**Name: Abinash Satapathy**

**Reg. No.: 16BCE0081**

**Slot: L27 + L28**

**Sub: Networking Lab (CSE1004)**

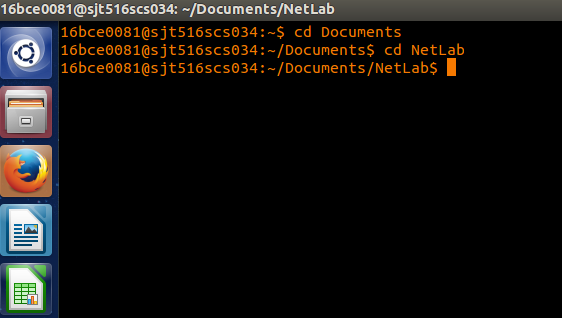
**Basic Linux commands**

1. Command: cd

Description: Change to new directory

Syntax: cd Documents

Output:

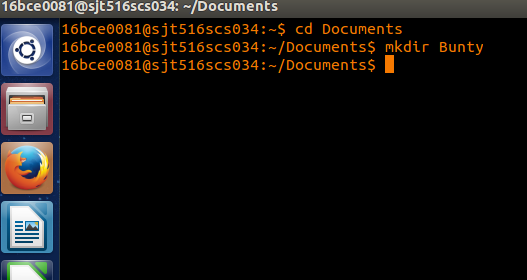


2. Command: mkdir

Description: create new directory

Syntax: mkdir Bunty

Output:

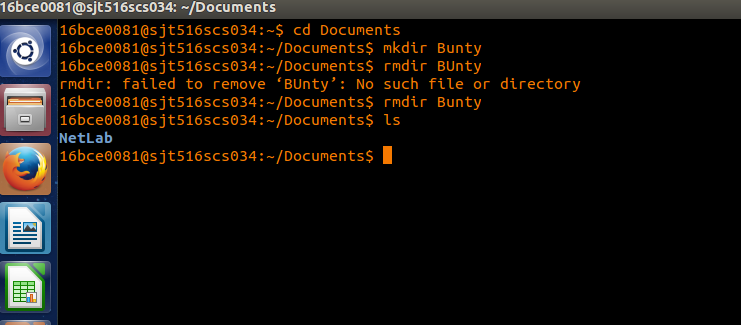


3. Command: rmdir

Description: remove empty directory (remove files first)

Syntax: rmdir Bunty

Output:



4. Command: mv

Description: change name of directory

Syntax: mv <Source> <Dest>

5. Command: pwd

Description: show current directory

Syntax: pwd

6. Command: date

Description: show date and time

Syntax: date

7. Command: history

Description: list of previously executed commands

Syntax: history

8. Command: cal

description: Prints a calendar for the specified month of

the specified year.

