

TKM COLLEGE OF ENGINEERING KOLLAM-5
DEPARTMENT OF COMPUTER APPLICATIONS
(Affiliated to APJ Abdul Kalam Kerala Technological University)



LAB MANUAL

Subject Name : **NETWORKING & SYSTEM ADMINISTRATION LAB**

Subject Code : 20MCA136

Branch : MCA

Semester: 2

By:

Prof.Vaheetha Salam &

Dr. Fousia M Shamsudeen

VISION

Excellence in computer application development and research with robust environmental and social outlook.

MISSION

- *Impart quality education in computer applications and mould competent professionals for industrial and societal needs.*
- *Encourage research and take up collaborative projects*
- *Inculcate the habits of continual learning, sustainable computing and ethical responsibilities*

PROGRAM EDUCATIONAL OBJECTIVES

PEO 1

Excel in professional career and / pursue higher education and research utilizing the knowledge gained in mathematics and computational domain.

PEO 2

Ability to analyse real world problems and develop economically feasible and environmentally acceptable solutions.

PEO 3

Work in multidisciplinary teams with robust ethical and sustainable computing perspectives, adaptable to the changing trends in technology and society by engaging in lifelong learning.

PROGRAM OUTCOMES

MCA Graduates will be able to have :

1. **Computational Knowledge** : Apply knowledge of computing fundamentals, computing specialization, mathematics, and domain knowledge appropriate for the computing specialization to the abstraction and conceptualization of computing models from defined problems and requirements.
2. **Problem analysis** : Identify, formulate, research literature, and solve complex computing problems reaching substantiated conclusions using fundamental principles of mathematics, computing sciences, and relevant domain disciplines.
3. **Design/development of solutions** : Design and evaluate solutions for complex computing problems, and design and evaluate systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems** : Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5. **Modern tool usage** : Create, select, adapt and apply appropriate techniques, resources and modern computing tools to complex computing activities, with an understanding of the limitations.
6. **Professional Ethics** : Understand and commit to professional ethics and cyber regulations, responsibilities, and norms of professional computing practices.
7. **Life-long Learning** : Recognize the need, and have the ability, to engage in independent learning for continual development as a computing professional.
8. **Project management and finance** : Demonstrate knowledge and understanding of the computing and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
9. **Communication Efficacy** : Communicate effectively with the computing community, and with society at large, about complex computing activities by being able to comprehend and write effective reports, design documentation, make effective presentations and give and understand clear instructions.
10. **Societal and Environmental Concern** : Understand and assess societal, environmental, health, safety, legal and cultural issues within local and global contexts and the consequential responsibilities relevant to professional computing practices.
11. **Individual and Team Work** : Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary environments.
12. **Innovation and Entrepreneurship** : Identify a timely opportunity and using innovation to pursue that opportunity to create value and wealth for the betterment of the individual and society at large.

GUIDELINES TO STUDENTS

1. Students are supposed to occupy the computers allotted to them and maintain strict discipline in the Lab.
2. Student should get the record of previous program duly signed before starting the new experiment.
3. If a student is absent for any lab, they need to be completed the same experiment before attending next lab.
4. Students must use the equipment with care. Any damage caused by student is punishable
5. Turn off the computer before leaving the lab unless a member of lab staff has specifically told you not to do so
6. Students are not allowed to use their cell phones/pen drives/ CDs in labs.
7. Students should adhere to proper dress code along with ID Card.
8. Keep your mobile phone switched off before entering the lab
9. Don't use internet, internet chat of any kind in your regular lab schedule.
10. No games are allowed in the lab sessions.
11. No hardware including USB drives can be connected or disconnected in the labs without prior permission of the lab in-charge.
12. Do not leave the lab without permission from the lab in charge.
13. If you are having problems or questions, please consult with either the faculty or the supporting staff. They will help you. We need your full support and cooperation for smooth functioning of the lab.

INDEX

SL.NO	PROGRAMS	PAGE.NO
	Syllabus	
	Cycle 1	8
1	Familiarisation to computer hardware	
2	Specification of desktop and server computers	
3	Installation of linux on virtual box in windows 10	
4	Installation of windows on virtual box in ubuntu	
5	Installation of windows operating system on a bare machine	
6	Installation of linux over windows operating system	
7	Installation of server system	
	Cycle 2	21
8	Study on command line text editors	
9	Basic Linux Commands	
10	Familiarisation with Vi Editor	
11	Familiarisation with EMACS editor	
12	Linux File System	
	Cycle 3	40
13	Familiarisation to linux shell and shell scripting	
14	Linux shell scripting problems	
	Cycle 4	54
15	Installation of LAMP	
16	Installation of Laravel	
	Cycle 5	59
17	Familiarisation to command Line tool for networking	
18	Familiarisation of Static and Dynamic IP	
19	Concept of subnets and CIDR address scheme	
20	Concept of subnet mask	
21	Setting up of firewall on LAN	
22	Wireshark and TCP Dump	
23	Analyse Packets using wireshark	
	Cycle 6	75
24	Familiarisation to hypervisors and Virtual machines	
25	Familiarisation to Xen or KVM	
26	Installation of software from source code	
27	Deploy Linux virtual machine using Ansible Playbook	

20MCA136 NETWORKING & SYSTEM ADMINISTRATION LAB

Preamble: This laboratory course is intended to provide the background knowledge required for a software professional in the fields of networking and system administration. Students will acquire necessary knowledge to deploy and administer systems.

Prerequisite: Basic understanding of computer programming, Internet and operating systems.

Course Outcomes: After the completion of the course the student will be able to

CO 1	Install and configure common operating systems.
CO 2	Perform system administration tasks.
CO 3	Install and manage servers for web applications.
CO 4	Write shell scripts required for system administration.
CO 5	Acquire skill sets required for a DevOps.

Mapping of course outcomes with program outcomes

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
CO 1	1		2		2							
CO 2	1		2									
CO 3			2		2							
CO 4					2							
CO 5	2				2							

Assessment Pattern

Bloom's Category	Continuous Assessment Tests		End Semester Examination
	1	2	
Remember(K1)			
Understand(K2)			
Apply(K3)	10	10	10
Analyse(K4)	10	10	10
Evaluate(K5)	10	10	10
Create(K6)	20	20	20

Mark distribution

Total Marks	CIE	ESE	ESE Duration
100	50	50	3 hours

Continuous Internal Evaluation Pattern:

Continuous Internal Evaluation Pattern (Maximum Marks: 50)	
Attendance	15%
Maintenance of daily lab record and GitHub management	20%
Regular class viva	15%
Timely completion of day to day tasks	20%
Tests/Evaluation	30%

End Semester Examination Pattern:

End Semester Evaluation Pattern			
Verification of Daily program record and Git Repository			5 marks
Viva			10 marks
Problem solving (Based on difficulty level, one or more questions may be given)	Flowchart / Algorithm / Structured description of problem to explain how the problem can be solved / Interface Design	15%	35 marks
	Program correctness	50%	
	Code efficiency	15%	
	Formatted output and Pushing to remote Git repository	20%	
Total Marks			50 marks

Course Level Assessment Questions

Course Outcome 1

(CO1):

1. Install latest version of Ubuntu on a virtual box, set up a static ip address to it and install drupal environment.
2. You are given a computer with very low hardware resources. It is to be used as a kiosk. Identify and install a suitable Linux distribution. You can simulate it in a virtual environment.

Course Outcome 2 (CO2)

1. You are given a system which is connected to internet. However, users logging on to the system are unable to access internet from their browser. Trouble shoot the issue, clearly documenting the steps you have taken (Possible issues to look for are browser configuration, network connectivity, routing, ip address configuration, DNS resolution).
2. You are given a system which boots to a non graphical environment. You are also given a shell script which is designed for a specific task. Your task is to make sure that the script runs every time the system boots up. Write/modify necessary scripts for this.
3. You are required to add 100 users to a Linux system. Details of the users to be added were collected from a web form to a csv file. The csv may contain errors such as wrong case or missing fields. Write a script to add users using the data provided in the csv file with proper error checking.

Course Outcome 3 (CO3):

1. You are given a bare bone installation of latest version Ubuntu. Assume that the system is accessible from internet. Your task is to successfully install word press (or any other web application) on this server. Clearly indicate the steps taken and software installed for this task.
 2. Assume that you have an installation of old version Ubuntu. However, it does not have the latest version of virtual box (or some other application). The new version is available as a binary on a website. Upgrade to this version.
-

Course Outcome 4 (CO4):

1. Look at the system log files. Write a shell script to extract the last login details of a particular user and list out all failed logins. Store the results to a file. The user name should be given as a command line argument.
2. Write a shell script to display the details of a particular process currently running. Assume that you have necessary permissions. The process name/id is to be given as a command line argument.

Course Outcome 5 (CO5):

1. Capture network traffic on your system. Using wireshark find out all http and https traffic to a specific host.
2. Write an Ansible playbook to deploy a new Linux VM on a remote server.

Syllabus

Introduction to Computer hardware. Study of various peripherals. Study of common operating systems. File system organization in common operating systems. Study of command line environment in common operating systems. Study of command line tools for system administration. Shell scripting: bash shell, shell scripts for system management. Study of startup scripts. Study of server software for common applications such as http, ftp, dns, dhcp. Practical study of Ipv4 and Ipv6 networking protocols. Setting up firewalls. Virtual machines and containers. Configuration and deployment.

List of Lab Experiments/Exercises

To gain proficiency in command line tools and operations, it is highly recommended to use a terminal window instead of GUI tools. This will later help the student with latest approaches in maintaining cloud based infrastructure. virtualbox/queemu. may be used for this.

1. Introduction to Computer hardware: Physical identification of major components of a computer system such as mother board, RAM modules, daughter cards, bus slots, SMPS, internal storage devices, interfacing ports. Specifications of desktop and server class computers. Installation of common operating systems for desktop and server use. (Students may be asked to formulate specification for computer to be used as Desktop, Web server)
2. Study of a terminal based text editor such as Vim or Emacs. (By the end of the course, students are expected to acquire following skills in using the editor:

cursor operations, manipulate text, search for patterns, global search and replace) Basic Linux commands, familiarity with following commands/operations expected

1. man
 2. ls, echo, read
 3. more, less, cat,
 4. cd, mkdir, pwd, find
 5. mv, cp, rm ,tar
 6. wc, cut, paste
 7. head, tail, grep, expr
 8. chmod, chown
 9. Redirections & Piping
 10. useradd, usermod, userdel, passwd
 11. df,top, ps
 12. ssh, scp, ssh-keygen, ssh-copy-id
-
3. File system hierarchy in a common Linux distribution, file and device permissions, study of system configuration files in /etc, familiarizing log files for system events, user activity, network events.
 4. Shell scripting: study bash syntax, environment variables, variables, control constructs such as if, for and while, aliases and functions, accessing command line arguments passed to shell scripts. Study of startup scripts, login and logout scripts, familiarity with systemd and system 5 init scripts is expected.
 5. Installation and configuration of LAMP stack. Deploy an open source application such as phpmyadmin and Wordpress.
 6. Installation and configuration of common software frame works such as Laravel. (Student should acquire the capability to install and configure a modern framework)
 7. Build and install software from source code, familiarity with make and cmake utilities expected
 8. Introduction to command line tools for networking IPv4 networking, network commands: ping route traceroute, nslookup, ip. Setting up static and dynamic IP addresses. Concept of Subnets, CIDR address schemes, Subnet masks, iptables, setting up a firewall for LAN, Application layer (L7) proxies.
 9. Analyzing network packet stream using tcpdump and wireshark. Perform basic network service tests using nc.

10. Introduction to Hypervisors and VMs, Xen or KVM , Introduction to Containers: Docker, installation and deployment.

11. Automation using Ansible: Spin up a new Linux VM using Ansible playbook

Course Contents and Lecture Schedule

Topic	No.of hours
1 Introduction to Computer hardware: Physical identification of major components of a computer system such as mother board, RAM modules, daughter cards, bus slots, SMPS, internal storage devices, interfacing ports. Specifications of desktop and server class computers. Installation of common operating systems for desktop and server use. (Students may be asked to formulate specification for computer to be used as Desktop, Web server)	2
2. Study of a terminal based text editor such as Vim or Emacs. (By the end of the course, students are expected to acquire following skills in using the editor: cursor operations, manipulate text, search for patterns, global search and replace) Basic Linux commands, familiarity with following commands/operations expected 1. man 2. ls, echo, read 3. more, less, cat, 4. cd, mkdir, pwd, find 5. mv, cp, rm ,tar 6. wc, cut, paste 7. head, tail, grep, expr 8 chmod, chown 9. Redirections & Piping 10. useradd, usermod, userdel, passwd 11. df,top, ps 12 ssh, scp, ssh-keygen, ssh-copy-id	4
3. File system hierarchy in a common Linux distribution, file and device permissions, study of system configuration files in /etc, familiarizing log files for system events, user activity, network events.	2

4. Shell scripting: study bash syntax, environment variables, variables, control constructs such as if, for and while, aliases and functions, accessing command line arguments passed to shell scripts. Study of startup scripts, login and logout scripts, familiarity with systemd and system 5 init scripts is expected	6
5. Installation and configuration of LAMP stack. Deploy an open source application such as phpmyadmin and Wordpress.	4
6. Installation and configuration of common software frame works such as Laravel. (Student should acquire the capability to install and configure a modern framework)	4
7. Build and install software from source code, familiarity with make and cmake utilities expected	4
8. Introduction to command line tools for networking IPv4 networking, network commands: ping route traceroute, nslookup, ip. Setting up static and dynamic IP addresses. Concept of Subnets, CIDR address schemes, Subnet masks, iptables, setting up a firewall for LAN, Application layer (L7) proxies.	4
9. Analyzing network packet stream using tcpdump and wireshark. Perform basic network service tests using nc.	4
10. Introduction to Hypervisors and VMs, Xen or KVM , Introduction to Containers: Docker, installation and deployment.	4
11. Automation using Ansible: Spin up a new Linux VM using Ansible playbook	4

Cycle 1

MCA Department, TKMCE

1.Introduction to Computer hardware

Introduction to Computer hardware: Physical identification of major components of a computer system such as mother board, RAM modules, daughter cards, bus slots, SMPS, internal storage devices, interfacing ports. Specifications of desktop and server class computers. Installation of common operating systems for desktop and server use. (Students may be asked to formulate specification for computer to be used as Desktop, Web server)

MAJOR COMPONENTS OF COMPUTER SYSTEM

Mother board

A motherboard provides connectivity between the hardware components of a computer, like the processor (CPU), memory (RAM), hard drive, and video card. There are multiple types of motherboards, designed to fit different types and sizes of computers. Each type of motherboard is designed to work with specific types of processors and memory, so they don't work with every processor and type of memory. However, hard drives are mostly universal and work with the majority of motherboards, regardless of the type or brand.

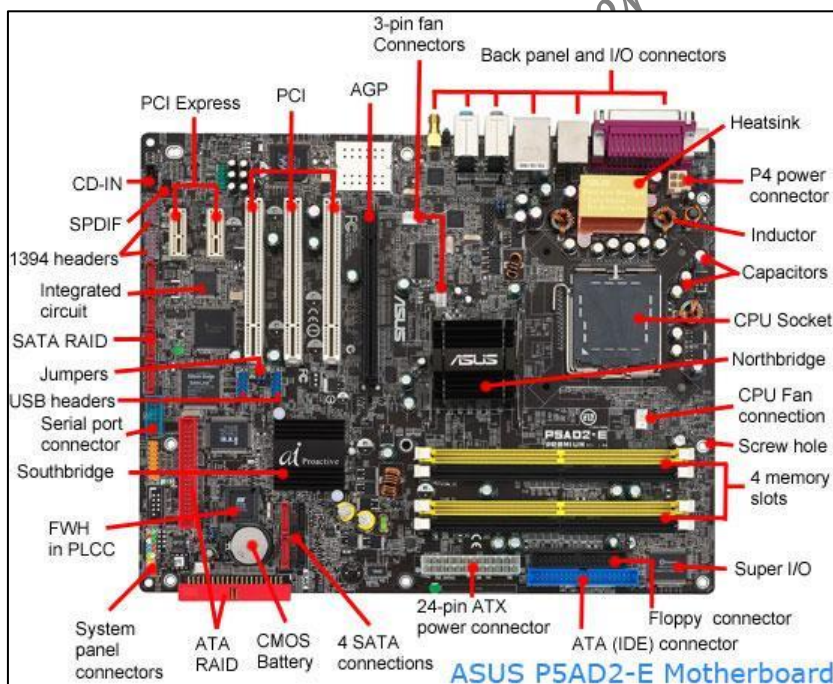


Figure 1: A standard Motherboard

A computer motherboard is located inside the computer case and is where most of the parts and computer peripherals connect. With tower computers, the motherboard is on the left or right side of the tower and is the biggest circuit board.

RAM modules

RAM(Random Access Memory) is a part of computer's Main Memory which is directly accessible by CPU. RAM is used to Read and Write data into it which is accessed by CPU randomly. RAM is volatile in nature, it means if the power goes off, the stored information is lost. RAM is used to store the data that is currently processed by the CPU. Most of the programs and data that are modifiable are stored in RAM.

Integrated RAM chips are available in two form:

1. *SRAM(Static RAM)* : Static memories(SRAM) are memories that consist of circuits capable of retaining their state as long as power is on. Thus this type of memories is called volatile memories.
2. *DRAM(Dynamic RAM)*: DRAM stores the binary information in the form of electric charges that applied to capacitors. The stored information on the capacitors tend to lose over a period of time and thus the capacitors must be periodically recharged to retain their usage. The main memory is generally made up of DRAM chips.

