Navigating Linux System Commands

A guide for beginners to the Linux Shell and GNU coreutils

Sayan Ghosh

June 13, 2024

IIT Madras
BS Data Science and Applications

Disclaimer
This document is a companion activity book for the System Commands (BSSE2001) course taught by **Prof. Gandham Phanikumar** at **IIT Madras BS Program.** This book contains resources, references, questions and solutions to some common questions on Linux commands, shell scripting, grep, sed, awk, and other system commands.

This was prepared with the help and guidance of the course instructors:

Santhana Krishnan and **Sushil Pachpinde**

Colophon
This document was typeset with the help of KOMA-Script and $\LaTeX$ using the kaobook class.
The source code of this book is available at:
`https://github.com/sayan01/se2001-book`
(You are welcome to contribute!)

Edition
Compiled on June 13, 2024

UNIX is basically a simple operating system, but you have to be a genius to understand the simplicity.

– Dennis Ritchie

Preface

Through this work I have tried to make learning and understanding the basics of Linux fun and easy. I have tried to make the book as practical as possible, with many examples and exercises. The structure of the book follows the structure of the course *BSSE2001 - System Commands*, taught by **Prof. Gandham Phanikumar** at **IIT Madras BS Program.** .

The book takes inspiration from the previous works done for the course,

- Sanjay Kumar's Github Repository
- Cherian George's Github Repository
- Prabuddh Mathur's TA Sessions

as well as external resources like:

- Robert Elder's Blogs and Videos
- Aalto University, Finland's Scientific Computing - Linux Shell Crash Course

The book covers basic commands, their motivation, use cases, and examples. The book also covers some advanced topics like shell scripting, regular expressions, and text processing using `sed` and `awk`.

This is not a substitute for the course, but a companion to it. The book is a work in progress and any contribution is welcome at `https://github.com/sayan01/se2001-book`

Sayan Ghosh

Contents

List of Figures

List of Tables

Networking and SSH 1

1.1 Networking

1.1.1 What is networking?

Have you ever tried to get some work done on a computer while the internet was down? It's a nightmare. Modern day computing relies highly on networking. But what is networking?

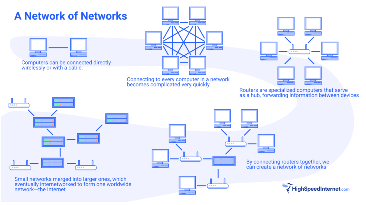
Definition 1.1.1 (Networking) A computer network comprises two or more computers that are connected—either by cables (wired) or wifi (wireless)—with the purpose of transmitting, exchanging, or sharing data and resources.

We have been using the computer, and linux, for a while now but the utility of a computer increases exponentially when it is connected to a network. It allows computers to share files and resources, and to communicate with each other. Current day world wide web is built on the internet.

Definition 1.1.2 (Internet) Internet is a global network of networks that connects millions of computers worldwide. It allows computers to connect to other computers across the world through a hierarchy of routers and servers.

Learning about networking and how networking works is useful, although we won't be devling into details in this book. It is left as an exercise for the reader to explore external resources if they are interested.

One succinct blogpost explaining how the internet works from which the figure Figure 1.1 is taken is available at `https://www.highspeedinternet.com/resources/how-the-internet-works`



A Network of Networks
Computers can be connected directly wirelessly or with a cable.
Connecting to every computer in a network becomes complicated very quickly.
Routers are specialized computers that serve as a hub, forwarding information between devices
Small networks merged into larger ones, which eventually internetworked to form one worldwide network—the Internet
By connecting routers together, we can create a network of networks
HighSpeedInternet.com

Figure 1.1: Types of Networks

1.1.2 Types of Networks

If the end goal is to connect computers with each other, one naive solution might be to connect all the computers with each other. Although this might seem intuitive at first, this quickly gets out of hand when the number of computers keep increasing.

If we have n computers, then the number of connections required to connect all the computers with each other is given by the formula

$$\frac{n(n-1)}{2} = \frac{n^2 - n}{2}$$

This is a quadratic function and grows very quickly.

This means it will cost a lot to connect all the computers to each other. This is applicable not only in computer networking with physical wires, but in many other fields. Take an examples of airlines and airplane routes.

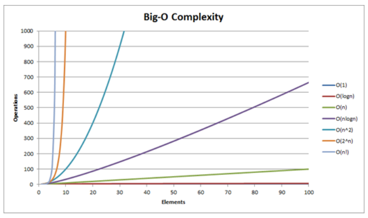


Figure 1.2: Growth Rate of Different Functions - Note how quickly n^2 grows

If there were n airports, then the number of routes required to connect all the airports is given by the same formula. This would be disastrous for the economy and the environment if we ran so many airplanes daily. So what gives?

Hub and Spoke Network

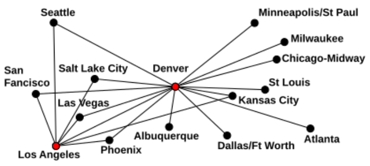


Figure 1.3: Hub and Spoke Model Employed by Airlines

The solution to this problem is to use a hub and spoke model, where there are one, or multiple, central hubs which connect to many other nodes. Any path from any node to another goes through one or more hubs. This reduces the number of connections required to connect all the nodes.

This is the solution used in airlines, and also in most computer networks. [1]

1: Although computer networks use a hub model for the local area network, the network of networks, especially the gateway routers follow a mesh model to ensure redundancy and make the network more robust.

Due to this, networks can be classified into three broad categories based on their geographical area coverage.

