The screenshot shows a terminal window titled "dlun@enpkln.polyu.edu.hk: /home/dlun/Desktop". The window contains the following text:

```
[dlun@enpkln Desktop]$ ls
Autostart          Red Hat Support.kdeInk  cdrom.kdeInk
Printer.kdeInk        Templates             floppy.kdeInk
Red Hat Errata.kdeInk Trash                www.redhat.com.kdeInk

[dlun@enpkln Desktop]$
[dlun@enpkln Desktop]$
[dlun@enpkln Desktop]$
[dlun@enpkln Desktop]$ ls -al
total 44
drwxr-xr-x  5 dlun    dlun      4096 May 17  2001 .
drwx----- 15 dlun   dlun      4096 Jan  4 15:25 ..
drwxr-xr-x  2 dlun    dlun      4096 May 17  2001 Autostart
-rw-r--r--  1 dlun    dlun      230  May 17  2001 Printer.kdeInk
-rw-r--r--  1 dlun    dlun     159  May 17  2001 Red Hat Errata.kdeInk
-rw-r--r--  1 dlun    dlun     153  May 17  2001 Red Hat Support.kdeInk
drwxr-xr-x  2 dlun    dlun      4096 May 17  2001 Templates
drwxr-xr-x  2 dlun    dlun      4096 May 17  2001 Trash
-rw-r--r--  1 dlun    dlun      388  May 17  2001 cdrom.kdeInk
-rw-r--r--  1 dlun    dlun      395  May 17  2001 floppy.kdeInk
-rw-r--r--  1 dlun    dlun      144  May 17  2001 www.redhat.com.kdeInk

[dlun@enpkln Desktop]$
```

A yellow callout box points to the word "Templates" in the second line of the output, which is highlighted in blue. The text inside the callout box reads: "Names in blue are directories, indicated by a letter d at the beginning of the line".

- Symbolic (soft) link
  - Not a real file, just a **link** to another file
  - Allow giving another name to a file without actually duplicates it
    - hence **save memory space**
- Special file (device)
  - Each hardware device, e.g. keyboard, hard disk, CD-ROM, etc is associated with at least one file
  - Usually stored in **/dev** directory
  - Applications can read and write any devices by reading and writing their associated file – hence the **access method** is known as **device independent**
  - Divide into two types: character special files, e.g. keyboard, and block special files, e.g. disk

Kernel

# Linux File

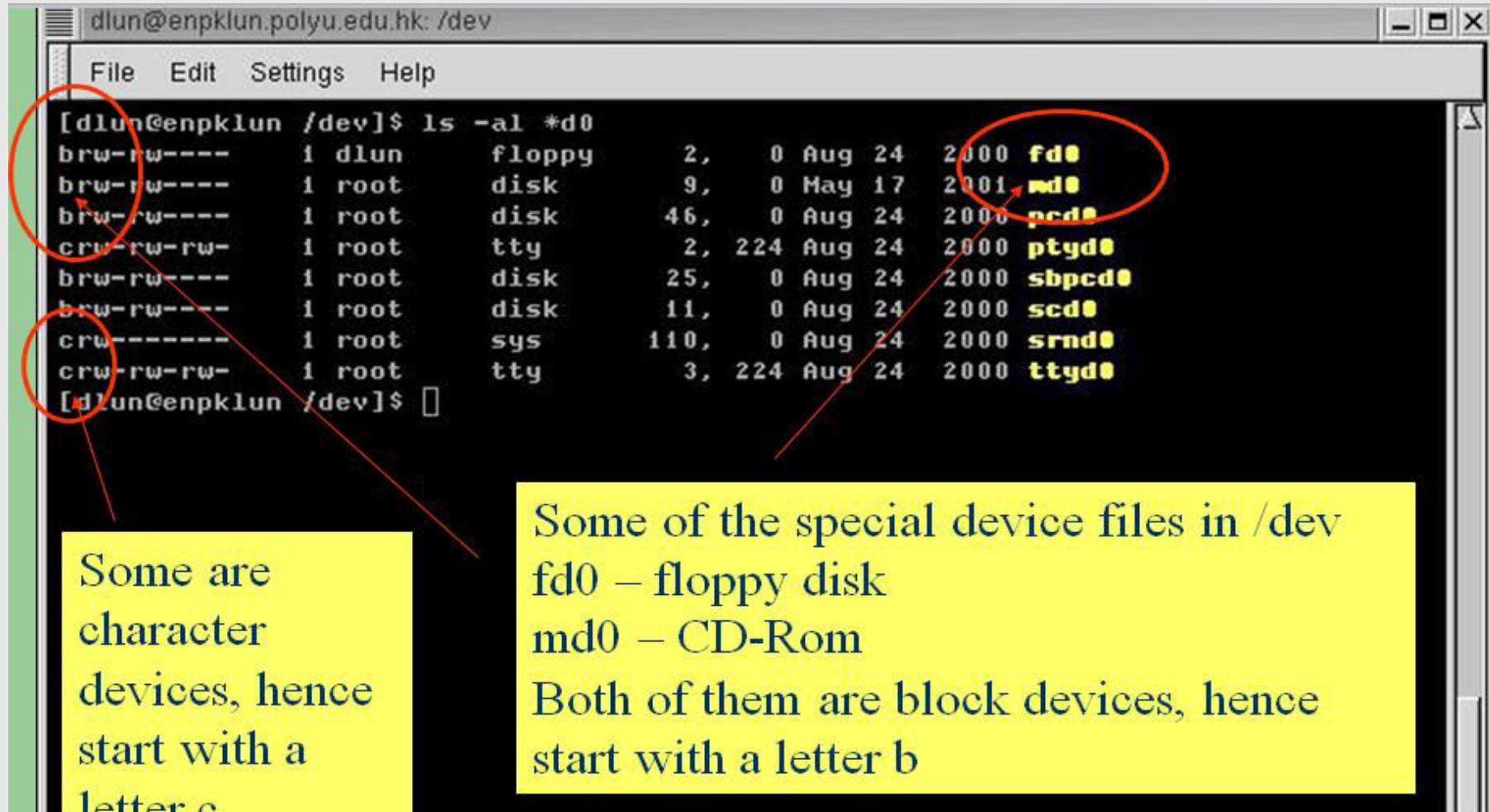
```
dlun@enpklun.polyu.edu.hk: /home/dlun/Desktop
File Edit Settings Help
[dlun@enpklun Desktop]$ ln -s .. /CUI anotherCUI
[dlun@enpklun Desktop]$ ls -al
total 44
drwxr-xr-x  5 dlun    dlun        4096 Jan  4 18:36 .
drwx----- 16 dlun    dlun        4096 Jan  4 18:26 ..
drwxr-xr-x  2 dlun    dlun        4096 May 17 2001 Autostart
-rw-r--r--  1 dlun    dlun       230 May 17 2001 Printer.kdelnk
-rw-r--r--  1 dlun    dlun      159 May 17 2001 Red Hat Errata.kdelnk
-rw-r--r--  1 dlun    dlun      153 May 17 2001 Red Hat Support.kdelnk
drwxr-xr-x  2 dlun    dlun        4096 May 17 2001 Templates
drwxr-xr-x  2 dlun    dlun        4096 May 17 2001 Trash
lrwxrwxrwx  1 dlun    dlun          6 Jan  4 18:36 anotherCUI -> .. /CUI
-rw-r--r--  1 dlun    dlun      300 May 17 2001 cdrom.kdelnk
-rw-r--r--  1 dlun    dlun      395 May 17 2001 floppy.kdelnk
-rw-r--r--  1 dlun    dlun      144 May 17 2001 www.redhat.com.kdelnk
[dlun@enpklun Desktop]$ 
```

symbolic link to a file  
called CUI to anotherCUI

A symbolic link begins with a letter *l*

File size is only 6 bytes

# Linux File



```
[dlun@enpklun ~]$ ls -al *d0
brw-rw---- 1 dlun    floppy      2,   0 Aug 24 2000 fd0
brw-rw---- 1 root     disk        9,   0 May 17 2001 md0
brw-rw---- 1 root     disk       46,   0 Aug 24 2000 pcd0
crw-rw-rw- 1 root     tty         2, 224 Aug 24 2000 ptyd0
brw-rw---- 1 root     disk       25,   0 Aug 24 2000 sbpcd0
brw-rw---- 1 root     disk       11,   0 Aug 24 2000 scd0
crw----- 1 root     sys        110,   0 Aug 24 2000 srnd0
crw-rw-rw- 1 root     tty         3, 224 Aug 24 2000 ttyd0
[dlun@enpklun ~]$
```

Some are character devices, hence start with a letter c.

Some of the special device files in /dev  
fd0 – floppy disk  
md0 – CD-Rom  
Both of them are block devices, hence start with a letter b



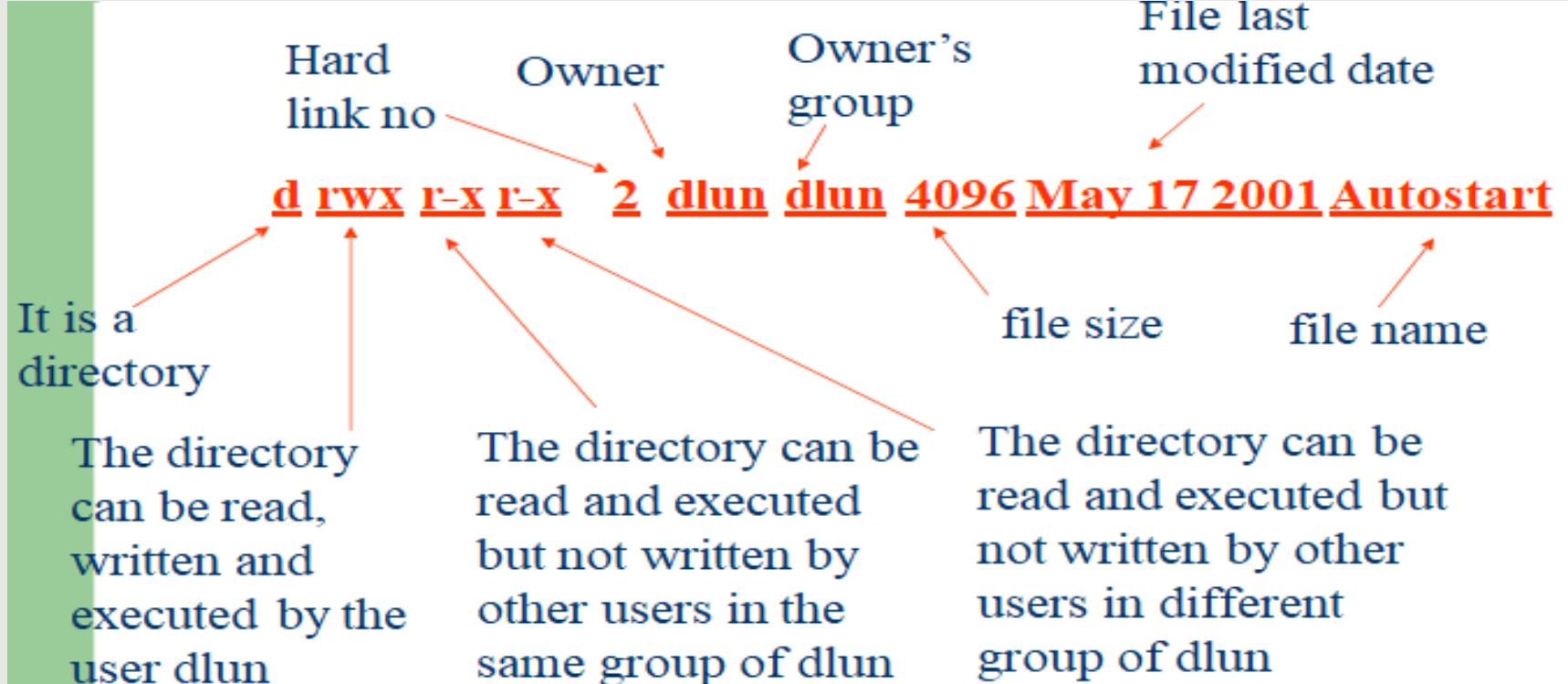
- Linux is a multiuser system, the files of all users are stored in a single file structure
- Mechanism is required to restrict one user to access the files of another user, if he is not supposed to
- User can impose **access permission** to each file to restrict its access
- The term “access permission” refers to
  - read permission
  - write permission
  - execute permission

# Linux File Access Privilege

```
File Edit Settings Help
[dlun@enpkln Desktop]$ ln -s ../CUI anotherCUI
[dlun@enpkln Desktop]$ ls -al
total 44
drwxr-xr-x    5 dlun    dlun   4096 Jan  4 18:36 .
drwx----- 16 dlun    dlun   4096 Jan  4 18:26 ..
drwxr-xr-x    2 dlun    dlun   4096 May 17 2001 Autostart
-rw-r--r--    1 dlun    dlun   230  May 17 2001 Printer.kdeLink
-rw-r--r--    1 dlun    dlun   159  May 17 2001 Red Hat Errata.kdeLink
-rw-r--r--    1 dlun    dlun   153  May 17 2001 Red Hat Support.kdeLink
drwxr-xr-x    2 dlun    dlun   4096 May 17 2001 Templates
drwxr-xr-x    2 dlun    dlun   4096 May 17 2001 Trash
lrwxrwxrwx    1 dlun    dlun      6 Jan  4 18:36 anotherCUI -> ../CUI
-rw-r--r--    1 dlun    dlun   388  May 17 2001 cdrom.kdeLink
-rw-r--r--    1 dlun    dlun   395  May 17 2001 floppy.kdeLink
-rw-r--r--    1 dlun    dlun   144  May 17 2001 www.redhat.com.kdeLink
[dlun@enpkln Desktop]$ 
```

The file access permission can be seen by using the command `ls -l` or `ls -al`

# Linux File Access Privilege





- Access permission can also be assigned to a directory
- Directory is also a file that contains the attributes of the files inside it.
- If **read permission** is not given to a directory
  - cannot show the structure of this directory
  - e.g. cannot use ls
- If **write permission** is not given to a directory
  - cannot modify anything of the directory structure
  - e.g. cannot copy a file into this directory since it will modify the directory structure by adding one more file
- If **execute permission** is not given to a directory
  - nearly nothing can be done with this directory, even cd



- The access permission of a file or directory can be changed by using the command

**chmod xyz filename/directory name**

- xyz refers 3 digit in octal form
  - E.g.

**660 : 110 110 000**

**⇒ rw- rw- ---**

**545 : 101 100 101**

**⇒ r-x r-- r-x**

# Linux File Access Privilege

```
File Edit Settings Help
[dlun@enpklun test]$ ls -l
total 12
-rw-r--r--    1 dlun      dlun          395 Jan  7 16:36 floppy.kdeLink
drw-----    2 dlun      dlun          4096 Jan  9 11:06 temp
-rw-rw-r--    1 dlun      dlun         16 Jan  7 16:05 testi.txt
[dlun@enpklun test]$
[dlun@enpklun test]$
[dlun@enpklun test]$ cd temp
bash: cd: temp: Permission denied
[dlun@enpklun test]$
[dlun@enpklun test]$
[dlun@enpklun test]$ chmod 700 temp
[dlun@enpklun test]$
[dlun@enpklun test]$ ls -l
total 12
-rw-r--r--    1 dlun      dlun          395 Jan  7 16:36 floppy.kdeLink
drwx-----    2 dlun      dlun          4096 Jan  9 11:06 temp
-rw-rw-r--    1 dlun      dlun         16 Jan  7 16:05 testi.txt
[dlun@enpklun test]$ cd temp
[dlun@enpklun temp]$ 
```

temp does not have execution right

even cd is not workable

execution right is added

now we can change the directory to temp



Symbol	Meaning
u	User
g	Group
o	Other
a	All
r	read
w	Write
x	Execute
+	Add permission
-	Take away permission

# Topics

- Introduction to Linux
- File System of the Linux
- General usage of Linux kernel & basic commands
- Linux users and group
- Permissions for file
- Directory and users
- **Searching a file & directory**
- Zipping and Unzipping concepts
- Linux for the Industry 4.0 Era
  - OPENIL and its advantages
  - Features of OPENIL

# Searching for Binary Files

## ➤ \$ which <binary file>

- If you are able to run a binary file and want to know where does this binary exist

## ➤ \$ *which shutdown*

```
17 root      rt  0      0      0      0      0 S  0.0  0.0  0:00.17 migration/
vibha@vibha-Inspiron-5547:~$ which shutdown
/usr/sbin/shutdown
vibha@vibha-Inspiron-5547:~$ █
```

## ➤ Note that this command searches for the binary using the \$PATH environment variable

- This means if the \$PATH does not have the folder of the binary, *which* will not be able to find it
- In other words, if you can not run the command, *which* will not be able to locate it

- \$ locate <filename>
- \$ locate <part of the path or filename>
- The *locate* command searches for the required file based on a pre-prepared database of all files in the system along with their full path
- When performing a search using *locate* command, the database is searched for the passed string
- Hence, you can search based on any string, even if it is only a part of the file path
- \$ *locate in/zi*
- This would result in,
- /usr/bin/zip
- /usr/bin/zipcloak
- /usr/bin/zipsplit

# Database for “locate”

- The database used by the locate command is created by the program ***updatedb***
- This program runs in a cron job that periodically (such as once daily) updates the database
- Accordingly the database may have incorrect or incomplete info,
  - Newly created files may not be in the database for sometime
  - Recently deleted files may still be in the database
  - Recently moved files, may be in the database using its old location
- This means that these changes will be missed in the search
- If you want to update the index manually,
  - ***\$ sudo updatedb***

# “locate” Command

- The **locate** command is a very useful command because:
  - It is very simple
  - You can search for a file based on its name, part of its name, part of its path, ...etc
  - It is very fast command (because it uses a pre-prepared database)
- However, it comes with some limitations,
  - It can only search based on file name (or path name)
  - It may not find new files
  - It may result in wrong results since it does not use the current tree in its search (instead it uses an old snapshot of the tree)
- The **locate** command is useful for searching for system files or traditional Linux files but can not be used in a lot of other scenarios

# A Smarter Search (**find** Command)

- The **find** is a very rich command that can search on files using different search filters, for example,
  - Search for a file based on its name
  - Search for a file based on its size
  - Search for a file based on its last modification date
  - Search for a file based on its type (normal file, directory, symbolic link, ....)
  - Search for a file based on other criteria
  - A combination of all of those search criteria
  - Limit the search scope to a certain directory (and its subdirectories)
  - Skip Some subdirectories from the search scope
- This command also allows for performing an action on all the files that match the search criteria

# Selecting the Search Scope

- The find command performs its search within the given scope
- For example:
  - **\$ find ~**  
This will list all files in the home directory and its subdirectories
  - **\$ find ~ | wc -l**  
Count files under my home directory
  - **\$ find /**  
List all files in the system starting from the root directory  
You may need to run this command with *sudo* to be able to search all files
  - **\$ find /bin /sbin /usr/bin**  
List all files in those 3 directories (scope can contain multiple directories)
  - **\$ find ~/Documents | grep ".pdf"**  
Find all pdf files under the Documents folder
  - **\$ find . > file-list.txt**  
Store a list of files under the current directory in the specified file

# Adding Filters to the Search

- So far, the ***find*** command finds all the files in the specified path, and lists them
- To limit the search to some files, we had to pipe the result to the ***grep*** command
- This gives a similar effect to the ***locate*** command (except that it is up to date)
- But we need more search options than searching by the filename or the path
- To do that, we need to add Search Filters to the ***find*** command



- **\$ find <scope> -type <type>**
- To generate a list of directories in the user home directory  
**\$ find ~ -type d**
- To generate a list of files only in the user home directory  
**\$ find ~ -type f**
- To generate a list for all symbolic links in the system,  
**\$ find / -type l**
- For Device files
  - \$ find / -type c**      (character devices)
  - \$ find / -type b**      (block device)

# Searching by File Name

- **\$ find <scope> -name <file name pattern>**
- **\$ find <scope> -iname <file name pattern>**
  
- To search for a file by its name
- **\$ find ~ -name my-file.txt**
- You can use a pattern for the file name (or even a regular expression)
- **\$ find ~ -name “\*.xml”**
- **\$ find ~ -name \\*\.\xml**
- For case insensitive search,
- **\$ find / -iname “\*.html”**
- **\$ find / -iname \\*\.\html**

# Searching by File Size

- **\$ find <scope> -size <file Size filter>**
- You can search by file size
- **\$ find ~ -size +1M**
  - + means bigger than
  - - means smaller than
  - Neither means, exactly this size
  - M means Megabyte
  - G means Gigabyte
  - K means Kilobyte
  - w means words (2 bytes)
  - c means characters (1 byte)
  - b means blocks (512 byte), this is the default, if no unit is passed

# And Much More

- The find command is a very rich command, and its use is not limited to the described scenarios
- Filters can be also using date of creation, date of modification, owning user, owning group, permissions, *etc* ...
  - Examples:
    - *\$ find ~ -perm 777*
    - *\$ find . -user root*
    - *\$ sudo find / -mtime -50*
    - *\$ sudo find / -atime*

# Mixing Search Criteria

- You can use multiple search filters to get a more sophisticated search criteria as follows,
  - To have multiple conditions “**ANDed**” together, which means a match happens only if all the conditions are satisfied
  - To have multiple conditions “**ORed**” together, which means a match happens if any of the conditions is satisfied
  - You can have any of the conditions **reversed**
  - You can have a more complicated search expressions using **parentheses**

# Using Multiple “ANDed” Filters

- You can have “ANDed” filters, in which a file match has to satisfy all the criteria
  - Just put the filters back to back
    - **\$ find ~ -type f -name “\*.jpg” -size +10M**
    - Use ‘-a’
      - **\$ find ~ -name ‘\*.jpg’ -a -size +10M**
    - Use ‘-and’
      - **\$ find ~ -name ‘\*.jpg’ -and -size +10M**



# Using Multiple “ORed” Conditions

- You can have “ORed” filters, in which a file match needs to satisfy any one of the filters
  - Use ‘-o’
    - *\$ find ~ -name ‘\*.jpg’-o -size +10M*
  - Use ‘-or’
    - *\$ find ~ -name ‘\*.jpg’-or -size +10M*

# Inverting Conditions

- You can have one of the filters inverted in the search criteria
  - Using “!”
    - *\$ find ~ \! -name “\*.pdf”*
  - Using “-not”
    - *\$ find ~ -not -name “\*.pdf”*

# Mix & Match

- You can mix and match any of the filters described so far to get a more sophisticated filter expression
- The use of parentheses “( )” becomes necessary as the expression gets more complicated
- Keep in mind that the parentheses need to be escaped
  - Use **\( expression \)**
  - *Example:*
  - **\$ find ~ \( -name tmp -o -name \*.txt \)**
  - **\$ find ~ \( -name tmp -o \( -not -user root\)\ )**

# Find & Perform Action

- The default action for the **find** command is to print the list of files matching the search criteria (file name with full path)
- However, you can set another action to apply on those files that meet the search criteria
  - Print the names with path (Default action)
    - **\$ find ~ -type d -print**
  - Delete the files
    - **\$ find ~ -name “\*.pdf” -delete**
  - Perform a detailed list of the files
    - **\$ find ~ -size +10M -ls**
  - Quit the search on the first match. This can be used if the purpose of the search is to just find if any file meets this criteria
    - **\$ find ~ -size +100M -quit**

# Skipping Files (The -prune Action)

- The **find** command performs its search recursively in all sub-directories
- In case we need to skip some search results, we can use a search filter with **the -prune** action with the other search filters
- Example:
  - *\$ find ~ -type l -prune -o -name '\*txt' -print*
  - *\$ find . -path ./misc -prune -o -name '\*txt' -print*
  - *\$ find . -type d \(-path dir1 -o -path dir2 -o -path dir3 \) -prune -o -print*



**Thanks!...**

**Any question?**



## MODULE II

# LINUX NETWORKING AND DOCKER

by: Dr. Ram Paul Hathwal  
Dept of CSE, ASET, AUUP

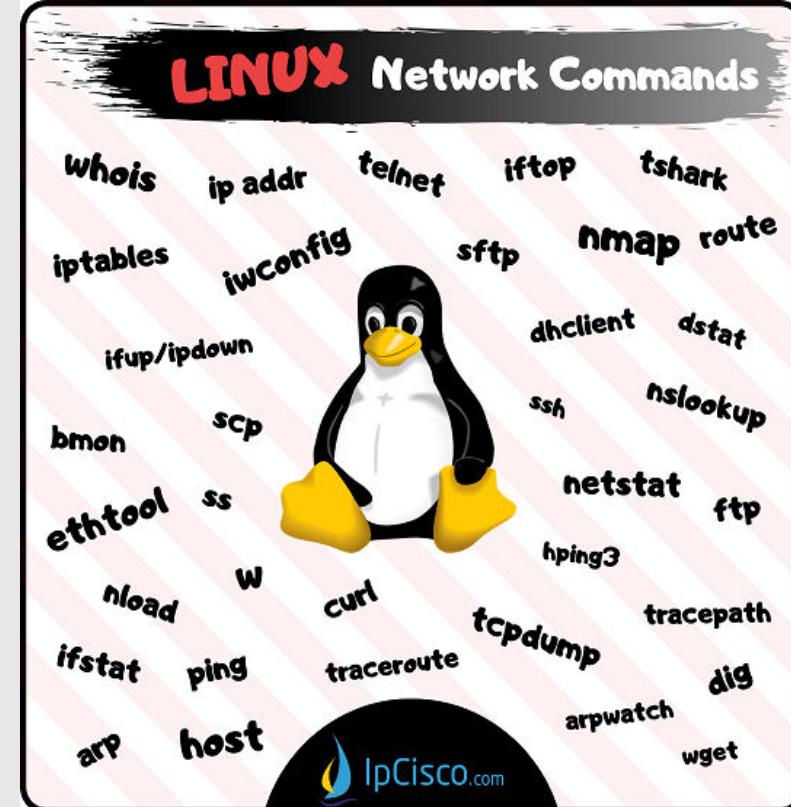


# Learning Objectives & Outcomes

Sub Topic	Learning Objectives	Learning Outcomes
Introduction to Networking in Linux	To understand the preliminaries of networking in Linux	Achieve glimpse of Networking in Linux
Network basics and tools	To understand network basics and tools	Understanding the basics of Network and tools involved
File transfer protocol in Linux	To understand File Transfer Protocol in Linux operating system	Understanding the usage of FTP in Linux OS
Network file system	To have an insight into Network file system	Understand the File system in networks
Domain Name Services	To understand DNS and its utility	Understanding of Domain name services
Dynamic hosting configuration Protocol	To understand the utility and importance of DHCP protocol	Understanding of Dynamic host configuration protocol
Network Information Services	To achieve understanding of Network Information Services	Understand network information services in Linux environment

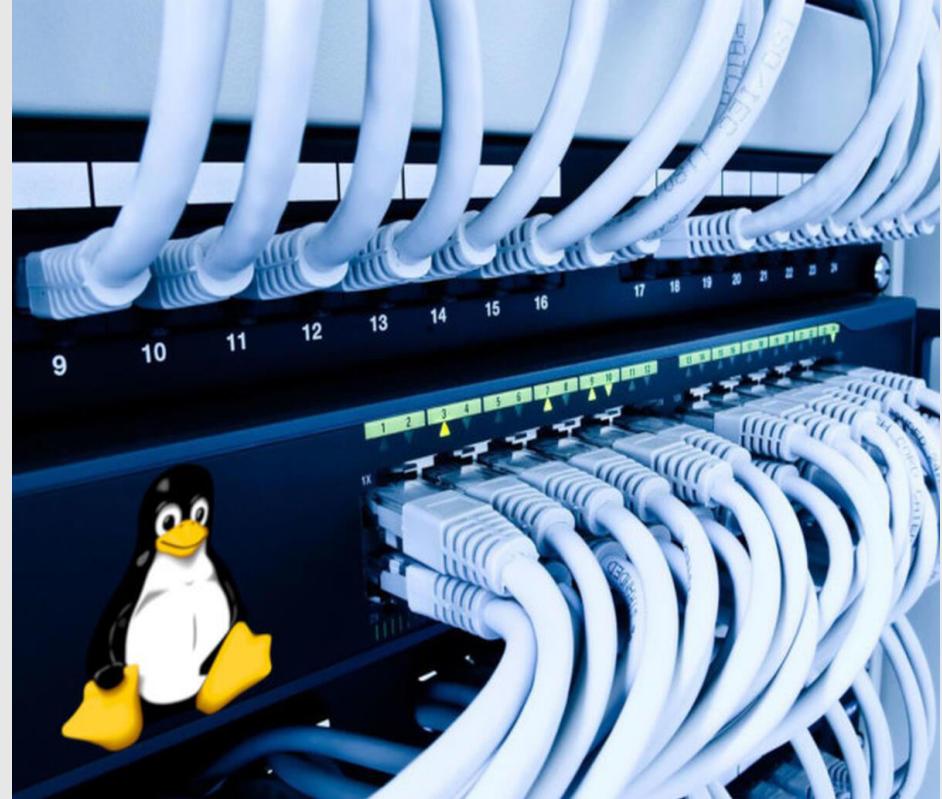
# Outline

- ✓ Introduction to Networking in Linux
- Network basics and tools
- File transfer protocol in Linux
- Network file system
- Domain Name Services
- Dynamic Host Configuration Protocol
- Network Information Services



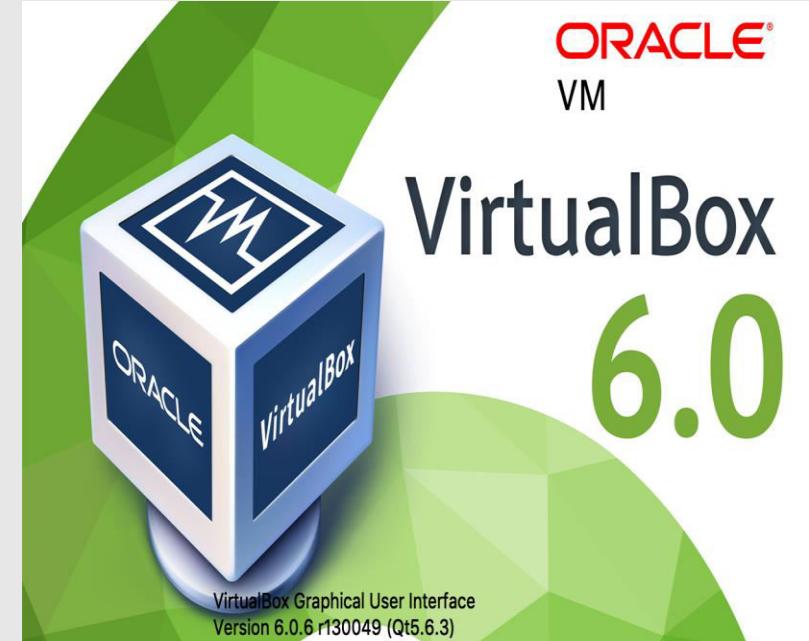
# Networking: Motivation

- Networking is an essential tool for Linux users/developers:
  - Setup machine to be able to connect to the Internet
  - Identify machine information such as interface, IP address, etc
  - Copy files to/from a remote machine
  - Remote access a remote machine



# Internet Access to VM

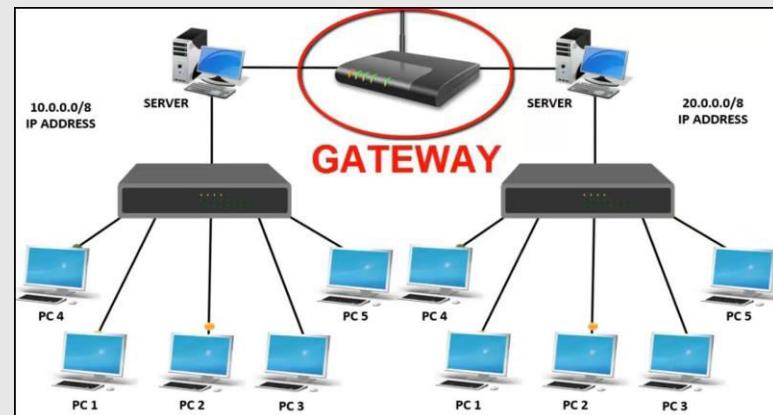
- Open **Virtualbox Manager**
- Select the machine you cannot get internet on in the left pane
- Click the **Settings** button in the Top menu
- Click **Network** in the left pane in the settings window
- Switched to **Bridged Adaptor** in the **attached to** drop-down menu
- Hit **OK** to save your changes
- Start your VM



# Networking: Introduction

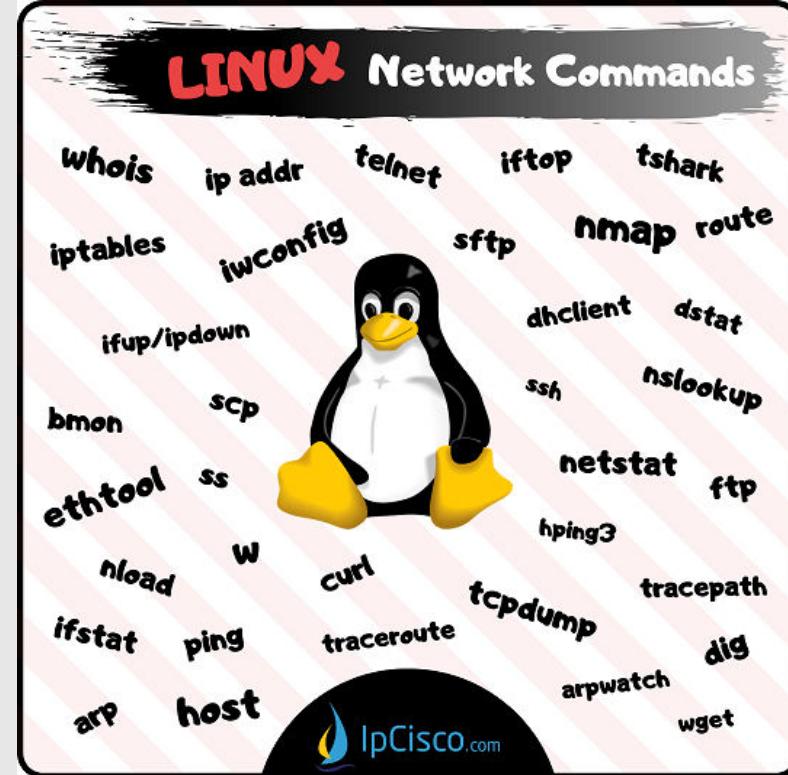
- **IP Address:** a unique string of numbers separated by periods that identifies each computer using the Internet Protocol to communicate over a network
- **Subnet mask** is a 32-bit number that masks an IP address and divides the IP address into network and host address
- Subnet mask is made by setting network bits to all "1"s and setting host bits to all "0"s.
- **Gateway**
- **Static Vs DHCP**
- **Interface: NIC**
- **Interface MAC**

