0:Unix Command 16-01-2020

**Problem 0 : Unix Command**

# AIM:

Try out the following unix commands(use manual and help features for support

1. echo, read

2. more, less

3. man

4. chmod, chown

5. cd, mkdir, pwd, ls, find

6. cat, mv, cp, rm

7. wc, cut, paste

8. head, tail, grep, expr

9. Redirections & Piping

10. useradd, usermod, userdel, passwd

11. tar

**PROGRAM CODE:**

|  |  |  |
| --- | --- | --- |
| echo , read | Echo displays line of text or strings that are passes as arguments. | |
|  | Syntax : Echo [string] |  |
|  | Read is used to get input from keyboard. | |
|  | Syntax : Read text |  |
| more , less | More is used to view the text file in command prompt displaying one screen | |
|  | at a time in case file is large. |  |
|  | Syntax : more -d sample.txt |  |
|  | Less is used to read contents of text file in page per time. | |
|  | Syntax : sudo | less |  |
| man | Man displays man pages for commands manmeors manual for command. | |
|  | Syntax : man chown |  |
| chmod, | Chmod set the permission flag . |  |
| chown | Syntax : chmod 766 ex.txt |  |

|  |  |  |
| --- | --- | --- |
|  | Chown allow change owner and group owner of file. | |
|  | Syntax : sudo chown clave : mary example.txt | |
|  |  |  |
| cd, mkdir, | Cd changes current directory. |  |
| pwd, ls, find | Syntax : cd directory\_name |  |
|  | Mkdir creates new directory . |  |
|  | Syntax : mkdir invoice |  |
|  | Pwd prints workind directory from root directory | |
|  | Syntax : pwd |  |
|  | Ls lists file and folder in directory |  |
|  | Syntax : ls -l |  |
|  | Find tracks down file. |  |
|  | Syntax : find . name \* ones \* |  |
|  |  | |
| cat, mv, cp, | Cat concatenates file and print to stdout. | |
| rm | Syntax : cat [OPTION]..[FILE] |  |
|  | Mv moves file or rename files. |  |
|  | Syntax : mv [OPTION]source |  |
|  | Cp copy files. |  |
|  | Syntax : cp [OPTION] source |  |
|  | Rm removes files and directories. |  |
|  | Syntax:rm [OPTION]…[FILE] |  |
|  |  | |
| wc, cut, | Wc is used for printing newline , word and byte counts for files. | |
| paste | Cut is used for cutting out sections for each line of files and writes result to | |
|  | standard output. |  |
|  | Paste is used to join files horizontally by outputting lines. | |
|  |  | |
| head, tail, | Head command prints lines from beginning of a file and tail prints lines from | |
| grep, expr | end of file. |  |
|  | Grep searches through a set of files for arbitrary text pattern through regular | |
|  | expression.Expr evaluates a given expression and displays corresponding | |
|  | output. |  |
|  |  | |
| Redirections, | Redirection is a feature when executes a command,we can change input or | |
| Piping | output devices.Piping is a form of redirection ie used in linux to send output | |
|  | of one command for further processing. | |
|  |  | |
| useradd, | Useradd is used to create new accounts in linux. | |
| usermod, | Usermod used to modify existing accounts in linux. | |
| userdel, | Userdel is used to delete account in linux. | |
| passwd | Passwd is used to assign password to local accounts of users. | |
|  |  | |
| tar | Tar stands for tap to achieve which is used to tape drive back up command | |
|  | used by linux. |  |
|  | Syntax : tar [OPTIONS] [ARCHIEVE-FILE] [FILE OD DIRECTORY TO BE | |
|  | ACHIEVED] |  |
|  |  |  |

1:Unix Command 16-01-2020

1:Welcome message 20-01-2020

**Problem 1 : Welcome Meassage**

# AIM:

Print a customized welcome message. Get the name of the user as input and attach the name to the welcome message. Eg. “Welcome Rahul”.