- **Local Area Network (LAN)**: A network that covers a small geographical area, like a home, office, or a building.
- **Metropolitan Area Network (MAN)**: A network that covers a larger geographical area, like a city or a town.
- **Wide Area Network (WAN)**: A network that covers a large geographical area, like a country or the entire world.

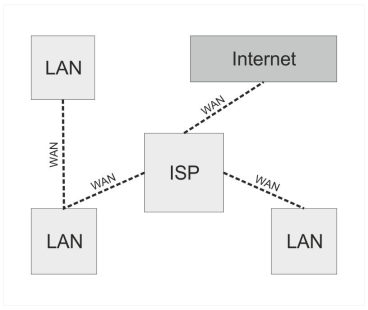


Figure 1.4: LAN and WAN connecting to the Internet

To connect these networks to computers and also to each other, we require some special devices.

1.1.3 Devices in a Network

In computer networks, this hub of the Hub and Spoke Model can either be a level 1 hub, a level 2 switch, or a level 3 router.

Hub

A hub will simply broadcast the message to all the connected nodes. This causes a lot of traffic to be generated and is not very efficient. Hub does not have the capability to identify which node is who. This is called a level 1 hub. [2]

2: To understand more about the levels, refer OSI Model

Switch

A switch is smarter than a hub. It can identify each device connected to it and can send the packets of data only to the intended recipient. This is more efficient than a hub. This is called a level 2 switch since it uses the level 2 of the OSI model (Data Link Layer) to identify the devices. This means that the devices are identified by their MAC addresses. Using this, you can only communicate with devices in your local network. This is useful for a home network or a office network. But we cannot communicate with the entire world using this, since it doesn't understand IP addresses.

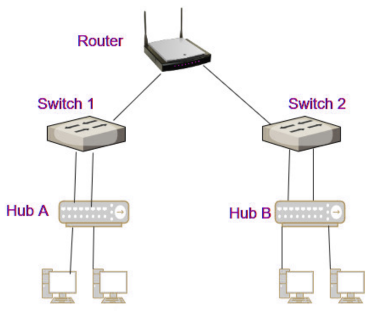


Figure 1.5: Hub, Switch, Router connecting to the Internet

Router

A router is even smarter than a switch. It can understand IP addresses and can route the packets from one network to another. This is called a level 3 router since it uses the level 3 of the OSI model (Network Layer) to identify the devices. This means that the networks are identified by their

IP addresses. This is what we use to connect to the internet. The internet is nothing but a whole lot of routers communicating with each other to find the optimal path to send the packets to its destination. Border Gateway Protocol (BGP) is the protocol used by routers to communicate with each other and find the optimal path. They usually are connected in a mesh network to ensure redundancy and robustness.

Level 3 Switch

A level 3 switch, or a routing switch, is a switch with the capabilities of a router. It can understand the language of IP addresses and can route the packets to different networks. This is useful in large organizations where there are many networks and each network can be divided into subnetworks called VLANs.

You can read more about the differences between these devices online.

So, in short, internet is a network of network of . . . networks. It is a hierarchical structure connecting all the computers in the world. Some computers are connected earlier in the hierarchy (usually the ones closer geographically) and some are connected later.

1.1.4 IP Addresses

So how do routers know which where to send the data packets? This is where IP addresses come in. To communicate over the internet, two computers need to know their public IP addresses. The routers then finds the optimal path to send the data packets to the destination network.

IP addresses are of two types: IPv4 and IPv6. The most common one is IPv4, which is a 32-bit address represented as four octets separated by dots. For example,

$$162.136.73.21$$

Here, each octet can take values from 0 to 255. [3] Technically all such combinations are possible IP addresses, resulting in $2^{32} = 4,294,967,296$ possible IP addresses. That is a lot of IP addresses, but not enough for the growing number of devices in the world.

3: An octet is a group of 8 bits. Since an IP address is 32 bits, it is represented as 4 groups of 8 bits. 8 bits can only represent numbers from 0 to 255. since $2^8 = 256$.

This is where IPv6 comes in. IPv6 is a 128-bit address represented as 8 groups of 4 hexadecimal digits separated by colons. For example,

$$2001 : 0db8 : 85a3 : 0000 : 0000 : 8a2e : 0370 : 7334$$

This results in $2^{128} = 340282366920938463463374607431768211456$ possible IP addresses, which is a lot more than IPv4.

Notice that there are some groups of zeros in the address. These can be compressed by writing only one zero in place of multiple zeros in each group. Further, any leading zeros can be omitted for each group. Making the above address as `2001:db8:85a3:0:0:8a2e:370:7334`. Further, if there are multiple groups of zeros, they can be compressed to `::`. This can be used only once in an address. Doing this, the above address can be compressed further to `2001:db8:85a3::8a2e:370:7334`.

1.1.5 Subnetting and CIDR

Legacy Classes

$$10.125.42.62 \longrightarrow 00001010.01111101.00101010.00111110$$

Recall that an IP address, although represented as four octets, is actually a 32-bit address. This means that in binary form, an IP address is a string of 32 1s and 0s. Using the first four bits, we can classify an IP address into five classes.

- **Class A**: The first bit is '0'. The IP addresses in the range 0.0.0.0 to 127.255.255.255.
- **Class B**: The first two bits are '10'. IP addresses in the range 128.0.0.0 to 191.255.255.255.
- **Class C**: The first three bits are '110'. IP addresses in the range 192.0.0.0 to 223.255.255.255.
- **Class D**: The first four bits are '1110'. IP addresses in the range 224.0.0.0 to 239.255.255.255. These are reserved for multicast addresses.
- **Class E**: The first four bits are '1111'. IP addresses in the range 240.0.0.0 to 255.255.255.255. These are reserved for experimental purposes.