	IP Address	netmask
Class A	16.1.1.1	255.0.0.0
	└── network    host	
Class B	132.147.1.1	255.255.0.0
	└── network    host	
Class C	221.138.62.1	255.255.255.0
	└── network    host	

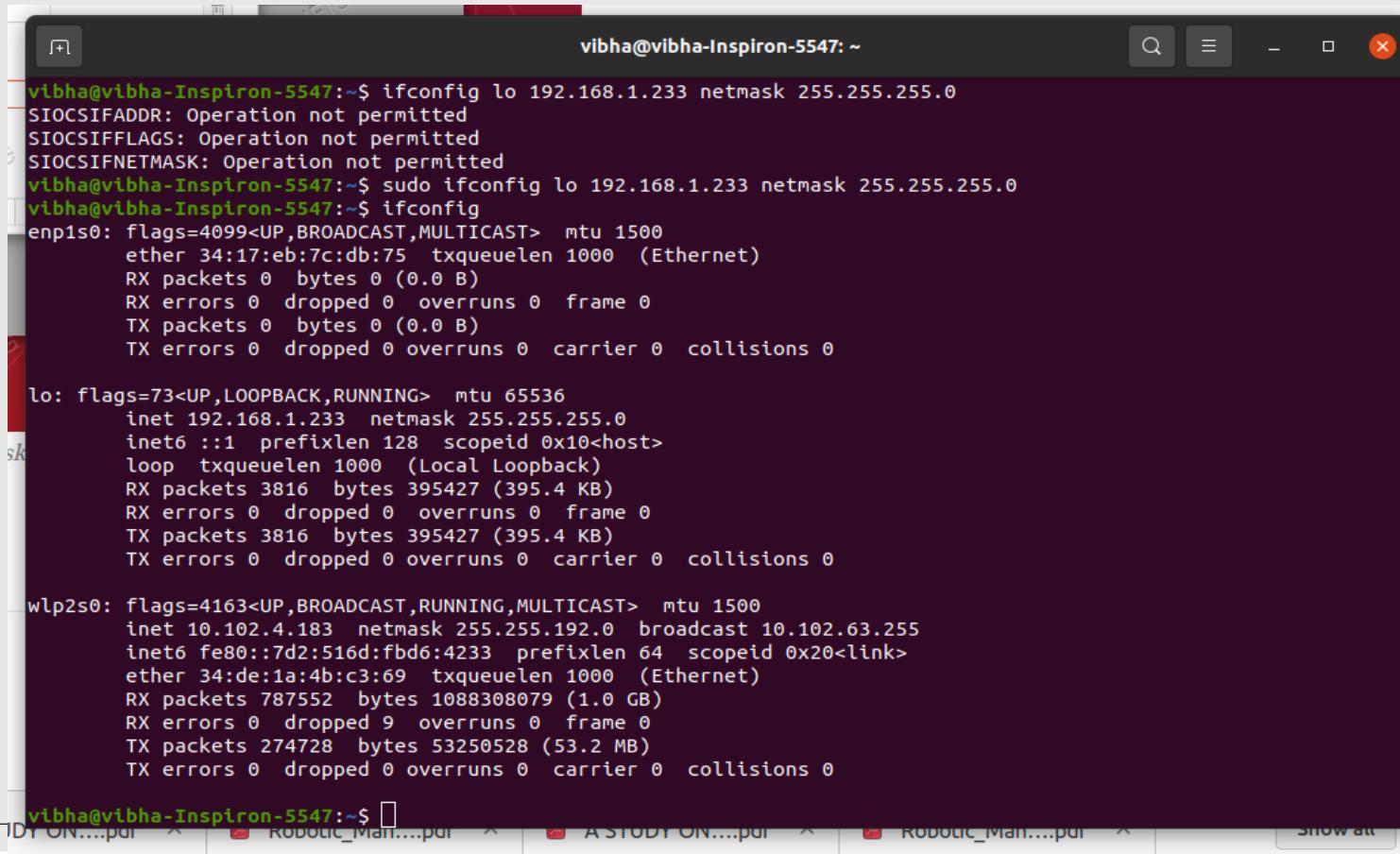


# Outline

- ✓ Introduction to Networking in Linux
- ✓ Network basics and tools
- File transfer protocol in Linux
- Network file system
- Domain Name Services
- Dynamic Host Configuration Protocol
- Network Information Services



- Interface Detection
- Assigning an IP address



The screenshot shows a terminal window titled "vibha@vibha-Inspiron-5547:~". The user runs several commands related to network interfaces:

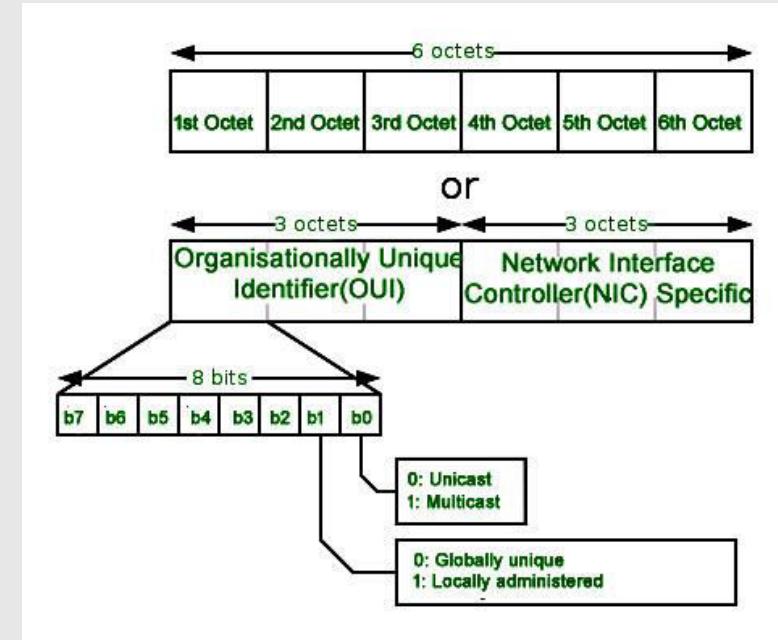
- `ifconfig lo 192.168.1.233 netmask 255.255.255.0`: This command attempts to change the IP and netmask of the loopback interface (lo). It fails with the error "SIOCSIFADDR: Operation not permitted".
- `sudo ifconfig lo 192.168.1.233 netmask 255.255.255.0`: This command uses sudo to run the same command successfully, changing the IP and netmask of the loopback interface.
- `ifconfig`: This command lists all network interfaces. It shows three interfaces: enp1s0 (Ethernet), lo (loopback), and wlp2s0 (wireless).
- `enp1s0`: Details for the Ethernet interface. It has flags indicating it is up, broadcast, and multicasted. MTU is 1500. MAC address is 34:17:eb:7c:db:75. Queueing discipline is txqueuelen 1000. RX and TX statistics show 0 errors, 0 dropped packets, 0 overruns, and 0 collisions.
- `lo`: Details for the loopback interface. It has flags indicating it is up, loopback, and running. MTU is 65536. It has two inet entries: one for the loopback address (inet 127.0.0.1) and one for the broadcast address (inet 127.255.255.255). Queueing discipline is txqueuelen 1000. RX and TX statistics show 0 errors, 0 dropped packets, 0 overruns, and 0 collisions.
- `wlp2s0`: Details for the wireless interface. It has flags indicating it is up, broadcast, running, and multicasted. MTU is 1500. MAC address is 34:de:1a:4b:c3:69. Queueing discipline is txqueuelen 1000. RX and TX statistics show 0 errors, 9 dropped packets, 0 overruns, and 0 collisions.

- Interface configuration files
  - /etc/nsswitch.conf - where it should resolve hostname to IP address
  - /etc/hostname - defines system Ip address and hostname
  - /etc/sysconfig/network - specifies hostname
  - /etc/sysconfig/network-scripts/ifcfg-nic – specify IP address, subnet mask and gateway
  - /etc.resolve.conf - specifies DNS server
- Network Commands
  - Ping
  - Ifconfig
  - Ifup or ifdown
  - Netstat –displays kernal IP routing table, displays destination, gateway and interface information
  - tcpdump

# The MAC Address

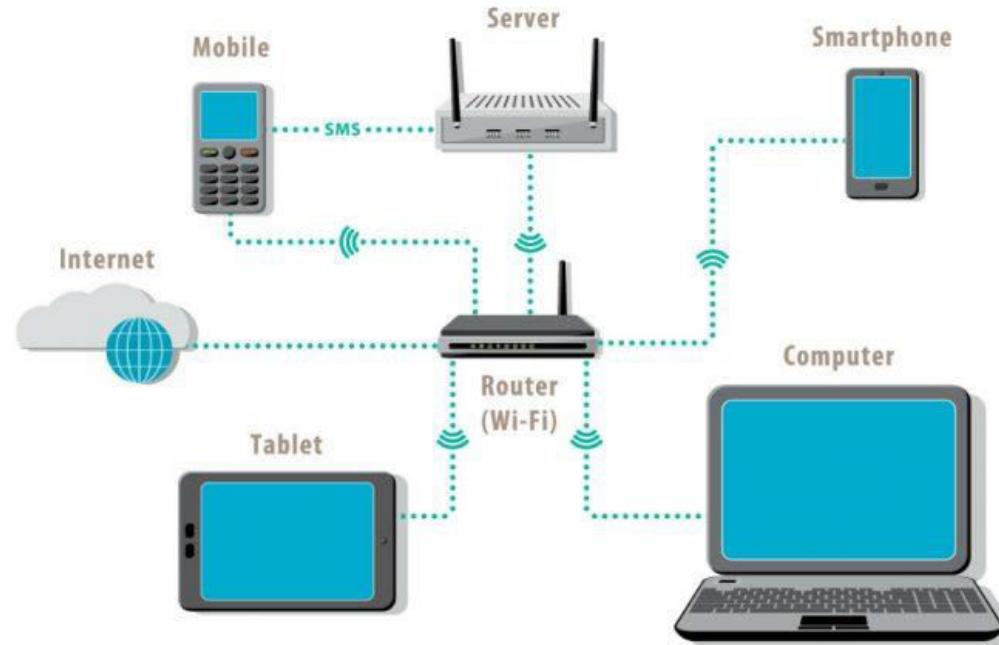
- The MAC address or hardware address, is a unique address for each NIC card.
- 48 bit number written in hexadecimal format such as: **68:05:CA:03:19:9C**

```
vibha@vibha-Inspiron-5547: /  
64 bytes from del12s04-in-f14.1e100.net (142.250.194.110): icmp_seq=13482 ttl=120 time=6.93 ms  
64 bytes from del12s04-in-f14.1e100.net (142.250.194.110): icmp_seq=13483 ttl=120 time=10.3 ms  
64 bytes from del12s04-in-f14.1e100.net (142.250.194.110): icmp_seq=13484 ttl=120 time=51.1 ms  
64 bytes from del12s04-in-f14.1e100.net (142.250.194.110): icmp_seq=13485 ttl=120 time=14.9 ms  
^C  
--- google.com ping statistics ---  
13485 packets transmitted, 13236 received, 1.8465% packet loss, time 31163057ms  
rtt min/avg/max/mdev = 1.562/16.000/1484.740/45.616 ms, pipe 2  
vibha@vibha-Inspiron-5547: /$ ifconfig  
enp1s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
      ether 34:17:eb:7c:db:75 txqueuelen 1000 (Ethernet)  
      RX packets 0 bytes 0 (0.0 B)  
      RX errors 0 dropped 0 overruns 0 frame 0  
      TX packets 0 bytes 0 (0.0 B)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
      inet 127.0.0.1 netmask 255.0.0.0  
      inet6 ::1 prefixlen 128 scopeid 0x10<host>  
      loop txqueuelen 1000 (Local Loopback)  
      RX packets 42137 bytes 5101623 (5.1 MB)  
      RX errors 0 dropped 0 overruns 0 frame 0  
      TX packets 42137 bytes 5101623 (5.1 MB)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 192.168.1.7 netmask 255.255.255.0 broadcast 192.168.1.255  
      inet6 fe80::66a8:7fc4:f30:cc9c prefixlen 64 scopeid 0x20<link>  
      ether 34:de:1a:4b:c3:69 txqueuelen 1000 (Ethernet)  
      RX packets 6362950 bytes 5998440064 (5.9 GB)  
      RX errors 0 dropped 52 overruns 0 frame 0  
      TX packets 4366200 bytes 2994066024 (2.9 GB)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```



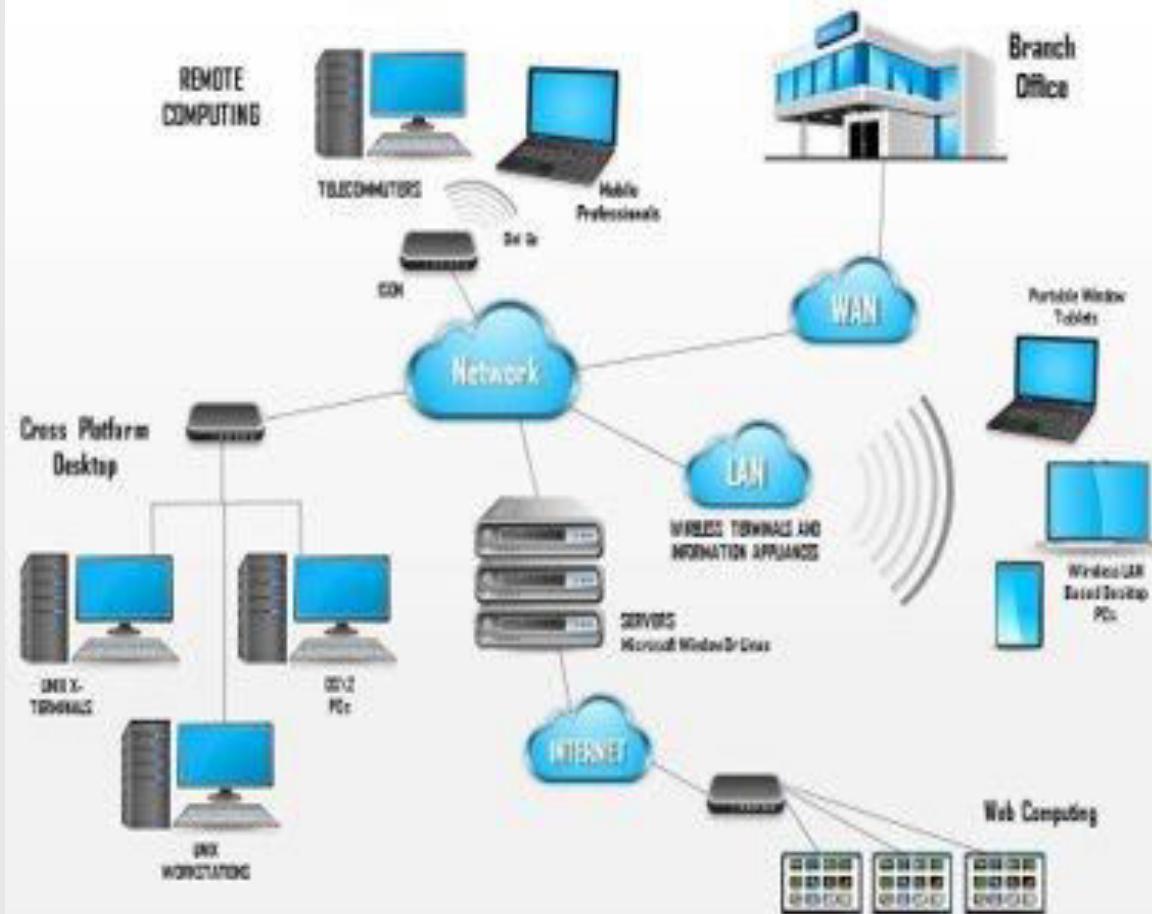
# Local Area Network

- A Local Area Network is a set of computers connected together via hub/switch.
- The devices are located in the same/small area.
- The devices have same type of network interface such as ethernet, wifi,...
- Data is transferred between machines based on their MAC addresses.



# A typical Network

Computer Networking



- Networks usually doesnot reside on same LAN
- A typical network is composed of group of LANs interconnected with each other
- Within each LAN, MAC address is used for communication inside.
- **MAC address is no longer enough** to communicate between the devices in different LANs, a more global addressing scheme required
- A network addressing which uses **IP address**

# IP Addressing

- TCP/IP currently use IP version 4 addressing (IPv4), which includes 4 groups of 8 bits for a combined IP address of length 32 bits.
- With 32 bits IP address, there are  $2^{32}$  or 4,294,967,296 addresses possible.
- IP Address Notation
  - Binary Notation
  - Dotted decimal Notation
  - Hexadecimal notation

## Internet Address (IP)

[Google IP4 Address](#)

**216.58.216.164**

[Google IP6 Address](#)

**2607:f8b0:4005:805::200e**

ComputerHope.com

# IP Addressing Notation

- IP address is represented with 4-byte (32-bit) word, which includes '4' 8-bit binary numbers separated with one or more spaces

## IPv4 address in dotted-decimal notation

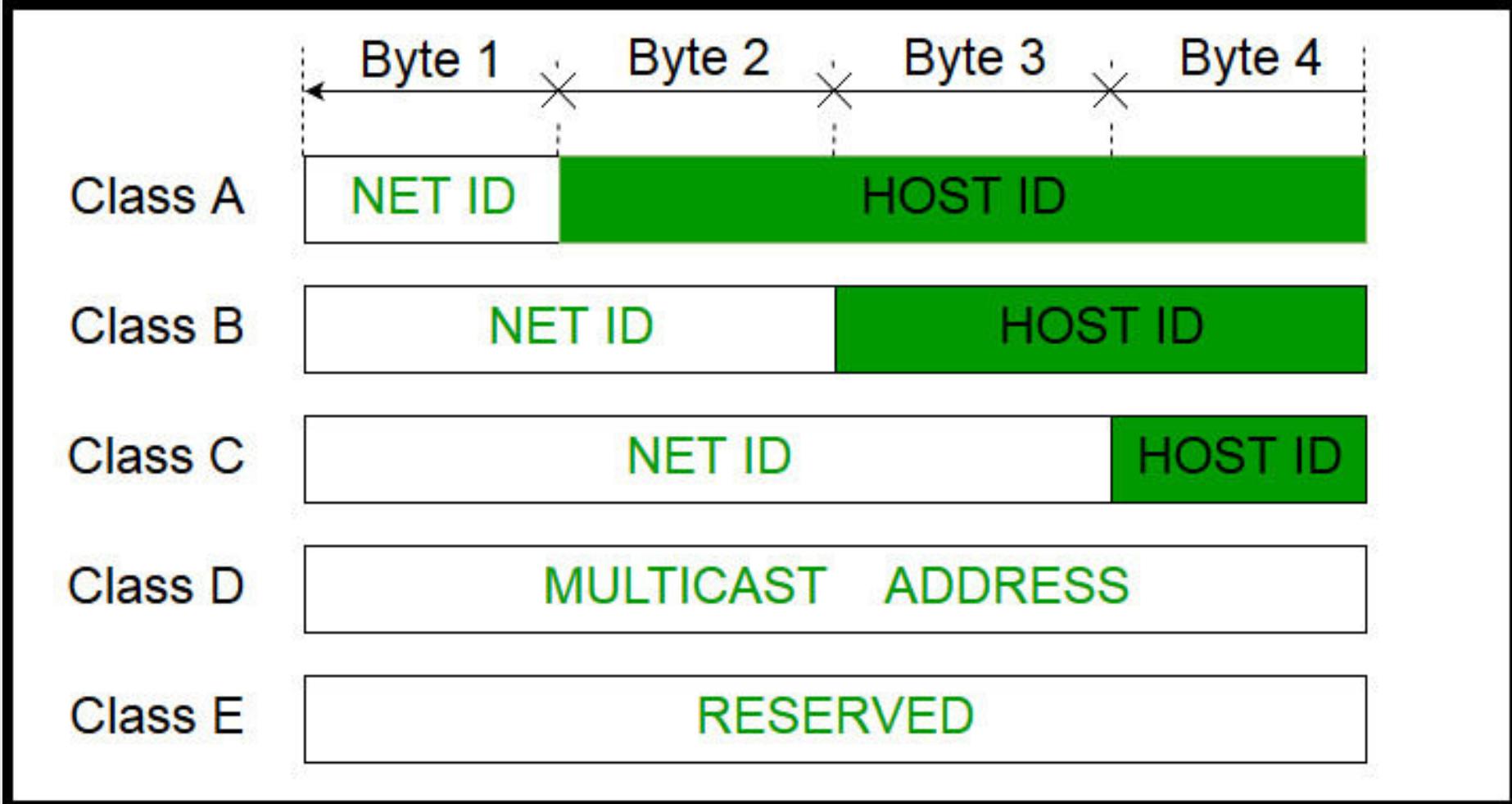
10101100.00010000.11111110.00000001

8 bits                    32 bits (4 bytes)

	0	8	16	24	32
Binary	11100011	01010010	10011101	10110001	
Hexadecimal	E3	52	9D	B1	
Dotted Decimal	227	82	157	177	

- In Hexadecimal, 32 bit binary word contains 4 groups of 8 bits, and each 8-bit grouping can be represented with 2-character hexadecimal number separated with spaces.
  - In Dotted-decimal notation, each 8-bit binary grouping is replaced by its decimal equivalent with decimal point (or dot) separating each number in form N.N.N.N, where N is any decimal value between 0 and 255.

# Classful IP Addressing

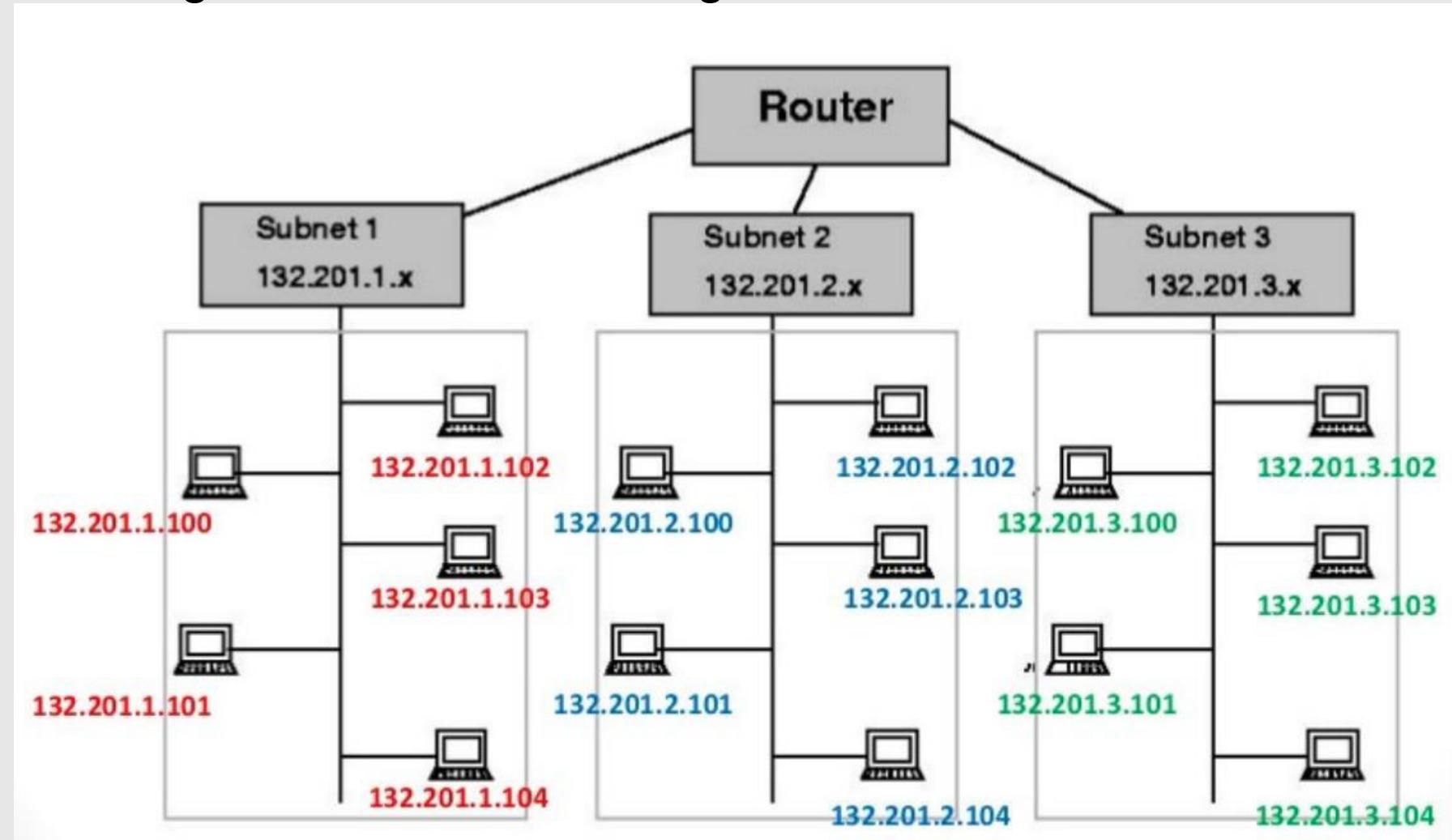


# Classful IP Addressing

CLASS	LEADING BITS	NET ID BITS	HOST ID BITS	NO. OF NETWORKS	ADDRESSES PER NETWORK	START ADDRESS	END ADDRESS
CLASS A	0	8	24	$2^7$ (128)	$2^{24}$ (16,777,216)	0.0.0.0	127.255.255.255
CLASS B	10	16	16	$2^{14}$ (16,384)	$2^{16}$ (65,536)	128.0.0.0	191.255.255.255
CLASS C	110	24	8	$2^{21}$ (2,097,152)	$2^8$ (256)	192.0.0.0	223.255.255.255
CLASS D	1110	NOT DEFINED	NOT DEFINED	NOT DEFINED	NOT DEFINED	224.0.0.0	239.255.255.255
CLASS E	1111	NOT DEFINED	NOT DEFINED	NOT DEFINED	NOT DEFINED	240.0.0.0	255.255.255.255

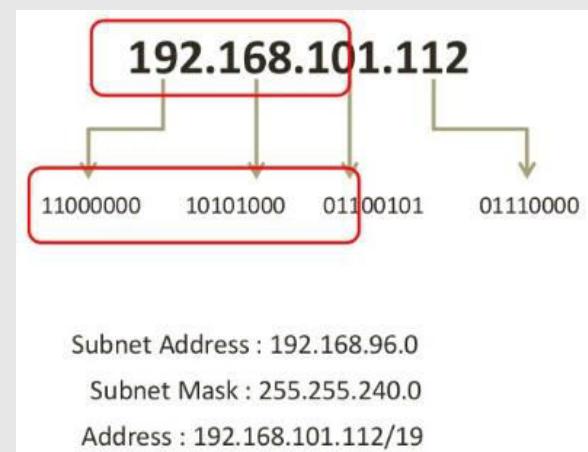
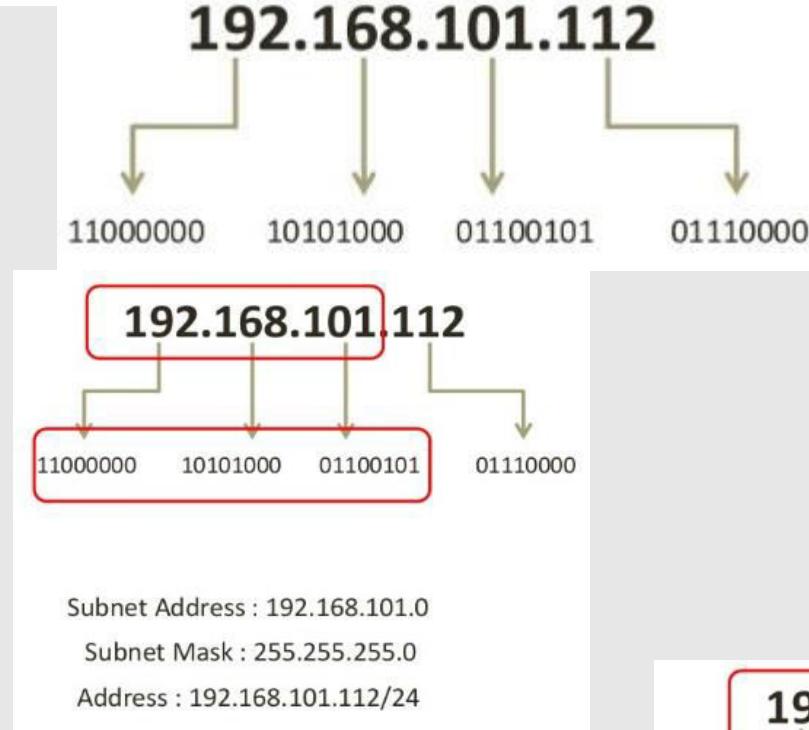
# Subnet

To facilitate routing, the machines are organized into small networks called **Subnets**.

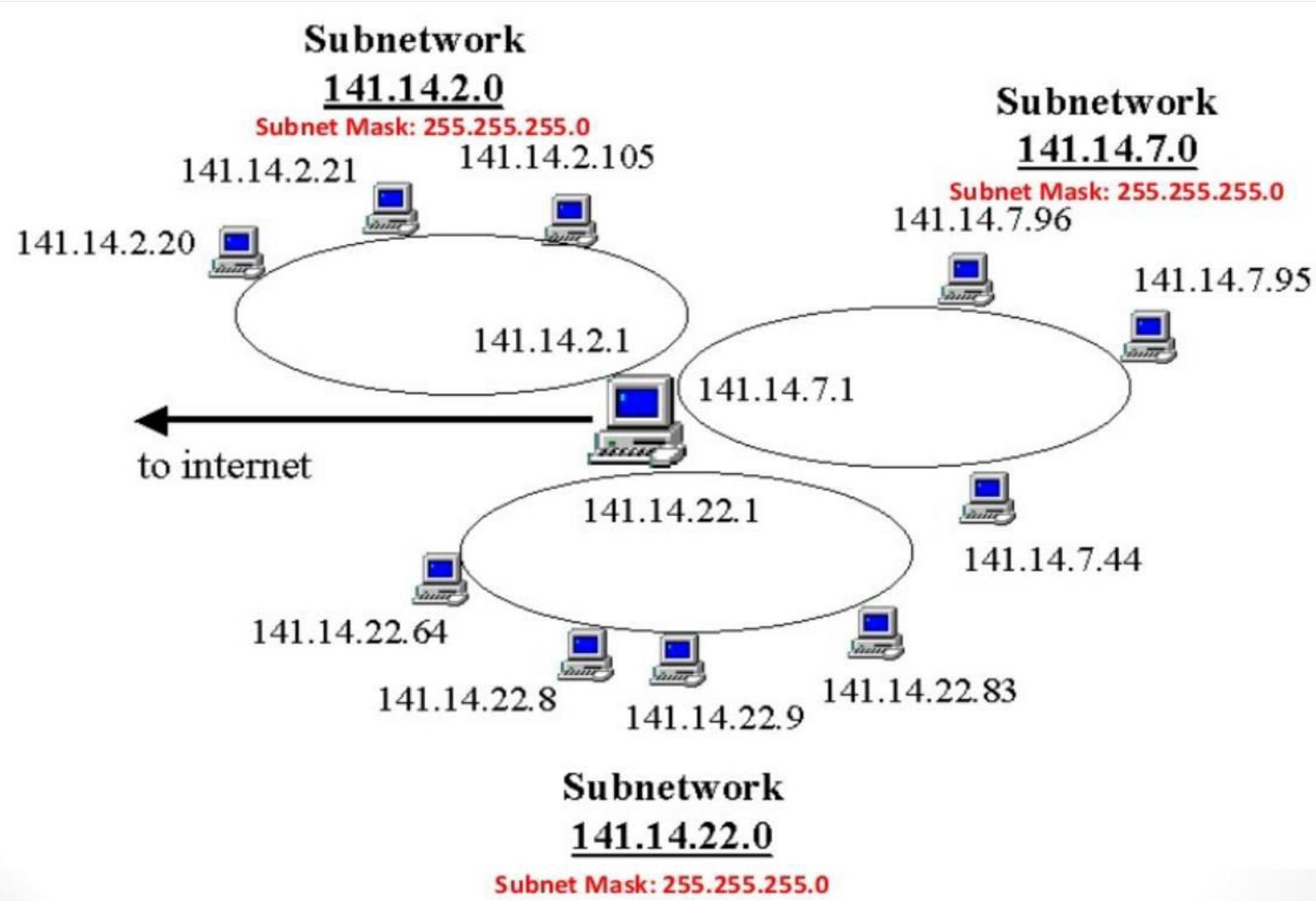


# Subnet

- Network interfaces within the same subnet share the upper part of the IP address, and differ in the lower part.
- The number of bits shared within the subnet control the subnet size.
- The part of the IP address shared among interfaces within the subnet, is expressed as the **Subnet mask**



# Subnet Mask



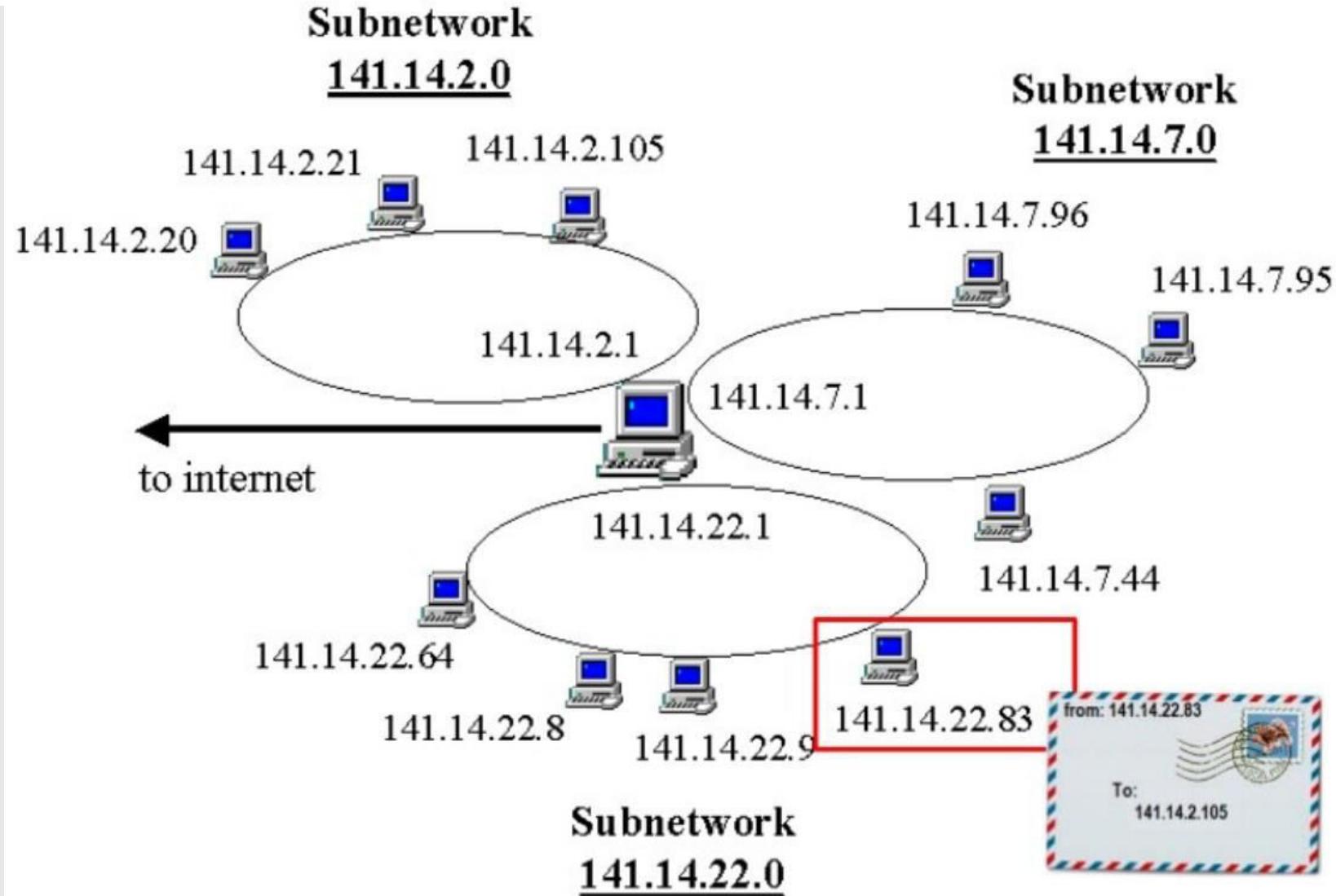
# Subnet Mask