**PROGRAM CODE:**

|  |
| --- |
| #!/bin/bash  read -p “enter your name“ name  echo “welcome” $name |

# RESULT

enter your name

saniya

Welcome saniya

2:Largest Number 20-01-2020

**Problem 2: Largest Number**

# AIM:

Take 2 numbers as input and print the greater of the two

**PROGRAM CODE:**

|  |
| --- |
| #!/bin/bash  read -p “enter a number“ n1  read -p “enter a number“ n2  if [$n1 –gt $n2]  then  echo “largest number is:”$n1  else  echo “largest number is:”$n2  fi |

# RESULT

enter a number

5

enter a number

8

largest number is:8

3:Odd Numbers 23-01-2020

**Problem 3: Odd Numbers**

# AIM:

Print the first 20 odd numbers

**PROGRAM CODE:**

|  |
| --- |
| #!/bin/bash  i=1  c=0  echo “Odd Numbers are:”  while[$i –lt 100]  do  echo $i  c=`expr $c+1`  if[$c –eq $20]  then  break  fi  i=`expr $i+2`  done |

# RESULT

Odd numbers are:

1

3

5

7

9

11

13

15

17

19

21

23

25

27

29

31

33

35

37

39

4:Array sum 23-01-2020

**Problem 4 : Array Sum**

# AIM:

Store 20 numbers in an array and print their sum

**PROGRAM CODE:**

|  |
| --- |
| #!/bin/bash  read -p “enter 20 numbers“ input  sum=0  for i in ${input[@]}  do  sum=`expr $sum+$i`  done  echo “the sum is:”$sum |

# RESULT

enter 20 numbers

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

The sum is:210

5:Text File 03-02-2020

**Problem 5 : Text File**

# AIM:

Create a text file with 20 lines of text

**PROGRAM CODE:**

(In terminal)

|  |
| --- |
| $Cat > file.txt  a  s  d  w  f  g  h  j  j  k  l  q  w  w  e  r  t  y  u  hi |

# Once you have entered all the text, hit enter to move to the next line and then use the Ctrl+D control to tell the system that you are done with entering the text

# RESULT

File created in the directory.Cat displays the content

6:Welcome message 03-02-2020

**Problem 6 : Welcome Meassage**

# AIM:

Open the file created in question 5 and replace any string with another without using stream editor

**PROGRAM CODE:**

|  |
| --- |
| #!/bin/bash  While read a;do  eco ${a//hi/hello}  done < file.txt > file1.txt.t  mv file.txt{.t,} |

# RESULT

hi is changed to hello

7:File copy 05-02-2020

**Problem 7 : File Copy**

# AIM:

Open the /etc/protocols file and copy the protocol number of the following protocols into another file named "favorite protocols" and format it in the same way as the original /etc/protocol file.

1. udp

2. idrp

3. skip

4. ipip

**PROGRAM CODE:**

|  |
| --- |
| #!/bin/bash  grep “udp\|idrp\|skip\|ipip”/etc/protocols > p.txt  cat p.txt |

# RESULT

udp 17 UDP

idrp 45 IDRP

skip 57 SKIP

ipip 94 IPIP

8: At And Batch 05-012-2020

**Problem 8 : at and batch**

# AIM:

Use "at" and "batch" to schedule tasks

**PROGRAM CODE:**

|  |  |
| --- | --- |
| At | #!/bin/bash  echo “sh execute.sh”|at now+2 minut  execute.sh->eho “Hello world” <cr.txt |
| batch | (in terminal)  $batch  Warning: commands will be executed using //bin/sh  At>echo “welcome….”>ou.txt  At>ctrl+d  Job2 at mon jan 27 20:56::00 2020  4cat ou.txt  Welcome… |