Syntax: cal <mon> <year>

9. Command: man

description: show online documentation by program name

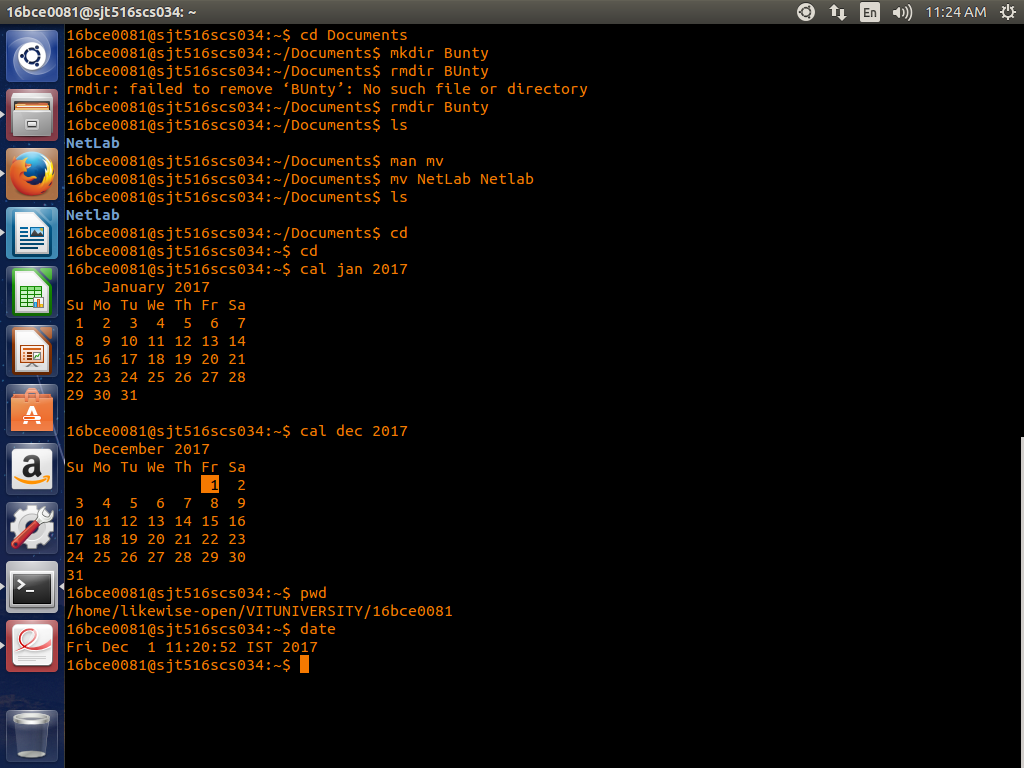
Syntax: man <command>

10. Command: w

Description: who is on the system and what they are doing

Syntax: w

Outputs from 4 to 10



11. Command: whoami

Description: who is logged onto this terminal

Syntax: whoami

12. Command: uptime

Description; show one line summary of system status

Syntax: uptime

13. Command: tty

Description: know the terminal name

Syntax: tty