```
vibha@vibha-Inspiron-5547: /  
  
64 bytes from del12s04-in-f14.1e100.net (142.250.194.110): icmp_seq=13482 ttl=120 time=6.93 ms  
64 bytes from del12s04-in-f14.1e100.net (142.250.194.110): icmp_seq=13483 ttl=120 time=10.3 ms  
64 bytes from del12s04-in-f14.1e100.net (142.250.194.110): icmp_seq=13484 ttl=120 time=51.1 ms  
64 bytes from del12s04-in-f14.1e100.net (142.250.194.110): icmp_seq=13485 ttl=120 time=14.9 ms  
^C  
--- google.com ping statistics ---  
13485 packets transmitted, 13236 received, 1.8465% packet loss, time 31163057ms  
rtt min/avg/max/mdev = 1.562/16.000/1484.740/45.616 ms, pipe 2  
vibha@vibha-Inspiron-5547:/$ ifconfig  
enp1s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
      ether 34:17:eb:7c:db:75 txqueuelen 1000 (Ethernet)  
      RX packets 0 bytes 0 (0.0 B)  
      RX errors 0 dropped 0 overruns 0 frame 0  
      TX packets 0 bytes 0 (0.0 B)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
      inet 127.0.0.1 netmask 255.0.0.0  
      inet6 ::1 prefixlen 128 scopeid 0x10<host>  
      loop txqueuelen 1000 (Local Loopback)  
      RX packets 42137 bytes 5101623 (5.1 MB)  
      RX errors 0 dropped 0 overruns 0 frame 0  
      TX packets 42137 bytes 5101623 (5.1 MB)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 192.168.1.7 netmask 255.255.255.0 broadcast 192.168.1.255  
      inet6 fe80::66a8:7fc4:f30:cc9c prefixlen 64 scopeid 0x20<link>  
      ether 34:de:1a:4b:c3:69 txqueuelen 1000 (Ethernet)  
      RX packets 6362950 bytes 5998440064 (5.9 GB)  
      RX errors 0 dropped 52 overruns 0 frame 0  
      TX packets 4366200 bytes 2994066024 (2.9 GB)  
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
vibha@vibha-Inspiron-5547:/$
```

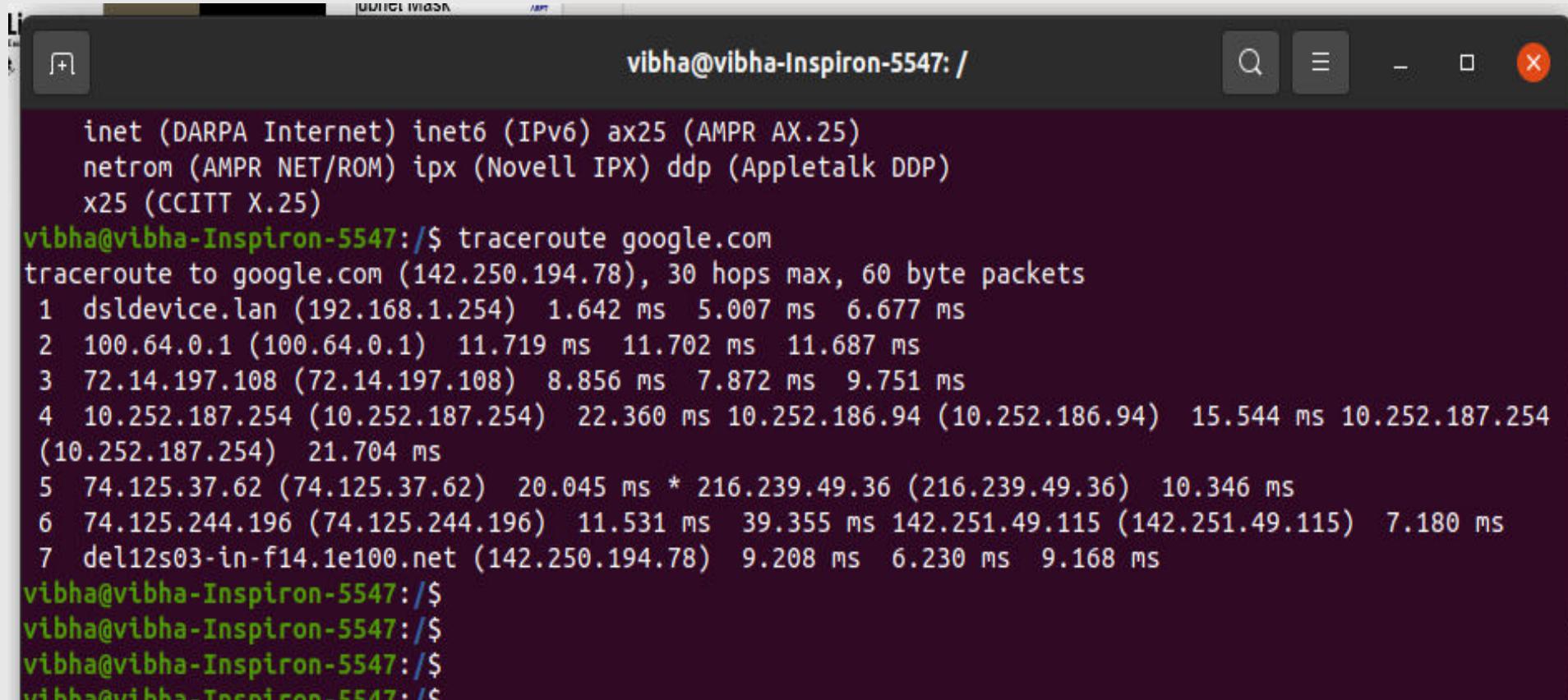
# IP Address Allocation

- IP address can be allocated in 2 different ways
  - **Statically**
    - Manual allocation or via a startup script
    - Address is always the same
  - **Dynamically**
    - At the startup, the machine asks a DHCP server to provide it with an IP address
    - The DHCP server responds with an unused IP address within the proper subnet
    - Hence, IP address can change every time the machine boots
    - The DHCP server can be configured so it will always assign a specific machine the same IP address.
- Using DHCP address allocation is more convenient and more flexible
- Since the DHCP server needs to be accessed before the IP address is allocated, it should reside within the same LAN as the machines

# Routing



# Routing



The screenshot shows a terminal window titled 'vibha@vibha-Inspiron-5547: /'. The window displays the output of the 'traceroute' command to 'google.com'. The output shows the path taken by the packets, starting from the local network and moving through several routers and finally reaching the destination at Google's IP address (142.250.194.78). The terminal window has a dark background with white text and includes standard Linux terminal icons.

```
inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)
x25 (CCITT X.25)

vibha@vibha-Inspiron-5547:/$ traceroute google.com
traceroute to google.com (142.250.194.78), 30 hops max, 60 byte packets
1 dsldevice.lan (192.168.1.254) 1.642 ms 5.007 ms 6.677 ms
2 100.64.0.1 (100.64.0.1) 11.719 ms 11.702 ms 11.687 ms
3 72.14.197.108 (72.14.197.108) 8.856 ms 7.872 ms 9.751 ms
4 10.252.187.254 (10.252.187.254) 22.360 ms 10.252.186.94 (10.252.186.94) 15.544 ms 10.252.187.254
(10.252.187.254) 21.704 ms
5 74.125.37.62 (74.125.37.62) 20.045 ms * 216.239.49.36 (216.239.49.36) 10.346 ms
6 74.125.244.196 (74.125.244.196) 11.531 ms 39.355 ms 142.251.49.115 (142.251.49.115) 7.180 ms
7 del12s03-in-f14.1e100.net (142.250.194.78) 9.208 ms 6.230 ms 9.168 ms

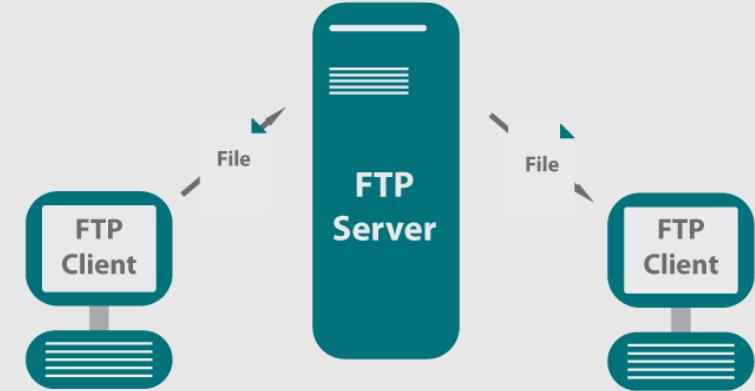
vibha@vibha-Inspiron-5547:/$
vibha@vibha-Inspiron-5547:/$
vibha@vibha-Inspiron-5547:/$
vibha@vibha-Inspiron-5547:/$
```

# Outline

- ✓ Introduction to Networking in Linux
- ✓ Network basics and tools
- ✓ File transfer protocol in Linux
- Network file system
- Domain Name Services
- Dynamic Host Configuration Protocol
- Network Information Services

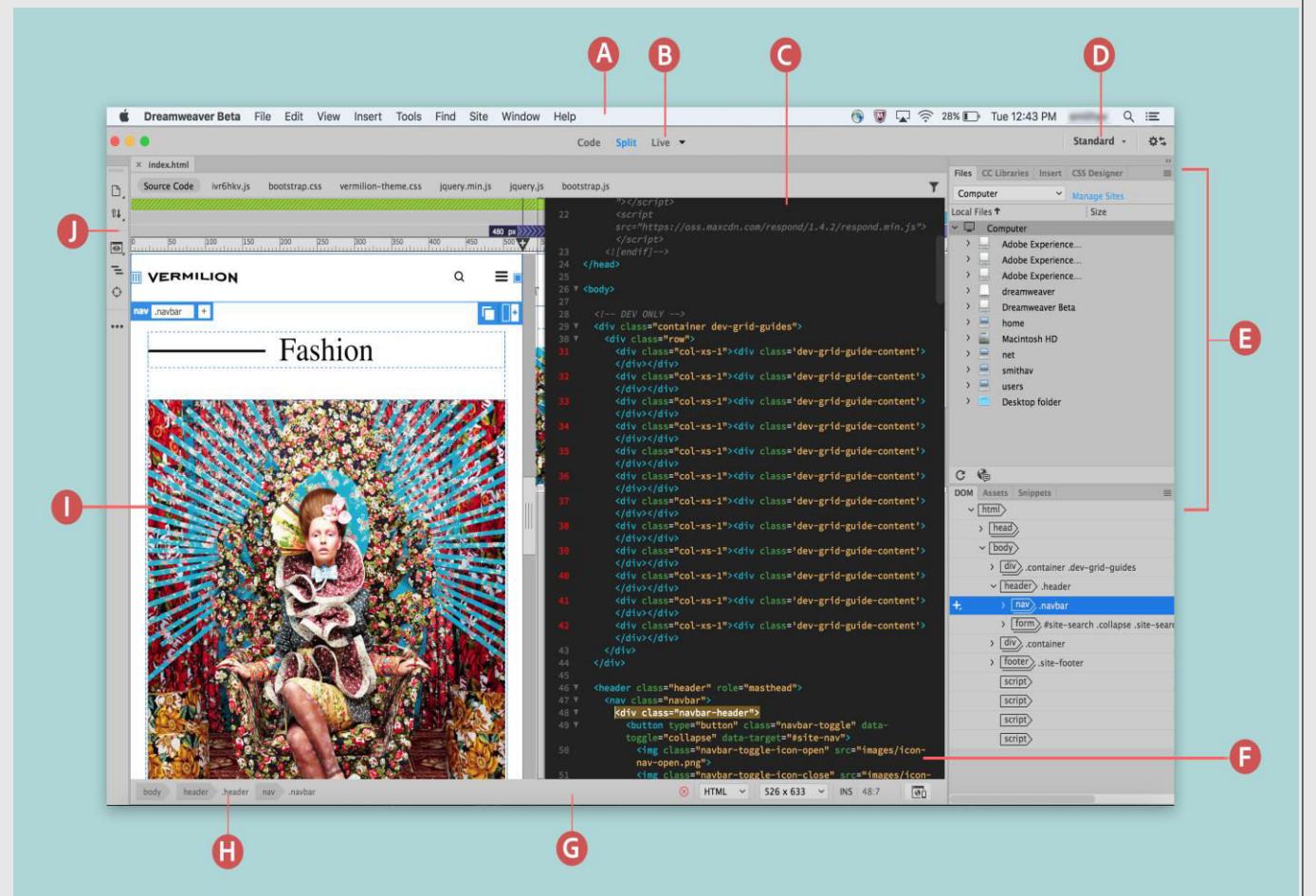


- FTP is the simplest and the most secure way to exchange files over the internet
- Transferring files from a client computer to a server computer is called **uploading** and transferring from a server to a client is **downloading**.
- To access an FTP server, users must be able to connect to the internet or an intranet with an FTP client program.
- FTP doesn't really move, it copies files from one computer to another.
- FTP is the file transfer protocol in the internet's TCP/IP protocol suite's application layer.



# FTP Clients

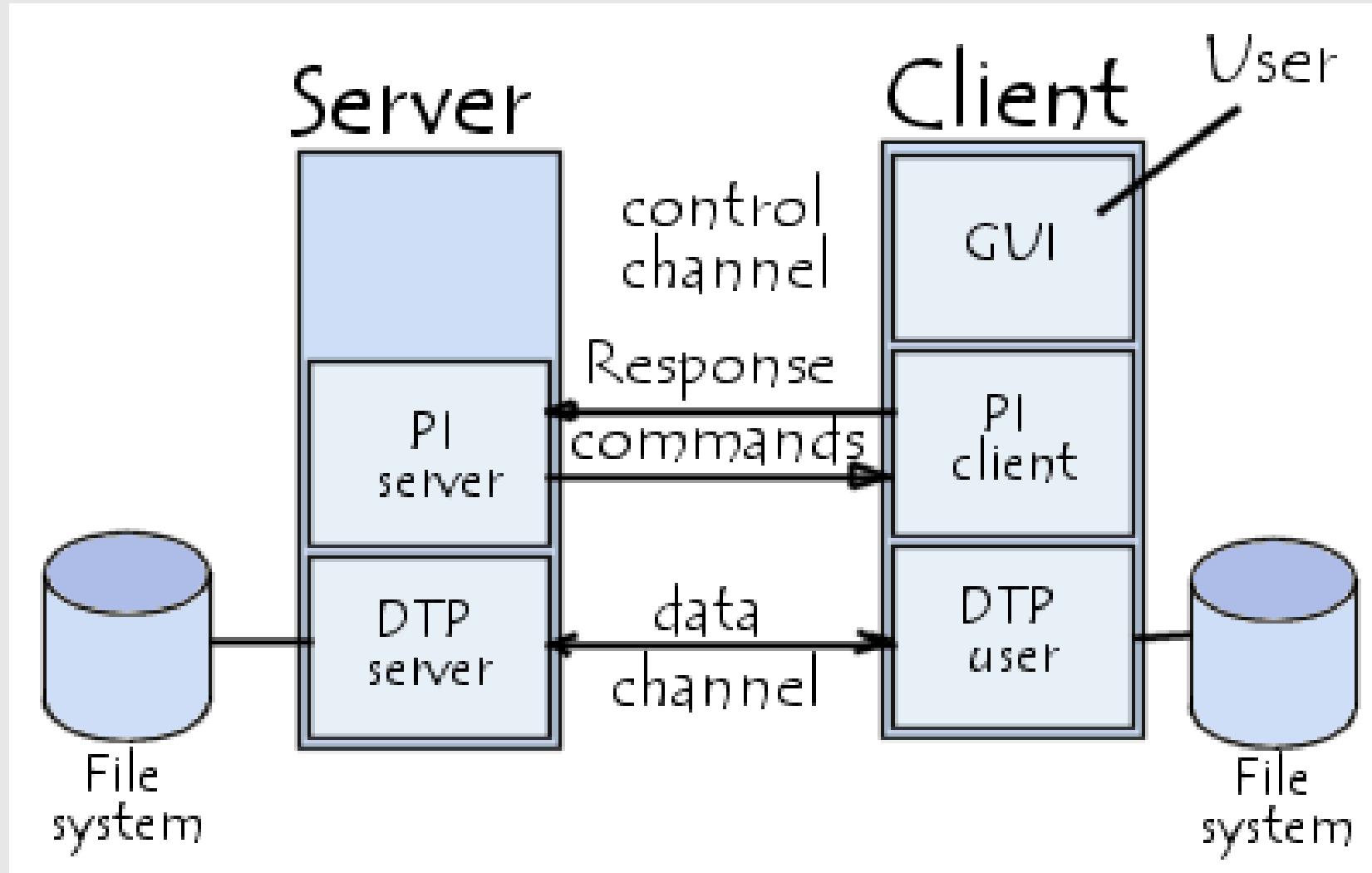
- Commonly used FTP clients include:
  - FileZilla
  - Fire FTP
  - Dreamweaver



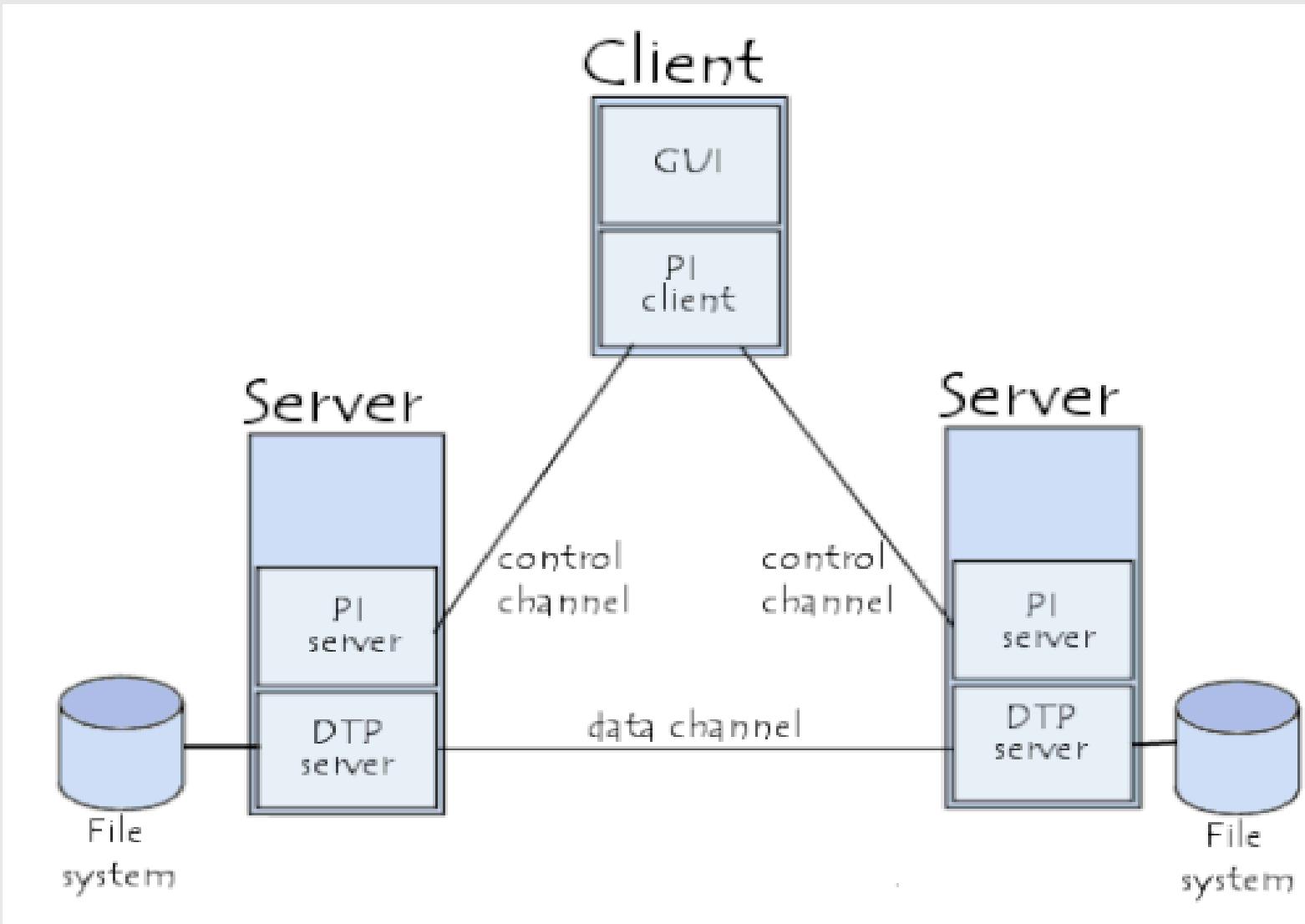
# FTP: Features

- FTP operates in a client/server environment , meaning that the remote machine is configured as a server, and consequently waits for other machine (client) to request a service from it.
- In UNIX, the service is provided by what is called a **daemon**, a small task that runs in the background. The FTP daemon is called **ftpd**.
- **The FTP protocol is used for transferring one file at a time.**
- The FTP protocol can also perform other actions, such as creating and deleting directories, listing files, deleting and renaming files, etc.
- FTP allows files to have ownership and access restrictions
- FTP hides the details of individual computer systems

# FTP model



# FTP model

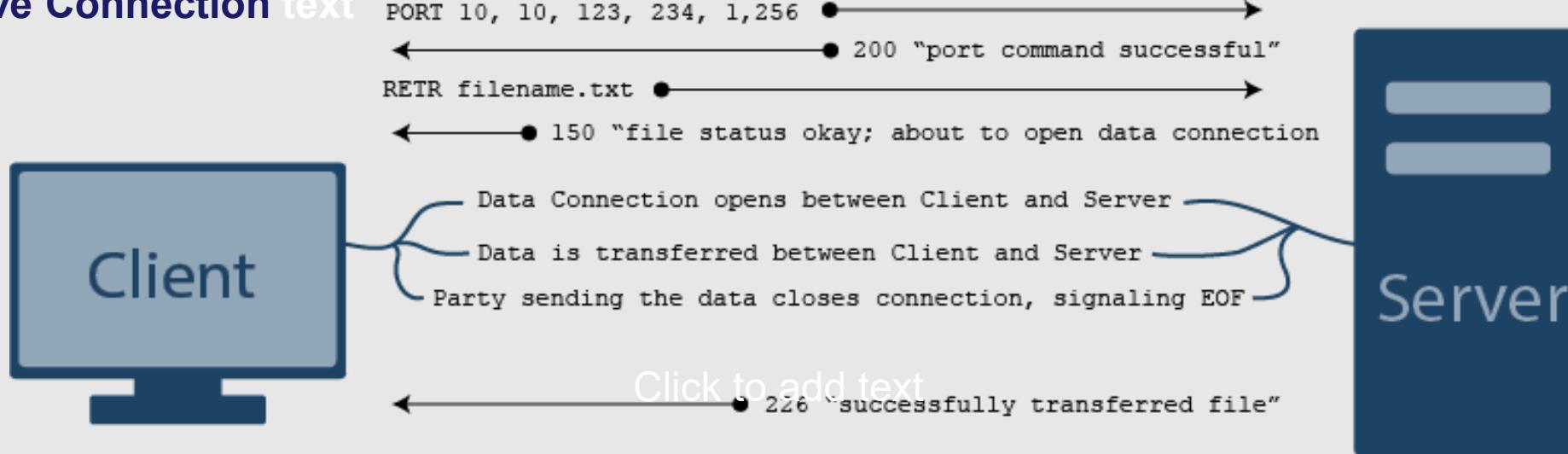


# Types of Connections

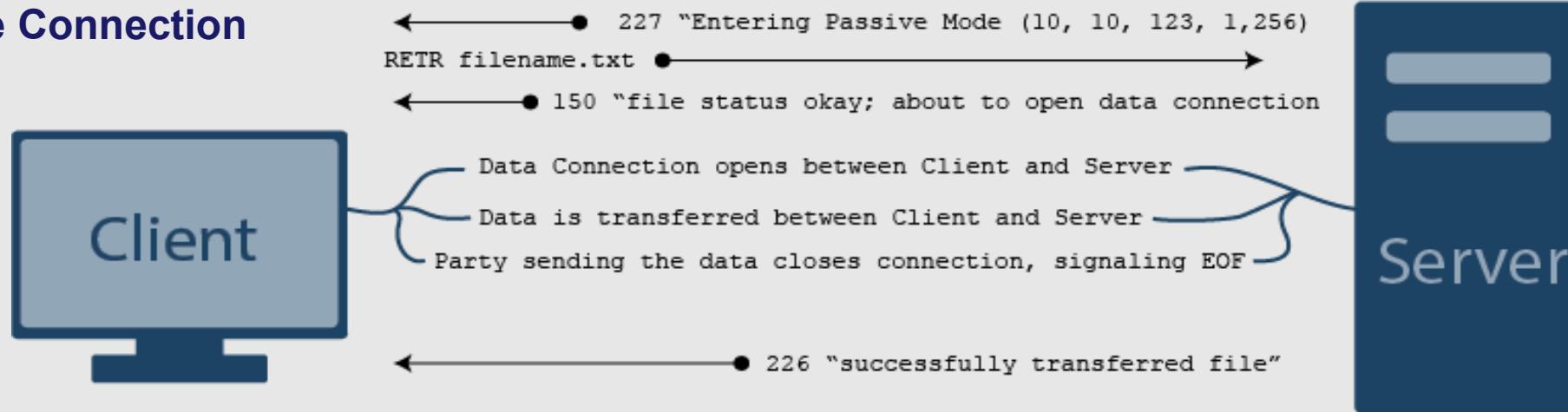
- When an FTP client connects to an FTP server, it opens a connection to the FTP control port 21. The client tell FTP server whether to establish an **active or passive** connection.
- Most modern FTP clients attempt to establish a passive connection when requesting data from servers.
- The type of connection choosen by the client determines how the server responds and on what ports transactions will occur.
- **Active connection:**
  - Server opens a data connection to the client from port 20 to a high range port on the client machine
  - All data from the server is then passed over this connection.
- **Passive Connection:**
  - Client asks the FTP Server to establish a passive connection port, which can be on any port higher than 10,000.
  - Server then binds to this higher numbered port for this particular session and relays that port number back to the client. Client then opens the newly bound port for the data connection.
  - Each data request the client makes results in a separate data connection.

# Types of Connections

## Active Connection

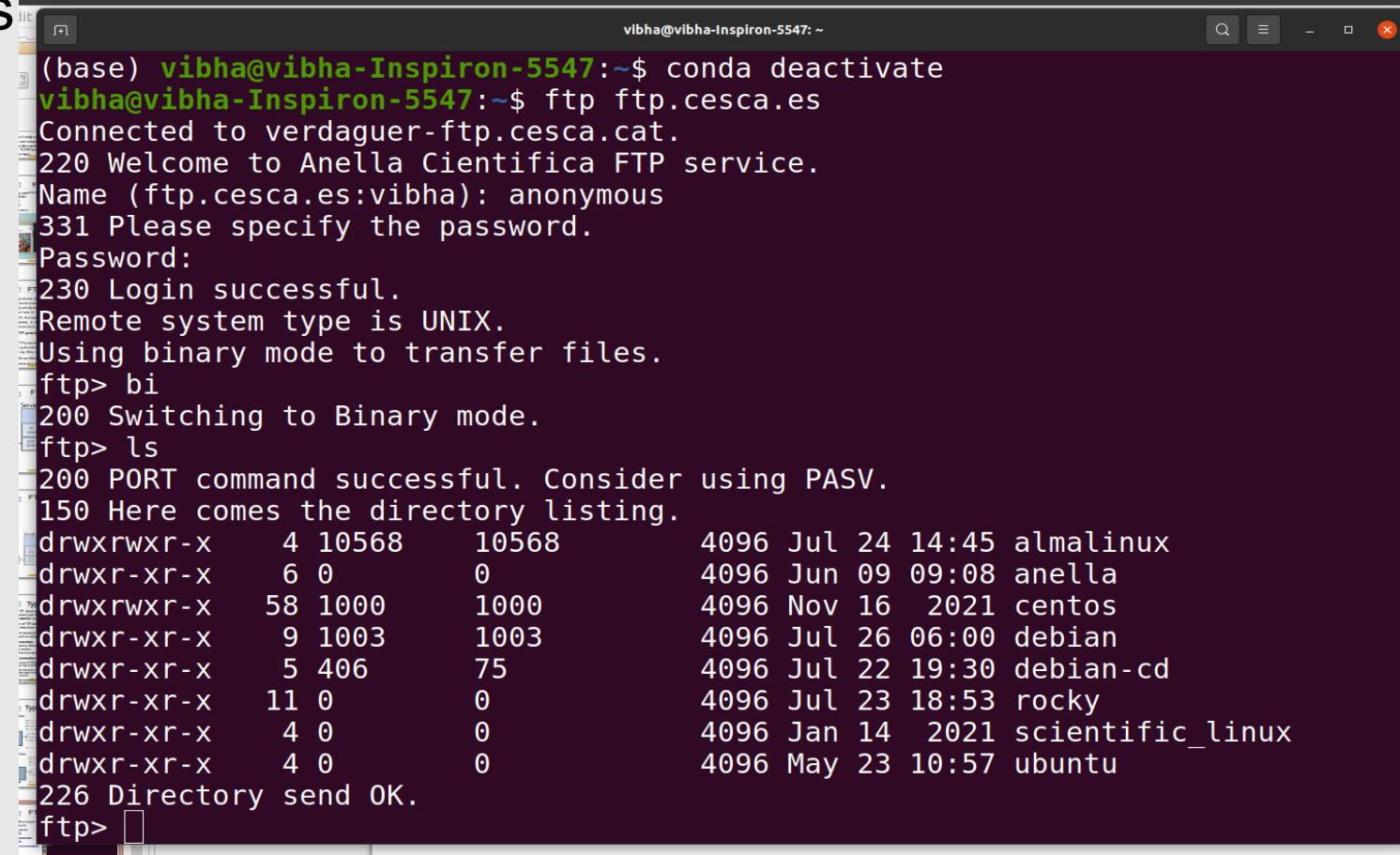


## Passive Connection



# FTP Commands

- There are 3 different types of FTP commands:
  - **Access control Commands**
  - **Transfer parameter commands**
  - **FTP service commands**



The screenshot shows a terminal window with a dark background and light-colored text. It displays an FTP session between a local machine (vibha@vibha-Inspiron-5547) and a remote server (ftp.cesca.es). The session starts with the user connecting anonymously and logging in successfully. The user then changes the transfer mode to binary and lists the contents of the directory. The directory listing shows several operating system distributions: almalinux, anella, centos, debian, debian-cd, rocky, scientific\_linux, and ubuntu. The output ends with a success message indicating the directory was sent.