# RESULT

Job 31 at mon feb 3 21:55:00 2020

A file specified in excute.sh will create after 2 minute with specified text

9:Cron 16-02-2020

**Problem 9 : Cron**

# AIM:

Use cron to schedule taks

**PROGRAM CODE:**

|  |
| --- |
| (in terminal)  cronetab –e  (In editor)  40 \*\*\*\* cd /home/Downloads/mca23/sdlab && sh execute.sh |

# RESULT

When time becomes 49 minutes execute.sh will execute. In exeute.sh we want to crete a file named hi.txt with text “hello”

10:Send Email 16-02-2020

**Problem 10 : Send Email**

# AIM:

Set up unix mail and use mail to send and receive mails to and from users using shell scripting

**PROGRAM CODE:**

|  |
| --- |
| #!/bin/bash  Echo “hi saniya”|mail-s  “subject of the message” saniyathahseen@gmail.com |

# RESULT

A mail is send to given mail id

11:Git Operations 23-02-2020

**Problem 11 : Git Operations**

# AIM:

0. Install and initialize git and perform the following operations

a. Create a text file in your git directory.

b. Configure your git with your credentials.

c. Configure the default editor to your favorite editor

d. Stage your files

e. Create your first commit

f. Push to remote repository

The following exercises must be done by a team of four students.

1. Create team account.

2. Create empty repository in any git remote repository service and add collaborators.

3. Leader must create the first commit.

4. All members must clone the remote repository.

5. Each member must create a feature brach each and add features to them(any mod)

6. Commit changes to branches.

7. Push the branches.

8. View Graph.

9. Leader must make changes to the master.

10. All member must rebase their branches to the position of latest commit in master.

10. Merge all branches to master.

11. Cherry pick commits from each branch created earlier.

12. View Status.

13. View History.

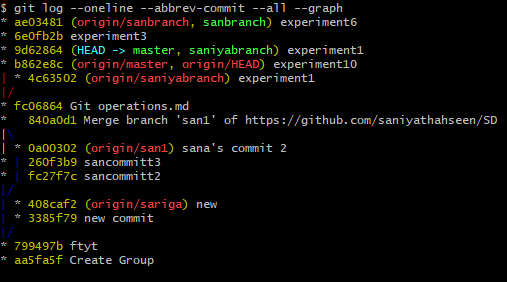
14. Delete all branches.

**Commands:**

|  |
| --- |
| * git clone <https://github.com/saniyathahseen/SD.git> * git add exp1.sh * git commit -m "Experiment 1" * git pull * git push origin master * git checkout -b saniya * git add exp2.sh * git commit -m "Experiment 2" * git push origin saniya * git log --graph * git rebase master saniyabranch * git checkout master * git merge saniyabranch * git cherry-pick fc06864 * git status * git log * git branch | grep -v '^\*' | xargs git branch -D |