14. Command: uname

Description: print system information

Syntax: uname

15. Command: cat

Description: view files

Syntax: cat <filename>

16. Command: ls

Description: list files in a directory and their attributes

Syntax: ls

17. Command: vi

Description: full-featured screen editor for modifying text files

Syntax: vi

18. Command: echo $$

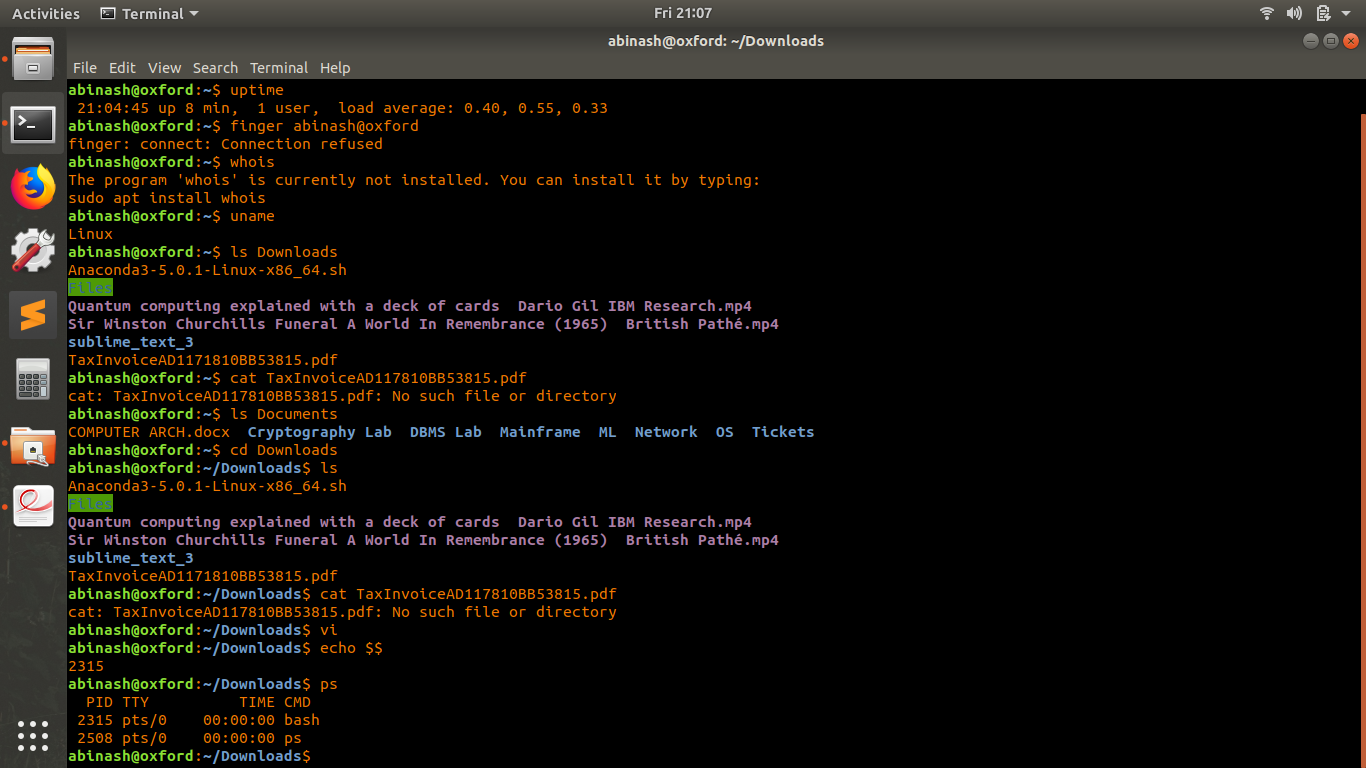
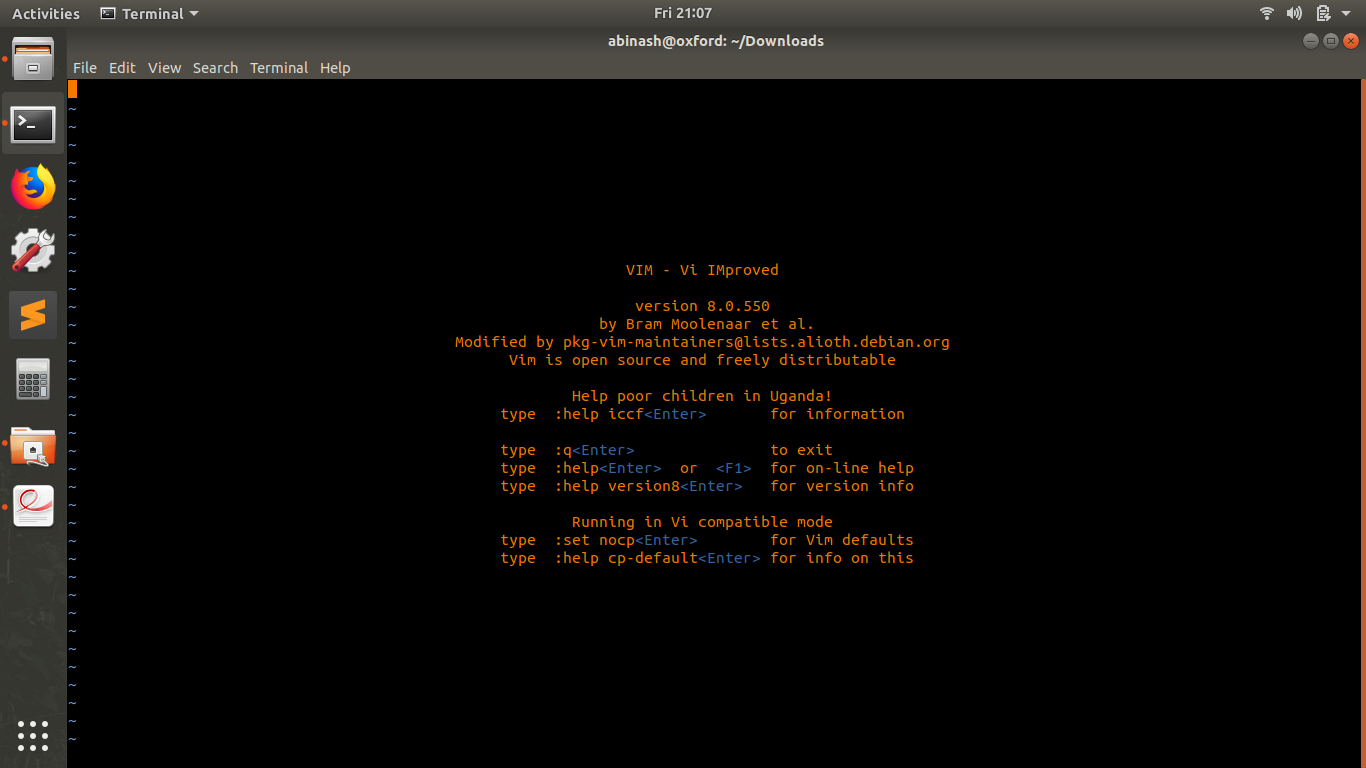
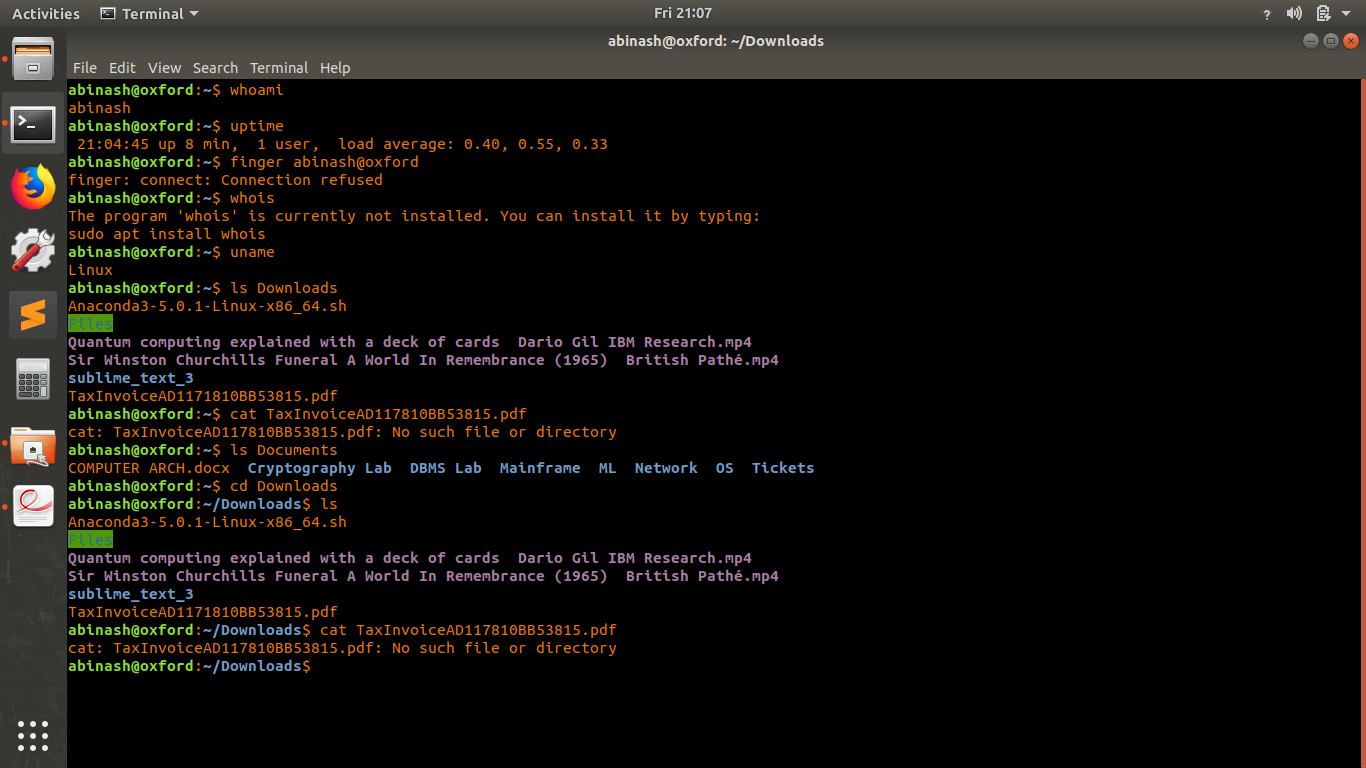
Description: process id of current shell