```
vibha@vibha-Inspiron-5547:~$ conda deactivate
vibha@vibha-Inspiron-5547:~$ ftp ftp.cesca.es
Connected to verdaguer-ftp.cesca.cat.
220 Welcome to Anella Cientifica FTP service.
Name (ftp.cesca.es:vibha): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bi
200 Switching to Binary mode.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxrwxr-x 4 10568 10568 4096 Jul 24 14:45 almalinux
drwxr-xr-x 6 0 0 4096 Jun 09 09:08 anella
drwxrwxr-x 58 1000 1000 4096 Nov 16 2021 centos
drwxr-xr-x 9 1003 1003 4096 Jul 26 06:00 debian
drwxr-xr-x 5 406 75 4096 Jul 22 19:30 debian-cd
drwxr-xr-x 11 0 0 4096 Jul 23 18:53 rocky
drwxr-xr-x 4 0 0 4096 Jan 14 2021 scientific_linux
drwxr-xr-x 4 0 0 4096 May 23 10:57 ubuntu
226 Directory send OK.
ftp>
```

[https://www.w3.org/Protocols/rfc959/4\\_FileTransfer.html](https://www.w3.org/Protocols/rfc959/4_FileTransfer.html)

- **USER:** Character string allowing the user to be identified.
- **PASS:** Character string specifying the user's password.  
This command must immediately precede the USE command.
- **ACCT:** Character string representing the user's account. During the response accepting password, if the response is 230 this stage is not necessary, if the response is 332, it is.
- **CWD:** Change Working Directory, command enables the current directory to be changed. This command requires the directory's access path to be fulfilled as an argument.
- **CDUP:** Change to Parent Directory, command allows you to go back to the parent directory.
- **SMNT:** Structure Mount
- **REIN:** Reinitialize
- **QUIT:** Command enables the termination of the current session. The server waits to finish the transfer in progress if the need arises, then supplies a response before closing the connection.

- **PORT:** Character string allowing the port number used to be specified.
- **PASV:** Command making it possible to indicate to the DTP server to stand by for a connection on a specific port chosen randomly from among the available ports.
  - The response to this command is the IP address of the machine and the port
- **TYPE:** Command enables the type of format in which the data will be sent to be specified.
- **STRU:** Telnet character specifying the file structure
  - F for File, R for Record, P for Page
- **MODE:** Telnet character specifying data transfer method
  - S for Stream, B for Block, C for Compresses

# FTP Service commands

- **RETR:** Retrieve command asks the server DTP for a copy of the files whose access path is given in the parameters
- **STOR:** Store command asks the server DTP to accept the data sent over data channel and store them in a file bearing name given in the parameters.
  - If the file does not exist, the server creates it, if not it overwrites it.
- **STOU:** Command is identical to the previous one, only it asks the server to create a file where name is unique.
  - The name of the file is returned in the response
- **APPE:** Append, the data sent is concatenated into the file bearing the name given in the parameter if it already exists, if not, it is created.
- **ALLO:** Allocate, command asks the server to plan a storage space big enough to hold the file
- **REST:** restart, command enables a transfer to be restarted from where it stopped.
  - This command must immediately follow a transfer command

# FTP Service commands

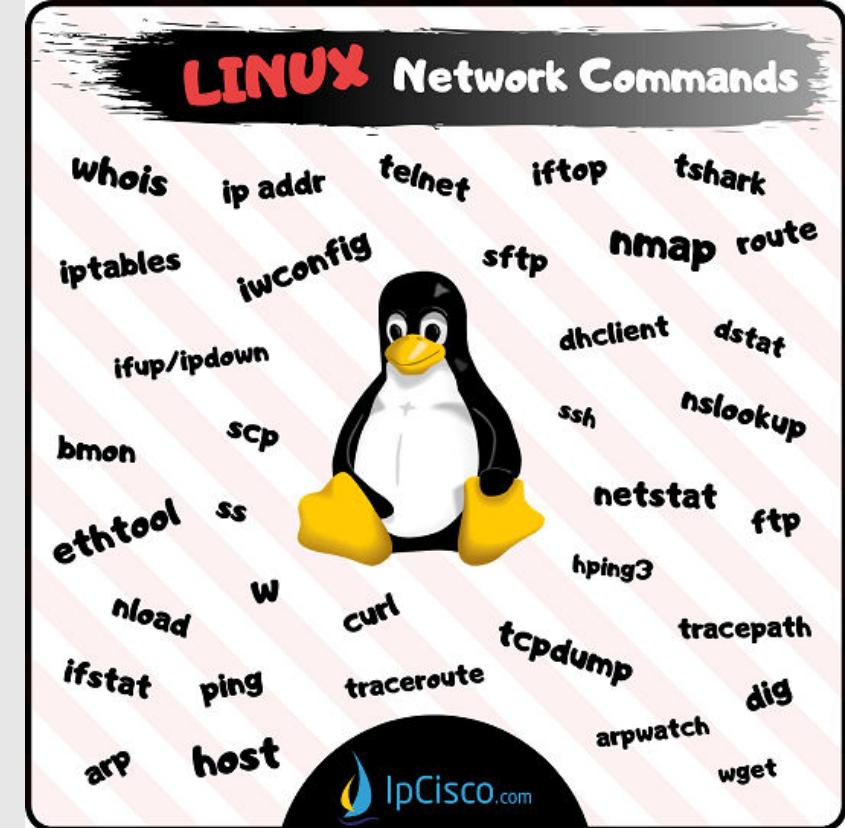
- RNFR: Rename from, command enables a file to be renamed.
  - Indicate name of the file to be renamed in the parameters and must be immediately followed by RNTO command
- RNTO: Rename to, Command enables a file to be renamed.
  - Indicate the name of file to be renamed in the parameters
- ABOR: Abort, command tells the server DTP to abondon all transfers associated with the previous command
  - If no data connection is open, the DTP server does nothing, else closes it
  - Control channel however remains open
- DELE: Delete, command allows a file to be deleted.
  - The command is irreversible, confirmation can only be given at client level.

# FTP Service commands

- NLST: name list, command enables the list of files and directories present in the current directory to be sent
- RMD: Remove directory, command enables a directory to be deleted.
- MKD: Make directory, command causes a directory to be created
- PWD: Print working directory, command prints the complete path of the current directory.
- LIST: command allows the list of files and directories present in the current directory.
- SITE: Site parameters, command causes the server to offer specific services not defined in FTP protocol.
- SYST: System, command allows information on the remote server to be sent.
- STAT: Status, command responds with status of the server.
- HELP: command gives all the commands understood by the server.
- NOOP: no operations, command is only used to obtain an OK command from the server.
  - It can only be used in order not to be disconnected after an excessive period of inactivity.

# Outline

- ✓ Introduction to Networking in Linux
- ✓ Network basics and tools
- ✓ File transfer protocol in Linux
- ✓ Network file system
- Domain Name Services
- Dynamic Host Configuration Protocol
- Network Information Services



- •The most commercially successful and widely available remote file system protocol
- •Designed and implemented by Sun Microsystems  
(Walash et al, 1985; Sandberg et al, 1985)
- •Reasons for success
  - The NFS protocol is public domain
  - Sun sells that implementation to all people for less than the cost of implementing it themselves
  - Evolved from version 2 to version 3 (which is the common implementation today).

# NFS Overview

- Views a set of interconnected workstations as a set of independent machines with independent file systems
- The goal is to allow some degree of sharing among these file systems (on explicit request)
- Sharing is based on client server relationships
- A machine may be both client and server
- The protocol is stateless
- Designed to support UNIX file system semantics
- The protocol design is transport independent

# Network File System

- Network File System is a NFS server client protocol used for sharing files and directories between Linux/unix to Unix/Linux vise versa.
- It is a popular distributed file system protocol that enables users to mount remote directories on their server.
- NFS enables you to mount a remote share locally.
- NFS allows a linux server to share directories with other UNIX clients over network.
  - NFS server exports a directory and NFS clients mounts this directory.
  - RHEL7 supports two versions of NFS- NFSv3 and NFSv4.
- RHEL7 provides the support for NFS versions 3, 4.0 and 4.1.
- NFS default port is 2049.
- NFS share can be mounted manually or automatically using AutoFS.

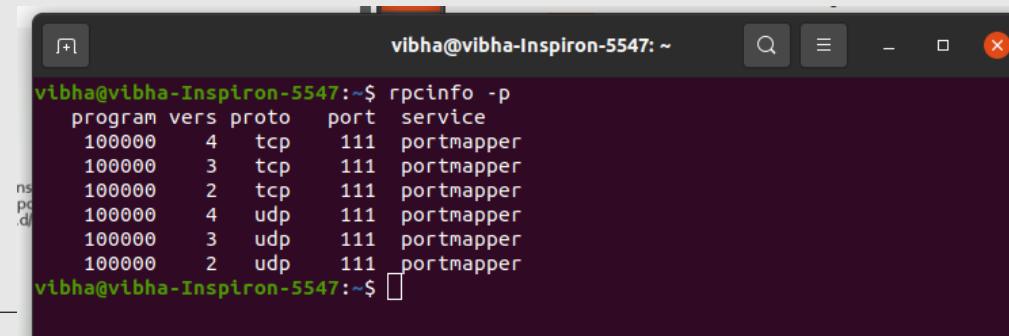
# NFS Services

- **rpcbind:** the rpc bind server converts RPC program numbers into universal addresses.
- **Nfs-server:** it enables the clients to access NFS shares.
- **Nfs-lock/ rpc-statd:** NFS file locking, implement file lock recovery when a NFS server crashes and reboots.
- **Nfs-idmap:** it translates user and group ids into names, and to translate user and group names into ids.

- The **rpcbind utility** maps RPC services to the ports on which they listen.
  - RPC processes notify rpcbind when they start, registering the ports they are listening on and the RPC program numbers they expect to serve.
- Because RPC-based services rely on rpcbind to make all connections with incoming client requests, rpcbind must be available before any of these services start.
  - *Command:* rpcinfo -p
- Starting the nfs-server process starts the nfs server and other RPC processes. RPC processes includes:
  - **Rpc.statd:** implements monitoring protocol (NSM) between the NFS client and NFS server
  - **Rpc.mountd:** NFS mount daemon that implements the server side of the mount requests from NFSv3 clients.
  - **Rpc.idmapd:** Maps NFSv4 names and local UIDs and GIDs
  - **Rpc.rquotad:** provides user quota information for remote users.

## Remote Procedure Call (RPC)

- RPC, defined by RFC 1057
- RPC is a set of function calls used by a client program to call functions in a remote server program.
- The port mapper program is the program used to keep track of which ports programs support RPC functions use.
  - The port mapper port is 111.
- In Redhat Linux, the port mapper daemon is started in the /etc/rc.d/init.d/portmap and the daemon program is called "portmap".



```
vibha@vibha-Inspiron-5547:~$ rpcinfo -p
    program  vers  proto   port  service
  100000    4    tcp    111  portmapper
  100000    3    tcp    111  portmapper
  100000    2    tcp    111  portmapper
  100000    4    udp    111  portmapper
  100000    3    udp    111  portmapper
  100000    2    udp    111  portmapper
vibha@vibha-Inspiron-5547:~$
```

- **/etc/exports** : Its a main configuration file of **NFS**, all exported **files and directories** are defined in this file at the **NFS Server** end.
- **/etc/fstab** : To mount a **NFS directory** on your system across the **reboots**, we need to make an entry in **/etc/fstab**.
- **/etc/sysconfig/nfs** - or **/etc/nfsconfig**: Configuration file of **NFS** to control on which port **rpc** and other services are listening and the **NFS version used** .

# Installation: NFS

## 1. Install NFS Kernel Server in Ubuntu

```
$ sudo apt update  
$ sudo apt install nfs-kernel-server
```

## 2. Create NFS Export Directory

```
$ sudo mkdir -p /mnt/nfs_share  
$ sudo chown -R nobody:nogroup /mnt/nfs_share/  
$ sudo chmod 777 /mnt/nfs_share/
```

## 3. Grant NFS Share Access to client System

```
$ sudo vim /etc/exports  
/mnt/nfs_share 192.168.43.0/24  
        (rw,sync,no_subtree_check)  
/mnt/nfs_share client_IP_1  
        (re,sync,no_subtree_check)
```

## 4. Export the NFS Share Directory

```
sudo exportfs -a  
sudo systemctl restart nfs-kernel-server
```

## 5. Allow NFS Access through the Firewall

```
$ sudo ufw allow from 192.168.43.0/24 to any  
port nfs  
$ sudo ufw enable  
$ sudo ufw status
```

[https://www.tecmint.com/  
install-nfs-server-on-  
ubuntu/](https://www.tecmint.com/install-nfs-server-on-ubuntu/)

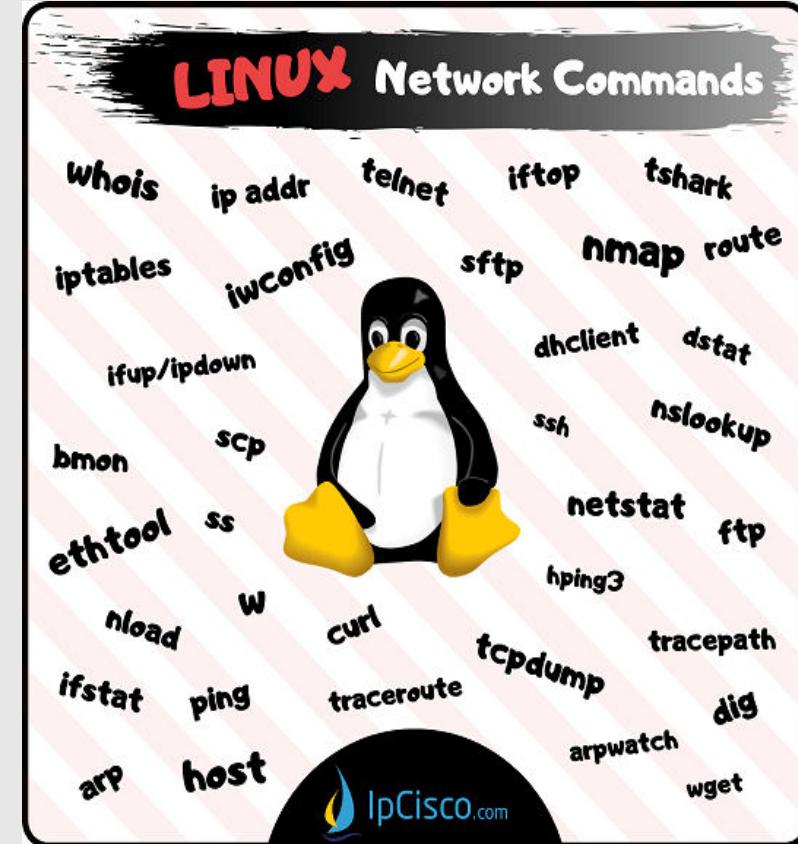
- **yum install nfs-utils nfs-utils-lib**
- **yum install portmap** (NFSv2 or NFSv3) -OR-
- **yum install rpcbind**( NFSv4)
- /etc/init.d/portmap start -OR- /etc/init.d/rpcbind start
- /etc/init.d/nfs start
- chkconfig --level 35 portmap (rpcbond) on
- [chkconfig --level 35 nfs on
- /etc/exports
- /nfsshare 172.27.0.0(rw,sync,no\_root\_squash)
- **exportfs -a** : Exports all shares listed in **/etc/exports**, or given name
- **exportfs -v** : Displays a list of shares **files** and **options** on a server
- **exportfs -u** : Unexports all shares listed in **/etc/exports**, or given name
- **exportfs -r** : Refresh the server's list after modifying **/etc/exports**

# NFS Options

- **ro**: With the help of this option we can provide **read only access** to the shared files i.e **client** will only be able to **read**.
- **rw**: This option allows the **client server** to both **read** and **write** access within the shared directory.
- **sync**: Sync confirms requests to the shared directory only once the **changes** have been committed.
- **no\_subtree\_check**: This option prevents the **subtree** checking. When a shared directory is the subdirectory of a larger file system, **nfs** performs scans of every directory above it, in order to verify its permissions and details. Disabling the **subtree** check may increase the reliability of **NFS**, but reduce **security**.
- **no\_root\_squash**: This phrase allows **root** to **connect** to the designated directory.

# Outline

- ✓ Introduction to Networking in Linux
- ✓ Network basics and tools
- ✓ File transfer protocol in Linux
- ✓ Network file system
- ✓ Domain Name Services
- Dynamic Host Configuration Protocol
- Network Information Services



# Purpose of naming

- Addresses are used to locate objects
- Names are easier to remember than numbers
- You would like to get to the address or other objects using a name
- DNS provides a mapping from names to resources of several types

# Names and addresses in general

- An address is how you get to an endpoint
  - Typically, hierarchical (for scaling):
    - 950 Charter Street, Redwood City CA, 94063
    - 204.152.187.11, +1-650-381-6003
- A “name” is how an endpoint is referenced
  - Typically, no structurally significant hierarchy
    - “David”, “Tokyo”, “itu.int”

# Naming History

## ➤ 1970's ARPANET

- Host.txt maintained by the SRI-NIC
- pulled from a single machine
- Problems
  - traffic and load
  - Name collisions
  - Consistency

## ➤ DNS created in 1983 by Paul Mockapetris (RFCs 1034 and 1035), modified, updated, and enhanced by a myriad of subsequent RFCs

- A lookup mechanism for translating objects into other objects
- A globally distributed, loosely coherent, scalable, reliable, dynamic database
- Comprised of three components
  - A “name space”
  - Servers making that name space available
  - Resolvers (clients) which query the servers about the name space

# DNS Features: Global Distribution

- Data is maintained locally, but retrievable globally
  - No single computer has all DNS data
- DNS lookups can be performed by any device
- Remote DNS data is locally cachable to improve performance

# DNS Features: Loose Coherency

- The database is always internally consistent
  - Each version of a subset of the database (a zone) has a serial number
    - The serial number is incremented on each database change
- Changes to the master copy of the database are replicated according to timing set by the zone administrator
- Cached data expires according to timeout set by zone administrator



- No limit to the size of the database
  - One server has over 20,000,000 names
    - Not a particularly good idea
- No limit to the number of queries
  - 24,000 queries per second handled easily
- Queries distributed among masters, slaves, and caches

# DNS Features: Reliability

- Data is replicated
  - Data from master is copied to multiple slaves
- Clients can query
  - Master server
  - Any of the copies at slave servers
- Clients will typically query local caches
- DNS protocols can use either UDP or TCP
  - If UDP, DNS protocol handles retransmission, sequencing, etc.



- Database can be updated dynamically
  - Add/delete/modify of any record
- Modification of the master database triggers replication
  - Only master can be dynamically updated
    - Creates a single point of failure

- The namespace needs to be made hierarchical to be able to scale.
- The idea is to name objects based on
  - location (within country, set of organizations, set of companies, etc)
  - unit within that location (company within set of company, etc)
  - object within unit (name of person in company)



## How names appear in the DNS

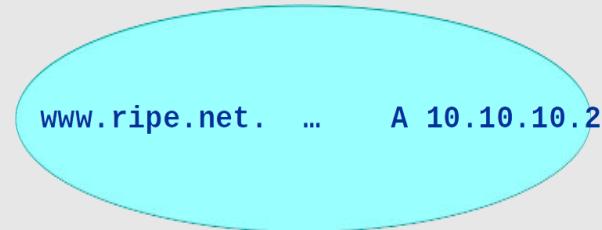
Fully Qualified Domain Name (FQDN)

[WWW.RIPE.NET](http://www.ripe.net)

- labels separated by dots
- DNS provides a mapping from FQDNs to resources of several types
- Names are used as a key when fetching data in the DNS

- The DNS maps names into data using Resource Records.

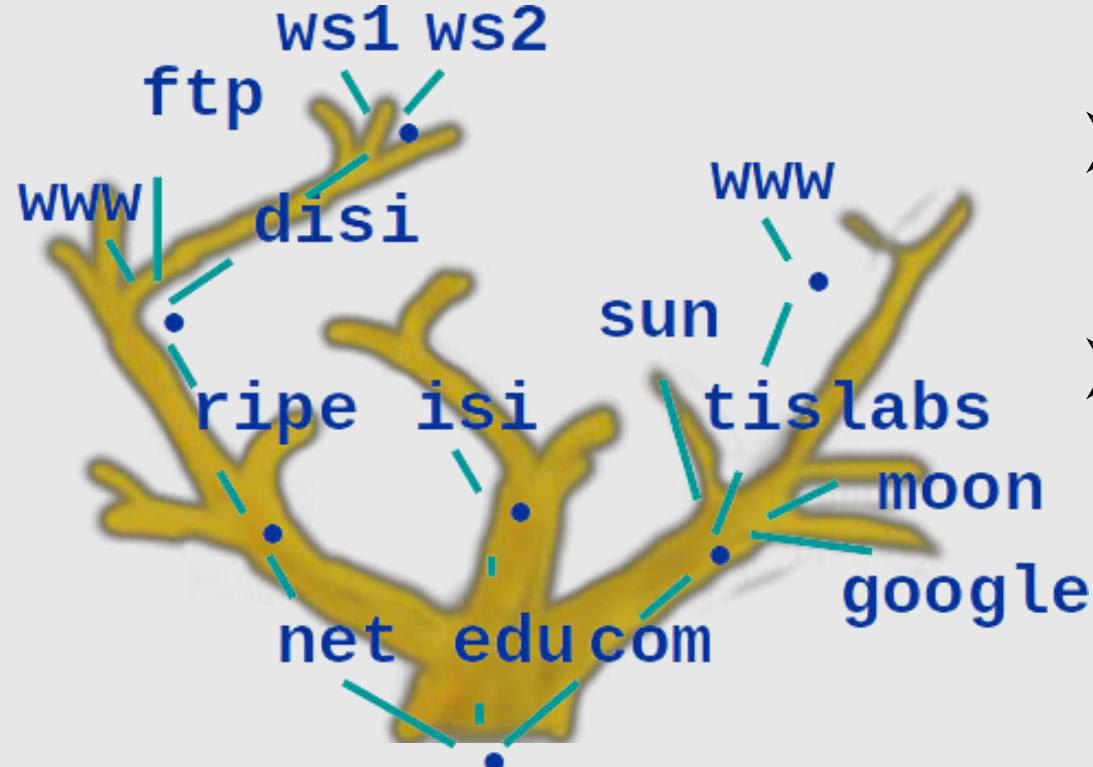
## Resource Record



## Address Resource

- More detail later

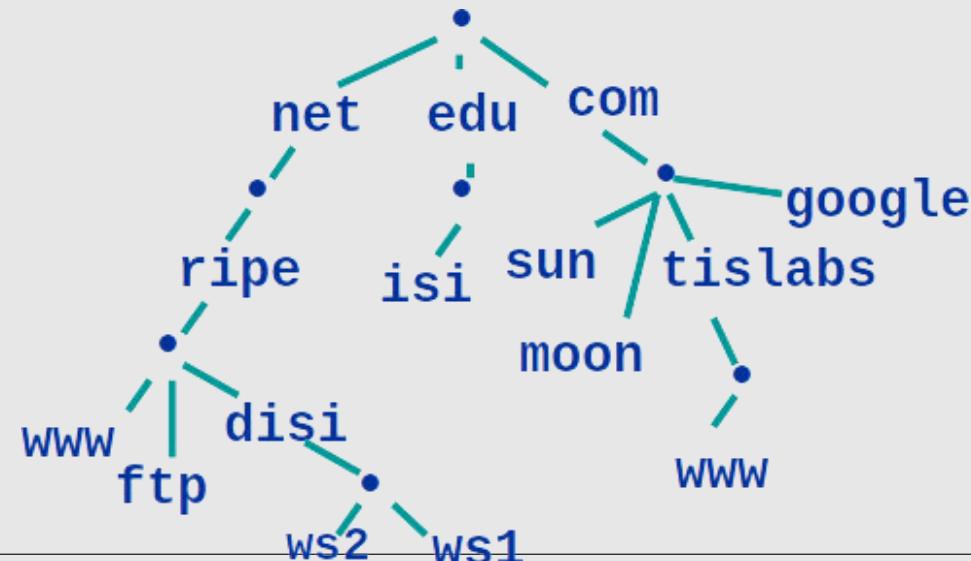
# Concept: DNS Names 3



- Domain names can be mapped to a tree.
- New branches at the 'dots'
- No restriction to the amount of branches.

# Concept: Domains

- Domains are “namespaces”
- Everything below .com is in the com domain.
- Everything below ripe.net is in the ripe.net domain and in the net domain.

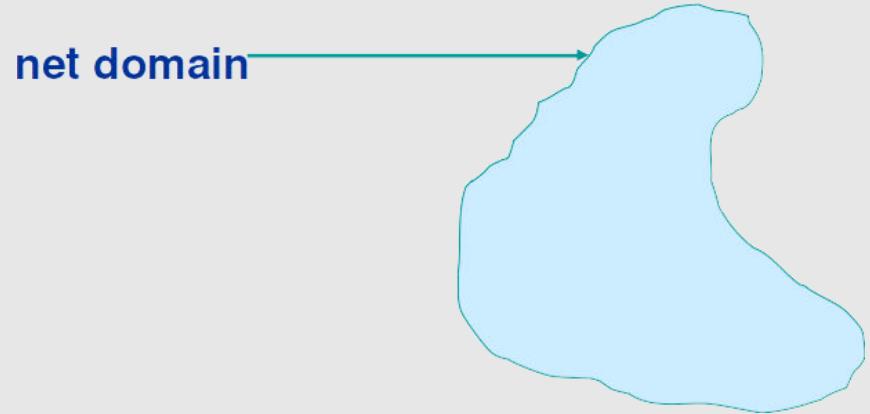


# Delegation

- Administrators can create subdomains to group hosts
  - According to geography, organizational affiliation or any other criterion
- An administrator of a domain can delegate responsibility for managing a subdomain to someone else
  - But this isn't required
- The parent domain retains links to the delegated subdomain
  - The parent domain “remembers” who it delegated the subdomain to

# Concept: Zones and Delegations

- Zones are “administrative spaces”
- Zone administrators are responsible for portion of a domain’s name space
- Authority is delegated from a parent and to a child



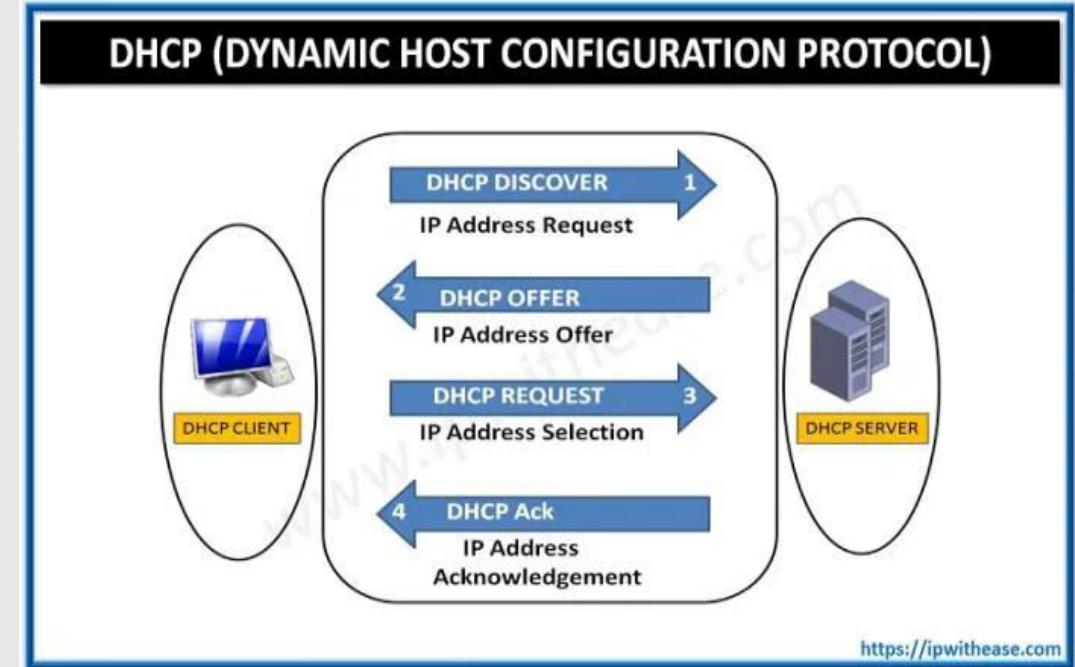
# Outline

- ✓ Introduction to Networking in Linux
- ✓ Network basics and tools
- ✓ File transfer protocol in Linux
- ✓ Network file system
- ✓ Domain Name Services
- ✓ Dynamic Host Configuration Protocol
- Network Information Services



# DHCP

- Dynamic Host Configuration Protocol
- It is a method for assigning Internet Protocol addresses permanently or to individual computers in an organizational setup
- DHCP lets a network administrator supervise and distribute IP addresses from a central point and automatically sends a new IP address when a computer is plugged into a different place in the network

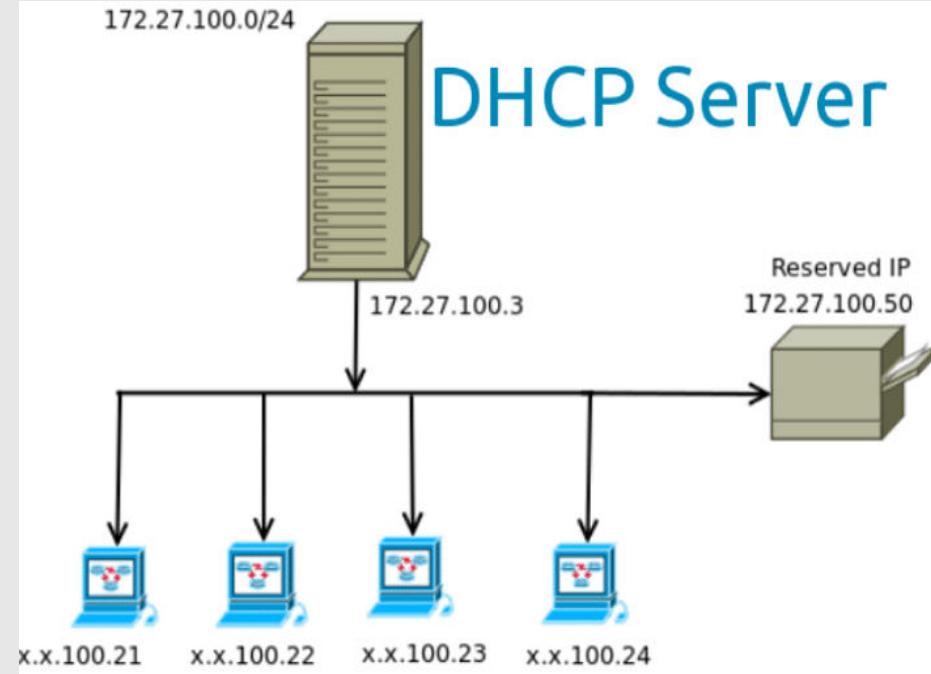


# DHCP

- DHCP was created by Dynamic Host Configuration working group of the IETF (Internet Engineering Task Force)
- Runs over UDP
- Utilizing Ports:
  - 67- connections to server
  - 68- connections to client
- Extension of BOOTP (protocol used for simple interaction)-  
DHCP enhances capabilities of BOOTP
- DHCP is basically used for dynamic configuration
- Uses client-server model

# Motivation for DHCP

- Configuration parameters for network hosts
  - IP address
  - Router
  - Subnet Mask
  - Others..



# DHCP: Objective

- DHCP temporarily binds IP address and other configuration parameters to DHCP client & Provides framework for passing configuration information to hosts
- DHCP was designed to provide computers with temporary address
- DHCP is well adapted to situation where hosts move from one location to another or are routinely connected and disconnected
- Thus, DHCP is mainly used to simplify the installation and maintenance of networked computers

# DHCP: Characteristics

- **Centralized IP address administration**
- **Backward compatible with BOOTP-** Thus a host running the BOOTP client software can request a static configuration from a DHCP server
- **Supports multiple servers**
- **Provides dynamic assignment**
- **Allows static assignment**
- **Does not interact with Domain name service (DNS)**

# 2 Types of IP Addresses

- DHCP is used to assign IP addresses to hosts or workstations on the network
- Two types of IP addresses:
  - Static
    - Is a number that is assigned to a computer by an Internet service provider (ISP) to be its permanent address on the Internet
  - Dynamic
    - The temporary IP address is called a dynamic IP address

# Why is DHCP Important?

- Important when it comes to adding a machine to a network
- When computer requests an address, the administrator would have to manually configure the machine
  - Mistakes are easily made
  - Causes difficulty for both administrator as well as neighbors on the network
- DHCP solves all the hassle of manually adding a machine to a network

# How does DHCP work?

- When a client needs to start up TCP/IP operations, it broadcasts a request for address information
- The DHCP server will not reallocate the address during the lease period and will attempt to return the same address every time the client requests an address
- The client can extend its lease or send a message to the server before the lease expires that it no longer needs the address so it can be released and assigned to another client on the network

# DHCP: Elements

## ➤ Client Software

- Installed in client machines
- To handle broadcast requests
- For automatic IP acquisition and acquiring other configurations

## ➤ Server Software

- Installed in server machines
- Designated to respond to client requests for IP address
- Manage pools of IP addresses and related configuration

## ➤ Relay agent software

- DHCP clients broadcasts requests onto local network
- Router block broadcasts to outer network => responses from DHCP servers must come from same network
- DHCP relay agents intercepts IP address requests
- Repackages the requests
- Rebroadcasts them as unicast messages to DHCP servers
- DHCP server sends its reply to relay agent which in turn forwards them to client requesting the IP address

<https://youtu.be/3Oth1Cx8nrw>



# Terminology

- **DHCP Databases:** DHCP server uses two databases
  - First acquires IP addresses manually and binds them permanently to hardware addresses similar to BOOTP
  - Other contains 1 or more blocks of IP addresses (address pools) that are dynamically assigned to clients on FCFS basis
- **DHCP leases:** DHCP issues a lease for dynamic IP address that expires at the end of lease time
  - After  $\frac{1}{2}$  the lease time, client can renew the lease time
  - Once lease has expired the client must either stop using the IP address or acquire a new IP address
  - If more than 1 DHCP server, each may offer IP address to client and it can select the best offer.
  - 3 Types of address leases
    - Manual lease- Network manager manually assigns all IP addresses
    - Automatic lease- DHCP server assigns
    - Dynamic lease- DHCP server assigns for a specific period of time.

# Advantages of DHCP

- DHCP minimizes the administrative burden
- By using DHCP there is no chance to conflict IP address
- By using DHCP relay agent you provide IP address to another network

# Disadvantages of DHCP

- When DHCP server is unavailable, client is unable to access enterprises network
- Your machine name does not change when you get a new IP address

# Security Problem

- DHCP is an unauthenticated protocol
  - When connecting to a network, the user is not required to provide credentials in order to obtain a lease
  - Malicious users with physical access to the DHCP-enabled network can instigate a denial-of-service attack on DHCP servers by requesting many leases from the server, thereby depleting the number of leases that are available to other DHCP clients

- DHCP servers are easy to administer and can be set-up in just a few minutes
- Client addresses are assigned automatically

# Limitations

- Some machines on your network need to be at fixed addresses, for example servers and routers
- You need to be able to assign a machine to run the DHCP server continually as it must be available at all times when clients need IP access

# Conclusions

- Assigning client addresses automatically is by far the easiest option of the two:
  - Set-up automatically by DHCP server
  - Set-up manually
- To set-up clients automatically all you need to do is set your TCP/IP control panels to receive automatically
- If you intend to set up your client computers manually, make sure that the assigned IP address is in the same range of your default router address and that it is unique to your private network

# References

[http://technet.microsoft.com/en-us/library/cc780347\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc780347(WS.10).aspx)

<http://www.networksorcery.com/enp/protocol/dhcp.htm>

<http://searchsecurity.techtarget.com.au/articles/33496-How-to-stop-rogue-DHCP-server-malware>

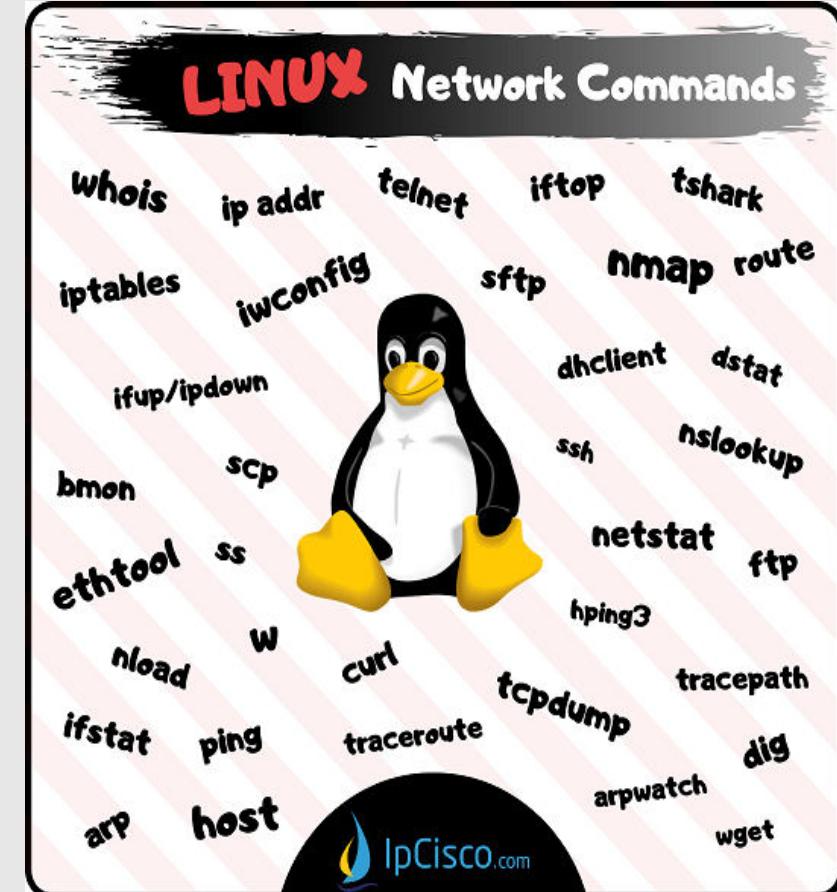
<http://www.spirit.com/Network/net0202.html>

<http://www.juniper.net/techpubs/software/erx/junose93/sw-rn-erx930/html/sw-rn-erx930-body11.html>

<http://technet.microsoft.com/en-us/library/bb680764.aspx>

# Outline

- ✓ Introduction to Networking in Linux
- ✓ Network basics and tools
- ✓ File transfer protocol in Linux
- ✓ Network file system
- ✓ Domain Name Services
- ✓ Dynamic Host Configuration Protocol
- ✓ Network Information Services



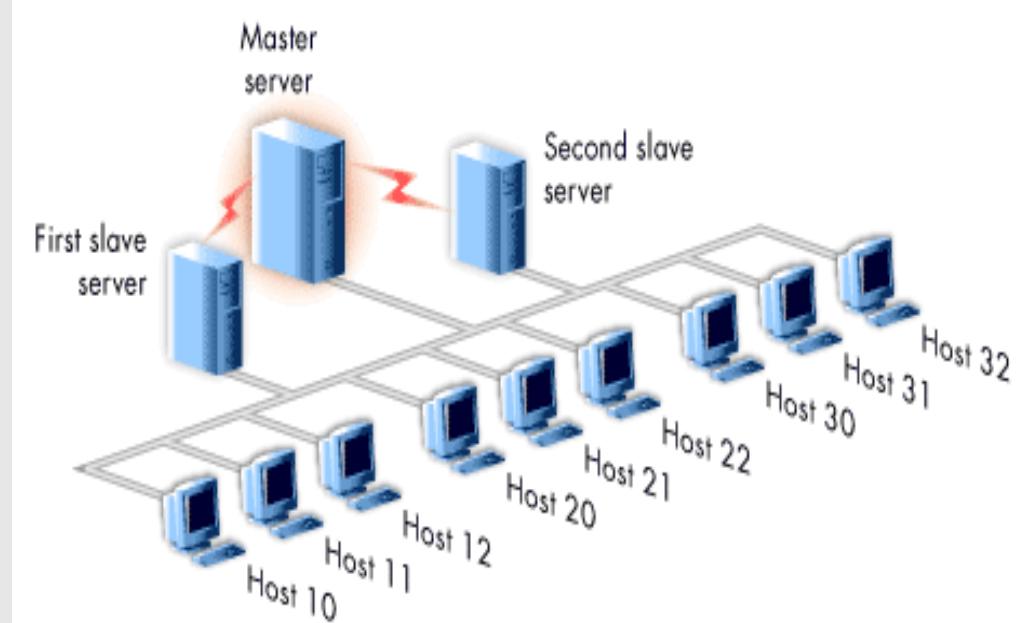
# What is NIS?

- Problems in running a distributed computing environment:
  - Each workstations has its own copies of common configuration files such as passwd, group and host files.
  - These files must be consistent and every changes to these common files must be propagated to every hosts on the network



# NIS: Introduction

- Network Information Service is a naming service.
- It is a mechanism for identifying and locating network objects and resources.
- It provides a uniform storage and retrieval method for network-wide information in a transport-protocol and media-independent fashion.



NIS also allows servers to have slave servers, which hold copies of the maps in the master server. The slaves are updated whenever the map changes. On large networks, NIS servers need slaves simply to handle all the information requests. A network can have a mix of masters and slaves on each map.

# NIS: Introduction

- By running NIS, the system administrator can distribute administrative databases called **Maps**, among variety of server (master and slaves)
- The administrator can update those databases from a **centralized location** in an automatic and reliable fashion to ensure that all clients share the **same naming service information** in a consistent manner throughout the network
- NIS is independent of DNS and has a slightly different focus
  - DNS focuses on making communication simpler by using machine names instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of network information.

# NIS: Introduction

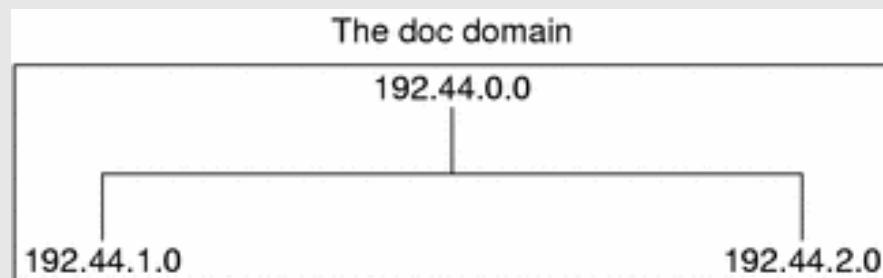
- NIS stores information not only about machine names and addresses, but also about users, the network itself, and network services. This collection of network **information** is referred to as NIS **namespace**.

<https://docs.oracle.com/cd/E19683-01/817-4843/6mkbebda8/index.html#anis1-24268>

<https://docs.oracle.com/cd/E19683-01/817-4843/anis1-20681/index.html>

# NIS Architecture

- NIS uses client-server architecture.
  - NIS servers provide services to NIS clients
- The principal servers called **master** server, and for reliability, they have backup, or **slave** server.
- Both master and slave servers use NIS information retrieval software and both store NIS maps.
- NIS uses domains to arrange the machines, users, and networks in its namespace.
- However, it does not use domain hierarchy, NIS namespace is flat.



# NIS Machine Types

- **NIS Servers-** contains set of maps that system administrator creates and updates
  - Master Server
  - Slave Server
- **NIS Clients-** runs the processes that request data from maps on the server
- Any machine can be NIS Client, but only machines with disks should be NIS servers, either master or slave
- NIS Server does not have to be same machine as NFS file server.
- Each NIS domain must have one and only one master server, which can propagate NIS updates with least performance degradation.
- System administrator designates one master server for all NIS maps.

# NIS Elements

- **NIS Domains** - collection of machines which share common set of NIS maps. Each domain has a domain name and each machine sharing in that domain share same maps
- **NIS Daemons**

Daemon	Function
ypserv	Server process
ypbind	Binding process
ypxfrd	High speed map transfer
rpc.yppasswd	NIS password update daemon
rpc.ypupdate	Modifies other maps such as publickey

## ➤ NIS Utilities

Utility	Function
makedbm	Creates dbm file for an NIS map
ypcat	Lists data in a map
ypinit	Builds and installs an NIS database and initializes NIS client's ypservers list.
ypmatch	Finds a specific entry in a map
yppoll	Gets a map order number from a server
yppush	Propagates data from NIS master to NIS slave server
ypset	Sets binding to a particular server
ypwhich	Lists name of the NIS server and nickname translation table
ypxfr	Transfers data from master to slave NIS server

# NIS Elements

- **NIS Maps:** information in NIS maps is stored in ndbm format.
- NIS maps were designed to replace UNIX /etc files, as well as other configuration files, so they store much more than names and addresses.
- NIS master server for each NIS domain maintains a set of NIS maps for other machines in the domain to query. NIS slave servers also maintain duplicates of the master server's maps. NIS client machines can obtain namespace information from either master or slave servers.
- NIS maps are essentially two-column tables. One column is the **key** and the other column is information related to the key.
- NIS finds information for a client by searching through the keys.
- Some information is stored in several maps because each map uses a different key.

# NIS-related Commands

Command	Description
ypserv	Services NIS clients' requests for information from an NIS map. ypserv is a daemon that runs on NIS servers with a complete set of maps. At least one ypserv daemon must be present on the network for NIS service to function.
ypbind	Provides NIS server binding information to clients. It provides binding by finding a ypserv process that serves maps within the domain of the requesting client. ypbind must run on all servers and clients.
ypinit	Automatically creates maps for an NIS server from the input files. It is also used to construct the initial /var/yp/binding/domain/ypservers file on the clients. Use ypinit to set up the master NIS server and the slave NIS servers for the first time.
make	Updates NIS maps by reading the Makefile (when run in the /var/yp directory). You can use make to update all maps based on the input files or to update individual maps. The <a href="#">ypmake(1M)</a> man page describes the functionality of make for NIS.

# NIS Binding

- NIS clients get information from an NIS server through the binding process, which can work in one of two modes: server-list or broadcast.
- **Server-list:**  
ypbind process queries the `/var/yp/binding/domain/ypservers` list for the names of all of the NIS servers in the domain. The ypbnd process binds only to servers in this file. The file is created by running `ypinit -c`.
- **Broadcast:** The ypbnd process can also use an RPC broadcast to initiate a binding. The servers themselves might exist throughout different subnets since map propagation works across subnet boundaries.
  - In a subnet environment, one common method is to make the subnet router an NIS server. This allows the domain server to serve clients on either subnet interface.



**Thanks!...**

**Any question?**

# **CSE 438: Linux For Devices**

## **Module 3**

### **The Linux Shell and File Structure**

- **Shell** is the user interface to the operating system
- **Functionality:**
  - Manage files (wildcards, I/O redirection)
  - Manage processes (build pipelines, multitasking)
- Most popular shells:
  - The Bourne shell (sh)
  - The Korn shell (ksh)
  - The C shell (csh)
  - The Bourne Again shell (bash)

- **The Bourne shell /bin/sh** ( S. R. Bourne, 1977)
  - powerful syntactical language
  - strong in controlling input and output
  - expression matching facilities
  - ⌚ interactive use: the use of shell functions.
- **The C-shell /bin/csh** (Bill Joy, UCB, 1978)
  - new concepts: job control and aliasing
  - much better for interactive use
  - ⌚ different input language:
    - out went the good control of input and output
    - too buggy to produce robust shell scripts

# A Bit of History

Amity School Of Engineering And Technology

---

- **/bin/tsch** (Ken Greer, Carnegie Mellon University, late 1970s)
  - User –oriented command line editing
  - Out most of the bugs
- **Korn shell /bin/ksh** (David Korn, AT&T, early 1980s)
  - Bourne shell language
  - C shell's features for interactive work
- ☹ You had to pay AT&T for it!
- GNU project: a free shell=> **/bin/bash (the Bourne again shell)**

# SELECTING A SHELL

**Amity School Of Engineering And Technology**

---