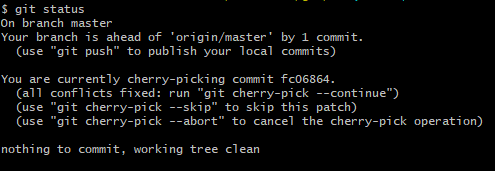
**RESULT:**

Graphical representation of git repository

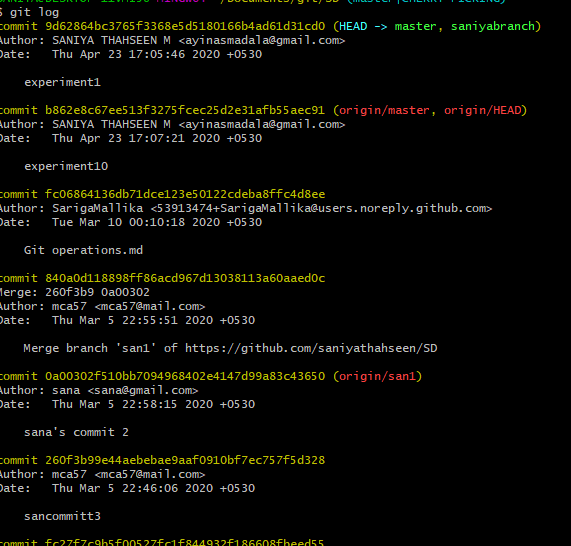


11.1

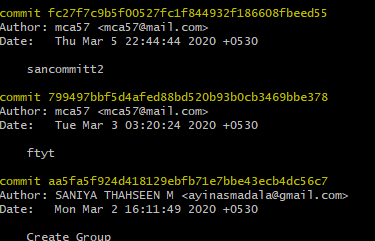
Git status



11.2



11.3



11.4

All branches are deleted



11.5

12:TCP client-server 12-03-2020

**Program 12 :TCP Client Server Communication**

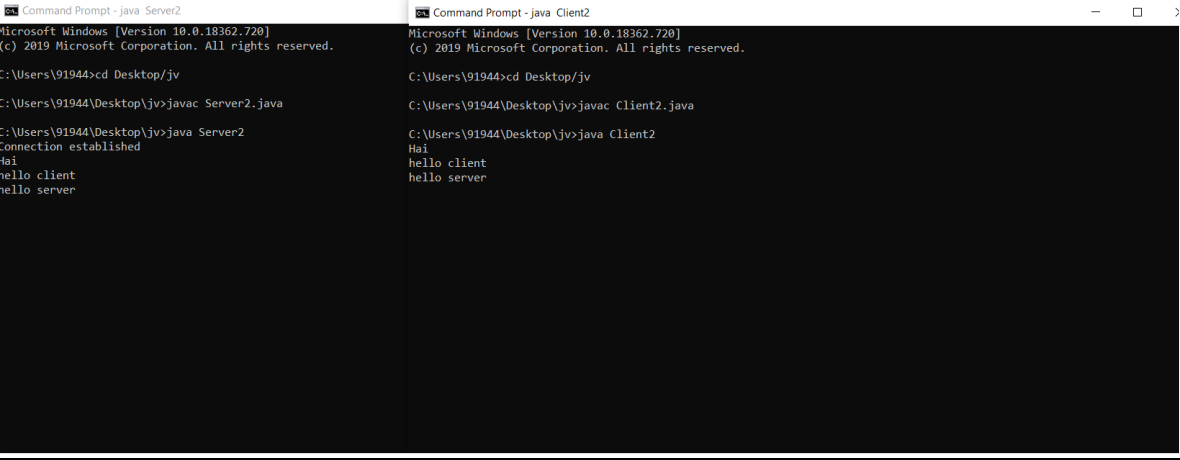
**AIM:**

Implement Bidirectional Client-Server communication using TCP.

**PROGRAM CODE:**

|  |  |
| --- | --- |
| Server.java | // Server2 class that  // receives data and sends data  import java.io.\*;  import java.net.\*;  class Server2 {  public static void main(String args[])  throws Exception  {  // Create server Socket  ServerSocket ss = new ServerSocket(888);  // connect it to client socket  Socket s = ss.accept();  System.out.println("Connection established");  // to send data to the client  PrintStream ps  = new PrintStream(s.getOutputStream());  // to read data coming from the client  BufferedReader br  = new BufferedReader(  new InputStreamReader(  s.getInputStream()));  // to read data from the keyboard  BufferedReader kb  = new BufferedReader(new InputStreamReader(System.in));  // server executes continuously  while (true) {  String str, str1;  // repeat as long as the client  // does not send a null string  // read from client  while ((str = br.readLine()) != null) {  System.out.println(str);  str1 = kb.readLine();  // send to client  ps.println(str1);  }  // close connection  ps.close();  br.close();  kb.close();  ss.close();  s.close();  // terminate application  System.exit(0);  } // end of while  }  } |
| Client.java | // Client2 class that  // sends data and receives also  import java.io.\*;  import java.net.\*;  class Client2 {  public static void main(String args[])  throws Exception  {  // Create client socket  Socket s = new Socket("localhost", 888);  // to send data to the server DataOutputStream dos  =new DataOutputStream( s.getOutputStream());  // to read data coming from the server  BufferedReader br= new BufferedReader(  new InputStreamReader(s.getInputStream()));  // to read data from the keyboard  BufferedReader kb=new BufferedReader(  new InputStreamReader(System.in));  String str, str1;  // repeat as long as exit  // is not typed at client  while (!(str = kb.readLine()).equals("exit")) {  //send to the server dos.writeBytes(str + "\n");  //receive from the server str1 = br.readLine();  System.out.println(str1);  }  //close connection. dos.close();  br.close();  kb.close();  s.close();  }  } |