However, these classes do not simply assign an IP to each machine. They are further divided into the network part and the host part.

Class A assigns the first octet to the network part, this is used to identify which network the machine is in. The remaining three octets are used to identify the host in that network. This means that a class A network can have $2^{24} - 2 = 16,777,214$ hosts. However, there can only be $2^7 = 128$ class A networks. Thus, class A networks are used by large organizations which have many hosts, but not many large organizations exist, so 128 networks are enough.

Similarly, class B assigns the first two octets to identify the network and the remaining two octets to identify the host. This means that a class B network can have $2^{16} - 2 = 65,534$ hosts. And there can be $2^{14} = 16,384$ class B networks. These are used by medium-sized organizations, which are plenty in number, and have a moderate number of hosts.

The same goes for class C networks, where the first three octets are used to identify the network and the last octet is used to identify the host. This network can have $2^8 - 2 = 254$ hosts. And there can be $2^{21} = 2,097,152$ class C networks. These are used by small organizations, which are plenty in number, and have a small number of hosts.

Subnet Masks

Definition 1.1.3 (Subnetting) The process of dividing a network into smaller network sections is called subnetting.

Usually, each network has only one subnet, which contains all the hosts in that network. However, the network can be further divided into smaller subnetworks, each containing a subset of the hosts. This is useful in large organizations where the network is divided into departments, and each department is given a subnetwork.

To indicate which part of the IP address is the network part and which part is the host part, we use a subnet mask. A subnet mask is a 32-bit number where the first n bits are 1s and the remaining bits are 0s. The number of 1s in the subnet mask indicates the number of bits used to identify the network. For example, for the IP Address `192.168.0.15`, it can be written in binary as

$$11000000 - 10101000 - 00000000 - 00001111$$

As we know, it belongs to the class C network, where the first three octets are used to identify the network, and the rest is used to identify the host. So the default network mask is

$$11111111 - 11111111 - 11111111 - 00000000$$

or `255.255.255.0` in decimal. The network portion of the IP address is found by taking the bitwise AND [4] of the IP address and the subnet mask. This results in the network address

4: The bitwise AND operation is a binary operation that takes two equal-length binary representations and performs the logical AND operation on each pair of corresponding bits. The result in each position is 1 if the first bit is 1 and the second bit is 1; otherwise, the result is 0.

$$11000000 - 10101000 - 00000000 - 00000000$$

which is `192.168.0.0` and the host address is `0000 1111` which is 15.

However, if we do not require all the 8 bits in the host space [5] then we can use some of the initial bits of the host space to identify subnetworks. This is called subnetting.

5: That is, if we have less than 254 hosts in the network.

For example, the netmask of `255.255.255.0` leaves 8 bits for the host space, or $2^8 - 2 = 254$ [6] hosts. If we want to split this network into two subnets, we can use the MSB of the host space for representing the subnetworks. This results in each subnet having $2^7 - 2 = 126$ hosts.

6: We subtract 2 from the total number of hosts to account for the network address and the broadcast address, which are the first(0) and the last(255) addresses in the network.

Remark 1.1.1 Observe that we effectively lost two available addresses from the total number of hosts in the network. Earlier we could have 254 hosts, but now we can have only $126 \times 2 = 252$ hosts. This is because each subnet also reserves the first and the last address for the network address and the broadcast address.

To do this, we change the subnet mask to

$$11111111 - 11111111 - 11111111 - 10000000$$

which can be represented as `255.255.255.128`.

This gives us two subnets, one with the address range of `192.168.0.1` to `192.168.0.127`, and another of `192.168.0.129` to `192.168.0.255`.

1.1.6 Private and Public IP Addresses

But what if we want to communicate with computers in our local network? This is where private IP comes in. Some ranges of IP addresses are reserved for private networks. These are not routable over the internet. Each LAN has a private IP address range, and the router translates these private addresses to the public IP address when sending the packets over the internet. The assignment of these private IP addresses is done by the DHCP server [7] in the router.

7: Dynamic Host Configuration Protocol (DHCP) is a network management protocol used on Internet Protocol networks whereby a DHCP server dynamically assigns an IP address and other network configuration parameters to each device on a network so they can communicate with other IP networks.

Each class of networks has a range of IP addresses that are reserved for private networks.

Table 1.1: Private IP Address Ranges

Class	Network Bits (CIDR)	Address Range	Number of Addresses
Class A	8	`10.0.0.0` - `10.255.255.255`	16,777,216
Class B	12	`172.16.0.0` - `172.31.255.255`	1,048,576
Class C	16	`192.168.0.0` - `192.168.255.255`	65,536

1.1.7 CIDR

However, this practice of subdividing IP addresses into classes is a legacy concept, and not followed anymore. Instead, we use CIDR (Classless Inter-Domain Routing) to announce how many bits are used to identify the network and how many bits are used to identify the host.

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 is helpful because not all organizations fit into the tight categorization of the legacy classes.

1.1.8 Ports

Ports usually refer to physical holes in a computer where you can connect a cable. However, in networking, ports refer to logical endpoints for communication.

Definition 1.1.4 (Port) A port or port number is a number assigned to uniquely identify a connection endpoint and to direct data to a specific service. At the software level, within an operating system, a port is a logical construct that identifies a specific process or a type of network service.

For any communication between two computers, the data needs to be sent to a specific port. This is because there are multiple services running on a computer, and the operating system needs to know which service to direct the data to.