Types of DRAM:

There are mainly 5 types of DRAM:

1. *Asynchronous DRAM (ADRAM)*: The DRAM described above is the asynchronous type DRAM. The timing of the memory device is controlled asynchronously. A specialized memory controller circuit generates the necessary control signals to control the timing.
2. *Synchronous DRAM (SDRAM)*: These RAM chips' access speed is directly synchronized with the CPU's clock. For this, the memory chips remain ready for operation when the CPU expects them to be ready.
3. *Double-Data-Rate SDRAM (DDR SDRAM)*: This faster version of SDRAM performs its operations on both edges of the clock signal; whereas a standard SDRAM performs its operations on the rising edge of the clock signal.
4. *Rambus DRAM (RDRAM)*: The RDRAM provides a very high data transfer rate over a narrow CPU-memory bus. It uses various speedup mechanisms, like synchronous memory interface, caching inside the DRAM chips and very fast signal timing. The Rambus data bus width is 8 or 9 bits.
5. *Cache DRAM (CDRAM)*: This memory is a special type DRAM memory with an on-chip cache memory (SRAM) that acts as a high-speed buffer for the main DRAM.

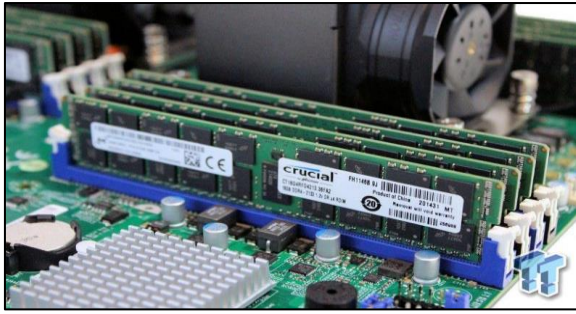


Figure 2: DDR4 RAM in RAM sockets

Daughter Cards

A Daughter Card (daughterboard) is an expansion board that connects directly to the motherboard and gives added functionality (e.g., modem). Today, these boards are not found or used in desktop computers. They were replaced with ISA cards, PCI cards, and onboard options. However, some laptops still use these boards.

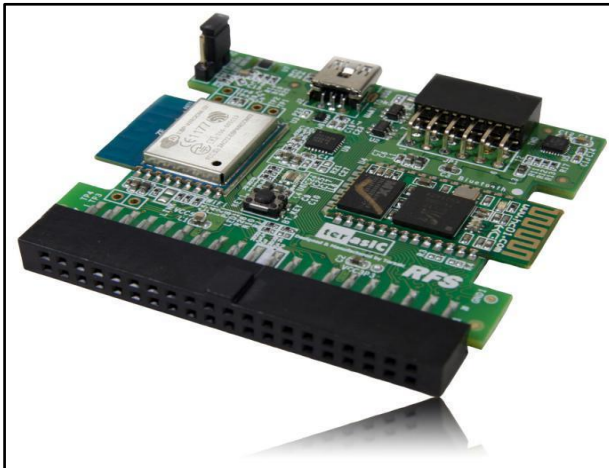


Figure 3: Daughter Card

Bus Slots

A bus slot or expansion port, an expansion slot is a connection or port inside a computer on the motherboard or riser card. It provides an installation point for a hardware expansion card to be connected. A list of bus slots present commonly in a motherboard are given below:

1. AGP - Video card.
2. AMR - Modem, sound card.
3. CNR - Modem, network card, sound card.
4. ISA - Network card, sound card, video card.
5. PCI - Network card, SCSI, sound card, video card.
6. PCI Express - Video card, modem, sound card, network card.

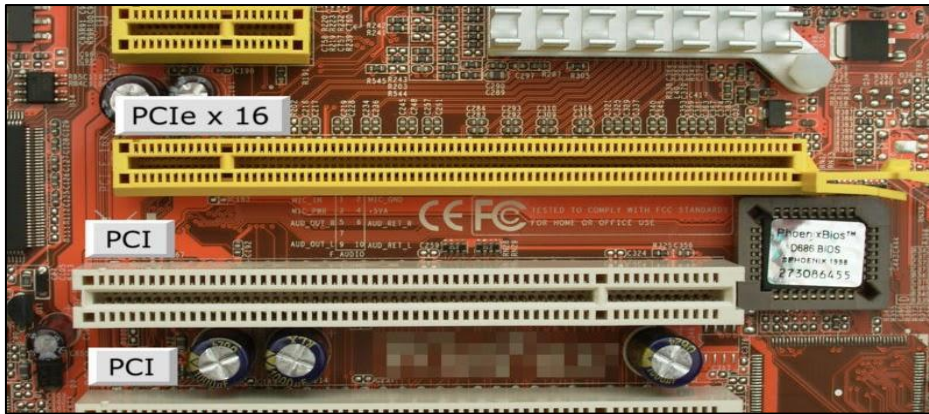


Figure 4: PCI and PCIe Bus Slots

SMPS

A switched-mode power supply (SMPS) is an electronic circuit that converts power using switching devices that are turned on and off at high frequencies, and storage components such as inductors or capacitors to supply power when the switching device is in its non-conduction state. Switching power supplies have high efficiency and are widely used in a variety of electronic equipment, including computers and other sensitive equipment requiring stable and efficient power supply. A switching regulator does the regulation in the SMPS. A series switching element turns the current supply to a smoothing capacitor on and off. The voltage on the capacitor controls the time the series element is turned. The continuous switching of the capacitor maintains the voltage at the required level.

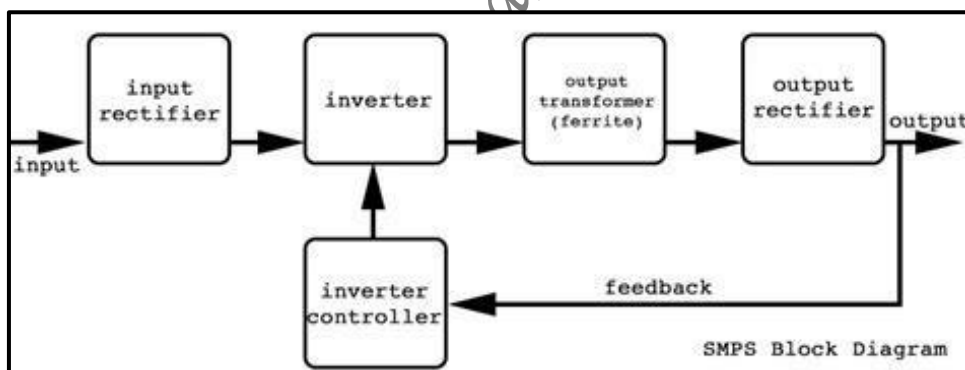


Figure 5: SMPS Block Diagram

Hard Disk Drive(Magnetic Storage Device)

HDD is an electro-mechanical storage device, which is an abbreviation of Hard Disk Drive. It uses magnetic storage for storing and retrieving the digital data. It is a non-volatile storage device. Hard Disk Drive is installed internally in our computer systems, which is connected directly to the disk controllers of the motherboard. Hard Disk Drive is a storage device which stores the operation system (OS), installed software, and the other computer files.



Figure 6: HDD

SSD

A solid-state drive (SSD) is a solid-state storage device that uses integrated circuit assemblies to store data persistently, typically using flash memory, and functioning as secondary storage in the hierarchy of computer storage. It is also sometimes called a solid-state device or a solid-state disk, even though SSDs lack the physical spinning disks and movable read–write heads used in hard disk drives (HDDs) and floppy disks.



Figure 7: SSD

2.SPECIFICATIONS OF DESKTOP AND SERVER CLASS COMPUTERS

- **Processor:** 10th or 11th Gen Intel Core i5, i7 or i9 Processor, or Apple M1 Processor (CPU)
- **Operating System:** Microsoft Windows 10 Home, Pro, Enterprise or Education version
- **Memory (RAM):** 8-16 GB of RAM
- **Warranty:** 3 year on-site (better) or depot warranty, accidental damage service highly recommended.
- **Storage:** All the files, programs and games on your PC are kept on its storage or hard drive, which you can think of as the computer's "long-term memory". The more storage available, the more you can keep on there. Desktop PCs usually have lots of storage, usually starting at 500GB to 1TB, which is plenty for the average user. Storage is measured in gigabytes (GB) and terabytes (TB), with 1TB equal to 1,000 GB
- **Monitor:** A desktop monitor is the screen you use to operate your PC. When buying a monitor. Much like a TV, the main things to consider are screen size, resolution and budge
- **Mouse:** The mouse has been the main way of operating PCs for decades, offering a simple and effective way to use the operating system, as well as for typing, browsing and gaming.
- **Keyboard:** The keyboard has been the main way of operating PCs for decades, offering a simple and effective way to use the operating system, as well as for typing, browsing and gaming.

INSTALLATION OF AN OPERATING SYSTEM TO YOUR DESKTOP/LAPTOP COMPUTER

1. Set up the display environment.

If you are not using the local DVD drive and monitor for the Tools and Drivers CD or OS installation CD, you have two options for your display environment: View system output serially, through the Embedded Lights Out Manager (LOM) service processor (SP) SSH or through the physical port. Use the remote KVMs Over IP feature of the Embedded LOM

2. Erase the primary boot disk.

If you have an operating system preinstalled on the server, you will need to remove it before installing a new operating system

To erase the primary boot hard disk:

1. Back up all data that you want to save on the hard drive.
2. Insert the Tools and Drivers CD into the server's optional DVD drive.
3. Boot the server from the CD.

4. When the main menu appears, select the following option from the Tools and Drivers CD main menu: Erase Primary Boot Hard Disk

3. Set up the BIOS.

You need to make sure that the BIOS is set up for the operating system that you plan to install.

4. Install the operating system.

5. Configure your server for RAID.

If you plan to configure your server for RAID operation, you will need to perform some setup tasks before installing the operating system.

6. Install the operating system, update the drivers, and run operating system updates, as necessary

HOW TO SELECT THE APPROPRIATE OPERATING SYSTEM FOR YOUR SYSTEM STABILITY AND ROBUSTNESS

➤ Memory Management

You want an OS with a good memory management, some OSs are well known as memory hogs. The same code can use twice as much memory on one OS compared to another.

➤ Cost and Support

If you want to install Windows on to your computer you need to purchase license. Each windows license comes up with an activation key which is good for one installation. Mostly Linux distributions are freely available to download and there are no such restrictions like single installation or one time installation.

➤ Stability and Robustness

Probably the most important features in an OS are stability and robustness. You are in an Internet business. You do not keep normal 9am to 5pm working hours like many conventional businesses you know. You are open 24 hours a day. You cannot afford to be off-line, for your customers will go shop at another service like yours (unless you have a monopoly :). If the OS of your choice crashes every day, first do a little investigation. There might be a simple reason which you can find and fix. There are OSs which won't work unless you reboot them twice a day. You don't want to use the OS of this kind, no matter how good the OS' vendor sales department. Do not follow flashy advertisements, follow developers advices instead.

➤ Memory Leaks

Some OSs and/or their libraries (e.g. C runtime libraries) suffer from memory leaks. A leak is when some process requests a chunk of memory for temporary storage, but then

does not subsequently release it. The chunk of memory is not then available for any purpose until the process which requested it dies. We cannot afford such leaks. A single `mod_perl` process sometimes serves thousands of requests before it terminates. So if a leak occurs on every request, the memory demands could become huge. Of course our code can be the cause of the memory leaks as well. Certainly, we can reduce the number of requests to be served over the process' life, but that can degrade performance.

➤ **Sharing Memory**

We want an OS with good memory sharing capabilities. As we have seen, if we preload the modules and scripts at server startup, they are shared between the spawned children (at least for a part of a process' life - memory pages can become "dirty" and cease to be shared). This feature can reduce memory consumption a lot!

➤ **OS Releases**

Actively developed OSs generally try to keep pace with the latest technology developments, and continually optimize the kernel and other parts of the OS to become better and faster. Nowadays, Internet and networking in general are the hottest topics for system developers. Sometimes a simple OS upgrade to the latest stable version can save you an expensive hardware upgrade. Also, remember that when you buy new hardware, chances are that the latest software will make the most of it.

3.Installation Of Linux On Virtual Box in Windows 10

VirtualBox is designed to run virtual machines on your physical machine without reinstalling your OS that is running on a physical machine. One more VirtualBox advantage is that this product can be installed for free. A virtual machine (VM) works much like a physical one.

WINDOWS 10

VirtualBox is a program which allows you to install an operating system without changing your computer's main operating system.

Ubuntu is a Linux distribution based on Debian and composed mostly of free and open-source software.

1. Download Virtual box from the
url: <https://www.virtualbox.org/wiki/Downloads>.
2. Install Virtual Box.
3. Download Ubuntu operating system from the
url: <https://ubuntu.com/download/desktop>.
4. After installation open Virtual box.
5. Click New Button.
6. Select the amount of memory (RAM) and click Next .
7. Choose the hard disk file type and click next.
8. Choose whether the new virtual hard disk file should grow as it is used dynamically or fixed size.
9. Select the location of the downloaded Ubuntu OS file and select the size of the virtual hard disk in mb. Create it
10. Now we can see Ubuntu has been added to the virtual box interface.

Any further correction can be made on settings including memorysize allocation mode.
Go to settings—Storage—empty—downloaded Ubuntu OS file.
Click Start and now we can boot to Ubuntu OS.\

4.Installation Of Windows On Virtual Box In Ubuntu

Step 1: Download Windows 10 ISO

First and foremost, you need to download a Windows 10 ISO. You can download Windows 10 32-bit or 64-bit, depending on your system.

Step 2: Install VirtualBox on Ubuntu

Install VirtualBox on Ubuntu using the command `sudo apt install virtualbox`

Step 3: Install Windows 10 in VirtualBox

5. Installation Of Windows Operating System On A Bare Machine

Our Objective is to install Windows 10 Operating system to our PC/Machine via a bootable medium. We will be installing the OS from scratch assuming that we want to update/format/fix the OS in our system by making a fresh installation.

Requirements :

1. PC/Machine with at least 4GB RAM
2. An 8GB DVD or USB Flash drive.
3. Rufus software (<https://rufus.ie/en-US/>).
4. Required Windows 10 OS/OEM.
5. Strong stable internet connection.

Step 1 : Acquiring Required Os/Oem

Step 2: Preparing Bootable Medium

Step 3 : System Booting & Installation

6.Installation Of Linux Over Windows Operating System

Requirements

- PC with at least 2GB RAM.
- 8GB Flash drive
- Rufus
- Linux ISO file

Steps involved

- Partition a Hard Drive in Windows 10
- Make a Linux Bootable USB
- Install linux from USB

Step1: Partition a Hard Drive in Windows 10

- open cmd
- Type the command “diskmgmt.msc” and hit enter.

Step2: Make a Linux Bootable USB

- Download a Linux distro in ISO format.
<https://ubuntu.com/>
- Insert the USB drive into your computer.
- Download Rufus
https://rufus.ie/en_US/
- Open Rufus and select your USB drive from the Device list.
- Under Boot Selection, click the Select button and choose the ISO file you downloaded.

Step 3:Install linux from USB

- Place the USB stick, reboot the machine and instruct the UEFI to boot-up from the USB by pressing a special function key (usually F12, F10 or F2 depending on the vendor specifications).
- Choose the language you wish to perform the installation and click on the Continue
- choose the first option “Normal Installation” and hit on the Continue button.
- check the Something else option and hit on the Continue button
- we'll create our custom partition layout for Ubuntu
- hit the Install Now button in order to apply changes to disk and start the installation process.
- set your machine physical location by selecting a city nearby from the map. When done hit Continue to move ahead.
- Pick up a username and password for your administrative sudo account, enter a descriptive name for your computer and hit Continue to finalize the installation.
- After the installation process reaches its end hit on the Restart Now button in order to complete the installation.
- Ubuntu has successfully installed on your system.

7. Installation Of Server System

Phase 1: Collecting information

In the first installation phase, the setup program asks for the preliminary information that it needs to begin the installation. A setup wizard prompts you for the following information

Language: Select your language, time-zone, and keyboard type.

Product Key: Enter the 25-character product key that came with the installation media. If setup says you entered an invalid product key, double-check it carefully. You probably just typed the key incorrectly.

Operating system type: The setup program lets you select Windows Server 2008 Standard Edition or Core. Choose Standard Edition to install the full server operating system; choose Core if you want to install the new text-only version.

License Agreement: The official license agreement is displayed. You have to agree to its terms in order to proceed.

Install Type: Choose an Upgrade or Clean Install type.

Disk Location: Choose the partition in which you want to install Windows.

Upgrade to NTFS: If you want to upgrade a FAT32 system to NTFS, you'll need to say so now

Phase 2: Installing Windows

In this phase, Windows setup begins the actual process of installing Windows. The following steps are performed in sequence:

- 1)**Copying Files:** Compressed versions of the installation files are copied to the server computer.
- 2)**Expanding Files:** The compressed installation files are expanded.
- 3)**Installing Features:** Windows server features are installed.
- 4)**Installing Updates:** The setup program checks Microsoft's website and downloads any critical updates to the operating system.
- 5)**Completing Installation:** When the updates are installed, the setup program reboots so it can complete the installation.

Configuring Your Server

After you've installed Windows Server 2008, the computer automatically reboots, and you're presented with the Initial Configuration Tasks Wizard. This wizard guides you through the most important initial tasks for configuring your new server

Cycle 2

MCA Department, TKMCE

8. Study On Command Line Text Editors

Text editors are program used to create and edit plain text files. The main purpose of the text editor is to create a file to be used by another program.

For example :

- HTML for a Web Browser
- Source code that a compiler can process.

There are two types of text editors:

1. Terminal Based editors

E.g.: Vi, Pico, Nano, Emacs.