```
[c33225@snowball ~]$ echo $SHELL
/bin/bash
[c33225@snowball ~]$ bash
[c33225@snowball ~]$ exit
exit
[c33225@snowball ~]$ ksh
$ exit
[c33225@snowball ~]$ sh
sh-3.2$ exit
exit
[c33225@snowball ~]$ csh
[c33225@snowball ~]$ exit
exit
```

# Changing a Shell

Amity School Of Engineering And Technology

---

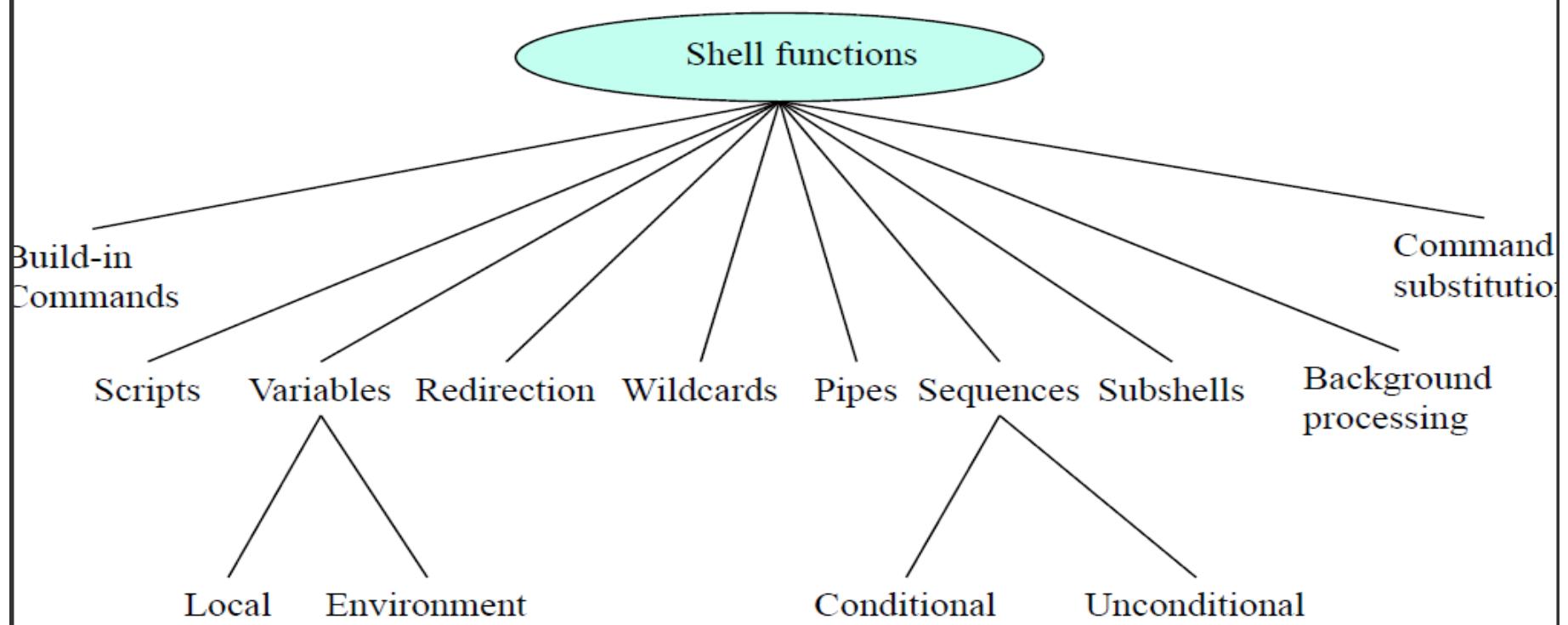
- To change your default shell use the ***chsh*** utility which requires full pathname of the new shell

```
[c33235@snowball ~]$ chsh
Changing shell for c33235.
Password:
New shell [/bin/bash]: /bin/tcsh
Shell changed.
[c33235@snowball ~]$ █
```

---

# Core Shell Functionality

Amity School Of Engineering And Technology



# Invoking the Shell

Amity School Of Engineering And Technology

---

- A shell is invoked, either
  - automatically upon login, or
  - manually from the keyboard or script

# What does the shell do?

Amity School Of Engineering And Technology

---

- The following takes place:
  - (1) reads a special startup file (.cshrc for csh in the user's home directory) and executes all the commands in that file
  - (2) displays a prompt and waits for a user command
  - (3) If user enters CTRL-D (end of input) the shell terminates, otherwise it executes the user command(s)

- ls (list files), ps (process info),
- \ continues line, lp (send to printer)

```
$ ls
```

```
$ ps -ef | sort | ul -tdumb | lp
```

```
$ ls | sort | \  
lp
```

## UNIX Commands vs Built-in Shell Commands

- ✖ Shell locate the execute program in the “/bin” directory and executes it
  - Example: ls
  
- ✖ Shells have built-in commands, which it recognizes and executes internally
  - Example: echo, cd

## echo

- Built-in shell command
- Displays its arguments to standard output
- By default, appends a newline to the output
- Subtle behavior may vary among different shells

```
[c33235@snowball ~]$ echo Vancouver is a nice city
Vancouver is a nice city
[c33235@snowball ~]$ █
```

- **Metacharacters** are characters that specially processed by a shell.
- When you enter a command, shell
  - scans for metacharacters,
  - specially processes them,
  - then executes command.
- To turn off the meaning of a metacharacter: \

```
[c33225@snowball ~]$ echo hi>file
[c33225@snowball ~]$ cat file
hi
[c33225@snowball ~]$ echo hi \>file
hi >file
[c33225@snowball ~]$
```



Symbol	Meaning
>	Output redirection: writes standard output to a file.
>>	Output redirection: appends standard output to a file.
<	Input redirection: reads standard input from a file.
<< <i>tok</i>	Input redirection: reads standard input from script, up to <i>tok</i> .
*	File substitution wildcard; matches zero or more characters.
?	File substitution wildcard; matches any single character.
[ . . ]	File substitution wildcard; matches any character between brackets.
'command'	Command substitution; replaced by the output from <i>command</i> .
	Pipe symbol; sends output of one process to the input of another
;	Separates sequences of commands (or pipes) that are on one line.
&&	Conditional execution; executes the command only if the previous one succeeded.
	Conditional execution; executes the command if the previous one failed.
()	Groups commands
\	Prevents special interpretation of the next character.
&	Places a process into the background.
\$	Expands the value of a variable.

- **command > file**
  - Creates file if it doesn't exist; Overwrites if file exists
  - If file exists and hasn't write permission, an error occur
- **command >> file**
  - Creates file if it doesn't exist, Appends to file if file exists

```
[c33225@snowball ~]$ echo hi>file
[c33225@snowball ~]$ cat file
hi
[c33225@snowball ~]$ echo Hello. world >> file
[c33225@snowball ~]$ cat file
hi
Hello. world
```

# Input Redirection

Amity School Of Engineering And Technology

---

- **command < file**
  - If file doesn't exist or hasn't read permission, an error occur
- **command << word**
  - Copies its standard input up to (not including) the line starting with word into buffer which is used as input

## Wildcards: Filename Substitution

- **Globbing** is process of replacement of the pattern with metacharacters by an alphabetically sorted list of all matching filenames.

*	File substitution wildcard; matches zero or more characters.
?	File substitution wildcard; matches any single character.
[...]	File substitution wildcard; matches any character between brackets.

- If string is surrounded by single (double) quotes, **shell doesn't process wildcards** in the string.

# Wildcards Example

Amity School Of Engineering And Technology

---

```
[c33225@snowball ~]$ ls *.script
awk2.script awk.script
[c33225@snowball ~]$ ls aw?.script
awk.script
[c33225@snowball ~]$ ls [ef]*
ex1.tar ex1.txt ex2.txt file file1.tar file.tar
[c33225@snowball ~]$ ls *.perl
ls: *.perl: No such file or directory
```

- Pipe metacharacter : |
- **command1 | command 2 | command 3**
- Such sequence of commands is called **pipeline**
  - Large problem can be often solved by a chain of smaller processes,
  - each performed by a relatively small, reusable utility
- The standard error channel is not piped through a standard pipeline (some shell supports it).
- ls | wc -w

- Causes standard input to be copied to file and also sent to standard output.
- **tee -ia {fileName}+**
  - **-a**: appends the input to the file instead of overwriting
  - **-i**: ignores interrupts

<https://linuxize.com/post/linux-tee-command/>

# tee Example

Amity School Of Engineering And Technology

---

```
[c33225@snowball ~]$ who
c33225  pts/1      2010-06-29 22:13 (adsl-92-195-16.asm.bellsouth.net)
sou     pts/2      2010-06-29 11:46 (ascsc-1408-w7.cs.gsu.edu)
c33201  pts/3      2010-06-29 20:51 (c-98-219-41-48.hsd1.ga.comcast.net)
```

```
[c33225@snowball ~]$ who | tee who.capture | sort
c33201  pts/3      2010-06-29 20:51 (c-98-219-41-48.hsd1.ga.comcast.net)
c33225  pts/1      2010-06-29 22:13 (adsl-92-195-16.asm.bellsouth.net)
sou     pts/2      2010-06-29 11:46 (ascsc-1408-w7.cs.gsu.edu)
```

- A command in grave accents (`) is executed
- Its standard output is inserted in the command in its place
- Any newlines in the output are replaced with spaces.

```
[c33235@snowball ~]$ echo Today is `date`
Today is Tue Feb 23 02:51:54 EST 2010
[c33235@snowball ~]$ echo The perl is in `which perl`
The perl is in /usr/bin/perl
```

# Sequences

Amity School Of Engineering And Technology

---

- Commands or pipelines separated by semi-colons
- Executed from left to right

```
[c33225@snowball ~]$ date > date.txt; ls; pwd > pwd.txt; who | sort >
sort.txt; cat date.txt; cat pwd.txt; cat sort.txt
awk2.script  date.txt  file  practice  script.ksh  temp
awk.script   ex1.tar  file1.tar  pwd.txt  script.sh  testif.sh
databook1.txt  ex1.txt  file.tar  readme.sh  sort.txt  who.capture
databook.txt  ex2.txt  letter  script.csh  temo.txt.4014
Tue Jun 29 23:27:27 EDT 2010
/home/c33225
c33225  pts/1      2010-06-29 22:13 (adsl-92-195-16.asm.bellsouth.net)
sou    pts/2      2010-06-29 11:46 (ascsc-1408-w7.cs.gsu.edu)
```

# Conditional Sequences

- Every Unix process terminates with an exit value:
  - **0 means successful completion**
  - Non zero means failure
  - Built-in shell commands return **1 if they fail.**
- &&: executes the next commands only if the previous command returns 0.
- ||: executes the next command only if the previous command returns a nonzero exit code

```
[c33235@snowball ~]$ find *.txt && cp *.txt temp && ls ./temp
1.txt
date.txt
pwd.txt
sort.txt
1.txt date.txt pwd.txt sort.txt vi.txt
[c33235@snowball ~]$ rm temp || echo Directory is not empty
rm: cannot remove 'temp': Is a directory
Directory is not empty
[c33235@snowball ~]$
```

# Grouping Commands

- Commands can be grouped by placing them between ()
- Grouped commands are executed by a child shell (subshell).
- Grouped commands share standard input, standard output and standard error channels.
- Group can be redirected and piped.



# Grouping Example

```
[c33235@snowball ~]$ ( date ; ls ; pwd ; who | sort ) > sort.txt
[c33235@snowball ~]$ cat sort.txt
Tue Feb 23 03:36:53 EST 2010
1.crypt
1.txt
date.txt
pwd.txt
sort.txt
temp
tmp
tr
users
/home/c33235
c33212 pts/5      2010-02-22 16:03 (adsl-065-006-144-099.sip.asm.bellsouth.net)
c33235 pts/1      2010-02-23 03:26 (adsl-145-123-76.asm.bellsouth.net)
c33235 pts/6      2010-02-23 03:26 (adsl-145-123-76.asm.bellsouth.net)
sou    pts/3      2010-02-19 11:27 (ascsc-1408-w7.cs.gsu.edu)
sou    pts/4      2010-02-11 11:16 (ascsc-1408-w7.cs.gsu.edu)
```

- Command, or pipeline, or sequence, or group of commands **is followed by &:**
  - A subshell is created to execute the commands as a background process
  - Runs concurrently with a parent shell
  - Does not take control over keyboard
- Returns **user job number** and **system process number**

```
[c33225@snowball ~]$ find . -name '*.sh' -print &
[1] 4901
```

```
[c33225@snowball ~]$ ./readme.sh
./testif.sh
./script.sh
```

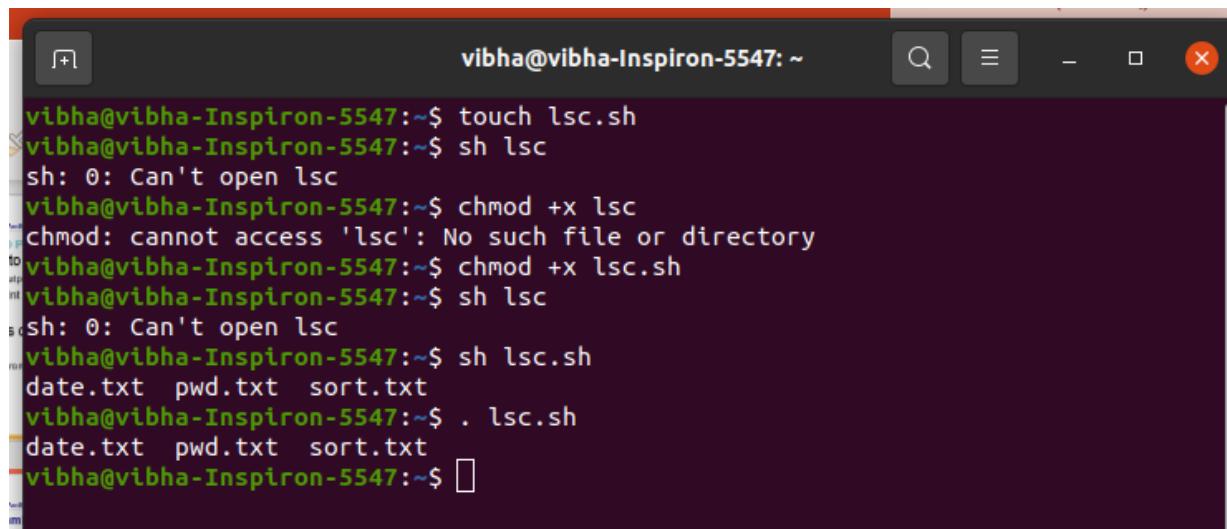
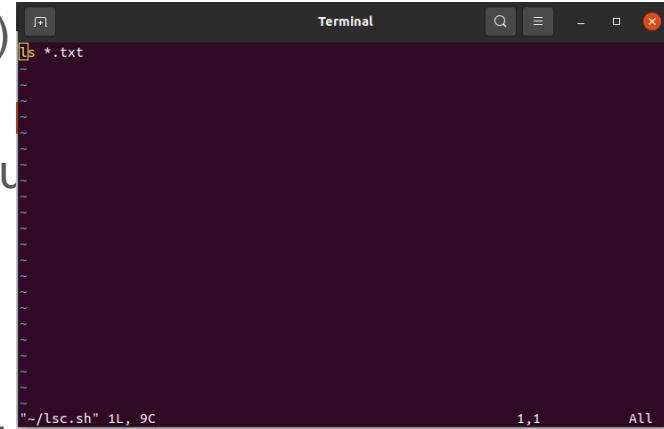
# BACKGROUND PROCESSING

- Redirect the output to a file (if desired)
  - prevents background output on terminal
  - find . – name ‘\*.c’ –print > find.txt &
- Background process cannot read from standard input
  - If they attempt to read from standard input; they terminate.

# Shell Programming: Scripts

Amity School Of Engineering And Technology

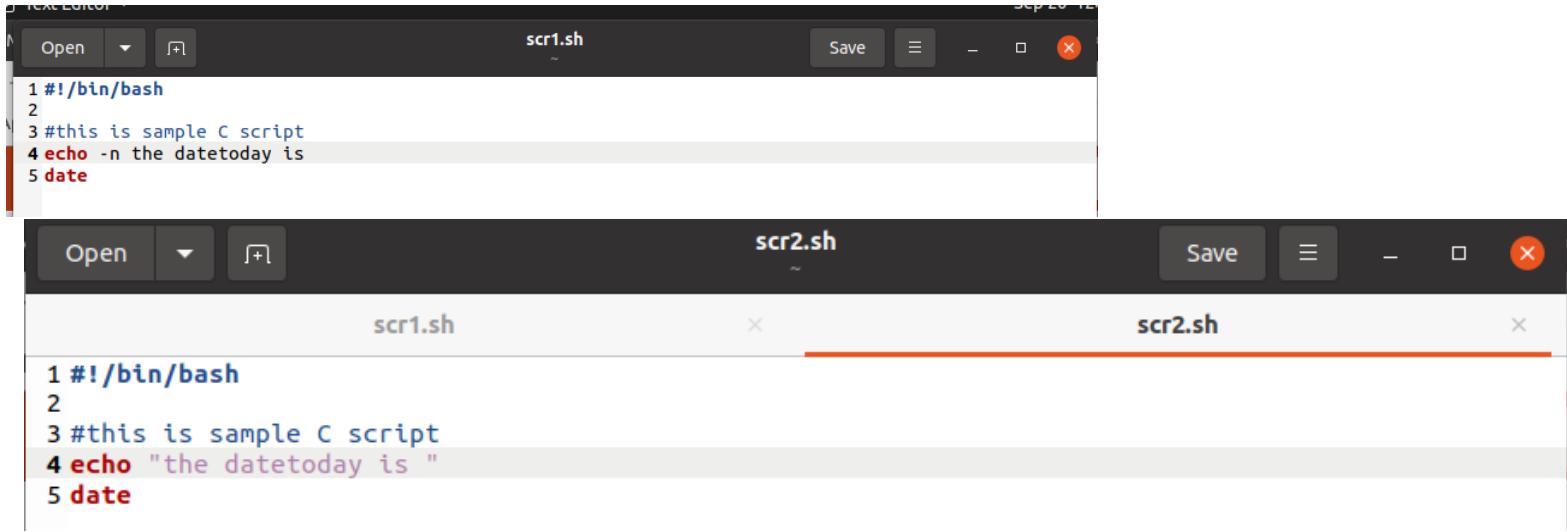
- **Script** is a text file containing series of shell commands for later execution.
- Execute permission (**chmod +x scriptName**)
- When script is run
  - System determines which shell should be used
  - Executes the shell with script as input
- If the first line is
  - **#**: current shell
  - **#!pathName**: the executable program *pathName*
  - otherwise: Bourne shell



```
vibha@vibha-Inspiron-5547:~$ touch lsc.sh
vibha@vibha-Inspiron-5547:~$ sh lsc
sh: 0: Can't open lsc
vibha@vibha-Inspiron-5547:~$ chmod +x lsc
chmod: cannot access 'lsc': No such file or directory
vibha@vibha-Inspiron-5547:~$ chmod +x lsc.sh
vibha@vibha-Inspiron-5547:~$ sh lsc
sh: 0: Can't open lsc
vibha@vibha-Inspiron-5547:~$ sh lsc.sh
date.txt  pwd.txt  sort.txt
vibha@vibha-Inspiron-5547:~$ . lsc.sh
date.txt  pwd.txt  sort.txt
vibha@vibha-Inspiron-5547:~$ 
```

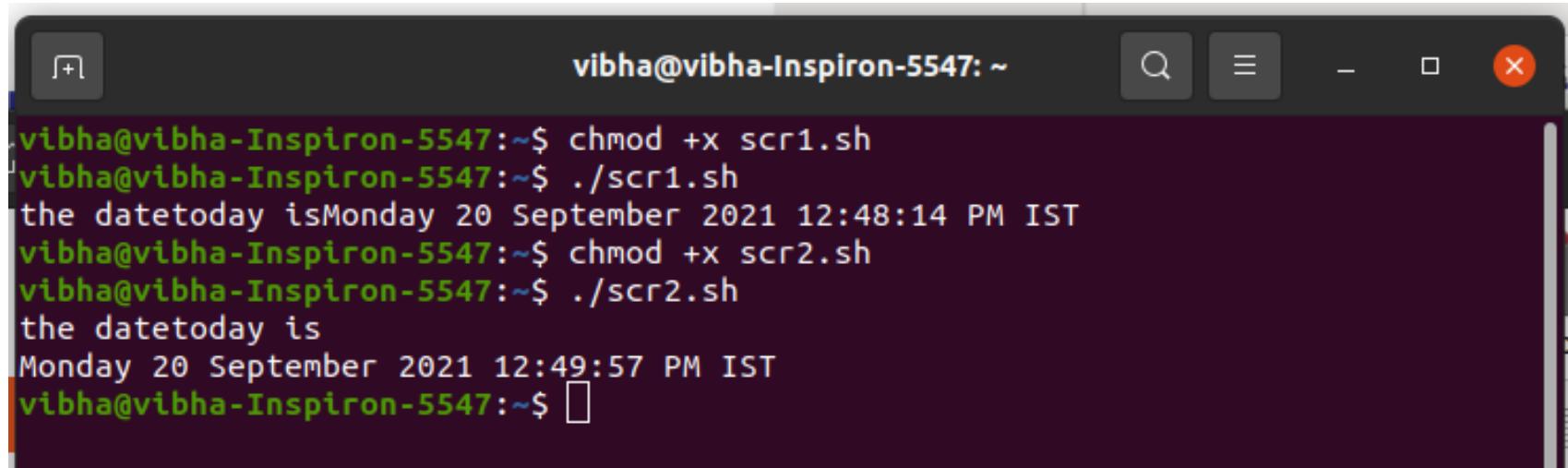
# Shell Programming: Scripts

Amity School Of Engineering And Technology



```
scr1.sh
1 #!/bin/bash
2
3 #this is sample C script
4 echo -n the datetoday is
5 date

scr2.sh
1 #!/bin/bash
2
3 #this is sample C script
4 echo "the datetoday is "
5 date
```



```
vibha@vibha-Inspiron-5547:~$ chmod +x scr1.sh
vibha@vibha-Inspiron-5547:~$ ./scr1.sh
the datetoday isMonday 20 September 2021 12:48:14 PM IST
vibha@vibha-Inspiron-5547:~$ chmod +x scr2.sh
vibha@vibha-Inspiron-5547:~$ ./scr2.sh
the datetoday is
Monday 20 September 2021 12:49:57 PM IST
vibha@vibha-Inspiron-5547:~$ 
```

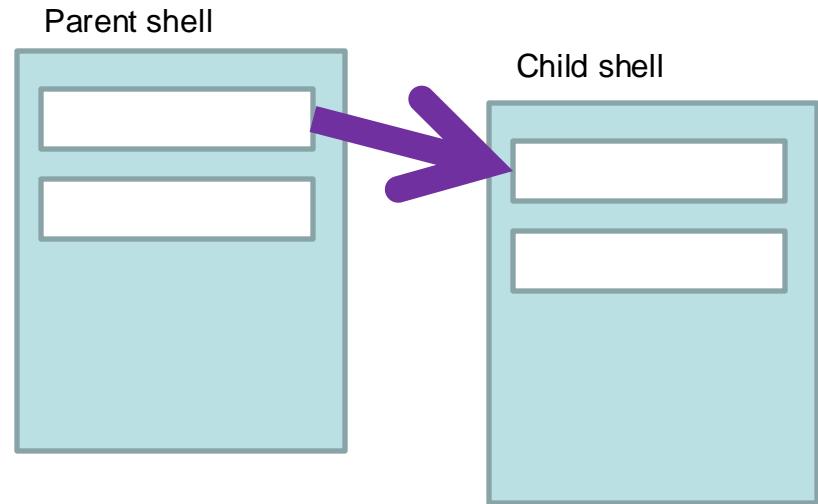
- Several ways a parent shell can create a child shell (**subshell**)
  - Group of commands are executed
  - Script is executed
  - Background job (concurrent execution)
- Parent shell sleeps till child shell terminates (execution in background can be useful)
- A subshell has its own working directory
  - `cd` commands in subshell do not change working directory of parent shell