There are $2^{16} = 65,536$ ports available for use. However, the first 1024 ports are reserved for well-known services. These are called the **well-known ports** and are used by services like HTTP, FTP, SSH, etc. The well known ports can be found in Table 1.2.

Other ports, from 1024 to 49151, are registered ports, these ports can be registered with the Internet Assigned Numbers Authority (IANA) by anyone who wants to use them for a specific service.

Ports from 49152 to 65535 are dynamic ports, these are used by the operating system for temporary connections and are not registered.

Whenever you send a request to a web server for example, the request is sent to the server's IP address and the port number `80` which is the default port for HTTP [8] but the port number from your (the client) side

8: or 443 for HTTPS.

is usually a random port number from the dynamic port range. This is a short-lived port number and is used to establish a connection with the server.

These kinds of ports which are short-lived and used to establish a connection are called **ephemeral ports**.

Table 1.2: Well-known Ports

Port Number	Service	Protocol
20	File Transfer Protocol (FTP) Data Transfer	TCP
21	File Transfer Protocol (FTP) Command Control	TCP
22	Secure Shell (SSH) Secure Login	TCP
23	Telnet remote login service, unencrypted text messages	TCP
25	Simple Mail Transfer Protocol (SMTP) email delivery	TCP
53	Domain Name System (DNS) service	TCP/UDP
67, 68	Dynamic Host Configuration Protocol (DHCP)	UDP
80	Hypertext Transfer Protocol (HTTP) used in the World Wide Web	TCP
110	Post Office Protocol (POP3)	TCP
119	Network News Transfer Protocol (NNTP)	TCP
123	Network Time Protocol (NTP)	UDP
143	Internet Message Access Protocol (IMAP) Management of digital mail	TCP
161	Simple Network Management Protocol (SNMP)	UDP
194	Internet Relay Chat (IRC)	TCP
443	HTTP Secure (HTTPS) HTTP over TLS/SSL	TCP
546, 547	DHCPv6 IPv6 version of DHCP	UDP

1.1.9 Protocols

Definition 1.1.5 (Protocols) In computing, a protocol is a set of rules that define how data is transmitted between devices in a network.

There are many protocols used in networking, some of the most common ones are

- **HTTP**: HyperText Transfer Protocol
- **HTTPS**: HyperText Transfer Protocol Secure
- **FTP**: File Transfer Protocol
- **SSH**: Secure Shell
- **SMTP**: Simple Mail Transfer Protocol
- **POP3**: Post Office Protocol
- **IMAP**: Internet Message Access Protocol
- **DNS**: Domain Name System
- **DHCP**: Dynamic Host Configuration Protocol
- **NTP**: Network Time Protocol
- **SNMP**: Simple Network Management Protocol
- **IRC**: Internet Relay Chat
- **BGP**: Border Gateway Protocol
- **TCP**: Transmission Control Protocol
- **UDP**: User Datagram Protocol

These protocols act on the different layers of the OSI model. For example, HTTP, HTTPS, FTP, SSH, etc. are application layer protocols, while TCP, UDP, etc. are transport layer protocols.

1.1.10 Firewalls

Definition 1.1.6 (Firewall) A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules.

A fireewall acts as a barrier between your computer and the internet. It monitors the incoming and outgoing traffic and blocks any traffic that does not meet the security rules. This is useful to prevent unauthorized access to your computer and to prevent malware from entering your computer and/or communicating with the outside world.

```
$ sudo ufw enable # Enable the firewall
$ sudo ufw allow 22 # Allow SSH
$ sudo ufw allow 80 # Allow HTTP
$ sudo ufw allow 443 # Allow HTTPS
$ sudo ufw status # Check the status of the firewall
```

1.1.11 SELinux

We can have additional security by using SELinux in addition to the firewall. SELinux is a security module that provides access control security policies. SELinux is short for Security-Enhanced Linux. It provides a flexible Mandatory Access Control (MAC) that restricts the access of users and processes to files and directories.

Least Privilege Principle

Definition 1.1.7 (Least Privilege Principle) The principle of least privilege (POLP) is an important concept in computer security, promoting minimal user profile privileges on computers based on users' job necessity.

This principle states that a user should have only the minimum privileges required to perform their job. This principle is applied throughout linux, and also in SELinux.

You can check if **SELinux** is enabled by running

```
$ sestatus
```

If SELinux is enabled, you can check the context of a file or a directory using

```
$ ls -lZ
```

However, if SELinux is not enabled, it will show a ? in the context.

If SELinux is enabled, you can set the context of a file or a directory using the `chcon` command.

RBAC Items

Definition 1.1.8 (Role-Based Access Control (RBAC)) Role-Based Access Control (RBAC) is a policy-neutral access control mechanism defined around roles and privileges. The components of RBAC such as role-permissions, user-role, and role-role relationships make it simple to perform user assignments.

SELinux uses the concept of RBAC to control the access of users and processes to files and directories.

There are four components in the SELinux context that are used to control the access of users and processes to files and directories, as shown in Figure 1.6.

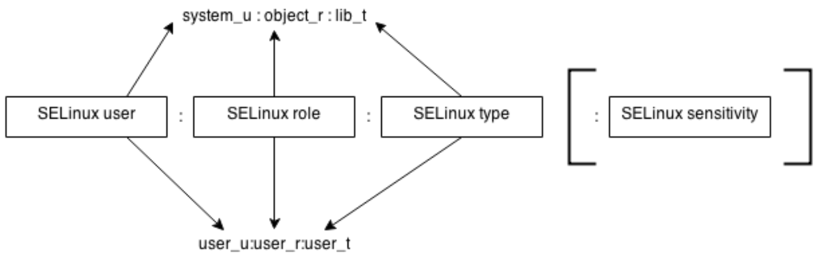


Figure 1.6: SELinux Context

- **User**: The user who is trying to access the file or directory.
- **Role**: The role of the user, which defines the permissions of the user.
- **Type**: The type of the file or directory.
- **Domain**: The domain [9] of the process trying to access the file or directory.