2. GUI editors

Eg: Gedit, Kwrite

Emacs Editor

Emac is a very popular editor on Unix until recently. Emacs has so many available features like a calculator, calendar, email client. Commands in emacs are either control characters (hold down the <Ctrl> key while typing another character). Syntax: emacs filename

Pico Editor

The UNIX Pico editor is a full screen editor which is very easy to use. Syntax: Pico filename

Nano Editors

The nano is improved open source of pico available for GNU/Linux. Syntax: nano filename

Vim Editor

The vi command-line text editor is included with most versions of UNIX and Linux. Vim is a improved version of vi. The vi commands is now linked to the vim commands. There are three modes in vi editor:

1. Command mode: Use key combinations as commands instead of typing text.
2. Insert mode: Typed text is displayed on screen.
3. Extend mode: Used for more advanced commands, such as saving files, exiting vim, or searching and replacing text.

9. Study On Command Line Text Editors

Study of a terminal based text editor such as Vim or Emacs. (By the end of the course, students are expected to acquire following skills in using the editor: cursor operations, manipulate text, search for patterns, global search and replace) Basic Linux commands, familiarity with following commands/operations expected

1. man
2. ls, echo, read
3. more, less, cat,

4. cd, mkdir, pwd, find
5. mv, cp, rm ,tar
6. wc, cut, paste
7. head, tail, grep, expr
- 8 chmod, chown
9. Redirections & Piping
10. useradd, usermod, userdel, passwd
11. df,top, ps
- 12 ssh, scp, ssh-keygen, ssh-copy-id

Here is a list of basic Linux commands students are expected to know. You may take this set as a starting point and explore each of them.

1. pwd

Use the **pwd** command to find out the path of the current working directory (folder) you're in. The command will return an absolute (full) path, which is basically a path of all the directories that starts with a forward slash (/). An example of an absolute path is **/home/username**.

2. cd

To navigate through the Linux files and directories, use the **cd** . It requires either the full path or the name of the directory, depending on the current working directory that you're in.

Let's say you're in **/home/username/Documents** and you want to go to **Photos**, a subdirectory of **Documents**. To do so, simply type the following command: **cd Photos**.

Another scenario is if you want to switch to a completely new directory, for example, **/home/username/Movies**. In this case, you have to type **cd** followed by the directory's absolute path: **cd /home/username/Movies**.

There are some shortcuts to help you navigate quickly:

- **cd ..**(with two dots) to move one directory up
- **cd**to go straight to the home folder
- **cd-** (with a hyphen) to move to your previous directory

On a side note, Linux's shell is case sensitive. So, you have to type the name's

directory exactly as it is.

3. ls

The **ls** command is used to view the contents of a directory. By default, this command will display the contents of your current working directory.

If you want to see the content of other directories, type **ls** and then the directory's path. For example, enter **ls /home/username/Documents** to view the content of **Documents**.

There are variations you can use with the **ls** command:

- **ls -R** will list all the files in the sub-directories as well
- **ls -a** will show the hidden files
- **ls -al** will list the files and directories with detailed information like the permissions, size, owner, etc.
- **ls -t** lists files sorted in the order of "last modified"
- **-r** option will reverse the natural sorting order. Usually used in combination with other switches such as **ls -tr**. This will reverse the time-wise listing.

4. cat

cat (short for concatenate) is one of the most frequently used commands in Linux. It is used to list the contents of a file on the standard output stdout. To run this command, type **cat** followed by the file's name and its extension. For instance: **cat file.txt**.

Here are other ways to use the **cat** command:

- **cat > filename** creates a new file
- **cat filename1 filename2 > filename3** joins two files (1 and 2) and stores the output of them in a new file (3)
- to convert a file to upper or lower case use, **cat filename | tr a-z A-Z > output.txt**

Use the **cp** command to copy files from the current directory to a different directory. For instance, the command **cp scenery.jpg /home/username/Pictures** would create a copy of **scenery.jpg** (from your current directory) into the **Pictures** directory.

- **cp -i** will ask for user's consent in case of a potential file overwrite.
- **cp -p** will preserve source files' mode, ownership and timestamp.
- **cp -r** will copy directories recursively.
- **cp -u** copies files only if the destination file is not existing or the source file is newer than the destination file.

6. mv

The primary use of the **mv** command is to move files, although it can also be used to rename files.

The arguments in **mv** are similar to the **cp** command. You need to type **mv**, the file's name, and the destination's directory. For example: **mv file.txt /home/username/Documents**.

To rename files, the Linux is **mv oldname.extnewname.ext**

7. mkdir

Use **mkdir** command to make a new directory — if you type **mkdir Music** it will create a directory called **Music**.

There are extra **mkdir** commands as well:

- To generate a new directory inside another directory, use this Linux basic command **mkdir Music/Newfile**
- use the **p** (parents) option to create a directory in between two existing directories. For example, **mkdir -p Music/2020/Newfile** will create the new “2020” file.

8. rmdir

If you need to delete a directory, use the **rmdir** command. However, **rmdir** only allows you to delete empty directories.

9. rm

The **rm** command is used to delete directories and the contents within them. If you only want to delete the directory — as an alternative to **rmdir** — use **rm -r**.

Note: Be very careful with this command and double-check which directory you are in. This will delete everything and there is no undo.

10. touch

The **touch** command allows you to create a blank new file through the Linux command line. As an example, enter **touch /home/username/Documents/Web.html** to create an HTML file entitled **Web** under the **Documents** directory.

11. locate

You can use this command to **locate** a file, just like the search command in Windows. What's more, using the **-i** argument along with this command will make it case-insensitive, so you can search for a file even if you don't remember its exact name.

To search for a file that contains two or more words, use an asterisk (*). For example, **locate -i school*note** command will search for any file that contains the word "school" and "note", whether it is uppercase or lowercase.

12. find

Similar to the **locate** command, using **find** also searches for files and directories. The difference is, you use the **find** command to locate files within a given directory.

As an example, **find /home/ -name notes.txt** command will search for a file called **notes.txt** within the home directory and its subdirectories.

Other variations when using the **find** are:

- To find files in the current directory use, **find . -name notes.txt**
- To look for directories use, **/ -type d -name notes.txt**

13. grep

Another basic Linux command that is undoubtedly helpful for everyday use is **grep**. It lets you search through all the text in a given file.

To illustrate, **grep blue notepad.txt** will search for the word blue in the notepad file. Lines that contain the searched word will be displayed fully. You should refer to some grep tutorial

Useful for command line use as well. Usually output of a previous command is piped into the grep command. For example **ls -l | grep "kernel"**

14. sudo

Short for "**SuperUser Do**", this command enables you to perform tasks that require administrative or root permissions. You must have sufficient permissions to use this command.

15. df

Use **df** command to get a report on the system's disk space usage, shown in percentage and KBs. If you want to see the report in megabytes, type **df -m**.

16. du

If you want to check how much space a file or a directory takes, the **du** (Disk Usage) command is the answer. However, the disk usage summary will show disk block numbers instead of the usual size format. If you want to see it in bytes, kilobytes, and megabytes, add the **-h** argument to the command line.

17. head

The **head** command is used to view the first lines of any text file. By default, it will show the first ten lines, but you can change this number to your liking. For example, if you only want to show the first five lines, type **head -n 5 filename.ext.** (Read the manual)

18. tail

This one has a similar function to the head command, but instead of showing the first lines, the **tail** command will display the last ten lines of a text file. For example, **tail -n filename.ext.**

19. diff

Short for difference, the **diff** command compares the contents of two files line by line. After analyzing the files, it will output

20. tar

The **tar** command is the most used command to archive multiple files into a **tarball**— a common Linux file format that is similar to zip format, with compression being optional.

This command is quite complex with a long list of functions such as adding new files into an existing archive, listing the content of an archive, extracting the content from an archive, and many more. Read some tutorial on net.

21. chmod

chmod is another Linux command, used to change the read, write, and execute permissions of files and directories. Read about permissions and how to manipulate them .

22. chown

In Linux, all files are owned by a specific user. The **chown** command enables you to change or transfer the ownership of a file to the specified username. For instance, **chown linuxuser2 file.ext** will make **linuxuser2** as the owner of the **file.ext**.

it the lines that do not match. Programmers often use this command when they need to make program alterations instead of rewriting the entire source code.

The simplest form of this command is **diff file1.ext file2.ext**

Ps command will display all current processes along with their process ids (PID) .
Read manuals for various options.

24. Kill

If you have an unresponsive program, you can terminate it manually by using the **kill** command. It will send a certain signal to the misbehaving app and instructs the app to terminate itself.

There is a total of sixty-four signals that you can use, but people usually only use two signals:

- **SIGTERM (15)** — requests a program to stop running and gives it some time to save all of its progress. If you don't specify the signal when entering the kill command, this signal will be used.
- **SIGKILL (9)** — forces programs to stop immediately. Unsaved progress will be lost.

Besides knowing the signals, you also need to know the process identification number (PID) of the program you want to **kill**. If you don't know the PID, simply run the command **ps ux**. After knowing what signal you want to use and the PID of the program, enter the following syntax:

kill [signal option] PID.

You can issue kill -9 PID

25. ping

Use the **ping** command to check your connectivity status to a server. For example, by simply entering **ping google.com**, the command will check whether you're able to connect to Google and also measure the response time.

26. wget

The Linux command line is super useful — you can even download files from the internet with the help of the **wget** command. To do so, simply type **wget** followed by the download link.

27. uname

The **uname** command, short for Unix Name, will print detailed information about your Linux system like the machine name, operating system, kernel, and so on.

28. top

As a terminal equivalent to Task Manager in Windows, the **top** command will display a list of running processes and how much CPU each process uses. It's very useful to monitor system resource usage, especially knowing which process needs to be terminated because it consumes too many resources.

29. history

When you've been using Linux for a certain period of time, you'll quickly notice that you can run hundreds of commands every day. As such, running **history** command is particularly useful if you want to review the s you've entered before.

30. man

Confused about the function of certain Linux commands? Don't worry, you can easily learn how to use them right from Linux's shell by using the **man** command. For instance, entering **man tail** will show the manual instruction of the tail command.

Use the command: **man man** to start learning about man utility.

31. echo

This command is used to move some data into a file. For example, if you want to add the text, "Hello, my name is John" into a file called name.txt, you would type **echo Hello, my name is John >> name.txt**

32. zip, unzip

Use the **zip** command to compress your files into a zip archive, and use the **unzip** command to extract the zipped files from a zip archive. (This program should be installed , some distributions may not have them. You can also look at gzip and bzip commands)

33. hostname

If you want to know the name of your host/network simply type **hostname**. Adding a **-I** to the end will display the IP address of your network.

34. useradd, userdel

(This is available only to system admins) Since Linux is a multi-user system, this means more than one person can interact with the same system at the same time. **useradd** is used to create a new user, while **passwd** is adding a password to that user's account. To add a new person named John type, **useradd John** and then to add his password type, **passwd 123456789**.

To remove a user is very similar to adding a new user. To delete the users account type, `userdelUserName`

10.Familiarisation With Vi Editor

- The vi editor comes with every version of Linux or Unix.
- Using vi is similar to using other editors in which you can see your file on the screen .
- The vi editor is the most popular editor in linux.
- The current version is really "vim", but to invoke it simply type "vi".
- Before vi, the primary editor used was the line editor
- Users were able to see/edit only one line of the text at a time .
- The vi editor is not a text formatter (like MS Word, Word Perfect, etc.)
- You cannot set margins - center headings.

Vi Modes

There are mainly two modes in vi:

1. Command mode

This mode enables you to perform administrative tasks such as saving the files, executing the commands, moving the cursor, cutting (yanking) and pasting the lines or words, as well as finding and replacing. In this mode, whatever you type is interpreted as a command. vi always starts in the command mode.

2. Input mode

Accessed by typing "i". This mode permits insertion of new text, editing of existing text or replacement of existing text. This mode enables you to insert text into the file. Everything that is typed in this mode is interpreted as input and is placed in the file. To come out of the insert mode, press the Esc key, which will take you back to the command mode.

Common vi Commands

You will be using the following Vi control commands to work around files in Vi editor:

Command	Description
Editing a File	

i	Use this command to insert text before the current cursor location
I	Use this command to insert text at the beginning of the line
a	Use this command to insert text after the current cursor location
o	Use this command to create a new line for text below the current cursor location
Deleting characters	
x	Use this command to delete the character under the current location
X	Use this command to delete the character before the current location
dw	Use this command to delete from the current location to the next word
D	Use this command to delete from current location till end of the line
dd	Use this command to delete the entire line
Copying and pasting	
yy	Use this command to copy the current line
p	Use this command to paste the copied text after the cursor
P	Use this command to paste the yanked(cut) text before the cursor
Changing text	
cc	Use this command to remove contents of the line

s	Use this command to replace the character with the character you write
r	Use this command to replace the character under the cursor and return to command mode

11.Familiarisation With EMACS Editor

Emacs Editors

- Very popular editor
- Most complex and flexible text editor than other text editors.
- Freedom to the users to editing text files
- The emacs program can be running either in a terminal or in a GUI
- There are two major “brands” of emacs :GNU Emacs and Xemacs, and they have very similar functions.
- GNU Emacs –most popular version of Emacs which was created by Richard Stallman for the GNU Project.

Xemacs is a variant that branched from GNU Emacs In 1991

Some Features:

- Faster
- Powerful
- Simple

According to Vi editor, the emacs editor does not use an insert mode .

Commands in emacs are either control characters(hold down the Key while typing another character

- **Installing emacs Editor on Ubuntu**

```
$sudo apt-get install emacs
```

12.Linux File System

File system hierarchy in a common Linux distribution, file and device permissions, study of system configuration files in /etc, familiarizing log files for system events, user activity, network events.

Once installed, stay in your terminal window and run *tree* like this:

```
$ tree /
```

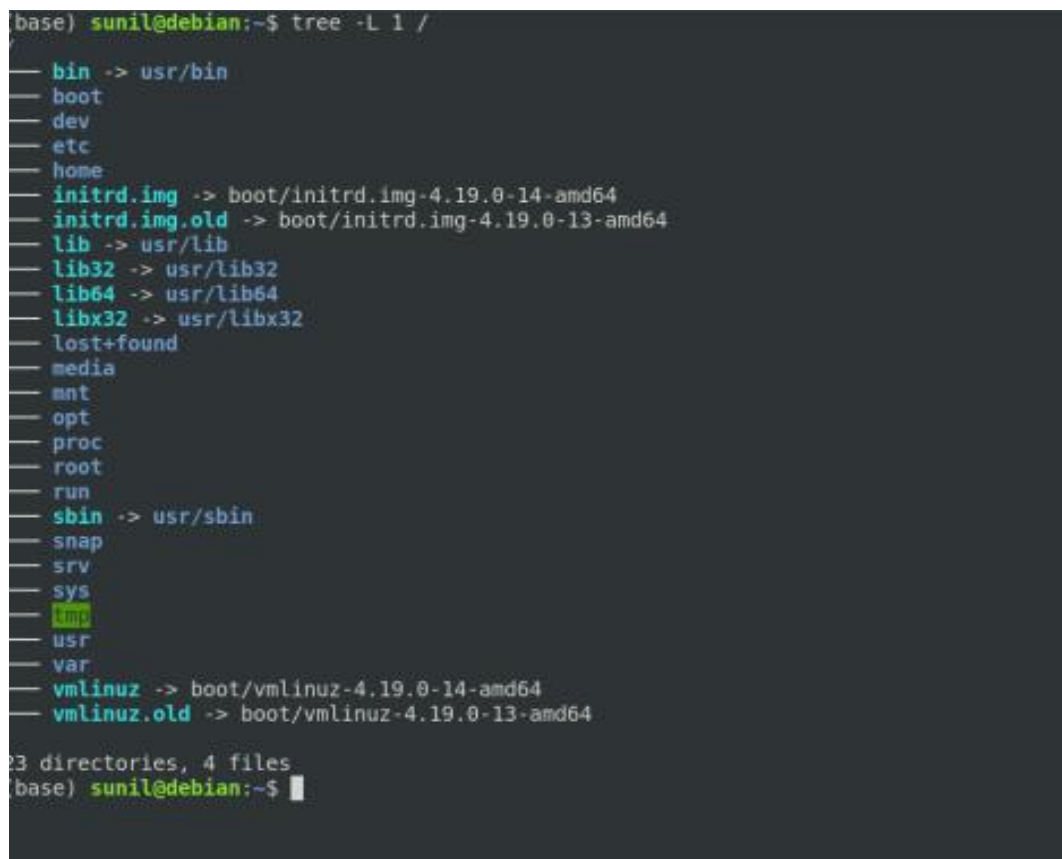
The `/` in the instruction above refers to the *root* directory. The root directory is the one from which all other directories branch off from. When you run `tree` and tell it to start with `/`, you will see the whole directory tree, all directories and all the subdirectories in the whole system, with all their files, fly by.

If you have been using your system for some time, this may take a while, because, even if you haven't generated many files yourself, a Linux system and its apps are always logging, caching, and storing temporary files. The number of entries in the file system can grow quite quickly.

Instead, try this:

```
tree -L 1 /
```

0And you should see a listing similar to what is shown in Figure 1.



```
(base) sunil@debian:~$ tree -L 1 /
tree
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── initrd.img -> boot/initrd.img-4.19.0-14-amd64
├── initrd.img.old -> boot/initrd.img-4.19.0-13-amd64
├── lib -> usr/lib
├── lib32 -> usr/lib32
├── lib64 -> usr/lib64
├── libx32 -> usr/libx32
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── snap
├── srv
├── sys
├── tmp
├── usr
├── var
├── vmlinuz -> boot/vmlinuz-4.19.0-14-amd64
├── vmlinuz.old -> boot/vmlinuz-4.19.0-13-amd64
└── 23 directories, 4 files
(base) sunil@debian:~$
```

The instruction above can be translated as “*show me only the 1st Level of the directory tree starting at / (root)*“. The `-L` option tells `tree` how many levels down you want to see.

Most Linux distributions will show you the same or a very similar layout to what you can see in the image above. This means that even if you feel confused now, master this, and you will have a handle on most, if not all, Linux installations in the whole wide

world.