Syntax: echo $$

19. Command: ps

Description: process status

Syntax: ps



**Basic network commands**

1. ping

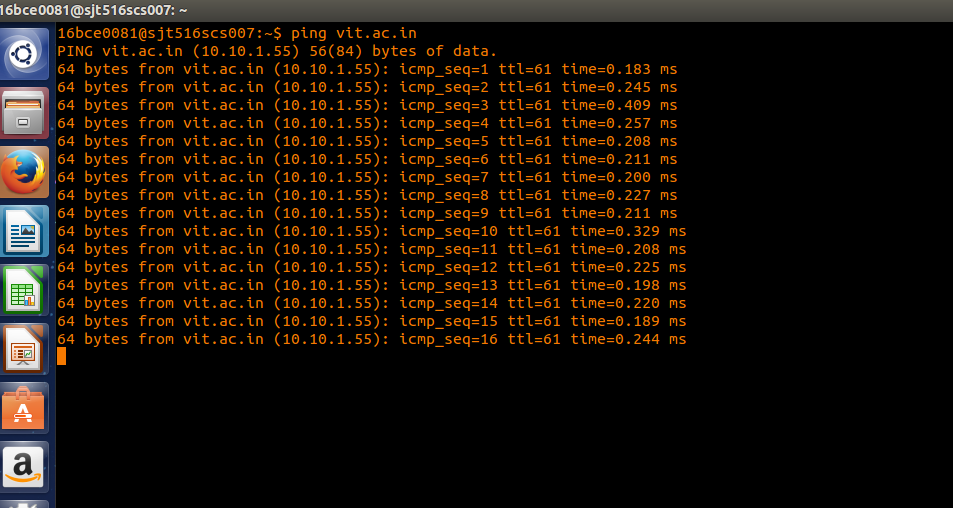
Description: sends echo requests to the host specified on the command line, and lists the responses

received.

Command name: ping

Command Syntax: ping ipAddress or hostname

Output:



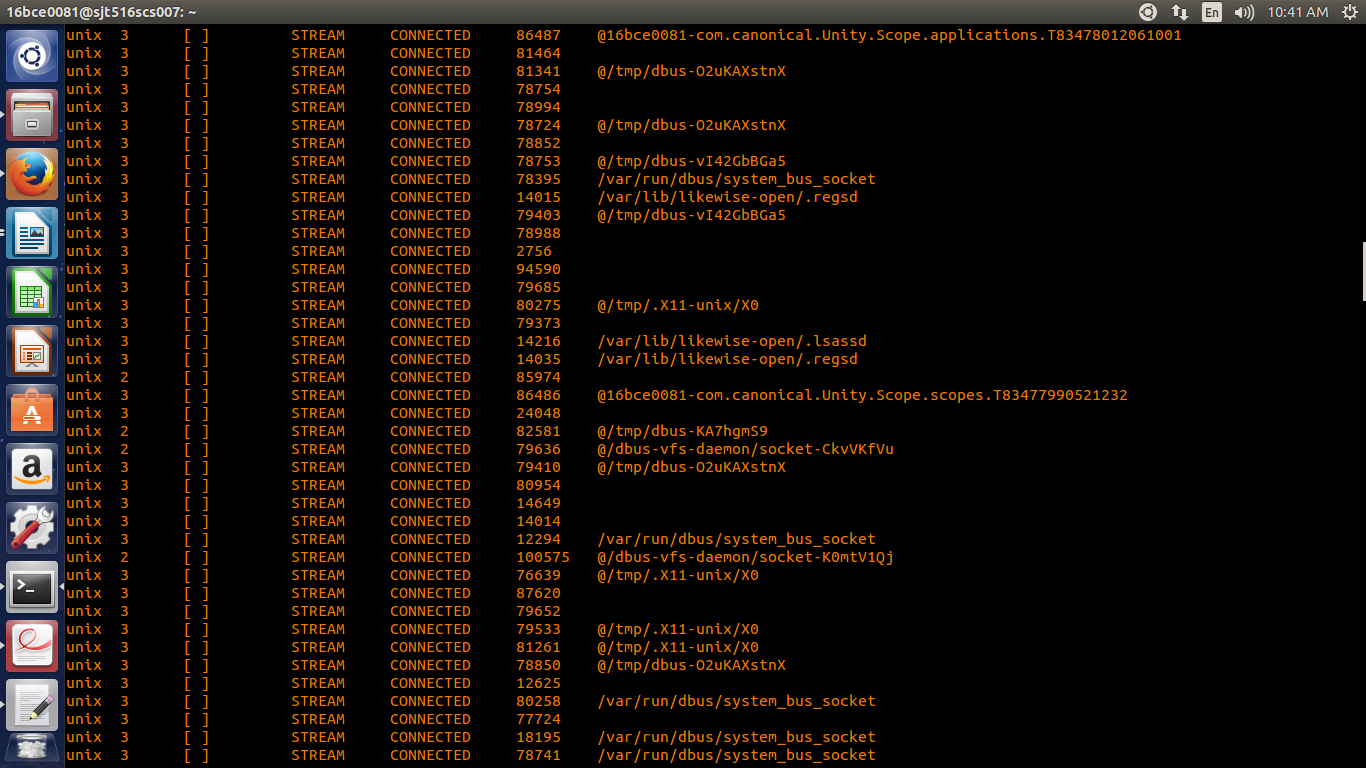
2. netstat

Description: It will tell us what the status of ports are ie. open, closed, waiting connections. It is used to display the TCP/IP network protocol statistics and information.

Command name: netstat

Command Syntax: netstat

Output:



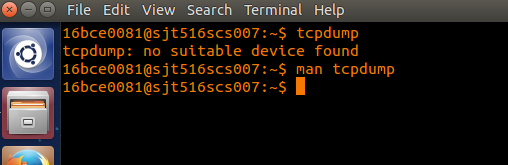
3. tcpdump

Description: This is a sniffer, a program that captures packets off a network interface and interprets them.

Command name: tcpdump

Command Syntax: tcpdump

Output:



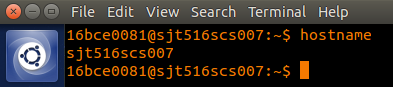
4. hostname

Description: Tells the user the host name of the computer they are logged into.

Command name: hostname

Command Syntax: hostname

Output:



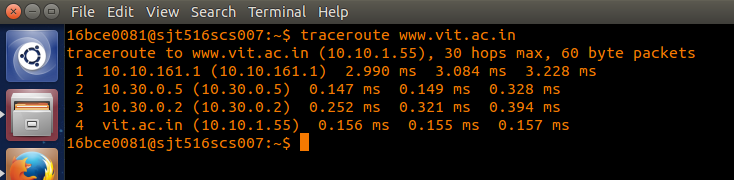
5. traceroute

Description: traceroute will show the route of a packet. It attempts to list the series of hosts through which our packets travel on their way to a given destination.

Command name: traceroute

Command Syntax: traceroute machineName or ip

Output:



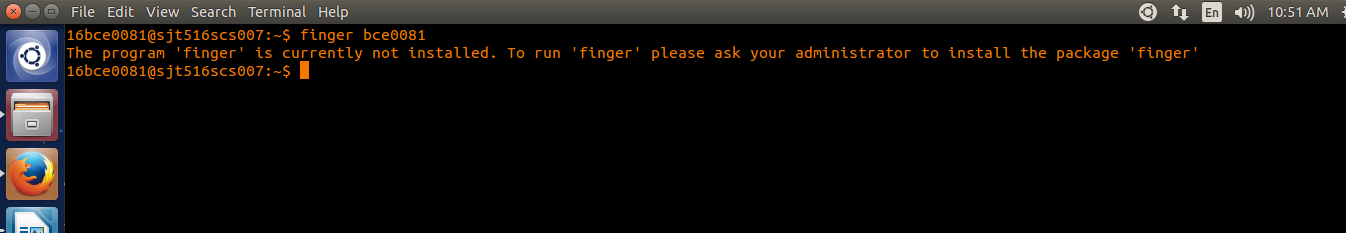
6. finger

Description: Retrieves information about the specified user.

Command name: finger

Command Syntax: finger bce0081

Output:



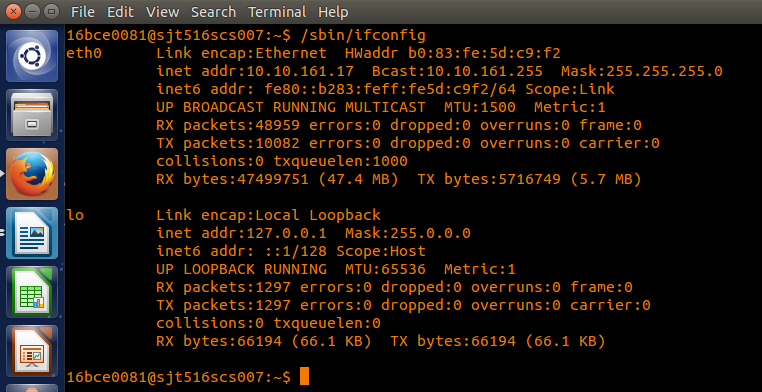
7. ifconfig

Description: This command is used to configure network interfaces, or to display their current configuration.