9: or type or sensitivity

Modes of SELinux

- **Enforcing**: In this mode, SELinux is enabled and actively enforcing the security policies.
- **Permissive**: In this mode, SELinux is enabled but not enforcing the security policies. It logs the violations but does not block them.
- **Disabled**: In this mode, SELinux is disabled and not enforcing any security policies.

You can change the mode of SELinux by editing the `/etc/selinux/config` file.

```
$ sudo vim /etc/selinux/config
```

Tools for SELinux

- **sestatus**: Check the status of SELinux.
- **semamange**: Manage the SELinux policy.
- **restorecon**: Restore the context of files and directories.

1.1.12 Network Tools

There are a lot of tools in GNU/Linux used for managing, configuring, and troubleshooting networks. Some of the important tools are listed in Table 1.3.

ip

To find out the private IP address of the NICs of your system, you can run the `ip addr` command. [10]

10: `ip a` also works.

Table 1.3: Network Tools

Tool	Description
`ip`	Show / manipulate routing, devices, policy routing and tunnels
`ping`	To see if the remote machine is up
`traceroute`	Diagnostics the hop timings to the remote machine
`nslookup`	Ask for conversion of IP address to name
`dig`	DNS lookup utility
`netstat`	Print network connections
mxtoolbox	Public accessibility of your server
whois	Information about the domain
`nmap`	Network port scanner
`wireshark`	Network protocol analyzer and packet sniffer

```
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
    UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
     state UP group default qlen 1000
    link/ether 1c:1b:0d:e1:5d:61 brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 192.168.0.109/24 brd 192.168.0.255 scope global dynamic
    eno1
       valid_lft 7046sec preferred_lft 7046sec
    inet6 fe80::68e2:97e0:38ec:4abc/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Here you can see there are two interfaces, `lo` and `eno1`. The `lo` interface is the loopback interface, and the `eno1` interface is the actual network interface. The IP address of the **lo** interface is usually always `127.0.0.1`. This address is used to refer to the same system in terms of IP address without knowing the actual private IP of the system in the LAN.

The IP address of the **eno1** interface is the private IP address allocated by your router. This is not your public IP address, which is the address of your router on the internet. Usually public IPs are statically assigned by ISPs and are not changed often. It is configured in your router.

Private IPs however often needs to be assigned dynamically since devices can connect and disconnect from the network at any time. This is done by the DHCP server in your router.

Remark 1.1.2 The NIC name can be different in different systems. For a ethernet connection, it is usually `eno1` or `eth0` which is the legacy name. For a wifi NIC, it is usually `wlan0`.

Remark 1.1.3 Earlier the tool used to check the network status was `ifconfig`. However, this tool is deprecated now and should not be used. The new tool to check the network status is `ip`.

ping

The `ping` command is used to check if a remote server is up and running. It sends an **ICMP**[11] packet to the remote server and waits for a response.

11: Internet Control Message Protocol (ICMP) is a supporting protocol in the Internet protocol suite. It is used by network devices, including routers, to send error messages and operational information indicating success or failure when communicating with another IP address.

Remark 1.1.4 Only a positive response from the server indicates that the server is up and running. A negative response does not necessarily mean that the server is down. Servers can be configured to not respond to ICMP packets.

```
$ ping -c 4 google.com # Send 4 ICMP packets to google.com
PING google.com (172.217.163.206) 56(84) bytes of data.
64 bytes from maa05s06-in-f14.1e100.net (172.217.163.206):
    icmp_seq=1 ttl=114 time=45.6 ms
64 bytes from maa05s06-in-f14.1e100.net (172.217.163.206):
    icmp_seq=2 ttl=114 time=45.4 ms
64 bytes from maa05s06-in-f14.1e100.net (172.217.163.206):
    icmp_seq=3 ttl=114 time=45.3 ms
64 bytes from maa05s06-in-f14.1e100.net (172.217.163.206):
    icmp_seq=4 ttl=114 time=45.8 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 45.316/45.524/45.791/0.181 ms
```

The response of the `ping` command shows the time taken for the packet to reach the server and also the resolved IP address of the server.

nslookup

Another tool to lookup the associated IP address of a domain name is the `nslookup` command.

```
$ nslookup google.com
Server:         192.168.0.1
Address:        192.168.0.1#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.163.206
Name:   google.com
Address: 2404:6800:4007:810::200e
```

Here you can see the resolved IP address of the domain is `172.217.163.206`. If you copy this IP address and paste it in your browser, you can see that the website of google opens up. The second address returned is the IPv6 IP Address.

The first lines mentioning the **Server** is the **DNS Server** which returned the resolution of the IP address from the queried domain name.

Remark 1.1.5 Notice that the DNS Server mentioned in the above output is actually a private IP. This is the IP address of the router in the LAN which acts as the DNS Server cache. However if you type the domain of a website which you have not visited, or have visited long ago into `nslookup`, then the DNS Server mentioned will be the public address of the DNS Server, which might be your ISP's DNS

Server, or some other public DNS Server.

You can also use **mxtoolbox** to check the IP address of your server from the public internet.

dig

Another tool to lookup the associated IP address of a domain name is the `dig` command. It can also reverse lookup the IP address to find the associated domain name.