Now, let's look at what each directory is used for. While we go through each, you can peek at their contents using *ls*.

Directories

From top to bottom, the directories you are seeing are as follows.

/bin

/bin is the directory that contains *binaries*, that is, some of the applications and programs you can run. You will find the *ls* program mentioned above in this directory, as well as other basic tools for making and removing files and directories, moving them around, and so on. There are more *bin* directories in other parts of the file system tree, but we'll be talking about those in a minute.

/boot

The */boot* directory contains files required for starting your system. If you mess up one of the files in here, you may not be able to run your Linux and it is a pain to repair. On the other hand, don't worry too much about destroying your system by accident: you have to have superuser privileges to do that.

/dev

/dev contains *device* files. Many of these are generated at boot time or even on the fly. For example, if you plug in a new webcam or a USB pendrive into your machine, a new device entry will automatically pop up here.

/etc

/etc is the directory where names start to get confusing. */etc* gets its name from the earliest Unixes and it was literally "et cetera" because it was the dumping ground for system files administrators were not sure where else to put.

Nowadays, it would be more appropriate to say that *etc* stands for "Everything to configure," as it contains most, if not all system-wide configuration files. For example, the files that contain the name of your system, the users and their passwords, the names of machines on your network and when and where the partitions on your hard disks should be mounted are all in here. Again, if you are new to Linux, it may be best if you don't touch too much in here until you have a better understanding of how things work.

/home

/home is where you will find your users' personal directories. In my case, under */home*

there are two directories: */home/paul*, which contains all my stuff; and */home/guest*, in case anybody needs to borrow my computer.

/lib

/lib is where *libraries* live. Libraries are files containing code that your applications can use. They contain snippets of code that applications use to draw windows on your desktop, control peripherals, or send files to your hard disk.

There are more *lib* directories scattered around the file system, but this one, the one hanging directly off of */* is special in that, among other things, it contains the all-important kernel modules. The kernel modules are drivers that make things like your video card, sound card, WiFi, printer, and so on, work.

/media

The */media* directory is where external storage will be automatically mounted when you plug it in and try to access it. As opposed to most of the other items on this list, */media* does not hail back to 1970s, mainly because inserting and detecting storage (pendrives, USB hard disks, SD cards, external SSDs, etc) on the fly, while a computer is running, is a relatively new thing.

/mnt

The */mnt* directory, however, is a bit of remnant from days gone by. This is where you would manually mount storage devices or partitions. It is not used very often nowadays.

/opt

The */opt* directory is often where software you compile (that is, you build yourself from source code and do not install from your distribution repositories) sometimes lands. Applications will end up in the */opt/bin* directory and libraries in the */opt/lib* directory.

A slight digression: another place where applications and libraries end up in is */usr/local*. When software gets installed here, there will also be */usr/local/bin* and */usr/local/lib* directories. What determines which software goes where is how the developers have configured the files that control the compilation and installation process.

/proc

/proc, like */dev* is a virtual directory. It contains information about your computer, such as information about your CPU and the kernel your Linux system is running. As with

/dev, the files and directories are generated when your computer starts, or on the fly, as your system is running and things change.

/root

/root is the home directory of the superuser (also known as the “Administrator”) of the system. It is separate from the rest of the users’ home directories BECAUSE YOU ARE NOT MEANT TO TOUCH IT. Keep your own stuff in you own directories, people.

/run

/run is another new directory. System processes use it to store temporary data for their own nefarious reasons.

/sbin

/sbin is similar to */bin*, but it contains applications that only the superuser (hence the initial *s*) will need. You can use these applications with the `sudo` command that temporarily concedes you superuser powers on many distributions. */sbin* typically contains tools that can install stuff, delete stuff and format stuff. As you can imagine, some of these instructions are lethal if you use them improperly, so handle with care.

/usr

The */usr* directory was where users’ home directories were originally kept back in the early days of UNIX. However, now */home* is where users kept their stuff as we saw above. These days, */usr* contains a mish-mash of directories which in turn contain applications, libraries, documentation, wallpapers, icons and a long list of other stuff that need to be shared by applications and services.

You will also find *bin*, *sbin* and *lib* directories in */usr*. What is the difference with their root-hanging cousins? Not much nowadays. Originally, the */bin* directory (hanging off of root) would contain very basic commands, like `ls`, `mv` and `rm`; the kind of commands that would come pre-installed in all UNIX/Linux installations, the bare minimum to run and maintain a system. */usr/bin* on the other hand would contain stuff the users would install and run to use the system as a work station, things like word processors, web browsers, and other apps.

But many modern Linux distributions just put everything into */usr/bin* and have */bin* point to */usr/bin* just in case erasing it completely would break something. So, while Debian, Ubuntu and Mint still keep */bin* and */usr/bin* (and */sbin* and */usr/sbin*) separate; others, like Arch and its derivatives just have one “real” directory for binaries, */usr/bin*, and the rest or **bins* are “fake” directories that point to */usr/bin*.

/srv

The */srv* directory contains data for servers. If you are running a web server from your

Linux box, your HTML files for your sites would go into `/srv/http` (or `/srv/www`). If you were running an FTP server, your files would go into `/srv/ftp`.

`/sys`

`/sys` is another virtual directory like `/proc` and `/dev` and also contains information from devices connected to your computer.

In some cases you can also manipulate those devices. I can, for example, change the brightness of the screen of my laptop by modifying the value stored in the `/sys/devices/pci0000:00/0000:00:02.0/drm/card1/card1-eDP-`

`l/intel_backlight/brightness` file (on

your machine you will probably have a different file). But to do that you have to become superuser. The reason for that is, as with so many other virtual directories, messing with the contents and files in `/sys` can be dangerous and you can trash your system. DO NOT TOUCH until you are sure you know what you are doing.

`/tmp`

`/tmp` contains temporary files, usually placed there by applications that you are running. The files and directories often (not always) contain data that an application doesn't need right now, but may need later on.

You can also use `/tmp` to store your own temporary files — `/tmp` is one of the few directories hanging off `/` that you can actually interact with without becoming superuser.

`/var`

`/var` was originally given its name because its contents was deemed *variable*, in that it changed frequently. Today it is a bit of a misnomer because there are many other directories that also contain data that changes frequently, especially the virtual directories we saw above.

Be that as it may, `/var` contains things like logs in the `/var/log` subdirectories. Logs are files that register events that happen on the system. If something fails in the kernel, it will be logged in a file in `/var/log`; if someone tries to break into your computer from outside, your firewall will also

log the attempt here. It also contains *spools* for tasks. These “tasks” can be the jobs you send to a shared printer when you have to wait because another user is printing a long document, or mail that is waiting to be delivered to users on the system.

Your system may have some more directories we haven't mentioned above. In the screenshot, for example, there is a `/snap` directory. That's because the shot was captured on an Ubuntu system. Ubuntu has recently incorporated [snap](#) packages as a way of distributing software. The `/snap` directory contains all the files and the software

installed from snaps.

/var/log/syslog contains lot of system related logfiles.

That is the root directory covered, but many of the subdirectories lead to their own set of files and subdirectories. Figure 2 gives you an overall idea of what the basic file system tree looks like (the image is kindly supplied under a CC By-SA license by Paul Gardner) and [Wikipedia has a break down with a summary of what each directory is used for](#).

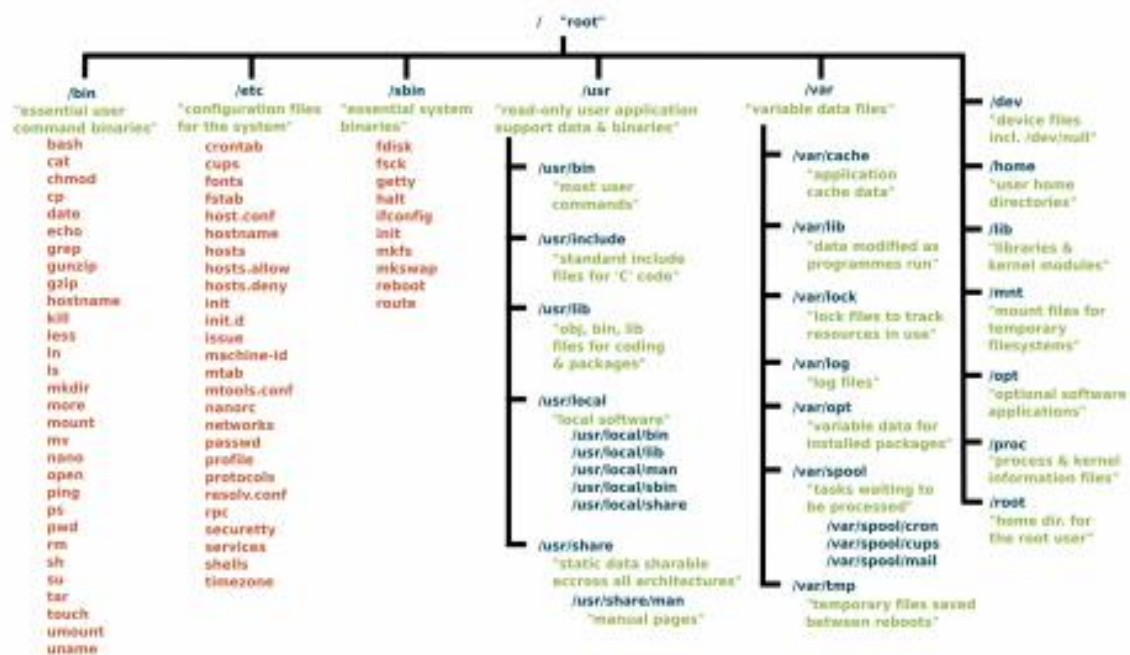


Figure 2: Standard Unix filesystem hierarchy.

To explore the filesystem yourself, use the cd command:

`cd`

will take you to the directory of your choice (`cd` stands for *change directory*. If you get confused,

`pwd`

will always tell you where you (`pwd` stands for *print working directory*). Also, `cd` with no options or parameters, will take you back to your own home directory, where things are safe and cosy.

Finally,

`cd ..`

will take you up one level, getting you one level closer to the /root directory. If you are in */usr/share/wallpapers* and run `cd ..`, you will move up to */usr/share*. To see what a directory contains, use

`ls`: to list the contents of the directory you are in right now.

And, of course, you always have `tree` to get an overview of what lays within a directory. Try it on */usr/share* — there is a lot of interesting stuff in there.

Although there are minor differences between Linux distributions, the layout for their filesystems are mercifully similar. So much so that you could say: once you know one, you know them all. And the best way to know the filesystem is to explore it. So go forth with `tree`, `ls`, and `cd` into uncharted territory.

You cannot damage your filesystem just by looking at it, so move from one directory to another and take a look around

Material adapted from <https://www.linux.com/training-tutorials/linux-file-system-explained/>

MCA Department, TKMCE

Cycle 3

MCA Department, TKMCE

13.Familiarization With Linux, Shell, Shell Scripting

Shell Scripting is an open-source computer program designed to be run by the Unix/Linux shell. Shell Scripting is a program to write a series of commands for the shell to execute. It can combine lengthy and repetitive sequences of commands into a single and simple script that can be stored and executed anytime which, reduces programming efforts.

What is shell?

Shell is a UNIX term for an interface between a user and an operating system service. Shell provides users with an interface and accepts human-readable commands into the system and executes those commands which can run automatically and give the program's output in a shell script.

An Operating is made of many components, but its two prime components are -

- Kernel
- Shell

A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one.

A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it.

Types of Shell

There are two main shells in Linux:

1. The Bourne Shell: The prompt for this shell is \$ and its derivatives are listed below:

- POSIX shell also is known as sh
- Korn Shell also knew as sh
- Bourne Again SHell also knew as bash (most popular)

2. The C shell: The prompt for this shell is %, and its subcategories are:

- C shell also is known as csh
- Tops C shell also is known as tcsh

Creating a shell script

Shell Scripts are written using text editors. On your Linux system, open a text editor program, open a new file to begin typing a shell script or shell programming, then give the shell permission to execute your shell script and put your script at the location from where the shell can find it.

Let us understand the steps in creating a Shell Script:

1. Create a file using a vi editor(or any other editor). Name script file with extension .sh
2. Start the script with `#!/bin/sh`
3. Write some code.
4. Save the script file as filename.sh
5. For executing the script type `bash filename.sh`

"#!" is an operator called shebang which directs the script to the interpreter location. So, if we use "`#!/bin/sh`" the script gets directed to the bourne-shell.

Let's create a small script -

`#!/bin/sh`

ls

Adding shell comments

Commenting is important in any program. In Shell programming, the syntax to add a comment is ,

`#comment`

Shell variables

Variables store data in the form of characters and numbers. Similarly, Shell variables are used to store information and they can be used by the shell only.

Two types of variables

- System variable- Created and maintained by linux itself. This type of variable defined in capital letters.
- User Defined Variables(UDV)- Created and maintained by user. This type of variable defined in small letters.

Read statement

Use to get input (data from user) from keyboard and store (data) to variable.

Syntax:

- `read variable1, variable2,...variableN`

Conditional statements

if condition

- Used for making the decisions in shell script
- If the condition is true , Command1 is executed.

Syntax:

if condition

then

command1 if condition is true or if exit status of condition is 0

fi

if else fi

- If the condition is true
- Command 1 is executed
- Otherwise Command 2 is executed
- fi represents loop termination.

Syntax

if [expression]

then

Statement1

else

Statement2

fi

if elif else fi

- To use multiple conditions in one if-else block, then elif keyword is used in shell.

Syntax

if [expression 1]

then

Statement1

Statement2

elif [expression2]

then

Statement3

Statement4

else

Statement5

fi

- If expression1 is true then it executes statement 1 and 2, and this process continues.
- If none of the condition is true then it processes else part.

Loops in Shell scripting

Bash supports two types of loop

- for loop
- while loop

Note that each and every loop

- a. First the variable used in loop condition must be initialized, then execution of the loop begins.
- b. A test(condition) is made at the beginning of each iteration.
- c. The body of the loop ends with a statement that modifies the value of the test(condition) variable.

for loop

Syntax

```
for ( exp1;exp2;exp3)
```

```
do
```

```
    repeat all statements between do and done until exp2 is true
```

```
done
```

While loop

Syntax

```
while [ condition ]
```

```
do
```

```
    command1
```

```
    command2
```

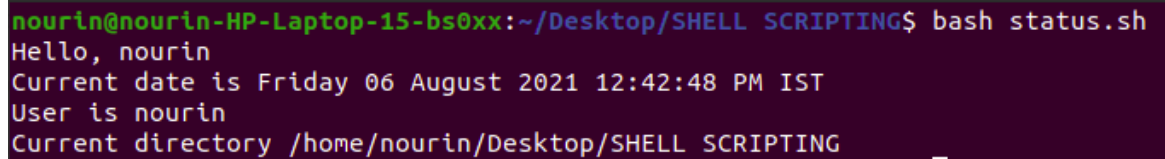
```
    ...
```

```
done
```

14.Linux Shell Scripting Problems

1. Write a shell script to get current date, time, username and current working?

```
#!/bin/bash
echo "Hello, $LOGNAME"
echo "Current date is $(date)"
echo "User is $(whoami)"
echo "Current directory $(pwd)"
```



```
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash status.sh
Hello, nourin
Current date is Friday 06 August 2021 12:42:48 PM IST
User is nourin
Current directory /home/nourin/Desktop/SHELL SCRIPTING
```

2. How to print the login names of all users on a system?

There are two options

Open the **etc/passwd** file by typing the command:

```
cat etc/passwd
```

Alternatively you can use the less command

```
less etc/passwd
```

```
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

3. How can we pass arguments to a script in Linux? And how to access these arguments from within the script?

We can write a bash script that can accept arguments from the command line in the following manner.

```
$ ./scriptName "arg1" "arg2"..."argn"

#!/bin/bash
#Call this script with at least 3parameters
echo "First parameter is $1"
echo "Second parameter is $2"
echo "Third parameter is $3"
```

```
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash parameters.sh 20 15 41
"First parameter is 20"
"Second parameter is 15"
"Third parameter is 41"
```

4.How to set an array in Linux?

1.IndirectDeclaration

In Indirect declaration, We assigned a value in a particular index of Array Variable. Noneed to first declare.

```
ARRAY NAME[INDEXNR]=value
```

2.ExplicitDeclaration

In Explicit Declaration, First We declare array then assigned the values.

```
declare -a ARRAYNAME
```

3.CompoundAssignment

In Compound Assignment, We declare array with a bunch of values. We can add other values later too.

```
ARRAYNAME=(value1 value2 .... valueN)
```

5.How to check if a directory exists?

```
if [ -d "/path/to/dir" ]  
then  
    echo "Directory /path/to/dir exists."  
else  
    echo "Error: Directory /path/to/dir does not exists."
```

6.What is the difference between \$* and \$@?

\$@ treats each quoted arguments as separate arguments but \$* considers

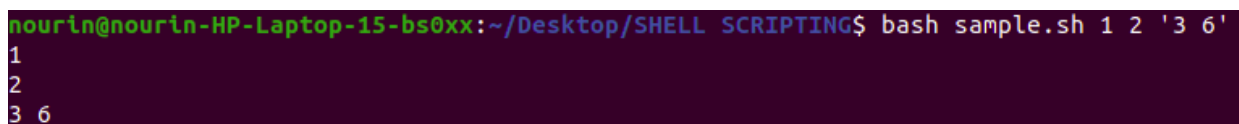
the entire set of positional parameters as a single string.

```
for var in "$@"
```

```
do
```

```
    echo "$var"
```

```
Done
```



```
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash sample.sh 1 2 '3 6'  
1  
2  
3 6
```

for var in \$*

```
do
```

```
    echo "$var"
```

```
Done
```

```
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash sample.sh 1 2 '3 4'
1
2
3
4
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$
```

7. Use the sed command to replace the content of the file?