Command name: ifconfig

Command Syntax: ifconfig

Output:



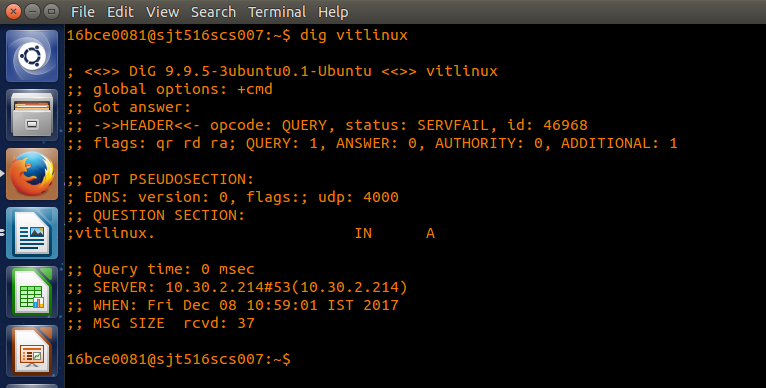
8. dig

Description: The "domain information groper" tool. If a hostname is given as an argument, it outputs information about that host, including it's IP address, hostname and various other information.

Command name: dig

Command Syntax: dig vitlinux

Output:



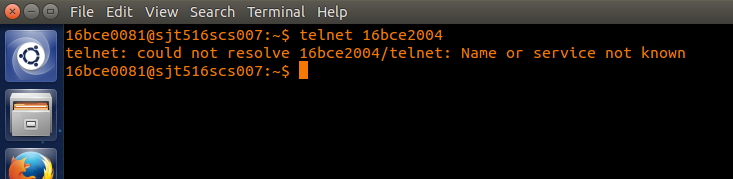
9. telnet

Description: telnet allows you to log in to a computer, just as if you were sitting at the terminal. Once your username and password are verified, you are given a shell prompt. From here, you can do anything requiring a text console.

Command name: telnet

Command Syntax: telnet 16bce2004

Output:



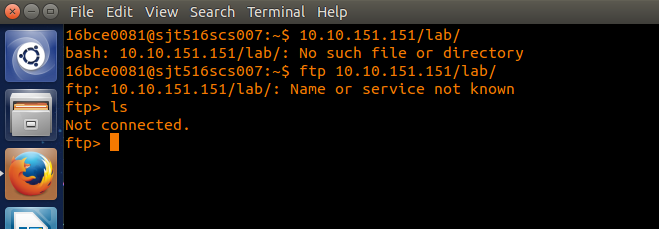
10. ftp

Description: To connect to an FTP server.

Command name: ftp

Command Syntax: ftp 10.10.151.151/lab/

Output:



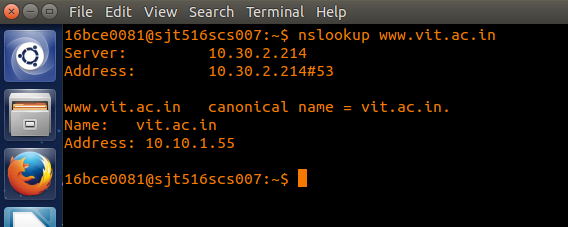
11. nslookup

Description: returns the ipaddress of the given hostname and vice versa.

Command name: nslookup

Command Syntax: nslookup www.vit.ac.in

Output:



**Switch**

A network switch (also called switching hub, bridging hub, officially MAC bridge) is a computer networking device that connects devices together on a computer network by using packet switching to receive, process, and forward data to the destination device.

Eg.

1. Cisco small business SG300-28 28-port Gigabit Ethernet rackmount switch
2. D-Link 24-Port 10/100 Mbps Unmanaged Fast Ethernet Switch DES-1024A Network Switch
3. TP-LINK TL-SG1024D 24-Port Gigabit Desktop Switch
4. Cisco Catalyst WS-C3650-24TS-L Ethernet Switch

**Bridge**

A network bridge is a computer networking device that creates a single aggregate network from multiple communication networks or network segments.

Eg.:

1. Quadro M32X
2. AXIS C8033 Network Audio Bridge
3. Cisco 1410 Series Bridge
4. G.Shdsl 4 Wire Router Bridge

**Hub**

A hub is a component of a network with a high-degree node. Hubs have a significantly larger number of links in comparison with other nodes in the network.