```
$ dig google.com

; <<>> DiG 9.18.27 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31350
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL:
     9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 3e5ff6a57c0fe2b3b5ce91b3666ae859ec9b6471261cecef (good)
;; QUESTION SECTION:
;google.com.            IN  A

;; ANSWER SECTION:
google.com.     50  IN  A   172.217.163.206

;; AUTHORITY SECTION:
google.com.     162911  IN  NS  ns1.google.com.
google.com.     162911  IN  NS  ns3.google.com.
google.com.     162911  IN  NS  ns4.google.com.
google.com.     162911  IN  NS  ns2.google.com.

;; ADDITIONAL SECTION:
ns2.google.com.     163913  IN  A   216.239.34.10
ns4.google.com.     163913  IN  A   216.239.38.10
ns3.google.com.     337398  IN  A   216.239.36.10
ns1.google.com.     340398  IN  A   216.239.32.10
ns2.google.com.     163913  IN  AAAA    2001:4860:4802:34::a
ns4.google.com.     163913  IN  AAAA    2001:4860:4802:38::a
ns3.google.com.     2787    IN  AAAA    2001:4860:4802:36::a
ns1.google.com.     158183  IN  AAAA    2001:4860:4802:32::a

;; Query time: 3 msec
;; SERVER: 192.168.0.1#53(192.168.0.1) (UDP)
;; WHEN: Thu Jun 13 18:18:52 IST 2024
;; MSG SIZE  rcvd: 331
```

And we can then feed the IP address to dig again, to find the domain name associated with the IP address.

```
$ dig -x 172.217.163.206

; <<>> DiG 9.18.27 <<>> -x 172.217.163.206
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15781
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL:
     9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 78a3c6e4e4b103f501380f62666ae89b3a3b52e8be388fe0 (good)
;; QUESTION SECTION:
;206.163.217.172.in-addr.arpa.  IN  PTR

;; ANSWER SECTION:
206.163.217.172.in-addr.arpa. 83966 IN  PTR maa05s06-in-f14.1e100.
    net.

;; AUTHORITY SECTION:
217.172.in-addr.arpa.   78207   IN  NS  ns4.google.com.
217.172.in-addr.arpa.   78207   IN  NS  ns2.google.com.
217.172.in-addr.arpa.   78207   IN  NS  ns1.google.com.
217.172.in-addr.arpa.   78207   IN  NS  ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.     340332  IN  A   216.239.32.10
ns2.google.com.     163847  IN  A   216.239.34.10
ns3.google.com.     337332  IN  A   216.239.36.10
ns4.google.com.     163847  IN  A   216.239.38.10
ns1.google.com.     158117  IN  AAAA    2001:4860:4802:32::a
ns2.google.com.     163847  IN  AAAA    2001:4860:4802:34::a
ns3.google.com.     2721    IN  AAAA    2001:4860:4802:36::a
ns4.google.com.     163847  IN  AAAA    2001:4860:4802:38::a

;; Query time: 3 msec
;; SERVER: 192.168.0.1#53(192.168.0.1) (UDP)
;; WHEN: Thu Jun 13 18:19:58 IST 2024
;; MSG SIZE  rcvd: 382
```

Note that the answer we got after running google.com through **dig** and then through **dig -x** (maa05s06-in-f14.1e100.net) is different from the original domain name.

This is because the domain name is resolved to an IP address, and then the IP address is resolved to a different domain name. This is because the the domain name is actually an alias to the cannonical name.

Remark 1.1.6 The IP address you would get by running **dig** or **nslookup** on google would be different from the IP address you get when using mxtoolbox. This is because google is a large company and they have multiple servers which are load balanced. So someone in India might get a different IP address compared to someone in the US.

To get the output of `dig` in a more readable and concise format, you can use the `+short` or `+noall` option.

```
$ dig +noall +answer google.com
google.com.     244 IN  A   172.217.163.206
```

netstat

The `netstat` command is used to print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.

It is useful to find what connections are open on your system, and what ports are being used by which applications.

```
$ netstat | head
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
      State
tcp        0      0 rex:53584               24.224.186.35.bc.:
    https TIME_WAIT
tcp        0      0 rex:56602               24.224.186.35.bc.:
    https TIME_WAIT
tcp        0      0 localhost:5037          localhost:43267
      TIME_WAIT
tcp        0      0 localhost:5037          localhost:46497
      TIME_WAIT
tcp        0      0 rex:35198               24.224.186.35.bc.:
    https TIME_WAIT
tcp        0      0 rex:44302               24.224.186.35.bc.:
    https TIME_WAIT
tcp        0      0 localhost:5037          localhost:55529
      TIME_WAIT
tcp        0      0 localhost:5037          localhost:38005
      TIME_WAIT
```

1.2 SSH

1.2.1 What is SSH?

The Secure Shell (SSH) Protocol is a protocol for secure communication between two computers over a compromised or untrusted network. [12] SSH uses encryption and authentication to secure the communication between the two computers.

12: like the internet.

SSH is now the ubiquitous protocol for secure remote access to servers, and is used by system administrators all over the world to manage their servers.

SSH lets a user of a computer to log into the computer from another computer over the network, to execute any command in the terminal that they have access to.

SSH can also be used to transfer files between two computers using the `scp` command.

1.2.2 History

It was initially developed by Tatu Ylönen in 1995 as a replacement for the insecure Telnet and FTP protocols when he found that someone had installed a packet sniffer on the server of his university.

There are multiple implementations of the SSH protocol, the most popular being OpenSSH, developed by the OpenBSD project. This is the implementation that is used in most of the linux distributions as well.

1.2.3 How does SSH work?

SSH works by using symmetric and asymmetric encryption. The data packets sent over the network are enctypted, usually using AES symmetric encryption. This ensures that even if the data packets are intercepted by a man-in-the-middle attacker, they cannot be read since they are encrypted.

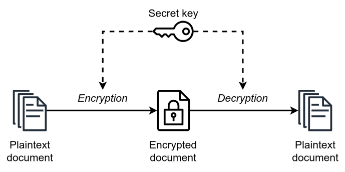


Figure 1.7: Symmetric Encryption

To login into a remote server, all you need to do is provide the username and the IP address or the domain name of the server to the `ssh` command.