```
sed -i 's/old-text/new-text/g' input.txt8
```

8. Write a script to compare numbers?

```
#!/bin/bash
# Script to do numeric comparisons
var1=10
var2=20
if [ $var2 -gt $var1 ]
then
    echo "$var2 is greater than $var1"
fi
# Second comparison
If [ $var1 -gt 30]then
echo "$var is greater than 30"
else
    echo "$var1 is less than 30"
Fi
```

```
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash compar.sh
20 is greater than 10
10 is less than 30
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$
```

9. Write a shell script to check to see if the file “file_path” exists. If it does exist, display “file_path passwords are enabled.” Next, check to see if you can write to the file. If you can, display “You have permissions to edit “file_path.””If you cannot, display “You do NOT have permissions to edit“file_path””?

```
#!/bin/bash
FILE="/home/svimukthi/Assignment/sanka"
if [ -e "$FILE" ]
then
    echo "$FILE passwords are enabled"
```

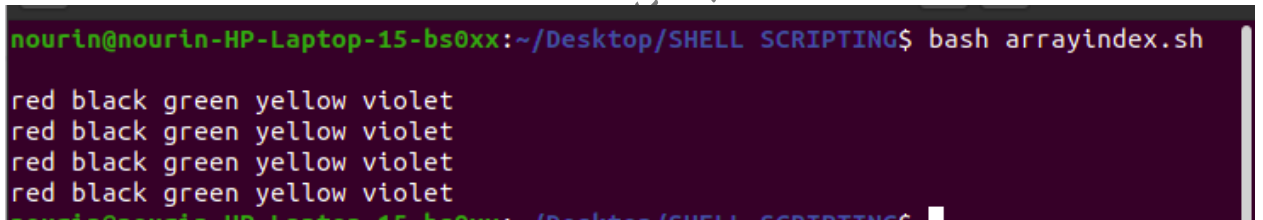
```
fi

if [ -x "$FILE" ]
then
    echo "You have permission to execute $FILE" else
    echo "You do Not have permissions to execute $FILE"
fi
```

10.How to print all array indexes?

```
echo ${ARRAYNAME[*]}
#!/bin/bash
# To declare static Array
arr=(red black green yellow violet)

# To print all elements of array
echo ${arr[@]}
echo ${arr[*]}
echo ${arr[@]:0}
echo ${arr[*]:0}
```



```
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash arrayindex.sh
red black green yellow violet
red black green yellow violet
red black green yellow violet
red black green yellow violet
```

11.Write a shell script to display the last updated file or the newest file in a directory?

```
#!/bin/bash

ls -lrt | grep ^- | awk 'END{print $NF}'

nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash test.sh
test.sh
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$
```

12.Write a shell script that adds an extension “.new” to all the files in directory.

running the test script.

```
#!/bin/bash
dir=$1
for file in `ls $1/*`
do
```

```
mv $file $file.new
done
```

13. Write a shell script to print a number in reverse order. It should support

- a. the following requirements.
- b. The script should accept the input from the command line.
- c. If you don't input any data, then display an error message to execute the script correctly

```
#!/bin/bash

if [ $# -ne 1 ]
then
    echo "Please provide the correct input in the below format."
    echo "Usage: $0 number"
    echo "    This script will reverse the given number."
    echo "    For eg. $0 1234, will print 4321"
    exit 1
fi

n=$1
rev=0
sd=0

while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
    rev=`expr $rev \* 10 + $sd`
    n=`expr $n / 10`
done
echo "Reverse number is $rev"
```

14. If you don't input any data, then display an error message to execute the script correctly

In Linux or Unix-like system, you may come across file names including the following special characters, white spaces, backslashes and more.

```
-
--
;
&
$
```

?
*

Bash shell considers most of the above special characters as commands. Thus, the “rm” command may not be able to delete such files. The simplest way to delete files having special characters in its name is by using the inode number

15. Write a shell script to find out the unique words in a file and also count the occurrence of each of these words. We can say that the file under consideration contains many lines, and each line has multiple words

```
$ cat animal.txt
```

```
tiger bear
elephant tiger bear
bear
```

Following test script/command will count the unique words.

```
$ awk '{for(i=1;i<=NF;i++)a[$i]++;}END{for(i in a){print i,
a[i];}}' animal.txt
```

16. Write a script to print the first 10 elements of Fibonacci series.

```
# Program for Fibonacci

# Series

# Static input for N
N=6

# First Number of the
# Fibonacci Series
a=0

# Second Number of the
# Fibonacci Series
b=1

echo "The Fibonacci series is : "

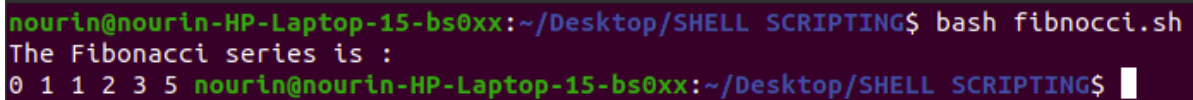
for (( i=0; i<N; i++ ))
do
    echo -n "$a "
    fn=$((a + b))
    a=$b
```



```

b=$fn
done
# End of for loop

```



```

nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash fibnocci.sh
The Fibonacci series is :
0 1 1 2 3 5 nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$

```

17. Write a shell script to get the total count of the word “Linux” in all the “.txt” files and also across files present in subdirectories.

The following is the test script/command which recursively searches for the “.txt” files and returns the total occurrences of a word <Linux>.

```

$ find . -name *.txt -exec grep -c 'Linux' '{}' \; | awk
'{x+=$0;}END{print x}'

```

18. Write a shell script to validate password strength. Here are a few assumptions for the password string.

1. Length – minimum of 8 characters.
2. Contain both alphabet and number.
3. Include both the small and capital case letters.

If the password doesn’t comply with any of the above conditions, then the script should report it as a <Weak Password>

If the password doesn’t comply with any of the above conditions, then the script should report it as a <Weak Password>

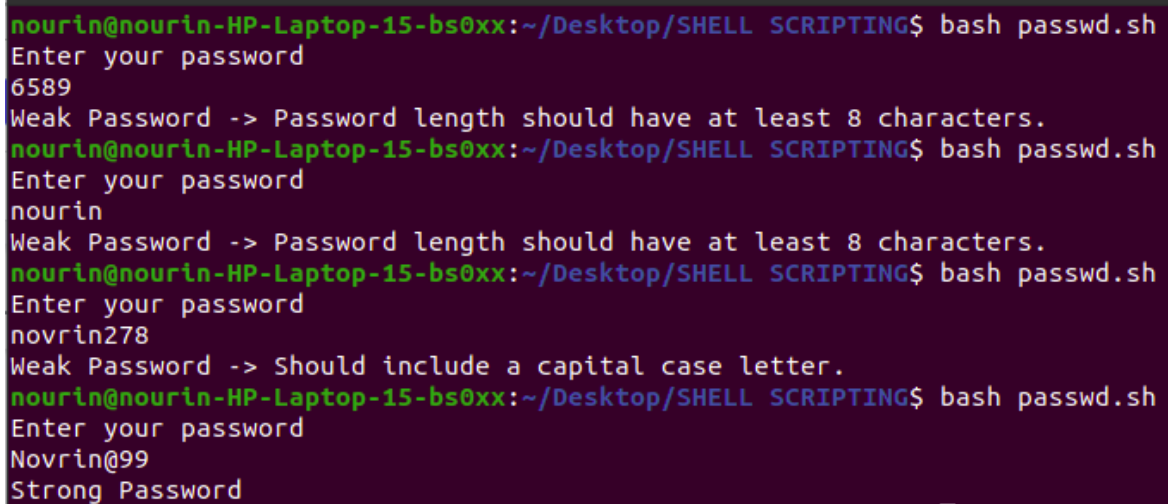
Password Validation Script

```

echo "Enter your password"
read password
len="${#password}"
if test $len -ge 8 ; then
    echo "$password" | grep -q [0-9]
    if test $? -eq 0 ; then
        echo "$password" | grep -q [A-Z]
        if test $? -eq 0 ; then
            echo "$password" | grep -q [a-z]
            if test $? -eq 0 ; then
                echo "Strong Password"
            fi
        fi
    fi
fi

```

```
        else
            echo "Weak Password -> Should include a lower case letter."
        fi
    else
        echo "Weak Password -> Should include a capital case letter."
    fi
else
    echo "Weak Password -> Should use numbers in your password."
fi
else
    echo "Weak Password -> Password length should have at least 8
characters."
fi
```



The screenshot shows a terminal window with a dark background. The prompt is `nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$`. The user runs `bash passwd.sh`. The script prompts for a password, and the user enters `6589`. The script outputs `Weak Password -> Password length should have at least 8 characters.`. The user runs `bash passwd.sh` again, enters `nourin`, and receives the same weak password message. The user runs `bash passwd.sh` a third time, enters `novrin278`, and receives the same weak password message. The user runs `bash passwd.sh` a fourth time, enters `Novrin@99`, and the script outputs `Strong Password`. A large, light-colored watermark "MCA D" is visible diagonally across the bottom half of the terminal window.

```
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash passwd.sh
Enter your password
6589
Weak Password -> Password length should have at least 8 characters.
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash passwd.sh
Enter your password
nourin
Weak Password -> Password length should have at least 8 characters.
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash passwd.sh
Enter your password
novrin278
Weak Password -> Should include a capital case letter.
nourin@nourin-HP-Laptop-15-bs0xx:~/Desktop/SHELL SCRIPTING$ bash passwd.sh
Enter your password
Novrin@99
Strong Password
```

Cycle 4

MCA Department, TKMCE

15.Installation Of Lamp Stack

Introduction

A “LAMP” stack is a group of open-source software that is typically installed together to enable a server to host dynamic websites and web apps. This term is actually an acronym which represents the **L**inux operating system, with the **A**pache web server. The site data is stored in a **M**ySQL database, and dynamic content is processed by **P**HP.

Step 1 — Installing Apache and Updating the Firewall

The Apache web server is a popular open source web server that can be used along with PHP to host dynamic websites. It’s well-documented and has been in wide use for much of the history of the web.

First, make sure your apt cache is updated with:

```
$ sudo apt update
```

Once the cache has been updated, you can install Apache with:

```
$ sudo apt install apache2
```

After entering this command, apt will tell you which packages it plans to install and how much extra disk space they’ll take up. Press **Y** and hit **ENTER** to confirm, and the installation will proceed.

Adjust the Firewall to Allow Web Traffic

Next, assuming that you have followed the initial server setup instructions and enabled the UFW firewall, make sure that your firewall allows HTTP and HTTPS traffic. You can check that UFW has an application profile for Apache like so:

```
$ sudo ufw app list
```

If you look at the Apache Full profile details, you’ll see that it enables traffic to ports 80 and 443:

```
$ sudo ufw app info “Apache Full”
```

To allow incoming HTTP and HTTPS traffic for this server, run:

```
$ sudo ufw allow "Apache Full"
```

Step 2- Installing MySQL

Now that you have your web server up and running, it is time to install MySQL. MySQL is a database management system. Basically, it will organize and provide access to databases where your site can store information.

Again, use apt to acquire and install this software:

```
$ sudo apt install mysql-server
```

This command, too, will show you a list of the packages that will be installed, along with the amount of disk space they'll take up. Enter Y to continue.

When the installation is complete, run a simple security script that comes pre-installed with MySQL which will remove some dangerous defaults and lock down access to your database system. Start the interactive script by running:

```
$ sudo mysql_secure_installation
```

This will ask if you want to configure the **VALIDATE PASSWORD PLUGIN**

Answer **Y** for yes, or anything else to continue without enabling.

If you answer “yes”, you’ll be asked to select a level of password validation. Keep in mind that if you enter **2** for the strongest level, you will receive errors when attempting to set any password which does not contain numbers, upper and lowercase letters, and special characters, or which is based on common dictionary words.

When you’re finished, test if you’re able to log in to the MySQL console by typing:

```
$ sudo mysql
```

This will connect to the MySQL server as the administrative database user root, which is inferred by the use of **sudo** when running this command. You should see output like this:

To exit the MySQL console, type:

```
mysql> exit
```

Step3 - Installing PHP

Once again, leverage the apt system to install PHP. In addition to the php package, you’ll also need libapache2-mod-php to integrate PHP into Apache, and the php-mysql package to allow PHP to connect to MySQL databases. Run the following command to install all three packages and their dependencies:

```
$ sudo apt install php libapache2-mod-php php-mysql
```

This should install PHP without any problems. We’ll test this in a moment.

Step 4 — Testing PHP Processing on your Web Server

In order to test that your system is properly configured for PHP, create a PHP script called info.php. In order for Apache to find this file and serve it correctly, it must be saved to your web root directory.

Create the file at the web root you created in the previous step by running:

```
$ sudo nano /var/www/your_domain/info.php
```

This will open a blank file. Add the following text, which is valid PHP code, inside the file:

```
<?php  
phpinfo();
```

When you are finished, save and close the file.

Now you can test whether your web server is able to correctly display content generated by this PHP script. To try this out, visit this page in your web browser. You'll need your server's public IP address or domain name again.

The address you will want to visit is: **http://your_domain/info.php**

16.Installation of Laravel

- **Laravel** is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. The web application thus designed is more structured and pragmatic.
- Laravel offers a rich set of functionalities which incorporates the basic features of PHP frameworks like CodeIgniter, Yii and other programming languages like Ruby on Rails. Laravel has a very rich set of features which will boost the speed of web development.
- If you are familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It saves a lot of time if you are planning to develop a website from scratch. Moreover, a website built in Laravel is secure and prevents several web attacks.

ADVANTAGES OF LARAVEL

- Laravel offers you the following advantages, when you are designing a web application based on it –
- The web application becomes more scalable, owing to the Laravel framework.
- Considerable time is saved in designing the web application, since Laravel reuses the components from other frameworks in developing web applications.
- It includes namespaces and interfaces, thus helping to organize and manage resources.

COMPOSER

- Composer is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you.

INSTALLATION

Step1

- Visit the following URL and download composer to install it on your system.
- <https://getcomposer.org/download/>
- Remember to set the path of the composer to the php.exe file in the php/xampp/C:

Step2

- After the Composer is installed, check the installation by typing the Composer command in the command prompt as shown in the following screenshot.

Step 3:

- Create a new directory anywhere in your system for your new Laravel project.
- After that, move to path where you have created the new directory and type the following command there to install Laravel
- `composer create-project --prefer-dist laravel/laravel folder_name`
- But the latest version (currently 8.x) just requires you to type
- `composer create-project laravel/laravel folder_name`

Step 4:

- The above command will install Laravel in the current directory.
- Start the Laravel service by executing the following command.

`php artisan serve`

Step 5:

- After executing the above command, you will see a screen as shown below –

Step 6

- Copy the URL underlined in gray in the above screenshot and open that URL in the browser. If you see the following screen, it implies Laravel has been installed successfully.

Step 7

- The following screen indicates the laravel framework has been successfully installed in your device.

Cycle 5

MCA Department, TUMCE

17. INTRODUCTION TO COMMAND LINE TOOL FOR NETWORKING

NETWORK COMMANDS

The operating system consists of various built-in, command-line networking utilities that are used for network troubleshooting.

1. Ping
2. Hostname
- 3.netstat
4. If Config
5. Nslookup
6. Traceroute
- 7.Route
- 8.IP Config

PING

PACKET INTERNET PROPER

Most widely used utility tool to troubleshoot. It sends packets of information to the user-defined source. If the packets are received, the destination device sends packets back. Ping can be used for two purposes. Network connection can be established, Speed of the connection. Ping is used to testing a network host capacity to interact with another host. Just enter the command. Ping, followed by the target host's name or IP address.

HOSTNAME

Hostname :displays the machine hostname

Hostname -f :displays the fully qualified host and domain name

Hostname -I :displays the ip address for the current machine

netstat

Netstat command displays various network related information such as network

connections, routing tables, interface statistics, multicast memberships etc.

netstat -a : To show both listening and non-listening sockets

netstat -at : To list all tcp ports.

netstat -au : To list all udp ports.

netstat -l : To list only the listening ports.

netstat -lt : To list only the listening tcp ports

netstat -lu : To list only the listening udp ports.

netstat -lx : To list only the listening UNIX ports.

netstat -s : To list the statistics for all ports

If config

The if config commands is used for displaying current network configuration information ,setting up an ip address , netmask or broadcast address to an network interface , creatingan alias for network interface , setting up hardware address and enable or disable network interface

ifconfig -a :This option is used to display all the interfaces available, even if they are down.

ifconfig -s : Display a short list, instead of details

ifconfig interface up :This option is used to activate the driver for the given interface

ifconfig interface down :This option is used to deactivate the driver for the given interface.

Nslookup

Nslookup (stands for “Name Server Lookup”) is a useful command for getting informationfrom DNS server. It is a network administration tool for querying the Domain NameSystem (DNS) to obtain domain name or IP address mapping or any other specific DNSrecord. It is also used to troubleshoot DNS related problems.

Traceroute

Traceroute command prints the route that a packet takes to reach the host. This command is useful when we want to know about the route and about all the hops that a packet takes.

Route

Route command in Linux is used when you want to work with the IP/kernel routing table. It

is mainly used to set up static routes to specific hosts or networks via an interface. It is used for showing or update the IP/kernel routing table. Install net-tools by \$sudo apt-get install net-tools.

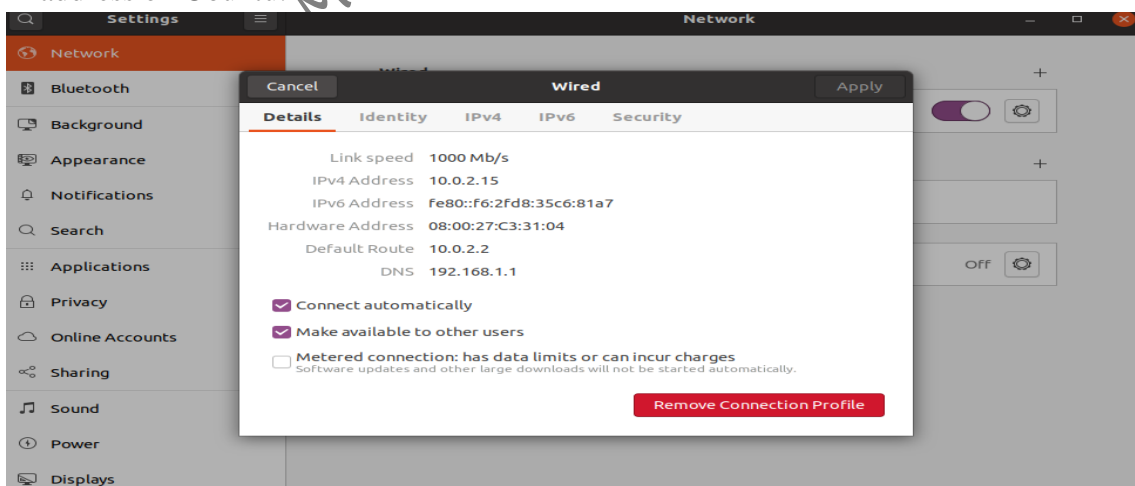
18.FAMILIARISATION OF STATIC AND DYNAMIC IP

Static ip

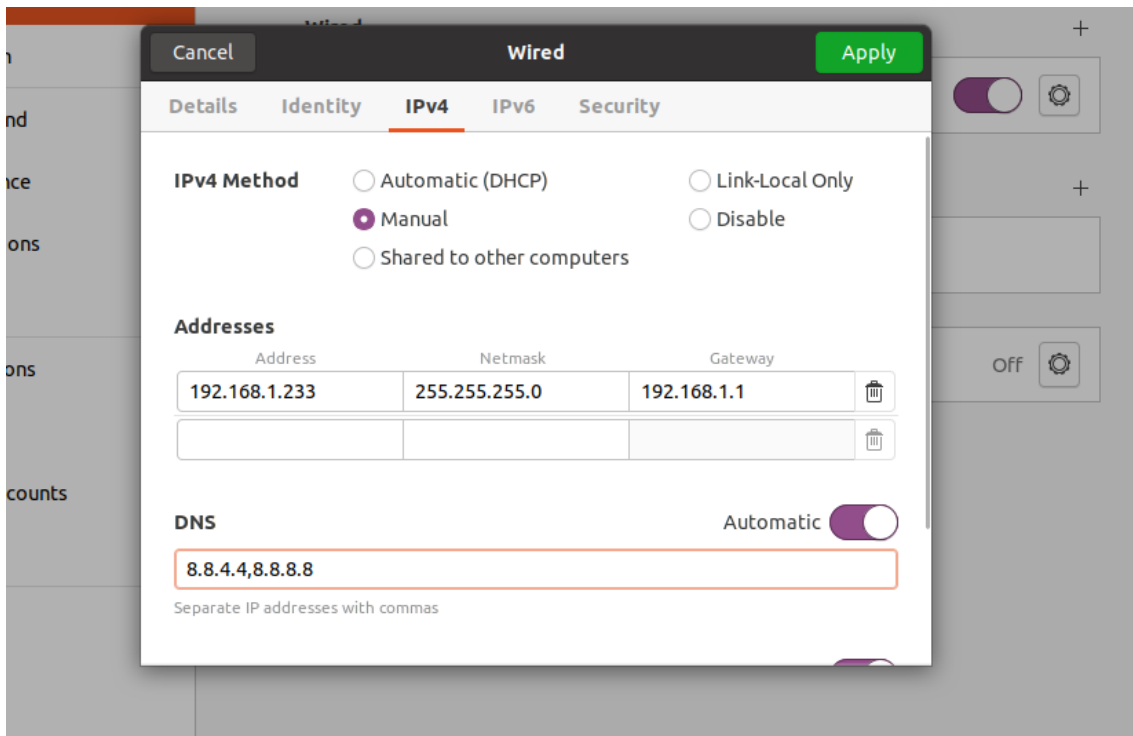
A fixed IP address is called static IP address , i.e. it never changes. It is required to set up an Ubuntu static IP address in order to access a device remotely and without losing a connection over the network. It is used to connect to an IP camera, home file server, game server, and many other devices. A static IP address is necessary only for servers and not for personal PCs

STEPS FOR SETTING STATIC IP

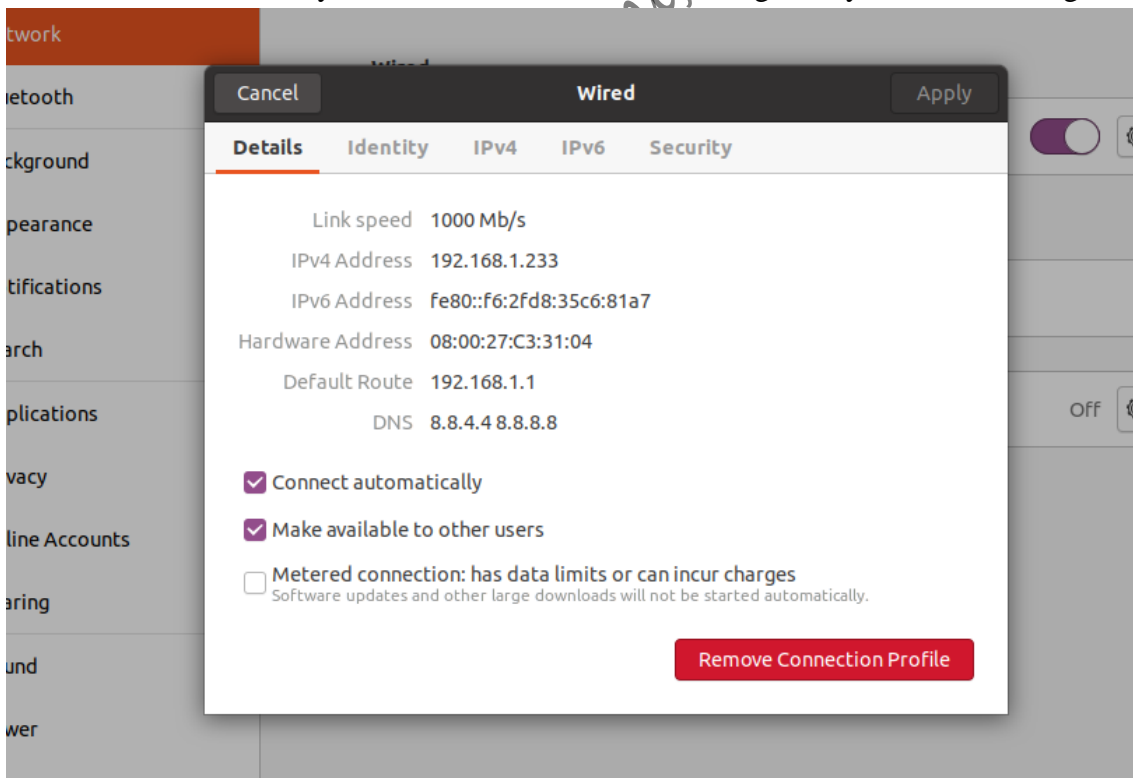
Click on the top right network icon and select settings of the network interface you wish to configure to use a static IP address on Ubuntu.



Click on the settings icon to start IP address configuration
Select IPv4 tab



Select manual and enter your desired IP address, netmask, gateway and DNS settings.



Once ready click Apply button.

Turn OFF and ON switch to apply your new network static IP configuration settings.

Click on the network settings icon once again to confirm your new static IP address settings.

DYNAMIC IP

A dynamic IP address as its name suggests is a temporary IP address assigned by a DHCP server for every new network.

A dynamic IP address is used due to the shortage of IP addresses on IPV4.

A single dynamic IP address can be used between many devices

Configuring a dynamic ip address

Step1: type the command in the terminal

```
sudo nano /etc/netplan/01-network-manager-all.yaml
```

step2: Now find the name of the network interface you want to configure and insert the following lines:

```
dhcp4: yes
```

```
dhcp6: yes
```

Step3: Apply the changes with sudo netplan apply command

19. Concept Of Subnets And Cidr Address Scheme

Subnets:

The process of dividing a network into smaller network sections is called subnetting. This can be useful for many different purposes and helps isolate groups of hosts from each other to deal with them more easily. By default, each network has only one subnet, which contains all of the host addresses defined within. A netmask is basically a specification of the amount of address bits that are used for the network portion. A subnet mask is another netmask within used to further divide the network.

Each bit of the address that is considered significant for describing the network should be represented as a “1” in the netmask. For instance, the address we discussed above, 192.168.0.15 can be expressed like this, in binary:

```
1100 0000 - 1010 1000 - 0000 0000 - 0000 1111
```

As we described above, the network portion for class C addresses is the first 3 octets, or the first 24 bits. Since these are the significant bits that we want to preserve, the netmask would be:

1111 1111 - 1111 1111 - 1111 1111 - 0000 0000

This can be written in the normal IPv4 format as 255.255.255.0. Any bit that is a “0” in the binary representation of the netmask is considered part of the host portion of the address and can be variable. The bits that are “1” are static, however, for the network or subnetwork that is being discussed. We determine the network portion of the address by applying a bitwise AND operation to between the address and the netmask. A bitwise AND operation will save the networking portion of the address and discard the host portion. The result of this on our above example that represents our network is:

1100 0000 - 1010 1000 - 0000 0000 - 0000 0000

This can be expressed as 192.168.0.0. The host specification is then the difference between these original value and the host portion. In our case, the host is 0000 1111 or 15. The idea of subnetting is to take a portion of the host space of an address, and use it as an additional networking specification to divide the address space again. For instance, a netmask of 255.255.255.0 as we saw above leaves us with 254 hosts in the network (you cannot end in 0 or 255 because these are reserved).

So, continuing with our example, the networking portion is: 1100 0000 - 1010 1000 - 0000 0000

The host portion is:

0000 1111

We can use the first bit of our host to designate a subnetwork. We can do this by adjusting the subnet mask from this:

1111 1111 - 1111 1111 - 1111 1111 - 0000 0000

To this:

1111 1111 - 1111 1111 - 1111 1111 - 1000 0000

In traditional IPv4 notation, this would be expressed as 192.168.0.128. What we have done here is to designate the first bit of the last octet as significant in addressing the network. This effectively produces two subnetworks. The first subnetwork is from 192.168.0.1 to 192.168.0.127. The second subnetwork contains the hosts 192.168.0.129 to 192.168.0.255.

CIDR Notation:

A system called Classless Inter-Domain Routing, or CIDR, was developed as an alternative to traditional subnetting. For example, we could express the idea that the IP

address 192.168.0.15 is associated with the netmask 255.255.255.0 by using the CIDR notation of 192.168.0.15/24. This means that the first 24 bits of the IP address given are considered significant for the network routing.

This allows us some interesting possibilities. We can use these to reference “supernets”. In this case, we mean a more inclusive address range that is not possible with a traditional subnet mask. For instance, in a class C network, like above, we could not combine the addresses from the networks 192.168.0.0 and 192.168.1.0 because the netmask for class C addresses is 255.255.255.0. However, using CIDR notation, we can combine these blocks by referencing this chunk as 192.168.0.0/23. This specifies that there are 23 bits used for the network portion that we are referring to. So the first network (192.168.0.0) could be represented like this in binary:

1100 0000 - 1010 1000 - 0000 0000 - 0000 0000

While the second network (192.168.1.0) would be like this:

1100 0000 - 1010 1000 - 0000 0001 - 0000 0000

The CIDR address we specified indicates that the first 23 bits are used for the network block we are referencing. This is equivalent to a netmask of 255.255.254.0, or:

1111 1111 - 1111 1111 - 1111 1110 - 0000 0000

As you can see, with this block the 24th bit can be either 0 or 1 and it will still match, because the network block only cares about the first 23 digits. CIDR allows us more control over addressing continuous blocks of IP addresses. This is much more useful than the subnetting we talked about originally.

20.CONCEPT OF SUBNET MASK

The subnet mask is used by the TCP/IP protocol to determine whether a host is on the local subnet or on a remote network. In TCP/IP, the parts of the IP address that are used as the network and host addresses aren't fixed. Unless you have more information, the network and host addresses above can't be determined. This information is supplied in another 32-bit number called a subnet mask. The subnet mask is 255.255.255.0 in this example. It isn't obvious what this number means unless you know 255 in binary notation equals 11111111. So, the subnet mask is 11111111.11111111.11111111.00000000.

Lining up the IP address and the subnet mask together, the network, and host portions of the address can be separated:

11000000.10101000.01111011.10000100 - IP address (192.168.123.132)

11111111.11111111.11111111.00000000 - Subnet mask (255.255.255.0)

The first 24 bits (the number of ones in the subnet mask) are identified as the network address. The last 8 bits (the number of remaining zeros in the subnet mask) are identified as the host address. It gives you the following addresses:

11000000.10101000.01111011.00000000 - Network address (192.168.123.0)

00000000.00000000.00000000.10000100 - Host address (000.000.000.132)

So now you know, for this example using a 255.255.255.0 subnet mask, that the network ID is 192.168.123.0, and the host address is 0.0.0.132. When a packet arrives on the 192.168.123.0 subnet (from the local subnet or a remote network), and it has a destination address of 192.168.123.132, your computer will receive it from the network and process it. Almost all decimal subnet masks convert to binary numbers that are all ones on the left and all zeros on the right.

Some other common subnet masks are:

Decimal	Binary
255.255.255.192	1111111.11111111.1111111.11000000
255.255.255.224	1111111.11111111.1111111.11100000

Internet RFC 1878 (available from InterNIC-Public Information Regarding Internet Domain Name Registration Services) describes the valid subnets and subnet masks that can be used on TCP/IP networks

21.Setting Up A Firewall For Lan

ufw-uncomplicated firewall

- The default firewall configuration tool for Ubuntu is ufw.
- Developed to ease iptables firewall configuration.
- ufw provides a user-friendly way to create an IPv4 or IPv6 host-based firewall.
- ufw by default is initially disabled.
- ufw is not intended to provide complete firewall functionality via its command interface, ut instead provides an easy way to add or remove simple rules.It is currently mainly used for host-based firewalls.

To ENABLE ufw

Step1:check current firewall status

sudo ufw status

Step2:to enable firewall

sudo ufw enable

Step3:to check status

sudo ufw status

To Disable ufw

Step1:To disable firewall

sudo ufw disable

Step2:to check status

sudo ufw status

22.Wireshark And Tcpdump

WIRESHARK

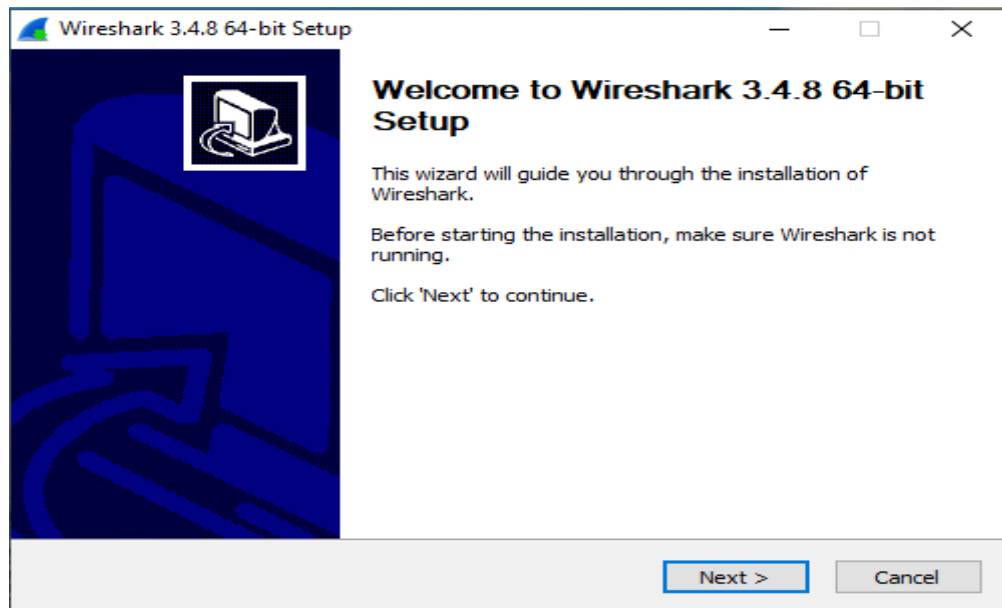
- Network packet protocol analyzer
- A network packet analyzer will try to capture network packets and try to display that packet data as detailed as possible.
- One of the best open source packet analyzers available today for UNIX and Windows

WHERE IT USE ?

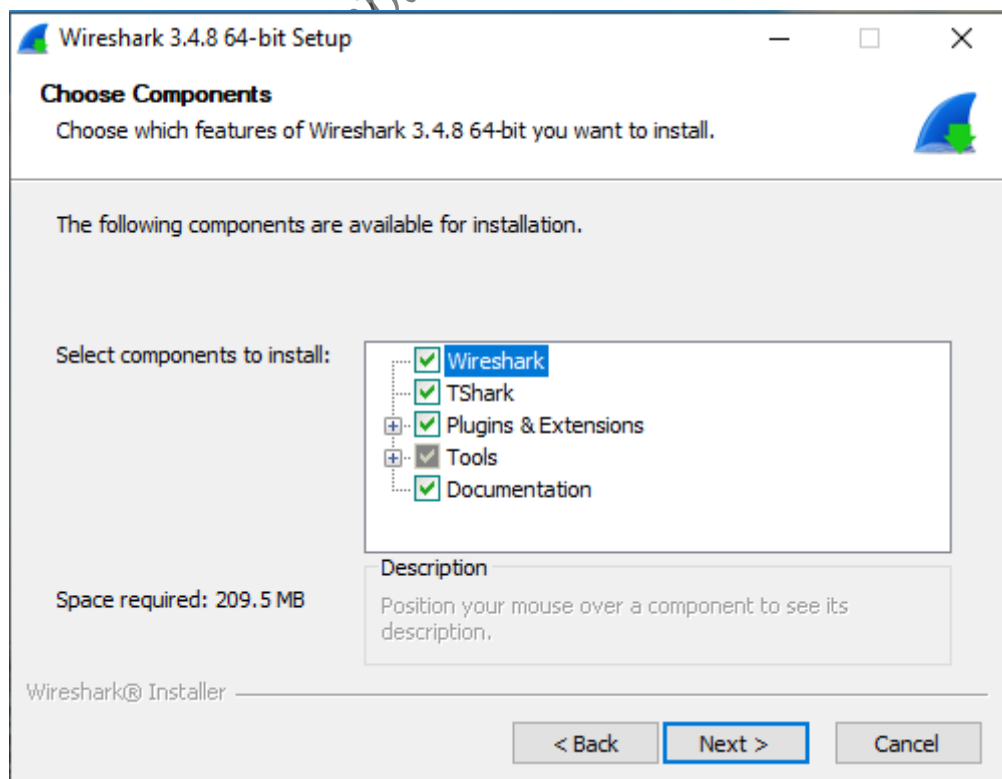
- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- Developers use it to debug protocol implementations
- Testers use it to detect defects
- People use it to learn network protocol internals

STEPS TO INSTALL WIRESHARK

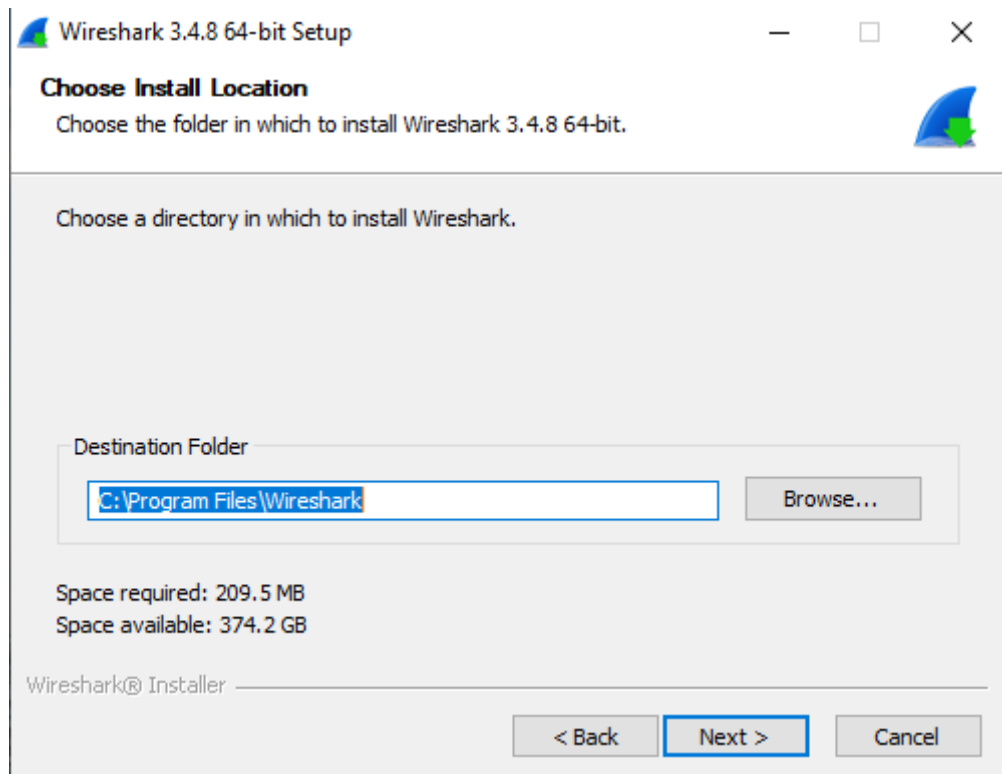
- Step1: Go to www.wireshark.org ->download 64bit package.
- Step 2 : run application and click on noted



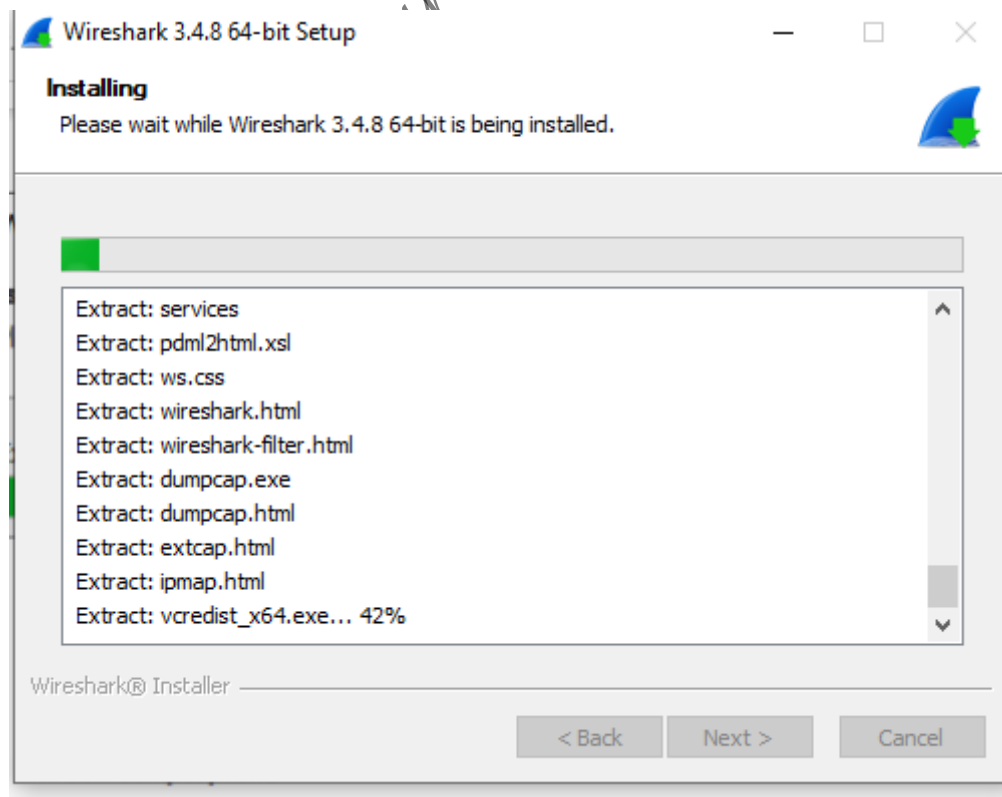
- Step 3 :select components and click next



- Step 4: choose default destination location

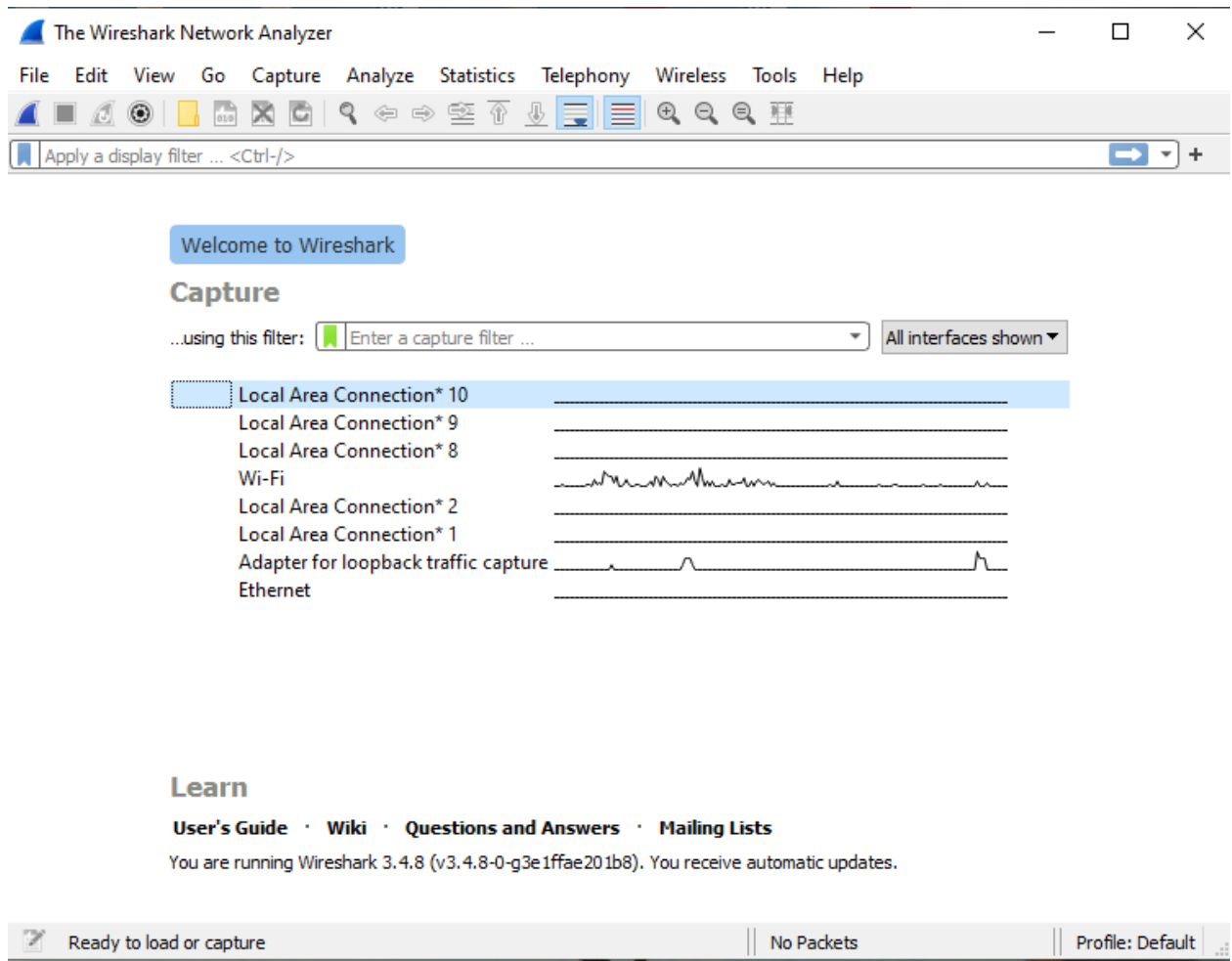


- Step 5: installation start running



- Step 6 : click on I agree
- Step 7 : complete installation part and click finish

HOMESCREEN



HOW IT CAPTURE PACKETS?

- Wireshark captures packets and lets you examine their contents
- Select any interface to capture its packet.
- No. shows the number of captured packet or index number.
- Time shows the time of capture
- Source shows the source ip of the packet or the packet is originally generated from which source ip.
- Destination shows the destination ip where the packet is going.
- Protocol shows which kind of protocol communication is held between the source and destination.
- Info shows the data payload in the packet

FEATURES OF WIRESHARK

- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Save packet data captured.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.

TCPDUMP

- It is an ip utility tool used for real-time packet sniffing(Network).
- Command line program comes in built in a Unix based system.
- Programs like ethereal(Wireshark) provide an alternative to Tcpcdump in GUI environment

STEPS TO INSTALL TCPDUMP

- Install tcpdump by entering the following commands in the terminal:

```
sudo apt update
```

```
sudo apt install tcpdump
```

tcpdump Command Examples

1. Display Available Interfaces

```
# tcpdump -D
```

2. Capture Packets from Specific Interface

```
# tcpdump -i any
```

- The command screen will scroll up until you interrupt and when we execute the tcpdump command it will captures from all the interfaces, however with -i switch only capture from the desired interface

3.Print Captured Packets in ASCII

- The below tcpdump command with the option -A displays the package in ASCII format. It is a character-encoding scheme format.
- # tcpdump -A -i any

4.Capture Only N Number of Packets

- When you run the tcpdump command it will capture all the packets for the specified interface, until you hit the cancel button. But using -c option, you can capture a specified number of packets.
- # tcpdump -c 5 -i any

5.Display Captured Packets in HEX and ASCII

- The following command with option -XX capture the data of each packet, including its link level header in HEX and ASCII format.
- # tcpdump -XX -i any

6.Capture and Save Packets in a File

- As we said, that tcpdump has a feature to capture and save the file in a .Pcap format, to do this just execute the command with -w option.
- # tcpdump -w 0001.Pcap -i any

Testing network services with Netcat [nc]

Use the netcat command, nc, to access the service. If you don't have nc installed, type the following command on the command line

Step 1: \$ sudo apt-get install netcat

Step 2: After the installation is done type 'nc -h'

Step3: Set up the server using netcat in listening mode.

We will use port 12345 and will specify the port number with -p option.

Step 4: Creating the server with netcat

- The command 'nc hostname port' puts netcat in client mode and connects to the specified hostname on the specified port. Open a new terminal window and type 'nc localhost 12345'

Step 5 : Now that we are connected to the server we can start chatting

23. Analyse Packets Using Wireshark

WIRESHARK LAB ASSIGNMENT

1. List 3 different protocols that appear in the protocol column in the unfiltered packet-listing window. Support your answer with an appropriate screenshot from your computer.
2. How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received? (By default, the value of the Time column in the packet-listing window is the amount of time, in seconds, since Wireshark tracing began. To display the Time field in time-of-day format, select the Wireshark View pull down menu, then select Time Display Format, then select Time-of-day.)
3. What is the Internet address of the gaia.cs.umass.edu? What is the Internet address of your computer? Support your answer with an appropriate screenshot from your computer.
4. Print the two HTTP messages (GET and OK) referred to in question 2 above. To do so, select Print from the Wireshark File command menu, and select the “Selected Packet Only” and “Print as displayed” radial buttons, and then click OK
5. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill of Rights?

Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request? What is the status code and phrase in the response?
6. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

Cycle 6

MCA Department, TKMCE

24.Familiarisation To Hypervisors And Virtual Machines

Virtual machine

- A virtual machine is a virtual representation, or emulation, of a physical computer. They are often referred to as a guest while the physical machine they run on is referred to as the host.
- Virtualization makes it possible to create multiple virtual machines, each with their own operating system (OS) and applications, on a single physical machine. A VM cannot interact directly with a physical computer. Instead, it needs a lightweight software layer called a hypervisor to coordinate between it and the underlying physical hardware.

Hypervisor

- Hypervisor is a software program that manages multiple operating systems (or multiple instances of the same operating system) on a single computer system.
- The hypervisor manages the system's processor, memory, and other resources to allocate what each operating system requires.
- Hypervisors are designed for a particular processor architecture and may also be called **virtualization managers**.

Hypervisor Types

- **Type 1: native (bare-metal) hypervisors**

The Hypervisor runs directly on the host's hardware to control the hardware and to manage guest operating systems.

E.g., Xen, VMWare ESXi, Microsoft Hyper-V

- **Type 2: hosted hypervisors**

These hypervisors run on a conventional operating system just as

other computer programs do. Eg. VMWare Workstation, VirtualBox

Benefits of hypervisor

- Speed
- Efficiency
- Flexibility
- Portability

Popular Hypervisors

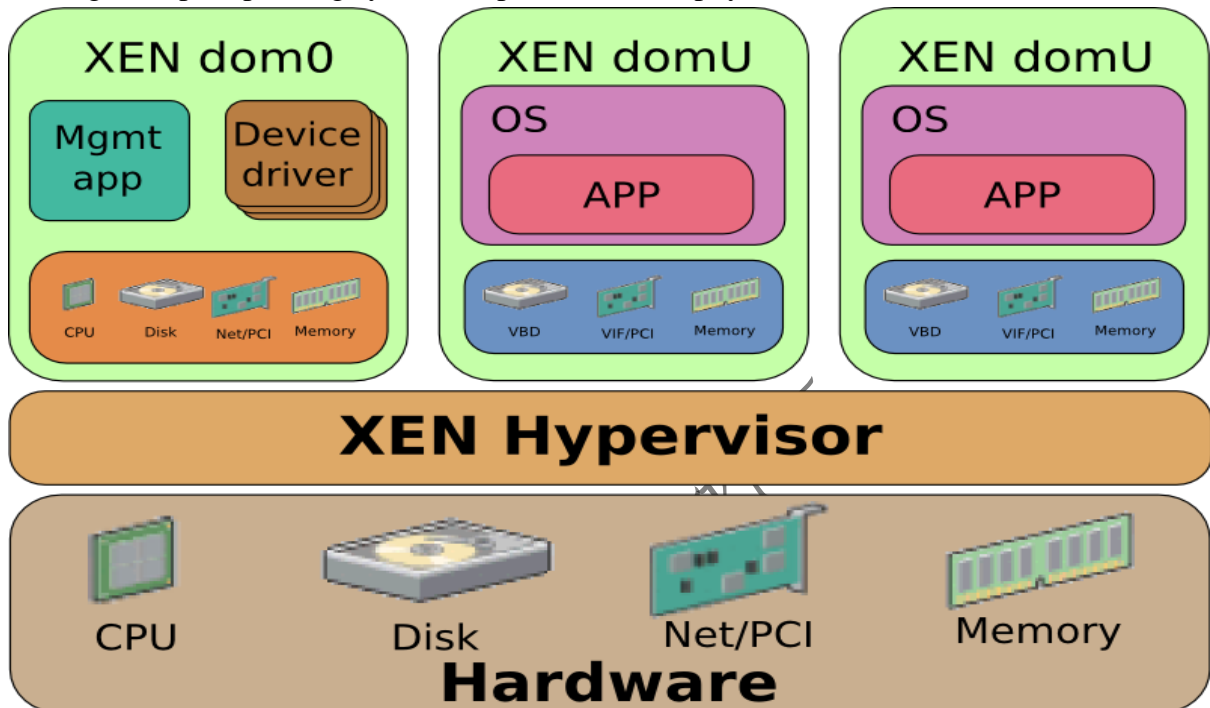
- Xen
- KVM(Kernel Based VM)

- SAN
- NAS

25.Familiarisation To Xen Or Kvm

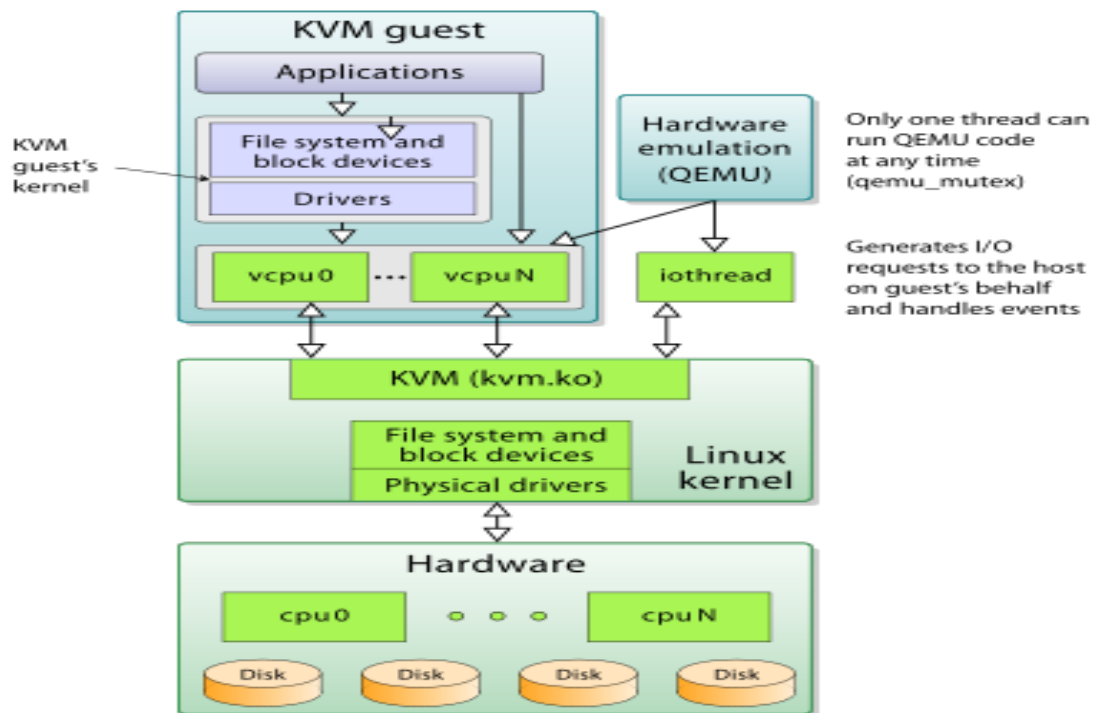
Xen

Xen is an open-source **paravirtualization** technology that provides a platform for running multiple operating systems in parallel on one physical hardware resource.



Kernel-based Virtual Machine (KVM)

- Kernel-based Virtual Machine (KVM) is an **open source virtualization** technology built into Linux. Specifically, KVM lets you turn Linux into a **hypervisor** that allows a host machine to run multiple, isolated virtual environments called guests or virtual machines (VMs).
- KVM converts Linux into a type-1 (bare-metal) hypervisor. All hypervisors need some operating system-level components—such as a memory manager, process scheduler, input/output (I/O) stack, device drivers, security manager, a network stack, and more—to run VMs.



DOCKER

- Docker is an open platform for developing, shipping, and running applications.
- Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
- With Docker, you can manage your infrastructure in the same ways you manage your applications.
- By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

What is a container?

- Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

What is a Docker image?

- A Docker image is a file used to execute code in a Docker container. Docker images act as a set of instructions to build a Docker container, like a template. Docker images also act as the starting point when using Docker. An image is comparable to a snapshot in virtual machine (VM) environments.

Docker installation

- `$ sudo apt install docker.io`
- **Version check**
`$ docker --version`
- **Check whether it is running or not**
`$ sudo systemctl status docker`
- **If not active**
`$ sudo systemctl enable --now docker`
- **List all the images you have locally**
`$ sudo docker images`
- **Pull an image from the Docker registry**
`$ sudo docker pull <image_name>:<tag>`
- **Run a docker**
`$ sudo docker run <image_name>:<tag>`
- **List all the running containers**
`$ sudo docker ps -a`

26.Installing Software From Source Code

- Source code software must be compiled and installed.
- Usually comes in a compressed archive, called a tarball with .tar or .tar.gz ending.
- Archive includes source, configure script, makefile, and install scripts.

Package Mangers

- Automate the installation, removal, and management of the software applications.
- Only track software installed using the package manger.
- Similar to Add/Remove programs control panel in MS Windows

Configure Script:

Inspects system for requirements and configures the “makefile” .

Make:

- Automates the compilation of programming source code for the target system.
- “makefile” define the necessary steps to build the application.
- They are far from perfect
- There is no central database to track applications installed with make.
- Removal of applications may or may not be supported by the make file.
- “makefile” contains installation parameters, variables, and setup instructions.
- “make” and “make install” commands are run to compile and install software.
- There is no central database to track applications installed with make.
- Removal of applications may or may not be supported by the make file.
- makefile” contains installation parameters, variables, and setup instructions.
- “make” and “make install” commands are run to compile and install software.

Make Command:

- Source code distributed as “gzipped tarballs”.
- After unpacking the code you must check the README file for specific install instructions.

\$ configure

\$ make

\$ make install

Installation Steps:

Step 1: Open the Linux terminal and enter