**RESULT:**

TCP Client and Server communication.

12.1: Communication between local client and server

13:Echo Server with TCP 16-03-2020

**PROGRAM 13 : Echo server with TCP**

**AIM:**

Implement Echo Server using TCP

**PROGRAM CODE:**

|  |  |
| --- | --- |
| Echoserver.java | import java.io.\*;  import java.net.\*;  public class EchoServer  {  public static void main(String args[]) throws Exception  {  try  {  int Port;  BufferedReader Buf =new BufferedReader(new  InputStreamReader(System.in));  System.out.print(" Enter the Port Address : " );  Port=Integer.parseInt(Buf.readLine());  ServerSocket sok =new ServerSocket(Port);  System.out.println(" Server is Ready To Receive a Message. ");  System.out.println(" Waiting ..... ");  Socket so=sok.accept();  if(so.isConnected()==true)  System.out.println(" Client Socket is Connected Succecfully.  ");  InputStream in=so.getInputStream();  OutputStream ou=so.getOutputStream();  PrintWriter pr=new PrintWriter(ou);  BufferedReader buf=new BufferedReader(new  InputStreamReader(in));  String str=buf.readLine();  System.out.println(" Message Received From Client : " + str);  System.out.println(" This Message is Forwarded To Client. ");  pr.println(str);  pr.flush();  }  catch(Exception e)  {  System.out.println(" Error : " + e.getMessage());  }  }  } |
| Echoclient.java | import java.io.\*;  import java.net.\*;  public class EchoClient  {  public static void main(String args[]) throws Exception  {  try {  int Port;  BufferedReader Buf =new BufferedReader(new  InputStreamReader(System.in));  System.out.print(" Enter the Port Address : " );  Port=Integer.parseInt(Buf.readLine());  Socket sok=new Socket("localhost",Port);  if(sok.isConnected()==true)  System.out.println(" Server Socket is Connected Succecfully.  ");  InputStream in=sok.getInputStream();  OutputStream ou=sok.getOutputStream();  PrintWriter pr=new PrintWriter(ou);  BufferedReader buf1=new BufferedReader(new InputStreamReader(System.in)); BufferedReader buf2=new BufferedReader(new InputStreamReader(in)); String str1,str2;  System.out.print(" Enter the Message : ");  str1=buf1.readLine();  pr.println(str1);  pr.flush();  System.out.println(" Message Send Successfully. "); str2=buf2.readLine();  System.out.println(" Message From Server : " + str2);  }  catch(Exception e)  {  System.out.println(" Error : " + e.getMessage());  }  }  } |