```
[c33235@snowball ~] $ pwd
/home/c33235
[c33235@snowball ~] $ (cd temp;pwd)
/home/c33235/temp
[c33235@snowball ~] $ pwd
/home/c33235
```

# Shell's Data Areas

Amity School Of Engineering And Technology

- Every shell has two data areas
  - environment space
  - local-variable space
- Child shell gets a copy of the parent's environment space
  - starts with an empty local-variable space.



- User defined variables within a shell are known as **Shell Variables**
- A shell supports two kinds of variables:
  - Local variables
  - Environment variables
  - Both hold data in string format
- Every shell has a set of pre-defined environment variables and local variables
- A variable name may be any set of alphabetic characters including underscore
  - A name may also include a number , but the number cannot be the first character in the name.
  - A name may not have any other type of character, such as ?, & or even space
  - A name may not include more than one word
  - Shell uses spaces on the command line to distinguish different components of a command such as options, arguments and name of the command
  - A Value to a variable is assigned using assignment operator
    - Poet=Virgil
- Accessing variables in all shells is done by prefixing the name with a \$ sign.



# Variables

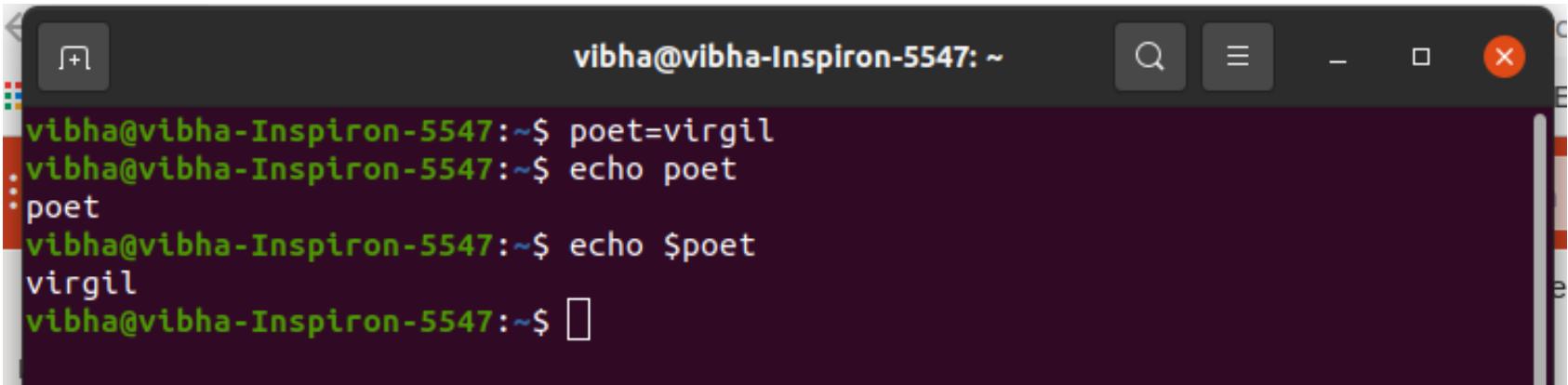
AMITY  
UNIVERSITY

Amity School Of Engineering And Technology

- Some pre-defined environment variables available in all shells:
  - **\$HOME** : full pathname of home directory
  - **\$PATH** : list of directories to search for commands
  - **\$MAIL** : the full pathname of mailbox
  - **\$USER** : your username
  - **\$SHELL** : the full pathname of login shell
  - **\$TERM** : the type of your terminal

```
vibha@vibha-Inspiron-5547:~$ $HOME
bash: /home/vibha: Is a directory
vibha@vibha-Inspiron-5547:~$ $PATH
bash: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/u
sr/local/games:/snap/bin: No such file or directory
vibha@vibha-Inspiron-5547:~$ $MAIL
vibha@vibha-Inspiron-5547:~$ $USER
vibha: command not found
vibha@vibha-Inspiron-5547:~$ $SHELL
vibha@vibha-Inspiron-5547:~$ $TERM
xterm-256color: command not found
vibha@vibha-Inspiron-5547:~$ 
```

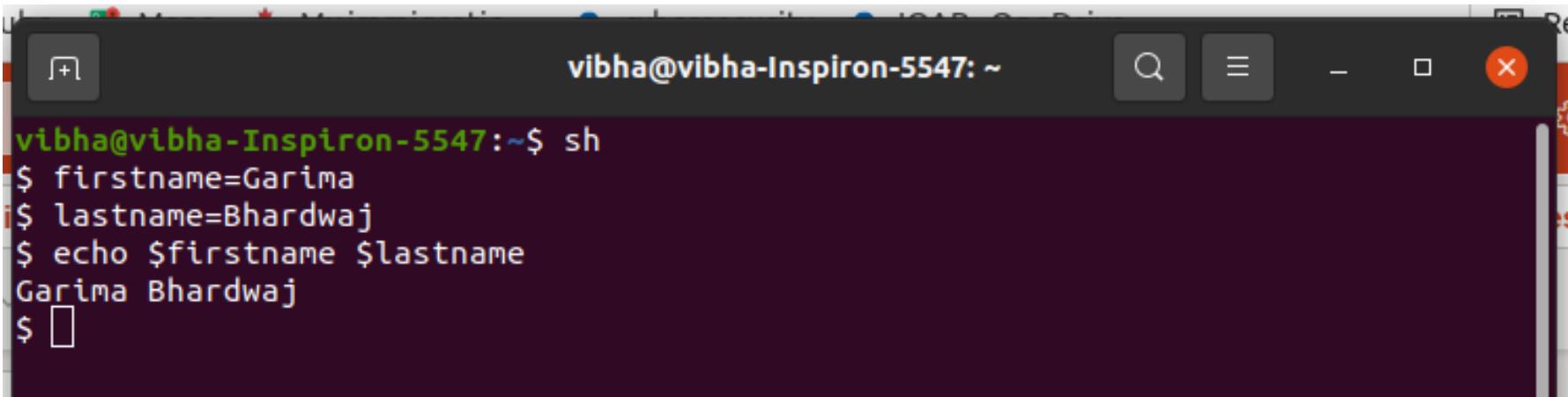
- Depends on shell:
  - sh, bash, ksh:
    - variable=value
    - variable="value"
  - **Notice no spaces around equal sign**
  - csh:
    - set variable=value
    - set variable="value"
- Access the variable: append prefix \$ to the name of a variable.
  - **\$variableName**



```
vibha@vibha-Inspiron-5547:~$ poet=virgil
vibha@vibha-Inspiron-5547:~$ echo poet
poet
vibha@vibha-Inspiron-5547:~$ echo $poet
virgil
vibha@vibha-Inspiron-5547:~$ 
```

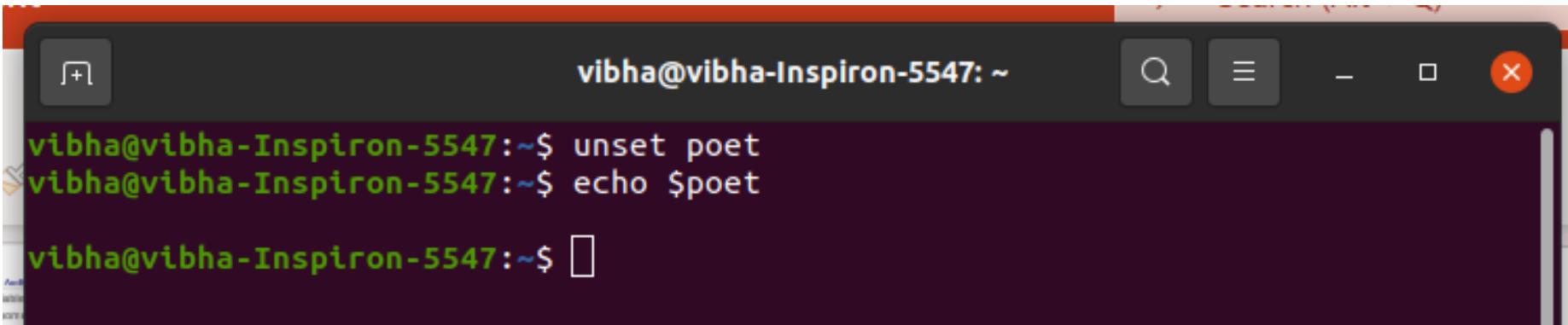
A screenshot of a terminal window titled "vibha@vibha-Inspiron-5547: ~". The window has a dark theme with light-colored text. It shows a command-line session where the user has assigned the value "virgil" to the variable "poet" using the command "poet=virgil". Then, they used the command "echo poet" to print the value of the "poet" variable, which output "poet". Finally, they used "echo \$poet" to print the value of the "poet" variable again, which output "virgil". The terminal window includes standard Linux-style window controls at the top right.

# Assigning Variables



```
vibha@vibha-Inspiron-5547:~$ sh
$ firstname=Garima
$ lastname=Bhardwaj
$ echo $firstname $lastname
Garima Bhardwaj
$ 
```

- **set** command is used to obtain list of all defined variables and their values.
- The **unset** command undefines a variables



```
vibha@vibha-Inspiron-5547:~$ unset poet
vibha@vibha-Inspiron-5547:~$ echo $poet

vibha@vibha-Inspiron-5547:~$ 
```

- **\$\$** the process ID of the shell
  - **\$0** the name of the shell script
  - **\$1..\$9** \$n refers to the n<sup>th</sup> command line argument
- ~~**\$\***~~ a list of all command-line arguments

# Example

```
vibha@vibha-Inspiron-5547:~$ cat script1.sh
echo the name of the file is $0
echo the first argumnet is $1
echo all arguments are $*
echo place information into temp file $1.$$
date>$1.SS
date>$1.$$
```

```
vibha@vibha-Inspiron-5547:~$ chmod +x script1.sh
vibha@vibha-Inspiron-5547:~$ ./script1.sh
```

```
the name of the file is ./script1.sh
```

```
the first argumnet is
```

```
all arguments are
```

```
place information into temp file .18117
```

```
./script1.sh: line 7: $'\030': command not found
```

```
vibha@vibha-Inspiron-5547:~$ ./script1.sh temo.txt 1 2 3 4 5
```

```
the name of the file is ./script1.sh
```

```
the first argumnet is temo.txt
```

```
all arguments are temo.txt 1 2 3 4 5
```

```
place information into temp file temo.txt.18128
```

```
./script1.sh: line 7: $'\030': command not found
```

```
vibha@vibha-Inspiron-5547:~$ □
```

- -Single quotes(') inhibit wildcard replacement, variable substitution and command substitution.
- -Double quotes (") inhibit wildcard replacement only.
- -When quotes are nested, only the outer quotes have any effect.

- 12

```
vibha@vibha-Inspiron-5547:~$ echo 3 * 4 = 12
3 abb.zip abc.sh abc.txt abc.zip anaconda3 Desktop Documents
 Downloads first.sh hello hello1.text hello1.txt MLCourse Mu
sic PandasTutorial.ipynb Pictures pt Public script1.sh snap
temo.txt.18128 temo.txt.SS Templates Videos VirtualBox VMs 4
 = 12
vibha@vibha-Inspiron-5547:~$ echo '3 * 4 = 12'
3 * 4 = 12
vibha@vibha-Inspiron-5547:~$ echo date is `date`
date is date
vibha@vibha-Inspiron-5547:~$ echo date is `date`
date is Tuesday 29 August 2023 11:37:30 AM IST
vibha@vibha-Inspiron-5547:~$ echo date is "`date`"
date is Tuesday 29 August 2023 11:37:37 AM IST
vibha@vibha-Inspiron-5547:~$ echo date is "'date'"
date is 'date'
vibha@vibha-Inspiron-5547:~$ □
```

- **ps** : allows to monitor a status of processes.
- **kill**: allows you to terminate a process on the basis of its ID number
- **wait**: allows a shell to wait for one of its child processes to terminate
- **sleep seconds**: sleeps for the specified number of seconds, then terminates

# ps Command

Amity School Of Engineering And Technology

---

- \$ ps -efl
  - - e: include all running processes
  - - f: include full listing
  - - l: include long listing
- PID : process ID

- **kill [signalId] {pid}+**
- **kill -l**
  - **-l**: list valid signal names
- Sends the signal with code signal-Id to the list of numbered processes.
  - **signalId** is the number or name of the signal
  - if signal is not specified the default signal is **SIGTERM (15)**
  - **SIGKILL (9)** is useful if the process refuses to die

```
[c33235@snowball ~]$ kill -l
HUP INT QUIT ILL TRAP ABRT BUS FPE KILL USR1 SEGV USR2 PIPE ALRM TERM STKFI
CHLD CONT STOP TSTP TTIN TTOU URG XCPU XFSZ VTALRM PROF WINCH POLL PWR SYS
RTMIN RTMIN+1 RTMIN+2 RTMIN+3 RTMAX-3 RTMAX-2 RTMAX-1 RTMAX
[c33235@snowball ~]$ █
```

- Processes may protect themselves from all signals except KILL
  - Process wouldn't be allowed to clean up and terminate normally, as it would do when it receives TERM
- The kill utility allows you to specify 0 as pid
  - All processes associated with shell would be terminated.

```
[c33235@snowball ~]$ ( sleep 1000 ; echo done ) &
[1] 2450
[c33235@snowball ~]$ ps
  PID TTY      TIME CMD
 2323 pts/1    00:00:00 tcsh
 2446 pts/1    00:00:00 sleep
 2450 pts/1    00:00:00 tcsh
 2451 pts/1    00:00:00 sleep
 2453 pts/1    00:00:00 ps
[c33235@snowball ~]$ kill -KILL 2450
[c33235@snowball ~]$ ps
  PID TTY      TIME CMD
 2323 pts/1    00:00:00 tcsh
 2446 pts/1    00:00:00 sleep
 2451 pts/1    00:00:00 sleep
 2454 pts/1    00:00:00 ps
[1] + Killed          ( sleep 1000; echo done )
[c33235@snowball ~]$ █
```

- **wait pid**

- -Causes the shell to suspend operation until the child process with the specified process ID terminates.
- -If no arguments are specified, shell waits for all of its child processes to terminate.

```
vibha@vibha-Inspiron-5547:~$ (sleep 30; echo done1) &
[1] 18644
vibha@vibha-Inspiron-5547:~$ (sleep 30; echo done2) &
[2] 18648
vibha@vibha-Inspiron-5547:~$ echo done1
do
[1]- Done                      ( sleep 30; echo done1 )
vibha@vibha-Inspiron-5547:~$ done2
[2]+ Done                      ( sleep 30; echo done2 )
vibha@vibha-Inspiron-5547:~$ echo done 3; wait; echo done 4
done 3
done 4
vibha@vibha-Inspiron-5547:~$ 
```

# Finding a Command: \$PATH

Amity School Of Engineering And Technology

-Shell checks if command is built-in.

- -If yes, executes it directly.
- Otherwise, checks whether it begins with /
  - -If yes, shell assumes that it is the absolute pathname of a command and tries to execute the file with such absolute pathname.
  - -If there is no such file or it's not executable, an error occurs.

```
vibha@vibha-Inspiron-5547:~$ /bin/ls
abb.zip      hello
abc.sh       hello1.text
abc.txt      hello1.txt
abc.zip      MLCourse
anaconda3    Music
Desktop      PandasTutorial.ipynb
Documents    Pictures
Downloads   pt
first.sh     Public
vibha@vibha-Inspiron-5547:~$ /bin/crypt
bash: /bin/crypt: No such file or directory
vibha@vibha-Inspiron-5547:~$ /bin/passwd
Changing password for vibha.
Current password:
passwd: Authentication token manipulation error
passwd: password unchanged
vibha@vibha-Inspiron-5547:~$ █
```

- Else, shell searches the directories (from left to right) in the PATH environment variable.
- If match is found, the file is executed.
- If match is not found, or it's not an executable, an error occurs.
- If PATH is not set or is empty string, only current directory is searched.

```
[c33235@snowball ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
[c33235@snowball ~]$ crypt
bash: crypt: command not found
[c33235@snowball ~]$ █
```

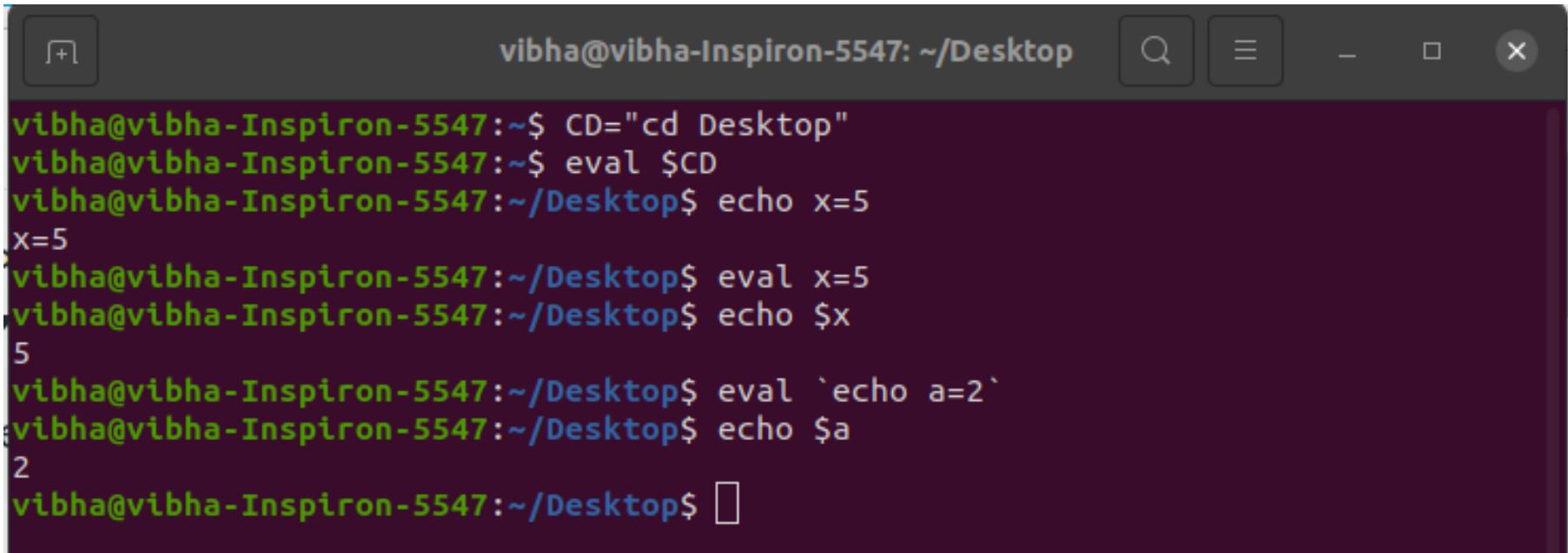
- Bourne, Korn and Bash shells: special variable **\$?** Which contains the value of the previous command's exit code.
- C shell: **\$status** holds the exit code.

```
dash: /bin/crypt: no such file or directory
vibha@vibha-Inspiron-5547:~$ /bin/passwd
Changing password for vibha.
Current password:
passwd: Authentication token manipulation error
passwd: password unchanged
vibha@vibha-Inspiron-5547:~$ echo $?
10
vibha@vibha-Inspiron-5547:~$ echo passwrd
passwrd
vibha@vibha-Inspiron-5547:~$ echo $?
0
vibha@vibha-Inspiron-5547:~$ █
```

- Any script written by you should contain the exit command:
  - *exit <number>*
- If the script does not exit with a exit code, the exit code of the last command is returned by default.

# Core Built-In Commands: eval

- **eval** is a built-in linux command which is used to execute arguments as shell Command.
  - Combines arguments into a single string and uses it as an input to the shell and execute the commands
  - Syntax: eval [arg ...]



vibha@vibha-Inspiron-5547:~/Desktop

```
vibha@vibha-Inspiron-5547:~$ CD="cd Desktop"
vibha@vibha-Inspiron-5547:~$ eval $CD
vibha@vibha-Inspiron-5547:~/Desktop$ echo x=5
x=5
vibha@vibha-Inspiron-5547:~/Desktop$ eval x=5
vibha@vibha-Inspiron-5547:~/Desktop$ echo $x
5
vibha@vibha-Inspiron-5547:~/Desktop$ eval `echo a=2`
vibha@vibha-Inspiron-5547:~/Desktop$ echo $a
2
vibha@vibha-Inspiron-5547:~/Desktop$ 
```

- Causes the shell's image to be replaced with command in the process' memory space.
- The **exec** command does not spawn a new process. Instead, exec command is executed in place of the current shell .
- If commands is successfully executed, the shell that performed the exec ceases to exist.
- If it was login shell, the login session is terminated when command terminates.

```
[c33235@snowball ~]$ exec date
Thu Feb 25 07:05:11 EST 2010
sh-3.2$ █
```

# Common Core Built-in commands

- exec command
  - The exec shell command causes the shell's image to be replaced with the command in the process' memory space.
  - As a result, if the command terminates, the shell also ceases to exist; If the shell was a login shell, the login session terminates.

- This command causes all of the positional parameters \$2..\$n to be renamed \$1..\$(n-1) and \$1 is lost.
- Useful in processing command line parameters. \$ cat script3.csh

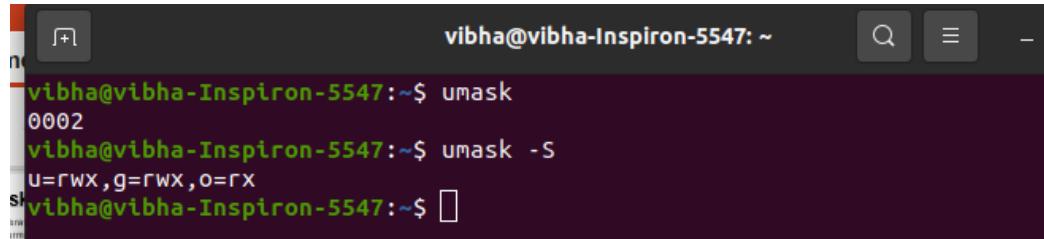
```
» #!/bin/csh
» echo first argument is $1, all args are $*
» shift
» echo first argument is $1, all args are $*
»
» $ script3.csh a b c d
» first argument is a, all args are a b c d
» first argument is b, all args are b c d
```

- On Linux and Unix operating system, all new files are created with a default set of permissions.
- The **umask utility** allows you to view or to set the file mode creation mask, which determines the permissions bits for newly created files or directories.
- It is used by mkdir, touch, [tee](#) , and other commands that create new files and directories.
- Every linux file is associated with an owner and a group and assigned with permission access rights for three different classes of users:
  - The file owner.
  - The group members.
  - Everyone else.
- There are three permissions types that apply to each class.

- Linux does not allow a file to be created with execute permissions
- Default permission values are
  - File 666 => read and write permission to users, group and others
  - Directory 777 => read, write and execute permission to users, group and others
- Default creation permissions can be modified using *umask* utility
- Every Unix process has a special quantity called umask value.
  - Default value: 022 octal

<https://linuxize.com/post/umask-command-in-linux/>

- To calculate the umask value, simply subtract the desired permissions from the default one:
  - Umask value:  $777 - 750 = 027$
- E.g. made by vi or by redirection
  - File permissions (usually 666) masked with umask value
  - Example: 022 to produce the permission 644
- To see current umask value:
  - `$ umask`
- To change umask value:
  - `$ umask octal/Value`



```
vibha@vibha-Inspiron-5547:~$ umask
0002
vibha@vibha-Inspiron-5547:~$ umask -S
u=rwx,g=rwx,o=rwx
vibha@vibha-Inspiron-5547:~$
```

	r	w	x	r	w	x	r	w	x
Original	1	1	0	1	1	0	1	1	0
Mask	0	0	0	0	1	0	0	1	0
Final	1	1	0	1	0	0	1	0	0

- Covered core shell functionality

- Built-in commands
- Scripts
- Variables
- Redirection
- Wildcards
- Pipes
- Subshells
- Background processing

## Linux for Devices

Module-4, Lecture-1

By: Dr. A K Yadav (9911375598)

Dept of CSE, ASET, AUUP

- Creating User Accounts
  - ✓ Adding user with User Manager
  - ✓ Adding user with useradd
  - ✓ Modifying users with usermod
  - ✓ Deleting users with userdel

User Manager

File Edit Help

Add User Add Group Properties Delete Refresh Help

Search filter:

Users Groups

User Name	User ID	Primary Group	Full Name	Login Shell	Home Director
cnegus	13597	cnegus	Chris Negus	/bin/bash	/home/cnegus
joe	13598	joe	Joe Jones	/bin/bash	/home/joe
bill	13599	bill	Bill Jones	/bin/bash	/home/bill

Add New User

User Name:	mary
Full Name:	Mary Walker
Password:	*****
Confirm Password:	*****
Login Shell:	/bin/bash
<input checked="" type="checkbox"/> Create home directory	
Home Directory: /home/mary	
<input checked="" type="checkbox"/> Create a private group for the user	
<input type="checkbox"/> Specify user ID manually:	13601
<input type="checkbox"/> Specify group ID manually:	13601

Cancel  OK

- Understanding Group Accounts
  - ✓ Using group accounts
  - ✓ Creating group accounts
- Setting permissions with Access Control Lists
  - ✓ Setting ACLs with setfacl
  - ✓ Setting default ACLs
  - ✓ Enabling ACLs

- Adding directories for users to collaborate
  - ✓ Creating group collaboration directories (set GID bit)
  - ✓ Creating restricted deletion directories (sticky bit)

Name	Numeric value	Letter value
Set user ID bit	4	u+s
Set group ID bit	2	g+s
Sticky bit	1	o+t



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

## Linux for Devices

Module-4, Lecture-2

By: Dr. A K Yadav (9911375598)  
Dept of CSE, ASET, AUUP

- Understanding Disk Storage
  - ✓ Partitioning Hard Disks
  - ✓ Viewing disk partitions
  - ✓ Creating a single-partition disk
  - ✓ Creating a multiple-partition disk

- Using Logical Volume Management Partitions
  - ✓ Checking an existing LVM
  - ✓ Creating LVM logical volumes
  - ✓ Growing LVM logical volumes

- Mounting Filesystems
  - ✓ Supported filesystems
  - ✓ Enabling swap areas
  - ✓ Disabling swap area
  - ✓ Using the fstab file to define mountable file systems
  - ✓ Using the mount command to mount file systems
  - ✓ Mounting a disk image in loopback
  - ✓ Using the umount command
  - ✓ Using the mkfs Command to Create a Filesystem

```
#| fdisk -c -u -l /dev/sdc
```

Disk /dev/sdc: 8021 MB, 8021606400 bytes

16 heads, 48 sectors/track, 20400 cylinders, total 15667200 sectors

Units = sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0xc3072e18

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1		2048	4196351	2097152	83	Linux
/dev/sdc2		4196352	5220351	512000	82	Linux swap / Solaris
/dev/sdc3		5220352	7317503	1048576	c	W95 FAT32 (LBA)
/dev/sdc4		7317504	15667199	4174848	5	Extended
/dev/sdc5		7319552	7729151	204800	83	Linux
/dev/sdc6		7731200	15667199	3968000	8e	Linux LVM

- Your first primary hard disk usually appears as /dev/sda.
- With RHEL and Fedora installations, there is usually at least one LVM partition, out of which other partitions can be assigned.
- So, the output of fdisk might be as simple as the following:

```
# fdisk -cul /dev/sda
Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ebb20

      Device Boot      Start        End      Blocks   Id  System
/dev/sda1  *        2048     411647      204800   83  Linux
/dev/sda2          411648  976773119    488180736   8e  Linux LVM
```

The general procedure:

1. Install the new hard drive or insert the new USB flash drive.
  2. Partition the new disk.
  3. Create the filesystems on the new disk.
  4. Mount the filesystems
- The following process takes you through partitioning a USB flash drive to be used for Linux that has only one partition

- For a USB flash drive, just plug it into an available USB port.
- Determine the device name for the hard disk: # tail -f /var/log/messages

```
# tail -f /var/log/messages
scsi 6:0:0:0: Direct-Access S31B1102 USB DISK 1100 PQ: 0 ANSI: 0 CCS
sd 6:0:0:0: Attached scsi generic sg2 type 0
sd 6:0:0:0: [sdc] 8342528 512-byte logical blocks: (4.27 GB/3.97 GiB)
sd 6:0:0:0: [sdc] Write Protect is off
sd 6:0:0:0: [sdc] Mode Sense: 43 00 00 00
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sdc: sdc1
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sd 6:0:0:0: [sdc1] Attached SCSI removable disk
```

- From the output, you can see that the USB flash drive was found and assigned to /dev/sdc etc.
- If the USB flash drive mounts automatically, unmount it: umount /dev/sdc1
- Use the fdisk command to create partitions on the new disk.  

```
# fdisk -c -u /dev/sdc
Command (m for help) :
```
- If you start with a new USB flash drive, it may have one partition that is entirely devoted to a Windows-compatible filesystem (such as VFAT).
- Use p to view all partitions and d to delete the partition

```
Command (m for help): p
```

```
...
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1		2048	15667199	7832576	c	W95 FAT32 (LBA)

```
Command (m for help): d
```

```
Selected partition 1
```

- To create a new partition, type the letter n. You are prompted for the type of partition
- Choose an extended (e) or primary partition (p). Type the letter p to choose primary.

- Type in the partition number. If you are creating the first partition (or for only one partition), type the number 1. You are prompted for the first sector to start the partition.
- Select the first available sector number (you can just press enter to choose it). You are prompted for the last sector.
- Enter the size of the partition. Because you are just creating one partition to consume the whole disk, choose the last available sector. To do that you can just press Enter to accept the default

- Double-check that the drive is partitioned the way you want by pressing p.

```
Command (m for help): p
```

```
...
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		2048	15667199	7832576	83	Linux

- To make changes to the partition table permanent, type w. This will write the changes, try to sync those changes with the Linux kernel, and quit fdisk.
- If fdisk cannot sync the partition table on the disk with the kernel, the most likely reason is that a partition from the disk is still mounted.

- Unmount the partition, and then try running the following command to sync the disk partition table with the kernel: `# partprobe /dev/sdc`
- If `partprobe` does not work, rebooting the computer will make sure the disk and kernel are in sync.
- Although the partitioning is done, the new partition is not yet ready to use. For that, you must create a filesystem on the new partition:

```
# mkfs -t ext4 /dev/sdc1
```

- To be able to use the new filesystem, you need to create a mount point and mount it to the partition.

```
# mkdir /mnt/test
# mount /dev/sdc1 /mnt/test
# df -h /mnt/test
Filesystem           Size   Used  Avail Use% Mounted on
/dev/sdc1            7.4G   17M   7.0G   1% /mnt/test
# mount | grep sdc1
/dev/sdc1 on /mnt/test type ext4 (rw)
```

- When you are done using the drive, you can unmount it with the umount command, after which you can safely remove the drive.

```
# umount /dev/sdc1
```

- Set up a USB flash drive to mount automatically every time the system boots.

Edit /etc/fstab and add a line describing what and where to mount. A line you might add: /dev/sdc1 /mnt/test ext4 defaults 0 1



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

- Understanding Group Accounts
  - ✓ Using group accounts
  - ✓ Creating group accounts
- Setting permissions with Access Control Lists
  - ✓ Setting ACLs with setfacl
  - ✓ Setting default ACLs
  - ✓ Enabling ACLs

- Adding directories for users to collaborate
  - ✓ Creating group collaboration directories (set GID bit)
  - ✓ Creating restricted deletion directories (sticky bit)

Name	Numeric value	Letter value
Set user ID bit	4	u+s
Set group ID bit	2	g+s
Sticky bit	1	o+t



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

## Linux for Devices

Module-4, Lecture-3

By: Dr. A K Yadav (9911375598)  
Dept of CSE, ASET, AUUP

- Understanding Disk Storage
  - ✓ Partitioning Hard Disks
  - ✓ Viewing disk partitions
  - ✓ Creating a single-partition disk
  - ✓ Creating a multiple-partition disk

- Using Logical Volume Management Partitions
  - ✓ Checking an existing LVM
  - ✓ Creating LVM logical volumes
  - ✓ Growing LVM logical volumes

- Mounting Filesystems
  - ✓ Supported filesystems
  - ✓ Enabling swap areas
  - ✓ Disabling swap area
  - ✓ Using the fstab file to define mountable file systems
  - ✓ Using the mount command to mount file systems
  - ✓ Mounting a disk image in loopback
  - ✓ Using the umount command
  - ✓ Using the mkfs Command to Create a Filesystem

```
#| fdisk -c -u -l /dev/sdc
```

Disk /dev/sdc: 8021 MB, 8021606400 bytes

16 heads, 48 sectors/track, 20400 cylinders, total 15667200 sectors

Units = sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0xc3072e18

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1		2048	4196351	2097152	83	Linux
/dev/sdc2		4196352	5220351	512000	82	Linux swap / Solaris
/dev/sdc3		5220352	7317503	1048576	c	W95 FAT32 (LBA)
/dev/sdc4		7317504	15667199	4174848	5	Extended
/dev/sdc5		7319552	7729151	204800	83	Linux
/dev/sdc6		7731200	15667199	3968000	8e	Linux LVM

- Your first primary hard disk usually appears as /dev/sda.
- With RHEL and Fedora installations, there is usually at least one LVM partition, out of which other partitions can be assigned.
- So, the output of fdisk might be as simple as the following:

```
# fdisk -cul /dev/sda
Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ebb20

      Device Boot   Start     End   Blocks Id System
/dev/sda1  *       2048  411647   204800  83  Linux
/dev/sda2          411648 976773119 488180736  8e  Linux LVM
```

The general procedure:

1. Install the new hard drive or insert the new USB flash drive.
  2. Partition the new disk.
  3. Create the filesystems on the new disk.
  4. Mount the filesystems
- The following process takes you through partitioning a USB flash drive to be used for Linux that has only one partition

- For a USB flash drive, just plug it into an available USB port.
- Determine the device name for the hard disk: # tail -f /var/log/messages

```
# tail -f /var/log/messages
scsi 6:0:0:0: Direct-Access S31B1102 USB DISK 1100 PQ: 0 ANSI: 0 CCS
sd 6:0:0:0: Attached scsi generic sg2 type 0
sd 6:0:0:0: [sdc] 8342528 512-byte logical blocks: (4.27 GB/3.97 GiB)
sd 6:0:0:0: [sdc] Write Protect is off
sd 6:0:0:0: [sdc] Mode Sense: 43 00 00 00
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sdc: sdc1
sd 6:0:0:0: [sdc] Assuming drive cache: write through
sd 6:0:0:0: [sdc1] Attached SCSI removable disk
```

- From the output, you can see that the USB flash drive was found and assigned to /dev/sdc etc.
- If the USB flash drive mounts automatically, unmount it: umount /dev/sdc1
- Use the fdisk command to create partitions on the new disk.  