```
sudo apt update
```

Step 2: Enter

```
sudo apt install build-essential
```

Step 3: Enter

```
cd /usr/local/src/
```

Step 4: Enter

```
sudo wget http://www.noip.com/client/linux/noip-due-linux.tar.gz
```

Step 5: Enter

```
sudo tar xf noip-due-linux.tar.gz and cd noip-2.1.9-1/
```

Step 6: Enter

```
sudo make install
```

27.DEPLOY LINUX VIRTUAL MACHINE USING ANSIBLE PLAYBOOK

Ansible

- Ansible is simple open source IT engine which automates application deployment, intra service orchestration, cloud provisioning and many other IT tools.
- Ansible is easy to deploy because it does not use any agents or custom security infrastructure.
- Ansible uses playbook to describe automation jobs, and playbook uses very simple language

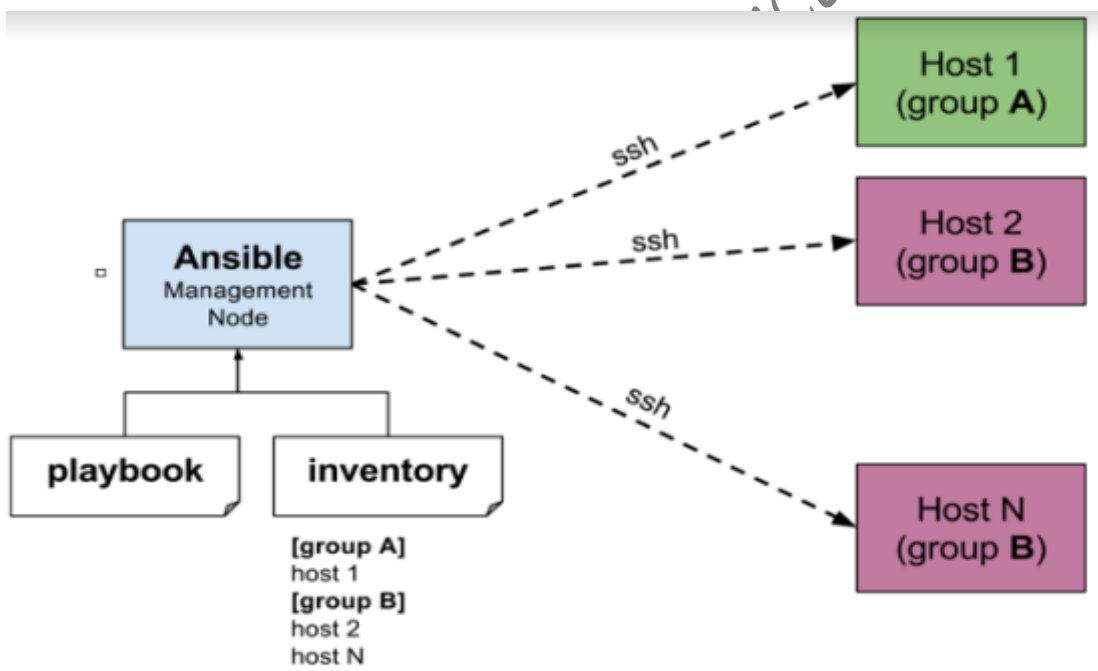
i.e. YAML (It's a human-readable data serialization language & is commonly used for configuration files, but could be used in many applications where data is being stored) which is very easy for humans to understand, read and write. Hence the

advantage is that even the IT infrastructure support guys can read and understand the playbook and debug if needed (YAML – It is in human readable form).

- Ansible is designed for multi-tier deployment. Ansible does not manage one system at time, it models IT infrastructure by describing all of your systems are interrelated. Ansible is completely agentless which means Ansible works by connecting your nodes through ssh(by default). But if you want other method for connection like Kerberos, Ansible gives that option to you.

How do Ansible playbooks work?

- Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. Ansible then executes these modules (over SSH by default), and removes them when finished. Your library of modules can reside on any machine, and there are no servers, daemons, or databases required.



Installation Process

- Ansible can be run from any machine with Python 2 (versions 2.6 or 2.7) or Python 3 (versions 3.5 and higher) installed.
- Ansible can be installed on control machine which have above mentioned requirements in different ways. You can install the latest release through Apt, yum, pkg, pip, OpenCSW, pacman, etc

Installation through Apt on Ubuntu Machine

- For installing Ansible you have to configure PPA on your machine. For this, you have to run the following line of code:

```
sudo apt-add-repository ppa:ansible/ansible
```

```
sudo apt update
```

```
sudo apt install ansible
```

Setting up Inventory File

```
ansible-inventory --list -y
```

Testing Connection

- After setting up the inventory file to include your servers, it's time to check if Ansible is able to connect to these servers and run commands via SSH.
- You can use the `-u` argument to specify the remote system user. When not provided, Ansible will try to connect as your current system user on the control node.
- `ansible all -m ping -u root`

PaaS (Platform-as-a-Service)

- Platform-as-a-service (PaaS) is a model of cloud service delivery where a third-party cloud service provider delivers some hardware and software tools, often those needed for application hosting or development, to customers over the internet. The key benefit of the PaaS model is that it enables users to access hardware and software that can be used to develop and run applications without having to purchase, install and maintain the infrastructure.
- Microsoft Azure, formerly known as Windows Azure, is Microsoft's public cloud computing platform. It provides a range of cloud services, including compute, analytics, storage and networking. Users can pick and choose from these services to develop and scale new applications, or run existing applications in the public cloud.

What is Microsoft Azure used for?

Microsoft Azure consists of numerous service offerings, its use cases are extremely diverse. Running virtual machines or containers in the cloud is one of the most popular uses for Microsoft Azure. These compute resources can host infrastructure components, such as domain

name system (DNS) servers; Windows Server services -- such as Internet Information Services (IIS); or third-party applications. Microsoft also supports the use of third-party operating systems, such as Linux.

AZURE SERVICES

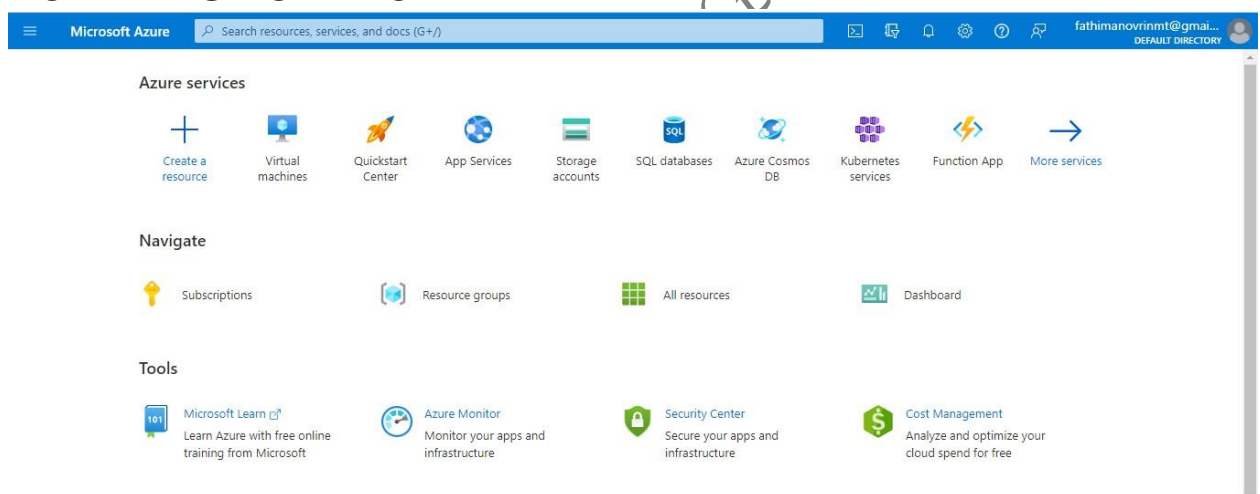
- **Compute.** These services enable a user to deploy and manage VMs, containers and batch jobs, as well as support remote application access. Compute resources created within the Azure cloud can be configured with either public IP addresses or private IP addresses, depending on whether the resource needs to be accessible to the outside world.
- **Mobile.** These products help developers build cloud applications for mobile devices, providing notification services, support for back-end tasks, tools for building application program interfaces (APIs) and the ability to couple geospatial context with data.
- **Web.** These services support the development and deployment of web applications. They also offer features for search, content delivery, API management, notification and reporting.
- **Storage.** This category of services provides scalable cloud storage for structured and unstructured data. It also supports big data projects, persistent storage and archival storage.
- **Analytics.** These services provide distributed analytics and storage, as well as features for real-time analytics, big data analytics, data lakes, machine learning (ML), business intelligence (BI), internet of things (IoT) data streams and data warehousing.
- **Networking.** This group includes virtual networks, dedicated connections and gateways, as well as services for traffic management and diagnostics, load balancing, DNS hosting and network protection against distributed denial-of-service (DDoS) attacks.

- **Media and content delivery network (CDN).** These CDN services include on-demand streaming, digital rights protection, encoding and media playback and indexing.
- **Integration.** These are services for server backup, site recovery and connecting private and public clouds.
- **Identity.** These offerings ensure only authorized users can access Azure services and help protect encryption keys and other sensitive information in the cloud. Services include support for Azure Active Directory and multifactor authentication (MFA).
- **Internet of things.** These services help users capture, monitor and analyze IoT data from sensors and other devices. Services include notifications, analytics, monitoring and support for coding and execution.
- **DevOps.** This group provides project and collaboration tools, such as Azure DevOps -- formerly Visual Studio Team Services -- that facilitate DevOps software development processes. It also offers features for application diagnostics, DevOps tool integrations and test labs for build tests and experimentation.
- **Development.** These services help application developers share code, test applications and track potential issues. Azure supports a range of application programming languages, including JavaScript, Python, .NET and Node.js. Tools in this category also include support for Azure DevOps, software development kits (SDKs) and blockchain.
- **Security.** These products provide capabilities to identify and respond to cloud security threats, as well as manage encryption keys and other sensitive assets.
- **Artificial intelligence (AI) and machine learning.** This is a wide range of services that a developer can use to infuse artificial intelligence, machine learning and cognitive computing capabilities into applications and data sets.
- **Containers.** These services help an enterprise create, register, orchestrate and manage huge volumes of containers in the Azure cloud, using common platforms such as Docker and Kubernetes.
- **Databases.** This category includes Database as a Service (DBaaS) offerings for SQL and NoSQL, as well as other database instances -- such as Azure Cosmos DB and Azure Database for PostgreSQL. It also includes Azure SQL Data Warehouse support, caching and hybrid database integration and migration features. Azure SQL is the platform's flagship database service. It is a relational

database that provides SQL functionality without the need for deploying a SQL server.

- **Migration.** This suite of tools helps an organization estimate workload Migration costs and perform the actual migration of workloads from local data centers to the Azure cloud.
- **Management and governance.** These services provide a range of backup, recovery, compliance, automation, scheduling and monitoring tools that can help a cloud administrator manage an Azure deployment.
- **Mixed reality.** These services are designed to help developers create content for the Windows Mixed Reality environment.
- **Blockchain.** The Azure Blockchain Service allows you to join a blockchain consortium or to create your own.

HOME PAGE OF AZURE



VM DEPLOYMENT WITHOUT ANSIBLE

1. Head to a virtual machine from the Azure homescreen.
2. Click on **create->virtual machine**.
3. Fill the details that you need to create a virtual machine and hit **Review+create**.

USING ANSIBLE

To Configure LINUX VM Using Ansible Playbook ,

- First we need an azure account, to get a free account:
-> <https://azure.microsoft.com/en-in/free/>
- Create a resource group
- Create a virtual network
- Create a public IP address
- Create a network security group
- Create a virtual network interface card
- Create a virtual machine

Open The Azure Shell, we can directly run Ansible in Azure Cloud Shell, where Ansible is **pre-installed**.

In case not installed,

#Update all packages that have available updates.

sudo yum update -y

Install Python 3 and pip.

sudo yum install -y python3-pip

Upgrade pip3.

sudo pip3 install --upgrade pip

Install Ansible. pip3 install "ansible==2.9.17"

Install Ansible azure_rm module for interacting with Azure.

pip3 install ansible[azure]

To check the ansible version installed,

ansible --version