```
$ ssh username@ipaddress
```

OR

```
$ ssh username@domainname
```

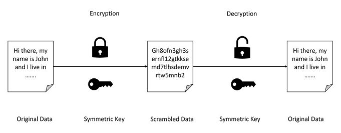

Encryption
Decryption
Hi there, my name is John and I live in ....
Gh8ofn3gh3s ernfl12gtkkse md7tlhsdemv rtw5mnb2
Hi there, my name is John and I live in ....
Original Data
Symmetric Key
Scrambled Data
Symmetric Key
Original Data

Figure 1.8: Symmetric Encryption

SSH allows user to login to a remote server using their username and password, but this is not encouraged since it lets the user to be vulnerable to brute-force attacks.

Another way to authenticate is by using public-private key pairs.

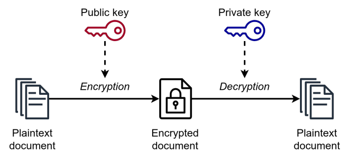


Figure 1.9: Asymmetric Encryption

1.2.4 Key-based Authentication

One of the most powerful features of SSH is its ability to use public-private key pairs for authentication. In our course, we emphasize the importance of this method. Instead of relying on passwords, which can be vulnerable to brute-force attacks, a pair of cryptographic keys is used. The public key is stored on the server, while the private key is kept secure on your local machine. This ensures a highly secure and convenient way of accessing remote servers without the need for constantly entering passwords.

1.2.5 Configuring your SSH keys

For this course, it is a must for you to not only create, but also understand SSH keys. Let us quickly see how to create a ssh key-pair which can be used to login into a remote server.

We need to use the `ssh-keygen` command to create a new public-private key pair.

```
$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/test1/.ssh/id_ed25519):
```

Here you have to either simply press enter to continue with the default location, or you can also type a custom location where you want to save the key. If this is your first time creating keys, it is recommended to use the default location.

Remark 1.2.1 There are multiple algorithms that can be used to generate a key pair. The most common ones are RSA, DSA, and ED25519. The ED25519 algorithm is the new default algorithm used by OpenSSH since it is shorter yet more secure than RSA. If you have an outdated version of OpenSSH, you might get the default RSA algorithm. To change the algorithm, you can use the `-t` flag along with the `ssh-keygen` command.

```
$ ssh-keygen -t rsa
```

will create a RSA key pair and using

```
$ ssh-keygen -t ed25519
```

will create a ED25519 key pair.

Next, it will ask you to enter a passphrase. You can enter a passphrase for added security, or you can simply press enter to continue without a passphrase. If you do add a passphrase, you will have to always enter the passphrase whenever you use the key. We can continue without a passphrase for now by pressing enter.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

Your identification has been saved in /home/username/.ssh/
    id_ed25519
Your public key has been saved in /home/username/.ssh/id_ed25519.
    pub
```

```
The key fingerprint is:
SHA256:n4ounQd6v9uWXAtMyyq7CdncMsh1Zuac5jesWXrndeA test1@rex
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|                 |
|                 |
|          .      |
|      . S+ .  .  |
|   . *.O o=... . |
|    =o=o*+++ .E .|
|    o.=Bo*O o. . |
|     +**XB.+.    |
+----[SHA256]-----+
```

Our key pair has been generated. The private key is stored in `/home/username/.ssh/id_-ed25519` and the public key is stored in `/home/username/.ssh/id_-ed25519.pub`.

Make sure to **never** share your private key with anyone. Ideally, you dont even need to see the private key yourself. You should only share the public key with the server you want to login to.

1.2.6 Sharing your public key

ssh-copy-id

Finally, to share the public key with the server, there are usually multiple ways. If the server allows you to login using a password, you can simply use the `ssh-copy-id` command. This command will take your username and password to login to the server, and then copy the public key which you provide to the server.

```
$ ssh-copy-id -i /key/to/public/key username@ipaddress
```

Remark 1.2.2 The `-i` flag is used to specify the path to the public key. You can drop the `.pub` from the path as well (making it the path to the private key), since `ssh-copy-id` will automatically look for the public key. However, this flag is not required if you are using the default location. This is why using the default location is recommended for beginners. The simplified syntax then becomes

```
  $ ssh-copy-id username@ipaddress
```

The same applies for logging into the server using the `ssh` command.

manual install

However, most servers do not allow password login at all, since it defeats the purpose of using a public-private key pair. In such cases, you need to somehow copy the public key to the server.

If you have physical access to the server, you can simply copy the public key to the server in the `/.ssh/authorized_keys` file of the server.

```
$ file ~/someoneskey.pub
~/someoneskey.pub: OpenSSH ED25519 public key
$ cat ~/someoneskey.pub >> ~/.ssh/authorized_keys
```

Remark 1.2.3 Make sure to use the » operator and not the > operator. The » operator appends the contents of the file to the end of the file, while the > operator overwrites the file, we do not want that.