```
# fdisk -c -u /dev/sdc
Command (m for help) :
```
- If you start with a new USB flash drive, it may have one partition that is entirely devoted to a Windows-compatible filesystem (such as VFAT).
- Use p to view all partitions and d to delete the partition

```
Command (m for help): p
```

```
...
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1		2048	15667199	7832576	c	W95 FAT32 (LBA)

```
Command (m for help): d
```

```
Selected partition 1
```

- To create a new partition, type the letter n. You are prompted for the type of partition
- Choose an extended (e) or primary partition (p). Type the letter p to choose primary.

- Type in the partition number. If you are creating the first partition (or for only one partition), type the number 1. You are prompted for the first sector to start the partition.
- Select the first available sector number (you can just press enter to choose it). You are prompted for the last sector.
- Enter the size of the partition. Because you are just creating one partition to consume the whole disk, choose the last available sector. To do that you can just press Enter to accept the default

- Double-check that the drive is partitioned the way you want by pressing p.

```
Command (m for help): p
```

```
...
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		2048	15667199	7832576	83	Linux

- To make changes to the partition table permanent, type w. This will write the changes, try to sync those changes with the Linux kernel, and quit fdisk.
- If fdisk cannot sync the partition table on the disk with the kernel, the most likely reason is that a partition from the disk is still mounted.

- Unmount the partition, and then try running the following command to sync the disk partition table with the kernel: `# partprobe /dev/sdc`
- If `partprobe` does not work, rebooting the computer will make sure the disk and kernel are in sync.
- Although the partitioning is done, the new partition is not yet ready to use. For that, you must create a filesystem on the new partition:

```
# mkfs -t ext4 /dev/sdc1
```

- To be able to use the new filesystem, you need to create a mount point and mount it to the partition.

```
# mkdir /mnt/test
# mount /dev/sdc1 /mnt/test
# df -h /mnt/test
Filesystem           Size   Used  Avail Use% Mounted on
/dev/sdc1            7.4G   17M   7.0G   1% /mnt/test
# mount | grep sdc1
/dev/sdc1 on /mnt/test type ext4 (rw)
```

- When you are done using the drive, you can unmount it with the umount command, after which you can safely remove the drive.

```
# umount /dev/sdc1
```

- Set up a USB flash drive to mount automatically every time the system boots.

Edit /etc/fstab and add a line describing what and where to mount. A line you might add: /dev/sdc1 /mnt/test ext4 defaults 0 1



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

# Understanding Server Administration

Department of Computer  
Science and Engineering

## Linux for Devices

Module-4, Lecture-4

By: Dr. A K Yadav (9911375598)  
Dept of CSE, ASET, AUUP

## Challenges as a server administrator:

- ✓ Remote access
- ✓ Diligent security
- ✓ Continuous monitoring

- Starting with Server Administration

- ✓ Step 1: Install the server
- ✓ Step 2: Configure the server
- ✓ Step 3: Start the server
- ✓ Step 4: Secure the server
- ✓ Step 5: Monitor the server

- Managing Remote Access with the Secure Shell Service

- ✓ Starting the openssh-server service
- ✓ Using SSH client tools
- ✓ Using key-based (passwordless) authentication

- Configuring System Logging
  - ✓ Enabling system logging with rsyslog
  - ✓ Watching logs with logwatch
- Checking System Resources with sar
- Checking System Space
  - ✓ Displaying system space with df
  - ✓ Checking disk usage with du

**TABLE 13.1: Commands to Determine sshd Status**

Distribution	Command to Determine sshd Status
RHEL	chkconfig --list sshd
Fedora	systemctl status sshd.service
Ubuntu	status ssh

**TABLE 13.2: Commands to Start sshd**

Distribution	Command to Start sshd
RHEL	service sshd start
Fedora	systemctl start sshd.service
Ubuntu	service ssh start

**TABLE 13.3: Commands to Start sshd at Boot**

Distribution	Command to Start sshd at Boot
RHEL	chkconfig sshd on
Fedora	systemctl enable sshd.service
Ubuntu	update-rc.d ssh defaults

- The following is an example of remotely logging in to johndoe's account on 10.140.67.23:

```
$ ssh johndoe@10.140.67.23
The authenticity of host '10.140.67.23 (10.140.67.23)' can't be established.
RSA key fingerprint is a4:28:03:85:89:6d:08:fa:99:15:ed:fb:b0:67:55:89.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.140.67.23' (RSA) to the list of known hosts.
johndoe@10.140.67.23's password: *****
```



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

## Linux for Devices

Module-4, Lecture-5

By: Dr. A K Yadav (9911375598)

Dept of CSE, ASET, AUUP

- Type of Networks
  - ✓ Wired network
  - ✓ Wireless network
- The general approach to configuring networking in three types of Linux systems
  - ✓ Desktop/laptop networking
  - ✓ Server networking
  - ✓ Enterprise networking

Group	Title	Description
ISOC	Internet Society	Professional membership organization of Internet experts that oversees boards and task forces dealing with network policy issues <b><a href="http://isoc.org">isoc.org</a></b>
IESG	The Internet Engineering Steering Group	Responsible for technical management of IETF activities and the Internet standards process <b><a href="http://ietf.org/iesg.html">ietf.org/iesg.html</a></b>
IANA	Internet Assigned Numbers Authority	Responsible for Internet Protocol (IP) addresses <b><a href="http://iana.org">iana.org</a></b>
IAB	Internet Architecture Board	Defines the overall architecture of the Internet, providing guidance and broad direction to the IETF <b><a href="http://iab.org">iab.org</a></b>
IETF	Internet Engineering Task Force	Protocol engineering and development arm of the Internet <b><a href="http://ietf.org">ietf.org</a></b>

---

**TABLE 34-1** TCP/IP Protocol Development Groups

- The three basic protocols are the Transmission Control Protocol (TCP), which handles receiving and sending out communications;
- The Internet Protocol (IP), which handles the actual transmissions; and
- The User Datagram Protocol (UDP), which also handles receiving and sending packets.
- The IP protocol, which is the base protocol that all others use, handles the actual transmissions, the packets of data with sender and receiver information in each.

- The TCP protocol is designed to work with cohesive messages or data.
- This protocol checks received packets and sorts them into their designated order, forming the original message.
- For data sent out, the TCP protocol breaks the data into separate packets, designating their order.
- The UDP protocol, meant to work on a much more raw level, also breaks down data into packets but does not check their order.

- The TCP/IP protocol is designed to provide stable and reliable connections that ensure that all data is received and reorganized into its original order
- UDP, on the other hand, is designed to simply send as much data as possible, with no guarantee that packets will all be received or placed in the proper order.
- UDP is often used for transmitting very large amounts of data of the type that can survive the loss of a few packets—for example, temporary images, video, and banners displayed on the Internet.

- Other protocols provide various network and user services. The Domain Name Service (DNS) provides address resolution.
- The File Transfer Protocol (FTP) provides file transmission, and the Network File System (NFS) provides access to remote file systems.
- Table 34-2 lists the different protocols in the TCP/IP protocol suite.
- These protocols make use of either the TCP or UDP protocol to send and receive packets, which, in turn, uses the IP protocol for actually transmitting the packets.

<b>Transport</b>	<b>Description</b>
TCP	Transmission Control Protocol; places systems in direct communication
UDP	User Datagram Protocol
IP	Internet Protocol; transmits data
ICMP	Internet Control Message Protocol; status messages for IP
<b>Routing</b>	<b>Description</b>
RIP	Routing Information Protocol; determines routing
OSPF	Open Shortest Path First; determines routing
<b>Network Address</b>	<b>Description</b>
ARP	Address Resolution Protocol; determines unique IP address of systems
DNS	Domain Name Service; translates hostnames into IP addresses
RARP	Reverse Address Resolution Protocol; determines addresses of systems

User Service	Description
FTP	File Transfer Protocol; transmits files from one system to another using TCP
TFTP	Trivial File Transfer Protocol; transfers files using UDP
Telnet	Remote login to another system on the network
SMTP	Simple Mail Transfer Protocol; transfers email between systems
RPC	Remote Procedure Call; allows programs on remote systems to communicate
Gateway	Description
EGP	Exterior Gateway Protocol; provides routing for external networks
GGP	Gateway-to-Gateway Protocol; provides routing between Internet gateways
IGP	Interior Gateway Protocol; provides routing for internal networks

Network Service	Description
NFS	Network File System; allows mounting of file systems on remote machines
NIS	Network Information Service; maintains user accounts across a network
BOOTP	Boot Protocol; starts system using boot information on server for network
SNMP	Simple Network Management Protocol; provides status messages on TCP/IP configuration
DHCP	Dynamic Host Configuration Protocol; automatically provides network configuration information to host systems

---

**TABLE 34-2** TCP/IP Protocol Suite

- Configuring Networks on GNOME and KDE
- Zero Configuration Networking (zeroconf): Avahi and Link Local Addressing
- IPv4 and IPv6
- TCP/IP Network Addresses
- IPv4 Network Addresses
- Class-Based IP Addressing
- Netmask
- Classless Interdomain Routing (CIDR)

---

**FIGURE 34-2**  
CIDR addressing

CIDR Addressing				
IP Address 192.168.4.6/22				
	Network		Host	
binary	11000000	10101000	000001	00 0000110
numeric	192	168	4	6
Netmask	255.255.252.0	22 bits		
binary	11111111	11111111	111111	00 0000000
numeric	255	255	252	000

- IPv4 CIDR Addressing
- IPv6 CIDR Addressing

<b>Short Form</b>	<b>Full Form</b>	<b>Maximum Number of Hosts</b>
/8	/255.0.0.0	16,777,215 (A class)
/16	/255.255.0.0	65,534 (B class)
/17	/255.255.128.0	32,767
/18	/255.255.192.0	16,383
/19	/255.255.224.0	8191
/20	/255.255.240.0	4095
/21	/255.255.248.0	2047
/22	/255.255.252.0	1023
/23	/255.255.254.0	511
/24	/255.255.255.0	255 (C class)
/25	/255.255.255.128	127
/26	/255.255.255.192	63
/27	/255.255.255.224	31
/28	/255.255.255.240	15
/29	/255.255.255.248	7
/30	/255.255.255.252	3

**TABLE 34-3** CIDR IPv4 Network Masks

<b>Subnetwork</b>	<b>CIDR Address</b>	<b>Binary Mask</b>
First subnet network address	.0/25	00000000
Second subnet network address	.128/25	10000000
First subnet broadcast address	.127/25	01111111
Second subnet broadcast address	.255/25	11111111
First address in first subnet	.1/25	00000001
First address in second subnet	.129/25	10000001
Last address in first subnet	.126/25	01111110
Last address in second subnet	.254/25	11111110



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

# Starting and Stopping Services

Department of Computer  
Science and Engineering

## Linux for Devices

Module-4, Lecture-6

By: Dr. A K Yadav (9911375598)  
Dept of CSE, ASET, AUUP

- Understanding the Linux init Daemon
  - Understanding the classic init daemons
  - Understanding the Upstart init daemon
  - Understanding systemd init
- Auditing Services
  - Auditing the classic SysVinit daemon
  - Auditing the Upstart init daemon
  - Auditing the systemd init

- Stopping and Starting Services
  - Stopping and starting the classic SysVinit daemon
  - Stopping and starting the Upstart init daemon
  - Stopping and starting the systemd daemon
- Configuring Persistent Services
  - Configuring the classic SysVinit daemon persistent services
  - Configuring Upstart init daemon persistent services
  - Configuring systemd init persistent services

- Configuring a Default runlevel or target unit
  - Configuring the classic SysVinit daemon default runlevel
  - Configuring the Upstart init daemon default runlevel
  - Configuring the systemd init default target unit
- Adding New or Customized Services
  - Adding new services to classic SysVinit daemon
  - Adding new services to the Upstart init daemon
  - Adding new services to systemd init

# Stopping and starting the classic SysVinit daemon

- The primary command for stopping and starting SysVinit services is the service command.
- With the service command, the name of the service you are wishing to control comes second in the command line.
- The last option is what you want to do to the service, stop, start, restart, and so on.
- The following example shows how to stop the cups daemon.

- Notice that an OK is given, which lets you know that cupsd has been successfully stopped.

```
# service cups status
cupsd (pid 5857) is running...
#
# service cups stop
Stopping cups: [ OK ]
#
# service cups status
cupsd is stopped
```

- To start a service, you simply add a start option instead of a stop option on the end of the service command as follows:

```
# service cups start
Starting cups: [ OK ]
#
# service cups status
cupsd (pid 6860) is running...
```

- To restart a SysVinit service, the restart option is used. This option will stop the service and then immediately start it again.

```
# service cups restart
Stopping cups:          [  OK  ]
Starting cups:          [  OK  ]
#
# service cups status
cupsd (pid 7955) is running...
```

- When a service is already stopped, a restart will generate a FAILED status on the attempt to stop it. However, as shown in the example that follows, the service will be successfully started when a restart is attempted.

```
# service cups stop
Stopping cups: [ OK ]
#
# service cups restart
Stopping cups: [FAILED]
Starting cups: [ OK ]
#
# service cups status
cupsd (pid 8236) is running...
```

- Reloading a service is different from restarting a service. When you reload a service, the service itself is not stopped. Only the service's configuration files are loaded again. The following example shows how to reload the cups daemon.

```
# service cups status
cupsd (pid 8236) is running...
#
# service cups reload
Reloading cups: [ OK ]
#
# service cups status
cupsd (pid 8236) is running...
```

- If a SysVinit service is stopped when you attempt to reload it, you will get a FAILED status. This is shown in the following example:

```
# service cups status
cupsd is stopped
#
# service cups reload
Reloading cups: [FAILED]
```

# Stopping and starting the Upstart init daemon

- The primary command for stopping and starting Upstart init services is the `initctl` command. The options are very similar to SysVinit's service command:
- Stopping a service with Upstart init — In the following example, the status of the cups daemon is checked and then stopped using the `initctl stop cups.service` command.

```
# initctl status cups
cups start/running, process 2390
#
# initctl stop cups
cups stop/waiting
#
# initctl status cups
cups stop/waiting
```

- Starting a service with Upstart init — In the following example, the cups daemon is stopped using the initctl stop cups.service command.

```
# initctl start cups
cups start/running, process 2408
#
# initctl status cups
cups start/running, process 2408
```

- Restarting a service with Upstart init — Restarting a service with Upstart init will stop and then start the service. However, the configuration file will not be reloaded.

```
# initctl restart cups
cups start/running, process 2490
#
# initctl status cups
cups start/running, process 2490
#
```

- Reloading a service with Upstart init — Reloading will not stop and start the service. It only loads the configuration file again. This is the option to use when you have made changes to the configuration file.

```
# initctl reload cups
#
# initctl status cups
cups start/running, process 2490
```

- Notice that the process ID (PID) is still 2490, which is the same as it was in the example for restarting the cups daemon because the process was not stopped and started in the reload process.

# Stopping and starting the systemd daemon

- For the systemd daemon, the systemctl command will work for stopping, starting, reloading, and restarting. The options to the systemctl command should look familiar.
- Stopping a service with system - In the example that follows, the status of the cups daemon is checked and then stopped using the systemctl stop cups.service command.
- Notice that when the status is taken, after stopping the cups daemon, the service is inactive (dead) but still considered enabled. This means that the cups daemon will still be started upon server boot.

```
# systemctl status cups.service
cups.service - CUPS Printing Service
  Loaded: loaded (/lib/systemd/system/cups.service; enabled)
  Active: active (running) since Mon, 30 Apr 2015 12:36:3...
    Main PID: 1315 (cupsd)
      CGrouп: name=systemd:/system/cups.service
              └─ 1315 /usr/sbin/cupsd -f

#
# systemctl stop cups.service
#
# systemctl status cups.service
cups.service - CUPS Printing Service
  Loaded: loaded (/lib/systemd/system/cups.service; enabled)
  Active: inactive (dead) since Tue, 01 May 2015 04:43:4...
    Process: 1315 ExecStart=/usr/sbin/cupsd -f
   (code=exited, status=0/SUCCESS)
      CGrouп: name=systemd:/system/cups.service
```

- Starting a service with system - Starting the cups daemon is just as easy as stopping it. The example that follows demonstrates this ease.

```
# systemctl start cups.service
#
# systemctl status cups.service
cups.service - CUPS Printing Service
   Loaded: loaded (/lib/systemd/system/cups.service; enabled)
     Active: active (running) since Tue, 01 May 2015 04:43:5...
       Main PID: 17003 (cupsd)
          CGrou...: name=systemd:/system/cups.service
                        └ 17003 /usr/sbin/cupsd -f
```

- After the cups daemon is started, using systemctl with the status option shows the service is active (running). Also, its process ID (PID) number, 17003, is shown

- Restarting a service with system - Restarting a service means that a service is stopped and then started again. If the service was not currently running, restarting it will simply start the service.

```
# systemctl restart cups.service
#
# systemctl status cups.service
cups.service - CUPS Printing Service
   Loaded: loaded (/lib/systemd/system/cups.service; enabled)
   Active: active (running) since Tue, 01 May 2015 04:45:2...
     Main PID: 17015 (cupsd)
        CGroup: name=systemd:/system/cups.service
                  └─ 17015 /usr/sbin/cupsd -f
```

- You can also perform a conditional restart of a service using `systemctl`. A conditional restart only restarts a service if it is currently running. Any service in an inactive state will not be started.
- Notice in the example that the cups daemon was in an inactive state. When the conditional restart was issued, no error messages were generated! The cups daemon was not started because conditional restarts will affect active services.
- Thus, it is always a good practice to check the status of a service, after stopping, starting, conditionally restarting, and so on.

```
# systemctl status cups.service
cups.service - CUPS Printing Service
 Loaded: loaded (/lib/systemd/system/cups.service; enabled)
 Active: inactive (dead) since Tue, 01 May 2015 06:03:32...
Process: 17108 ExecStart=/usr/sbin/cupsd -f
(code=exited, status=0/SUCCESS)
 CGroup: name=systemd:/system/cups.service

#
# systemctl condrestart cups.service
#
# systemctl status cups.service
cups.service - CUPS Printing Service
 Loaded: loaded (/lib/systemd/system/cups.service; enabled)
 Active: inactive (dead) since Tue, 01 May 2015 06:03:32...
Process: 17108 ExecStart=/usr/sbin/cupsd -f
(code=exited, status=0/SUCCESS)
 CGroup: name=systemd:/system/cups.service
```

- Reloading a service with system - Reloading a service is different from restarting a service. When you reload a service, the service itself is not stopped. Only the service's configuration files are loaded again
- Doing a reload of a service, instead of a restart, will prevent any pending service operations from being aborted.
- A reload is a better method for a busy Linux server.
- Now that you know how to stop and start services for troubleshooting and emergency purposes, you can learn how to enable and disable services

```
# systemctl reload cups.service
Failed to issue method call: Job type reload is
not applicable for unit cups.service.

#
# systemctl reload sshd.service
#
# systemctl status sshd.service
sshd.service - OpenSSH server daemon
   Loaded: loaded (/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Mon, 30 Apr 2015 12:35:2...
     Process: 17009 ExecReload=/bin/kill -HUP $MAINPID
    (code=exited, status=0/SUCCESS)
   Main PID: 786 (sshd)
      CGroup: name=systemd:/system/sshd.service
              └─ 786 /usr/sbin/sshd -D
```



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

# Configuring an NFS File Server

Department of Computer  
Science and Engineering

## Linux for Devices

Module-4, Lecture-7

By: Dr. A K Yadav (9911375598)  
Dept of CSE, ASET, AUUP

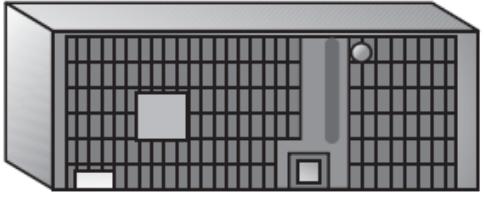
- Configuring an NFS File Server

- ✓ Installing an NFS Server
- ✓ Starting the NFS service
- ✓ Sharing NFS Filesystems
- ✓ Configuring the /etc/exports file
- ✓ Exporting the shared filesystems
- ✓ Securing Your NFS Server
- ✓ Opening your firewall for NFS
- ✓ Allowing NFS access in TCP wrappers
- ✓ Configuring SELinux for your NFS server
- ✓ Using NFS Filesystems
- ✓ Mounting an NFS filesystem at boot time
- ✓ Using autofs to mount NFS filesystems on demand
- ✓ Unmounting NFS filesystems

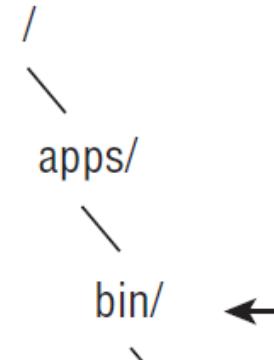
An NFS file server provides an easy way to share large amounts of data among the users and computers in an organization. An administrator of a Linux system that is configured to share its filesystems using NFS has to perform the following tasks to set up NFS:

1. Set up the network
2. Start the NFS service.
3. Choose what to share from the server
4. Set up security on the server
5. Mount the filesystem on the client

**/etc/exports** file:  
/apps/bin pins(rw) maple(rw) spruce(rw)



**oak**



file1 file2 file3

**# mount -t nfs oak:/apps/bin /oak/apps**



**pine**



# Installing an NFS Server

- `# yum install nfs-utils`
- To find out more about the nfs-utils package, you can run the following commands to see information about the package, configuration files, and commands, respectively:
  - `# rpm -qi nfs-utils`
  - `# rpm -qc nfs-utils`
  - `# rpm -ql nfs-utils | grep bin`

# Starting the NFS service

- The basic NFS service in Fedora is called nfs-server.

```
# systemctl start nfs-server.service
# systemctl enable nfs-server.service
# systemctl status nfs-server.service
systemctl status nfs-server.service
nfs-server.service - NFS Server
   Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled)
   Active: active (running) since Sat, 2 Jun 2012 08:40:25 -0400;
             1h 28min ago
     Main PID: 7767 (rpc.mountd)
        CGroup: name=systemd:/system/nfs-server.service
                  └─ 7767 /usr/sbin/rpc.mountd
```

- In Red Hat Enterprise Linux 6, you need the service and chkconfig commands to check, start, and enable the NFS service (nfs).
- The following commands show the nfs service not running currently and disabled:

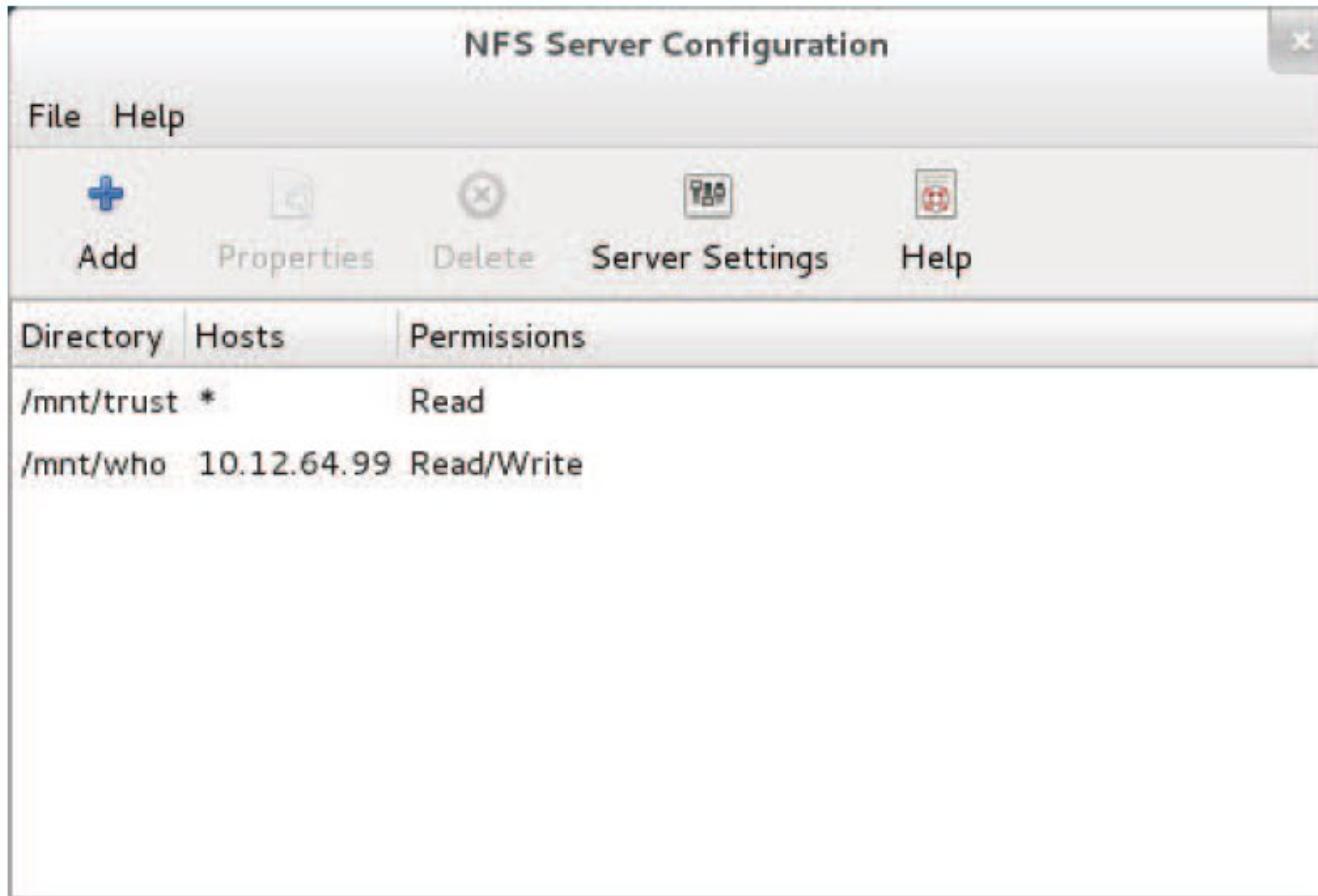
```
# service nfs status
rpc.svcgssd is stopped
rpc.mountd is stopped
nfsd is stopped
# chkconfig --list nfs
nfs 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

- As mentioned earlier, the rpcbind service must be running for NFS to work. So, you could use the following commands to start and permanently enable both the rpcbind and nfs services

```
# service rpcbind start
Starting rpcbind: [ OK ]
# service nfs start
Starting NFS services: [ OK ]
Starting NFS quotas: [ OK ]
Starting NFS daemon: [ OK ]
Starting NFS mountd: [ OK ]
# chkconfig nfs on
```

# Sharing NFS Filesystems

The NFS Server Configuration window (`system-config-nfs`) provides a graphical way of configuring NFS services.



- To share an NFS filesystem from your Linux system, you need to export it from the server system.
- Exporting is done in Linux by adding entries into the /etc/exports file.
- Each entry identifies a directory in your local filesystem that you want to share with other computers.
- The entry also identifies the other computers that can share the resource (or opens it to all computers) and includes other options that reflect permissions associated with the directory.

- Remember that when you share a directory, you are sharing all files and subdirectories below that directory as well (by default). So, you need to be sure that you want to share everything in that directory structure.
- There are still ways to restrict access within that directory structure

- To make a directory from your Linux system available to other systems, you need to export that directory. Exporting is done on a permanent basis by adding information about an exported directory to the /etc/exports file. The format of the /etc/exports file is:

*Directory    Host (Options...)    Host (Options...)    # Comments*

- where Directory is the name of the directory that you want to share and Host indicates the client computer to which the sharing of this directory is restricted.
- Options can include a variety of options to define the security measures attached to the shared directory for the host.
- Comments are any optional comments you want to add (following the # sign).

- As root user, you can use any text editor to configure /etc/exports to modify shared directory entries or add new ones. Here's an example of an /etc/exports file:

```
/cal      *.linuxtoys.net(rw)          # Company events
/pub      *(ro,insecure,all_squash)    # Public dir
/home    maple(rw,root_squash)  spruce(rw,root_squash)
```

# Exporting the shared filesystems

Here's an example of the `exportfs` command:

```
# /usr/sbin/exportfs -a -r -v
exporting maple:/pub
exporting spruce:/pub
exporting maple:/home
exporting spruce:/home
exporting *:/mnt/win
```

- The `-a` option indicates that all directories listed in `/etc/exports` should be exported.
- The `-r` resyncs all exports with the current `/etc/exports` file.
- The `-v` option says to print verbose output. In this example, the `/pub` and `/home` directories from the local server are immediately available for mounting by those client computers that are named (maple and spruce). The `/mnt/win` directory is available to all client computers.

Some of issues included:

- Remote root users
- Unencrypted communications
- User mapping
- Filesystem structure exposed

# Opening your firewall for NFS

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 111 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 111 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 2049 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 2049 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 20048 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 20048 -j ACCEPT
```