**RESULT:**

Communication between echo server and client via TCP



Figure 1 : TCP echo server and client

14:UDP chat server 20-03-2020

**PROGRAM 14: UDP chat Server**

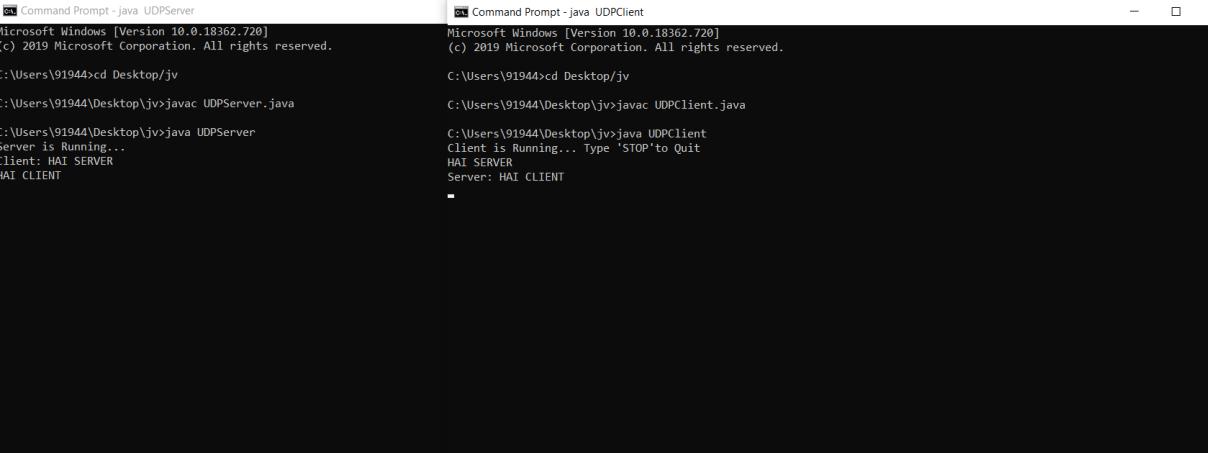
**AIM:**

Implement Chat Server using UDP

**PROGRAM CODE:**

|  |  |
| --- | --- |
| UDPClent.java | import java.io.\*;  import java.net.\*;  class UDPClient  {  public static DatagramSocket clientsocket;  public static DatagramPacket dp;  public static BufferedReader dis;  public static InetAddress ia;  public static byte buf[] = new byte[1024];  public static int cport = 789, sport = 790;  public static void main(String[] a) throws IOException  {  clientsocket = new DatagramSocket(cport);  dp = new DatagramPacket(buf, buf.length);  dis = new BufferedReader(new  InputStreamReader(System.in));  ia = InetAddress.getLocalHost();  System.out.println("Client is Running... Type 'STOP'to Quit");  while(true)  {  String str = new String(dis.readLine());  buf = str.getBytes();  if(str.equals("STOP"))  {  System.out.println("Terminated...");  clientsocket.send(new  DatagramPacket(buf,str.length(), ia,  sport));  break;  }  clientsocket.send(new DatagramPacket(buf,  str.length(), ia, sport));  clientsocket.receive(dp);  String str2 = new String(dp.getData(), 0,  dp.getLength());  System.out.println("Server: " + str2);  }  }  } |
| UDPServer.java | import java.io.\*;  import java.net.\*;  class UDPServer  {  public static DatagramSocket serversocket;  public static DatagramPacket dp;  public static BufferedReader dis;  public static InetAddress ia;  public static byte buf[] = new byte[1024];  public static int cport = 789,sport=790;  public static void main(String[] a) throws IOException {  serversocket = new DatagramSocket(sport);  dp = new DatagramPacket(buf,buf.length);  dis = new BufferedReader  (new InputStreamReader(System.in));  ia = InetAddress.getLocalHost();  System.out.println("Server is Running...");  while(true)  {  serversocket.receive(dp);  String str = new String(dp.getData(), 0,  dp.getLength());  if(str.equals("STOP"))  {  System.out.println("Terminated..");  break;  }  System.out.println("Client: " + str);  String str1 = new String(dis.readLine());  buf = str1.getBytes();  serversocket.send(new  DatagramPacket(buf,str1.length(), ia, cport));  }  }  } |

**RESULT:**

Figure explains chat communication between UDP client and server.

14.1 : UDP server and client communication