System Commands Course

However, in case of our course, you do not have access to the server as well. To submit your **public key**, you have to login into the website https://se2001.ds.study.iitm.ac.in/passwordless using your institute credentials, and then submit your public key in the form provided.

You can print out the contents of the public key using the `cat` command and copy the contents into the form.

```
[test1@rex ~]$ cat .ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDxh5EuvzQkGvsqlMQW3rOkY+wyo+2
    d6Y5CSqNGlLs2a test1@rex
```

You should copy the entire contents of the file, including your username and hostname.

1.2.7 How to login to a remote server

You can then login into the server using the `ssh` command.

```
$ ssh rollnumber@se2001.ds.study.iitm.ac.in
```

OR, if not using the default location of key

```
$ ssh -i /path/to/private/key rollnumber@se2001.ds.study.iitm.ac.
    in
```

If successful, you will be logged into the server and the prompt will change to the server's prompt.

```
[test1@rex ~]$ ssh 29f1001234@se2001.ds.study.iitm.ac.in
Last login: Mon Jun  3 07:43:22 2024 from 192.168.2.3
29f1001234@se2001:~$
```

Notice that the prompt has changed from `test1@rex` which was the prompt of your local machine, to `rollnumber@se2001` which is the prompt of the server.

1.2.8 Call an exorcist, there's a daemon in my computer

What is `sshd`? It is a daemon.

Definition 1.2.1 (Daemon) In multitasking computer operating systems, a daemon is a computer program that runs as a background process, rather than being under the direct control of an interactive user.

There are many daemons running in your computer. You can use `systemctl status` to see the loaded and active daemons in your computer.

```
$ systemctl status
* rex
    State: running
    Units: 419 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Thu 2024-06-13 12:55:42 IST; 7h ago
  systemd: 255.6-1-arch
   CGroup: /
           |-init.scope
           | '-1 /usr/lib/systemd/systemd --switched-root --system
     --deserialize=43
           |-system.slice
           | |-NetworkManager.service
           | | '-547 /usr/bin/NetworkManager --no-daemon
           | |-adb.service
           | | '-558 adb -L tcp:5037 fork-server server --reply-fd
     4
           | |-avahi-daemon.service
           | | |-550 "avahi-daemon: running [rex.local]"
           | | '-557 "avahi-daemon: chroot helper"
           | |-cronie.service
           | | '-621 /usr/sbin/crond -n
           | |-cups.service
           | | '-629 /usr/bin/cupsd -l
           | |-dbus-broker.service
           | | |-545 /usr/bin/dbus-broker-launch --scope system --
    audit
```

Here you can see some of the important daemons running, such as `NetworkManager` which is used to manage the network connections, `cronie` which is used to run scheduled tasks, `cups` which is used to manage printers, etc.

sshd

sshd is the daemon that runs on the server and listens to any incoming SSH connections. It is the daemon that lets you login into the server using the SSH protocol.

Your own system might not be running the `sshd` daemon, since you are not running a server. However, you can check if the `sshd` daemon is running using the `systemctl` command.

```
$ systemctl status sshd
* sshd.service - OpenSSH Daemon
     Loaded: loaded (/usr/lib/systemd/system/sshd.service;
    disabled; preset: disabled)
     Active: inactive (dead)
```

Here you can see that the `sshd` daemon is currently inactive. This is because I am not running a server and don't usually login remotely to my system. However, the output of the same command would be something like as shown below if it is enabled on your system.

```
$ systemctl status sshd
```

```
* sshd.service - OpenSSH Daemon
     Loaded: loaded (/usr/lib/systemd/system/sshd.service;
    disabled; preset: disabled)
     Active: active (running) since Thu 2024-06-13 19:48:44 IST;
    12min ago
   Main PID: 3583344 (sshd)
      Tasks: 1 (limit: 9287)
     Memory: 2.1M (peak: 2.3M)
        CPU: 8ms
     CGroup: /system.slice/sshd.service
             '-3583344 "sshd: /usr/bin/sshd -D [listener] 0 of
    10-100 startups"

Jun 13 19:48:44 rex systemd[1]: Started OpenSSH Daemon.
Jun 13 19:48:45 rex sshd[3583344]: Server listening on 0.0.0.0
    port 22.
Jun 13 19:48:45 rex sshd[3583344]: Server listening on :: port 22.
```

If we run the same command on the server, we can see that it is running. However, we wont be able to read the logs of the server, since we are not authorized.

```
$ ssh username@se2001.ds.study.iitm.ac.in
username@se2001:~$ systemctl status sshd
* ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled;
    vendor preset: enabled)
     Active: active (running) since Thu 2024-06-13 12:32:47 UTC; 1
    h 57min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 732 ExecStartPre=/usr/sbin/sshd -t (code=exited,
    status=0/SUCCESS)
   Main PID: 745 (sshd)
      Tasks: 1 (limit: 4557)
     Memory: 22.0M
        CPU: 8.769s
     CGroup: /system.slice/ssh.service
             '-745 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100
     startups"

Warning: some journal files were not opened due to insufficient
    permissions.
```

Remark 1.2.4 Notice that there are some differences in the output when run from my local system and from the system commands server. Such as, the name of the service is `ssh` on the server, while it is `sshd` on my local system. Also the full name is `OpenBSD Secure Shell server` on the server, while it is `OpenSSH Daemon` on my local system. The path of the service file is also different. This is because the server is running a ubuntu distribution whereas my local system runs an arch distribution. They have different packages for ssh, and hence the differences.

I have set the `LC_ALL` environment variable to the locale C while generating the above outputs to prevent latex errors. Ideally if you run the command, you will see a prettier unicode output.