```
RQUOTAD_PORT=49001
LOCKD_TCPPORT=49002
LOCKD_UDPPORT=49003
MOUNTD_PORT=49004
STATD_PORT=49005
STATD_OUTGOING_PORT=49006
```

tcp	0	0	0.0.0.0:49001	0.0.0.0:*	LISTEN	4682/rpc.rquotad
tcp	0	0	0.0.0.0:49002	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:49004	0.0.0.0:*	LISTEN	4698/rpc.mountd
tcp	0	0	:::49002	:::*	LISTEN	-
tcp	0	0	:::49004	:::*	LISTEN	4698/rpc.mountd
udp	0	0	0.0.0.0:49001	0.0.0.0:*		4682/rpc.rquotad
udp	0	0	0.0.0.0:49003	0.0.0.0:*		-
udp	0	0	0.0.0.0:49004	0.0.0.0:*		4698/rpc.mountd
udp	0	0	:::49003	:::*		-
udp	0	0	:::49004	:::*		4698/rpc.mountd



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

## Linux for Devices

Module-4, Lecture-8

By: Dr. A K Yadav (9911375598)  
Dept of CSE, ASET, AUUP

- Parallel Virtual File System (PVFS)
- Coda
- Red Hat Global File System (GFS and GFS 2)
  - GFS 2 Packages (Fedora Core 6 and On)
  - GFS 2 Service Scripts
  - Implementing a GFS 2 File System
  - GFS Tools

- GFS File System Operations
- GFS 1



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

# Linux for Devices

## Module-4, Lecture-9

By: Dr. A K Yadav (9911375598)  
Dept of CSE, ASET, AUUP

- Parallel Virtual File System (PVFS)
- Coda
- Red Hat Global File System (GFS and GFS 2)
  - GFS 2 Packages (Fedora Core 6 and On)
  - GFS 2 Service Scripts
  - Implementing a GFS 2 File System
  - GFS Tools

- GFS File System Operations
- GFS 1



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering



**AMITY**  
UNIVERSITY

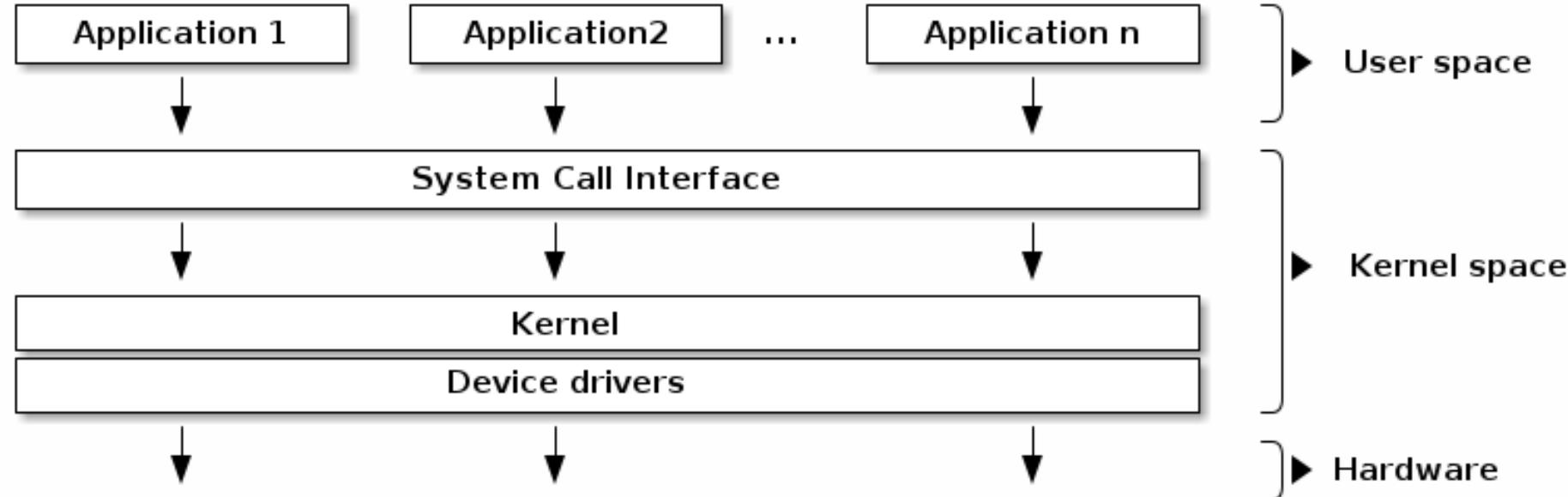
Department of Computer  
Science and Engineering

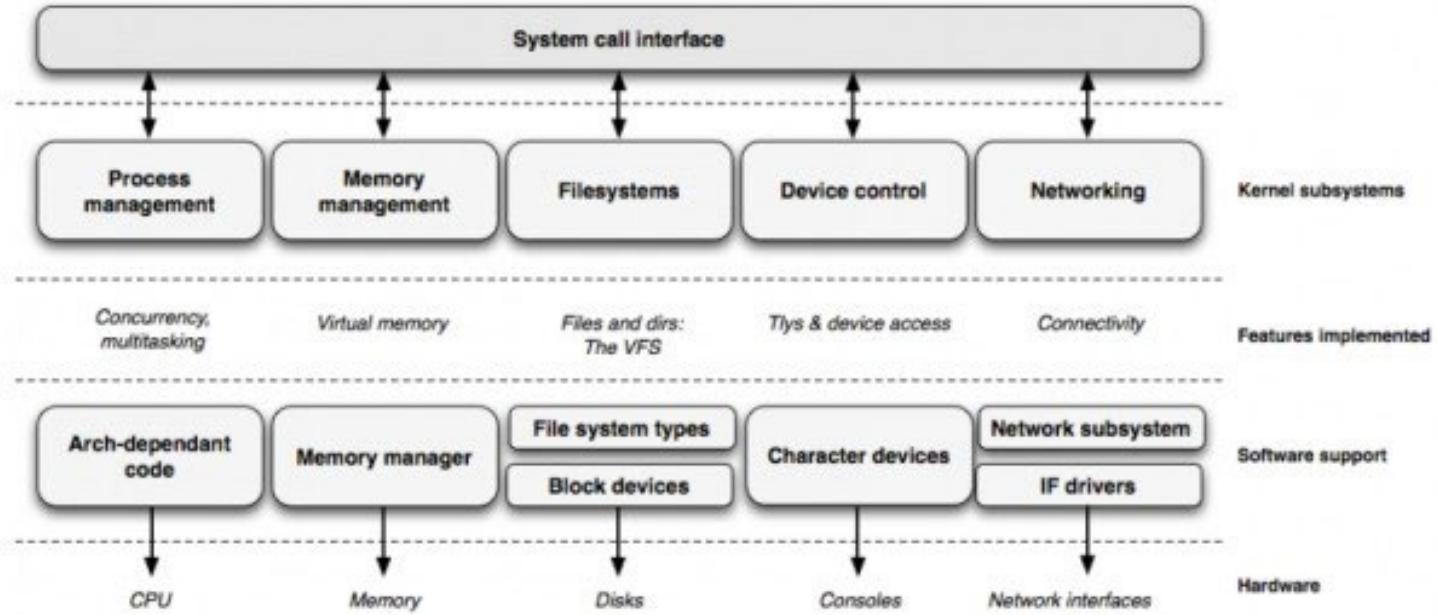
# Linux for Devices

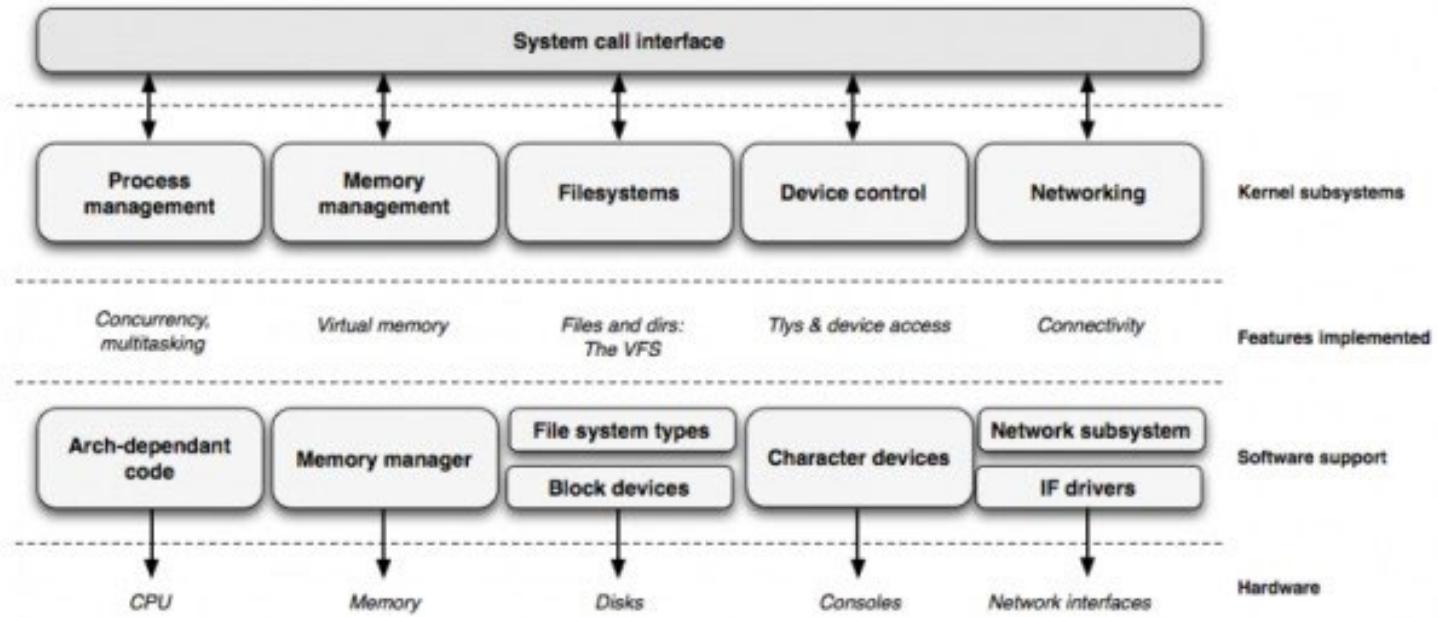
## Module-5

By: Dr. A K Yadav (9911375598)  
Dept of CSE, ASET, AUUP

# About Linux Device Driver







- In Linux system, several concurrent process perform different tasks
- Each process asks for the system resource, be it memory, networking or any other
- The kernel is the big chunk of executable code in charge of handling such request like:

-Process Management

-Memory Management

-File System

-Device Control

-Networking

- Linux device drivers are distinct “black boxes” that make a particular piece of hardware respond to a well-defined internal programming interface
- They completely hide the details of how the device works
- What is the need of device driver?

- User activities are performed by means of a set of standardized calls that are independent of the specific driver; mapping those calls to device-specific operations that act on real hardware is then the role of the device driver.

- Memory consists of RAM cells, whose contents can be accessed (i.e. read and written to) at extremely high speeds but are retained only temporarily (i.e. while in use or atmost while the power supply remains on)
- System memory in Linux can be divided into two distinct regions: Kernel space and User space

- Kernel space is where the core of the operating system runs and provides its services
- User space is that the set of memory locations in which user processes (i.e. everything other than the kernel) run. A process is an executing instance of a program. One of the roles of the kernel is to manage individual user processes within this space and to prevent them from interfering with each other.
- Device driver run in which space?

- Kernel space can be accessed by user processes only through the use of system calls

- Character Device: Accessed as stream of bytes
- Block Device: Accessed by the file system (/dev)
- Network Device

# Device Control

- Almost every system operation eventually maps to a physical device. With the exception of the processor, memory, and a very few other entities, any and all device control operations are performed by code that is specific to the device being addressed. That code is called device driver.
- The kernel must have embedded in it a device driver for every peripheral present on a system, from the hard drive to the keyboard and the tape drive.

# Linux Kernel Modules

- A piece of code that can be added to the base kernel is called a module
- As we have device drivers in Windows operating system, we have modules in Linux
- In short, device drivers are called modules in Linux. Both terms can be used interchangeably.

# Base Kernel

- Kernel code which is compiled and loaded along with basic device drivers as a single entity is called base kernel
- Note that all parts of base kernel will be loaded all the time
- Once you have working base kernel, it is good to leave it untouched as long as possible

# User Space and Kernel Space

- Kernel runs in kernel space and normal programs run in user space
- Normal programs in user space can't mess with memory (and other resources) owned by other programs or by the Linux kernel. This limits their ability to do corrupt things like crashing the machine
- The kernel is the core of the operating system. It normally has full access to the all memory and machine hardware (and everything else on the machine). To keep the machine as stable as possible, you normally want only the most trusted, well tested code to run in kernel mode/kernel space

# Kernel Modules

- Static modules
  - These are compiled as part of the kernel and are available at anytime
  - These make the kernel larger and has the disadvantage of requiring us to rebuild and reboot the kernel every time we want new functionality
- Dynamic Modules
  - These modules are the pieces of code that can be loaded and unloaded into the kernel upon demand
  - These are also called Loadable Kernel Modules (LKM)
  - They extend the functionality of the kernel without the need to reboot the system
  - The advantage that it uses the memory more efficiently than the statically linked drivers

# How exactly the dynamic module is loaded?

- Step 1: Assume you have written a dynamic module xyz. You will also write an application which tries to access xyz functionalities
- Step 2: Once Linux is up and running on your PC or device, you will dynamically insmod your xyz module using “insmod” command to RAM
- Step 3: Your application in userspace will try to access functionalities of your hardware through xyz module by calling Linux provided system calls. Normally they are called ioctl calls
- Once the required hardware functionality is retrieved by the user application, when xyz module is no more needed, we can request kernel to remove xyz module using rmmod command

# Disadvantages of building your module as static

- Assume you have written a static module “xyz” for a device driver. When you compile this as static module along with the kernel, you will be adding extra size to the kernel image permanently
- Whenever you modify your “xyz” device driver, you need to recompile your entire kernel to build it
- Also machine need to be rebooted for the changes to take effect
- Especially while the device driver is not stable, when you build it as static module, you will face issues which takes more time to debug
- Take a scenario when there is a memory overflow bug in your “xyz” static module. Once it is statically compiled and the kernel is loaded, at some point of time, the kernel may crash
- In order to debug it people may take hours or days to fix it. Also, your kernel which includes this “xyz” part of it will be useless till you fix the crash issue
- This doesn’t happen when you build your “xyz” as dynamic module

# Advantages of building your module as dynamic/LKM

- Assume you built your “xyz” module as dynamic
- With this, you don’t have to rebuild your kernel as often. It can be compiled separately
- It saves your time and spares you the possibility of introducing an error in rebuilding and reinstalling the base kernel
- It can be loaded onto kernel at run time without having the machine to reboot
- It saves you memory because you load them when you are actually using them
- It can be unloaded anytime and hence no permanent affect on kernel size since these are compiled and built separately from kernel, they are much faster to maintain and debug
- Since these are compiled and built separately from kernel, they are much faster to maintain and debug
- Each dynamic module is made up of object code (not linked into a complete executable) that can be dynamically linked to the running kernel by the insmod program and can be unlinked by the rmmod program

# Implementing & Running Driver Module

- Setting up environment: Install necessary packages required for compiling a kernel module
  - Switch on to the super user mode
  - Zypper install kernel-devel kernel-headers dkms make bzip2
    - kernel-devel: are files that are required for compiling kernel code that will run as part of the kernel, such as kernel modules
    - kernel-headers: contains files that describe the system environment and are used for compiling normal programs that run in userspace
    - dkms: Dynamic kernel module support package
    - make: make the tool necessary for kernel code build system
    - bzip2: zipping the tool
  - reboot the system

# Program

```
. #include<linux/init.h>
.
. #include<linux/module.h>
.
. MODULE_LICENSE("Dual BSD/GPL");
.
. static int hello_init(void)
. {
.     printk(KERN_ALERT "Hello World\n");
.     return 0;
.
. }
.
. static void hello_exit(void)
. {
.     printk(KERN_ALERT "Goodbye");
.
. }
.
. module_init(hello_init);
. module_exit(hello_exit);
```

# Kernel headers used in the module

- `#include<linux/init.h>`
- This header contains the definition of the functions used in this module
- `module_init()` and `module_exit()` are defined in this file
- `module_init()` is a macro which defines the function to be called at module insertion time or at system boot time
- `module_exit()` is a macro which defines the function to be called at module removal time. This function is called cleanup function
- Kernel modules must always contain these two functions `int_module` and `cleanup_module`

# Kernel headers used in module

- `#include<linux/module.h>`
- This header is included to add support for dynamic loading of module into the kernel
- It also contains the definitions of the special macros used in this module
- Special macros like `MODULE_LICENSE` is defined in this header
  - `MODULE_LICENSE`: This macro is used to tell the kernel that this module bears a free license, without such a declaration, the kernel complains when the module is loaded
- `printk`: This function is similar to `printf` in standard C. This is implemented to make kernel have its own `printf` function instead of having dependency on C library
  - When using `printk()` when `syslogd` and `klogd` daemons are running, the output of `printk()` will be appended to `/var/log/messages`. `Printk()` if called with low priority will only be appended to `/var/log/messages`. To ensure that it prints to the console use `KERN_ALERT` priority.
- `KERN_ALERT`: This string is the priority of message. `KERN_ALERT` is the highest priority set to specify kernel to print the message on console.

# Hello World Kernel module functions

- hello\_init and hello\_exit
- hello\_init()

-This particular function is called when the module is loaded

- hello\_exit()

-This particular function is called when module removed

# Makefile Code

- obj-m := hello.o
- KDIR := /lib/modules/`uname -a`/build
- PWD := \$(shell pwd)
- default:
- \$(MAKE) -C \$(KDIR) SUBDIRS=\$(PWD)

# Makefile

- Obj-m:=hello.o
  - This assignment states that there is one module to be built from object file hello.o
- The kernel build system in Linux kernel is designed in such a way to handle the rest of the parameters
  - KDIR := /lib/modules/`uname -a`/build Kernel build directory path
  - PWD := \$(shell pwd)
    - . Current directory path
- Default:
  - \$(MAKE) -C \$(KDIR) SUBDIRS=\$(PWD)
    - . This is the main make command which switches to your kernel build directory and invokes the make command to compile helloworld kernel module present in the sub directory
    - . -C option tells make command to switch to \$(KDIR) which is kernel directory

# Loading Kernel Modules

- Two kernel utilities to load the modules
  - insmod
  - modprobe
- insmod
  - Is a kernel utility that installs loadable kernel modules into the kernel
  - It actually loads the module code and data into the kernel land, links any unresolved symbols in the module to the symbol table of the kernel
  - insmod accepts a number of command line options and it can assign a value to parameters in a module before linking it to the current
  - Note that if a module is correctly designed, it can be configured at load time by passing arguments to insmod
  - Syntax: `insmod [filename] [module-options...]`
- For example: `insmod hello.ko`
- `insmod /path/to/my_module.ko param1=0x330`
- In order to know if your insmod command has caused any errors, try using “dmesg” command to see the output
- Note: insmod only accepts one file at a time

# Modprobe

- It's a linux utility which offers more features than basic insmod
- Advantages include
  - It has the ability to decide which modules to load
  - `/lib/modules/$(uname -r)`
  - It's aware of module dependencies
  - It supports resolution of recursive module dependencies
  - e.g module x dependent on module y which is dependent on module z
- Syntax:
  - `modprobe module_name`
  - `modprobe hello`
  - Note: no need to use .ko extension
  - If syntax is not working then you need to run depmod to re-create the module dependency list. Run:
    - `sudo depmod`

# Difference between insmod and modprobe

- While loading a module, insmod fails with “Unresolved symbol” error when an unresolved symbol is present in the loading module
- Modprobe looks at the module to be loaded to see whether it references any symbols that are not currently defined in the kernel. If any such references are found, modprobe looks for other modules in the current module search path that define the relevant symbols
- When modprobe finds these modules which are needed by the module being loaded, it loads them into the kernel as well
- In short, while loading a module into the kernel, if that module has any dependency on another module, modprobe finds those modules from its search path, it loads them into the kernel
- Note modprobe is aware of default location of module knows how to figure out the dependencies and load the modules in the right order
- We need to have root permissions for executing both insmod and modprobe commands
- When a module is loaded, any symbol exported by the module becomes part of kernel symbol table

# Kernel symbol table

- The kernel symbol table contains the addresses of global kernel items, functions and variables that are needed to implement modularized drivers

# Listing modules currently loaded in the kernel

- We use a linux utility called lsmod
- Lsmod command lists the modules currently loaded in the kernel. Lsmod works by reading the /proc/modules virtual file
- Information on currently loaded modules can also be found in the sysfs virtual file system under /sys/modules
- Syntax:
- Lsmod

# Unloading kernel module

- Two kernel utilities to unload the modules are:

- rmmod
- modprobe

rmmod

- Rmmod is a linux utility which unloads the loadable module
- Module should be already loaded to execute rmmod command
- Module fails if the kernel believes that the module is still in use or if the kernel has been configured to disallow module removal
- Syntax
  - rmmod [module\_name]
  - e.g rmmod hello
  - Note: we need to be a superuser for executing this command

Modprobe

- Syntax
  - modprobe -r [module\_name]
  - e.g modprobe -r hello

# Important question

- Is it possible to forcefully remove the kernel modules when they are in use?
- Yes, its possible to configure the kernel to allow forced removal of modules, even when they appear to be busy
- Commands normally used are

**-rmod -f module\_name**

**-modprobe -rf module\_name**

**-These commands only work when you enable  
the kernel configuration**

**CONFIG\_MODULE\_FORCE\_UNLOAD**

# Revise: Loadable Kernel Modules

- Linux has the ability of run time extension which means you can add functionality to the kernel while the system is up or running
- Each piece of code that can be added to the kernel at run time is called a module
- The module defines two functions: One to be invoked when the module is loaded into the kernel and other when the module is removed
- The `module_init()` and `module_exit()` special kernel macros to indicate the role of the two functions
- `MODULE_LICENSE()` macro is used to inform the kernel that this module bears a free license
- `printk()` is used to display output
- `cat /var/log/syslog` or `dmesg` will display your message

# Revise: Loadable Kernel Modules

- To build your module, you need to create Makefile which will give the .ko file as the resulting module
- After Makefile, insmod program loads the module code and data into the kernel
- insmod typically allocates kernel memory to hold the module. It then copies the module text into the memory region
- It performs the similar function to that of GNU ld(linker) and links any resolved symbol into the module to the symbol table of linux kernel and calls module initialization function
- rmmod() is used to unload the module from the kernel
- lsmod program produce a list of modules currently loaded in the kernel
- The \_\_init and \_\_initdata is used to tell the kernel that function is used only initialization, it will drop the init function after loading, making its memory available for another use

# Revise: User Space Vs Kernel Space

- Module runs in the kernel space, whereas application runs in the user space
- The kernel executes in the highest level, where everything is allowed and application executes at the lowest level, where the processor regulates direct access to hardware and unauthorized access to memory
- System calls are the interface where user space and kernel space can communicate with each other
- Kernel code executing a system call is working in the context of a process-it operates on behalf of the calling process and is able to access data in the process's address space
- Code that handles interrupts, on the other hand, is asynchronous with respect to processes and is not related to any particular process
- Some functions in the module are executed as part of the system calls, and some are in charge of interrupt handling

# Kernel modules Vs Applications

- Initialization
  - Most small and medium sized application programs perform a single task from beginning to end
  - Whereas every kernel module registers itself with kernel in order to serve future requests, and its initialization function terminates immediately
  - In other words, the task of the module's initialization function is to prepare for later invocation of the module's function when required

# Kernel modules Vs Applications

- Which are event driven KM or AP?
  - When a kernel module is loaded, it invokes init function (like `hello_init`) which conveys kernel “Here I am, and this is what I can do”
  - When kernel module is unloaded, it invokes exit function (like `hello_exit`) which conveys kernel “I am not there anymore; don’t ask me to do anything else”
  - This kind of approach to programming is similar to event driven programming, but while not all applications are event-driven, each and every kernel module is event driven

# Kernel modules Vs Applications

- About exit functions in KM and AP
  - Major difference between event-driven applications and kernel code is in the exit function
  - When an application terminates, it can be lazy in releasing resources or avoids clean up together, the exit function of a module must carefully undo everything otherwise there is a chance that few pieces remain around until the system is rebooted

# Kernel modules Vs Applications

- Ability to unload a module
  - Incidentally, the ability to unload a module is one of the features of modularization that you will most appreciate, because it helps cut down development time; you can test successive versions of your new driver without going through the lengthy shut down/reboot cycle each time
  - With application, you need to either exit it completely and return the application. You will not be sure if the memory allocated is deallocated or the task initiated is exited cleanly. It all depends on how you implement the application

# Kernel modules Vs Applications

- Handling faults

-Segmentation fault is harmless during application development and a debugger can always be used to trace the error to the problem in the source code; a kernel fault kills the current process atleast, if not the whole system

# Kernel symbol tables

- Symbol in Linux

-In Linux, symbols are nothing but variables and functions that are needed to implement modularized drivers. Note that each symbol has its address in memory

# How symbols are exported?

- Exporting kernel symbols is typically done with

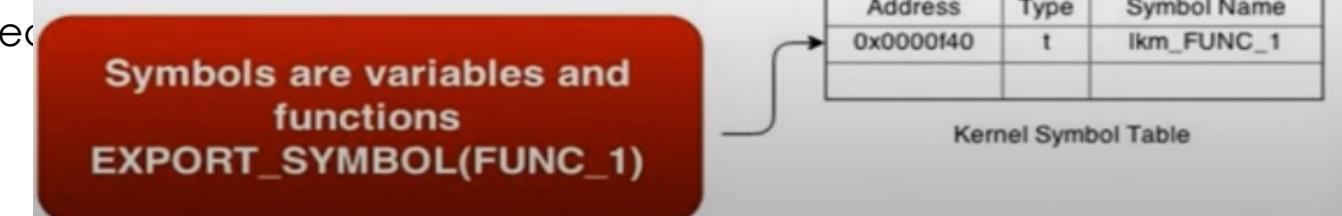
**-EXPORT\_SYMBOL()**

**-EXPORT\_SYMBOL\_GPL()**

- EXPORT\_SYMBOL(), which exports a given symbol to all loadable modules
- EXPORT\_SYMBOL\_GPL(), which exports a given symbol to only those modules that have a GPL compatible license
  - So whenever you write a driver, it should be GPL licensed. That will be great facility in case your module wants to use any other module's functions that are already exported using GPL

# Symbol and Symbol table relation

- Kernel symbol table is nothing but a look up table between symbol names and their address in memory
- When a module is loaded into kernel memory using insmod or modprobe utility, any symbol exported by the module becomes public kernel symbols



# More about symbol table

- While insmoding, the insmod utility resolves undefined symbols against the table of public kernel symbols
- Kernel symbol tables hold all the information needed to find program symbols, assign value to them and relocate them
- Primary task of symbol is to associate a string with a value. For ex, printk symbol represents the address of the printk function in virtual address space where the machine code resides
- This kernel symbol table is loaded into memory as part of the kernel boot process
-

# Basics of device driver types

- About /dev directory

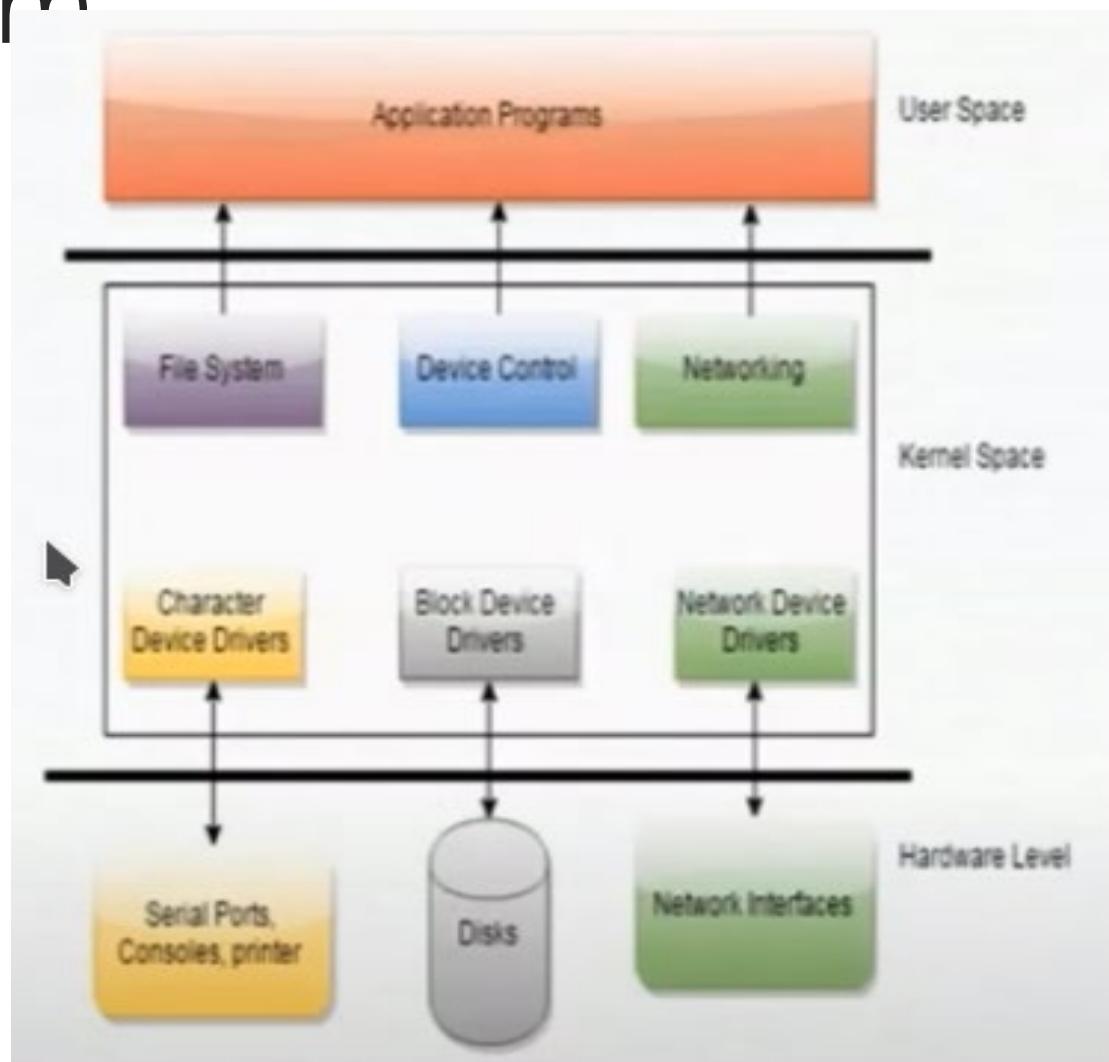
- The /dev directory on a Linux machine is where all the device files for the system are located
- Using this device file, a user program or application can access a specific hardware device
- /dev is a very interesting directory that highlights one important aspect of the Linux filesystem
- Take an example of a Printer driver in your Linux system. The device file of printer is /dev/lp0. Any data written to this file by your application will be redirected to your printer
- Similarly, data read/written to /dev/ttyS0 will allow you to communicate with a device like serial terminal or a modem device

- Device driver types

- In Linux, we have following device types
  - Character devices
  - Block devices
  - Network devices

# Device types diagram

- As shown in the diagram, all three device drivers fall under kernel space
- Each device has it's device driver
- These device driver has an entry of its device file under /dev special directory
- An application which resides in User space, communicates through device files under /dev and access the specific device it needs



# Introduction to Character Device Driver

- Character devices can be compared to normal files in which we can read/write arbitrary bytes at a time. Character device is one that can access as a stream of bytes like a file
- The difference between a char device and a regular file is that you can always move back and forth in the regular file, whereas most char devices are just data channels, which you can only access sequentially
- Character devices are accessed through names in the file system called Device files
- However, character devices are not limited to performing one character at a time(despite the name “character device”). For example, tape drivers (which is character device) frequently performs I/O operations in 10K chunks. You can also use a character device which it is necessary to copy data directly to or from a user process

# Character Device Driver

- Examples of character device drivers are:

-Serial port(/dev/ttys0)

-Text console(/dev/console)

-Mice

-Parallel printer ports

- Character devices are accessed by means of filesystem nodes, such as /dev/tty1 and /dev/lp0
- Since /dev contains all devices, how can we identify character device drivers under this directory?

# Block Devices

- Operate on blocks of data, not on arbitrary bytes. Usual block size is 512 bytes or larger powers of two
- Linux allows the application to read and write a block device like a char device. It permits the transfer of any number of bytes at a time
- As a result, block and char devices differ only in the way data is managed internally by the kernel
- Like a char device, each block device is accessed through a filesystem node, and the difference between them is transparent to the user
- Examples of block device drivers include

**-Hard drives**

**-CD-ROM drives**

- Since /dev contains all devices, how can we identify block device under this directory

# Network device

- Network device drivers receive and transmit data packets on hardware interface that connect to external systems, and provide a uniform interface that network protocol can access
- The difference between block device driver(disk) and Network device driver (packet-delivery interface) is that the block device exists as a special file in the /dev directory whereas network interface has no such entry point
- These network interfaces exist in their own namespace and export a different set of operations

# Minor and Major Numbers

- These are also called device numbers
- If you issue “ls -l” command in /dev directory, two numbers (separated by a comma) in the device file entries just before the date of the last modification, where the file length normally appears

# Major Number

- Is an identifier which identifies the driver associated with the device. So, each device will have a major number
- These major numbers can be defined by us (only if we know exactly which major number is available) or you can leave this job to kernel. Either way kernel knows to which major number our device is associated with

# Minor Number

- Is an identifier which is used by the kernel to determine exactly which device is being referred to
- Allocating minor number is same as major number but only difference is the kernel knows nothing about minor numbers beyond the fact that they refer to devices implemented by your drivers
- Note that there can be number of devices which can be controlled by a single driver. Multiple disks with a single block device driver is a good example
- Example ls /dev/sda-

# Representation of device numbers

- In order to represent major and minor numbers in specific format, Linux kernel developers have embedded major and minor numbers in a structure dev\_t type which is defined in <linux/types.h>

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

20 for the

# Revise: Major Number & Minor Number

- The major number identifies the driver associated with the device. Kernel uses the major number to invoke the driver.
- The minor is used by the driver to check exactly which device is referred to
- Within the kernel, the `dev_t` type (defined in `<linux/types.h>`) is used to hold device numbers-both the major and minor parts

# Allocating Major & Minor Number

- We can allocate the Major and Minor number in two ways:

- Statically

- . int register\_chrdev\_region(dev\_t first, unsigned int count, char \*name);
      - . The dev\_t type(defined in <linux/types.h>) is used to hold device numbers-both the major and minor parts.
      - . If you want to create the dev\_t structure variable for your major and minor number:
        - MKDEV(int major, int minor);
      - . To get your major number and minor number from dev\_t, use below method
        - MAJOR(dev\_t dev);
        - MINOR(dev\_t dev);
  - Dynamically
- . int alloc\_chrdev\_region(dev\_t \*dev, unsigned int firstminor, unsigned int count, char \*name);

# Allocating device numbers

- While setting up a char device, first thing our driver need to do is to obtain one or more device numbers to work with
- To do so, Linux kernel provides following functions

-int register\_chrdev\_region(dev\_t first,  
unsigned int count, char \*name);

-int alloc\_chrdev\_region(dev\_t \*dev, unsigned  
int firstminor, unsigned int count, char \*name);

# register\_chrdev\_region & alloc\_chrdev\_region

• int register\_chrdev\_region(dev\_t first, unsigned int count, char \*name);

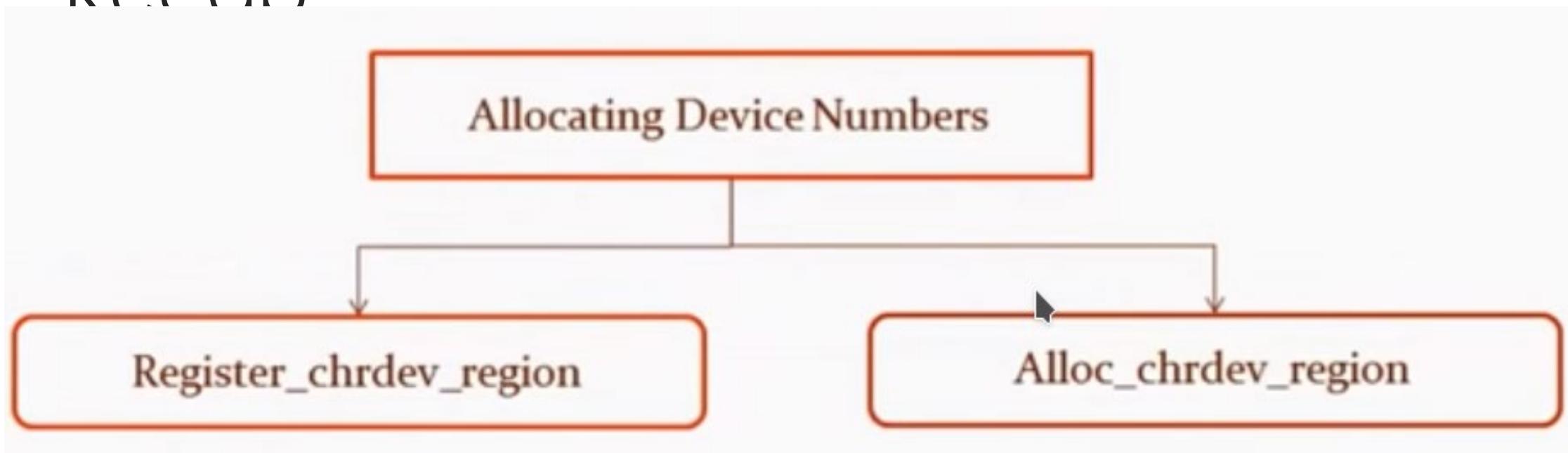
- first – is the beginning device number of the range you would like to allocate
- count – is the total number of contiguous device numbers you are requesting
- Name – is the name of the device that should be associated with this number range; it will appear in /procs/devices and sysfs
- Return value of this function will be 0 on success. In case of error, a negative error code will be returned and obviously you will not have access to the requested region

• int alloc\_chrdev\_region(dev\_t \*dev, unsigned int firstminor, unsigned int count, char \*name);

- Dev is an output-only parameter that will, on successful completion, hold the first number in your allocated range
- Firstminor – should be the requested first minor number to use; it is usually 0
- The count and name of parameters work like those given to request\_chrdev\_region

• Note: The usual place to call these functions would be in your module's init function

# Recon



# Freeing device numbers

- Now in order to free this region when its no longer required, kernel provides following function
  - Void unregister\_chrdev\_region(dev\_t first, unsigned int count);
  - The usual place to call unregister\_chrdev\_region would be in your module's clean-up function

# Best way to allocate device numbers

- As a device driver programmer, we have two choices to pick major number for a device

**-Pick a random number that appears to be unused**

**-Allocate major numbers in a dynamic manner**

- If we randomly pick a number that is unused, the driver works until it's used by only you. If your driver is widely used, a randomly picked major number will lead to conflict and cause more trouble
- So, it's always recommended to dynamically allocate major numbers for a new driver by using `alloc_chrdev_region` rather than `register_chrdev_region`

# How application communicate with the hardware device

- First application will open the device file which is created by device driver which we can create in two ways:
  - Manual: `mknod -m 666 /dev/char_device c 246 0`
  - Automatically:
    - Include the header file `linux/device.h` and `linux/kdev_t.h`
    - Create the struct class
      - Struct class **class\_create(struct module \*owner, const char \*name);**
    - Create device with the class
      - Struct device \***device\_create(struct \*class, struct device \*parent, dev\_t dev, const char \*fmt, ...)**
    - You can destroy the device using `device_destroy()`
      - Void **device\_destroy(struct class \*class, dev\_t devt);**



**AMITY**  
UNIVERSITY

Department of Computer  
Science and Engineering