Eg.:

1. SIGARAM 4 Port Usb Hub 2.0 Ultra
2. TECH SHOP 4 Port Ultra High Speed USB Hub 480 Mbps
3. eErlik 4 Port Ultra High Speed USB Hub 480 Mbps
4. Quantum 4 Port USB Hub

**Router**

A router is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet. A data packet is typically forwarded from one router to another router through the networks that constitute an internetwork until it reaches its destination node.

Eg.:

1. TP-LINK TL-WR740N Router
2. iball WRA300N3GT 300 Mbps Wireless 3G Router
3. TP-LINK TD-W8960N 300Mbps Wireless N ADSL2+ Modem Router
4. Alcatel MW40CJ 4G Wi-Fi Router

**Gateways**

A gateway is a node (router) in a computer network, a key stopping point for data on its way to or from other networks.

**Eg.:**

1. Maretron USB100 Nmea 2000 USB Gateway
2. Actisense NMEA Networks USB Gateway
3. Cisco VG248 - gateway ( VG248 )
4. LORA LORAWAN INDUSTRIAL GATEWAY 868MHZ

**Error Detection**

**Parity Check**

#include <stdio.h>

int getParity(int n)

{

int parity = 0;

while (n)

{

parity = !parity;

n = n & (n - 1);

}

return parity;

}

int convert(int dec){

if (dec == 0)

return 0;

else

return (dec % 2 + 10 \* convert(dec / 2));

}

int sender(){

int num;

printf("Enter a number for parity check\n");

printf("-----------------------------\n");

scanf("%d", &num);

return num;

}

void receiver(int m){

int result = getParity(m);

printf("-----------------------------\n");

printf("Binary equivalent of %d = %d\n", m, convert(m));

printf("-----------------------------\n");

if(result == 1)

printf("Parity of %d = odd\n", m);

else

printf("Parity of %d = even\n", m);

}

int main(){

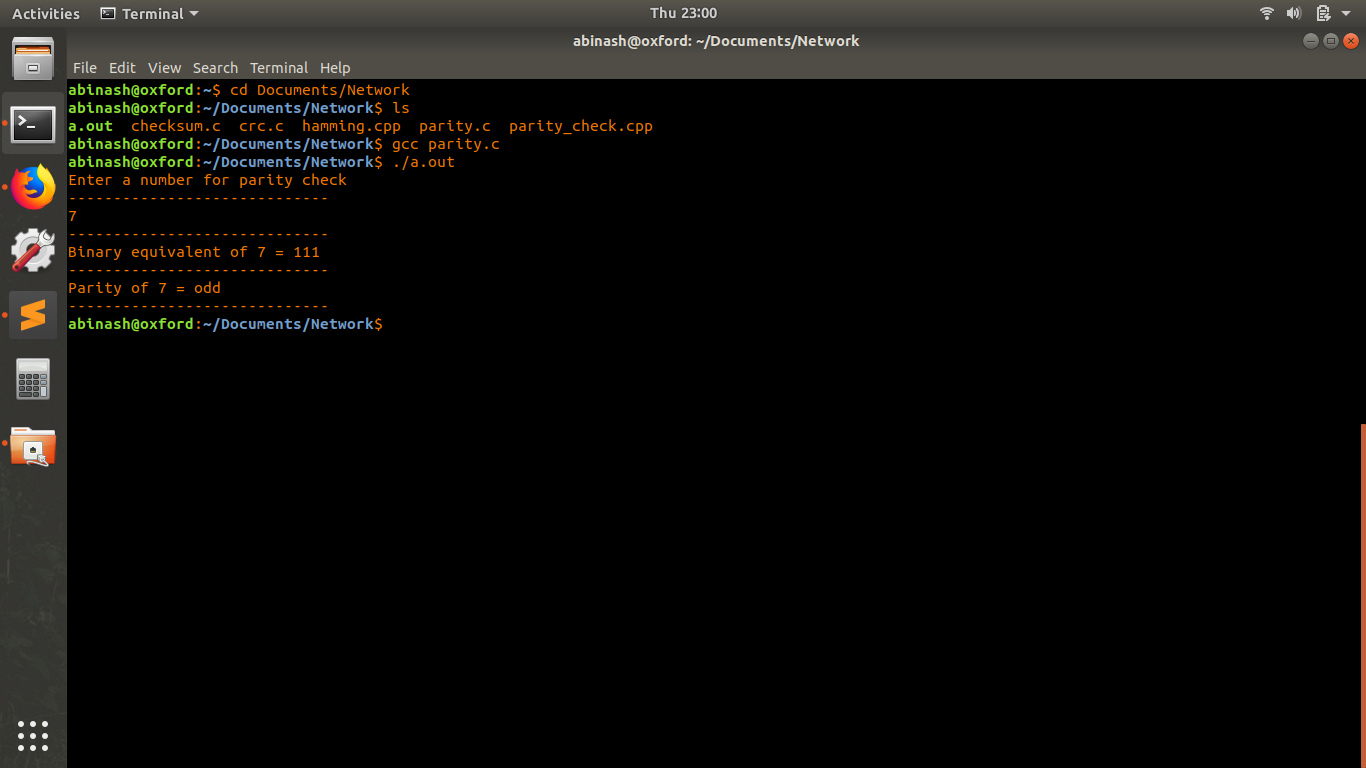
int sender\_val = sender();

receiver(sender\_val);

printf("-----------------------------\n");

return 0;

}



**CRC**

#include<stdio.h>

#include<string.h>

#define N strlen(g)

char t[28],cs[28],g[]="10001000000100001";

int a,e,c;

void xor(){

for(c=1;c<N;c++)

cs[c] = ((cs[c] == g[c])?'0':'1');

}

void crc(){

for(e=0;e<N;e++)

cs[e]=t[e];

do{

if(cs[0]=='1')

xor();

for(c=0;c<N-1;c++)

cs[c]=cs[c+1];

cs[c]=t[e++];

}while(e<=a+N-1);

}

void sender(){

printf("\nEnter data (only in binary format): ");

scanf("%s",t);

printf("\n----------------------------------------");

printf("\nGeneratng polynomial : %s",g);

a=strlen(t);

for(e=a;e<a+N-1;e++)

t[e]='0';

printf("\n----------------------------------------");

printf("\nModified data is : %s",t);

printf("\n----------------------------------------");

}

void testing(){

printf("\nChecksum is : %s",cs);

for(e=a;e<a+N-1;e++)

t[e]=cs[e-a];

printf("\n----------------------------------------");

printf("\nFinal codeword is : %s",t);

printf("\n----------------------------------------");

printf("\nTest error detection 0(yes) 1(no)? : ");

scanf("%d",&e);

}

void receiver(){

if(e==0){

do{

printf("\nEnter the position where error is to be inserted : ");

scanf("%d",&e);

}while(e==0 || e>a+N-1);

t[e-1]=(t[e-1]=='0')?'1':'0';

printf("\n----------------------------------------");

printf("\nErroneous data : %s\n",t);

}

}

int main(){

sender();

crc();

testing();

receiver();

crc();

for(e=0;(e<N-1) && (cs[e]!='1');e++);

if(e<N-1)

printf("\nError detected\n\n");

else

printf("\nNo error detected\n\n");

printf("\n----------------------------------------\n");

return 0;

}



**Checksum**

#include<stdio.h>

#include<math.h>

int sender(int b[10],int k)

{

int checksum,sum=0,i;

printf("\n\*\*\*\*SENDER\*\*\*\*\n");

printf("----------------------------------\n");

for(i=0;i<k;i++)

sum+=b[i];

printf("SUM IS: %d",sum);

printf("-------------\n");

checksum=~sum;

printf("\nSENDER's CHECKSUM IS:%d",checksum);

printf("-------------------------------\n");

return checksum;

}

int receiver(int c[10],int k,int scheck)

{

int checksum,sum=0,i;

printf("\n\n\*\*\*\*RECEIVER\*\*\*\*\n");

printf("-------------------------------\n");

for(i=0;i<k;i++)

sum+=c[i];

printf(" RECEIVER SUM IS: %d",sum);

printf("---------------------------\n");

sum=sum+scheck;

checksum=~sum;

printf("\nRECEIVER's CHECKSUM IS: %d",checksum);

printf("----------------\n");

return checksum;

}

int main()

{

int a[10],i,m,scheck,rcheck;

printf("\nENTER SIZE OF THE STRING:");

scanf("%d",&m);

printf("\nENTER THE ELEMENTS OF THE ARRAY:");

for(i=0;i<m;i++)

scanf("%d",&a[i]);

scheck=sender(a,m);

rcheck=receiver(a,m,scheck);

if(rcheck==0)

printf("\n\nNO ERROR IN TRANSMISSION\n\n");

else

printf("\n\nERROR DETECTED");

return 0;

}



**Error Correction**

**Hamming Code**

#include<iostream>

using namespace std;

int main() {

int data[10];

int dataatrec[10],c,c1,c2,c3,i;

cout<<"Enter 4 bits of data one by one\n";

cin>>data[0];

cin>>data[1];

cin>>data[2];

cin>>data[4];

//Calculation of even parity

data[6]=data[0]^data[2]^data[4];

data[5]=data[0]^data[1]^data[4];

data[3]=data[0]^data[1]^data[2];

cout<<"\nEncoded data is\n";

for(i=0;i<7;i++)

cout<<data[i];

cout<<"\n\nEnter received data bits one by one\n";

for(i=0;i<7;i++)

cin>>dataatrec[i];

c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];

c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];

c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];

c=c3\*4+c2\*2+c1 ;

if(c==0) {

cout<<"\nNo error while transmission of data\n";

}

else {

cout<<"\nError on position "<<c;

cout<<"\nData sent : ";

for(i=0;i<7;i++)

cout<<data[i];

cout<<"\nData received : ";

for(i=0;i<7;i++)

cout<<dataatrec[i];

cout<<"\nCorrect message is\n";

//if errorneous bit is 0 we complement it else vice versa

if(dataatrec[7-c]==0)

dataatrec[7-c]=1;

else

dataatrec[7-c]=0;

for (i=0;i<7;i++) {

cout<<dataatrec[i];

}

}

return 0;

}

