# 1. Introduction:

Using UNIX socket programming in connection oriented protocol (TCP) built a mail transfer system. We connect two or more computers. We assume one of them is as a server. Server holds the database about the all accounts and also usernames and passwords.

We divide the whole task in some phases, they are-

1. Login
2. Compose mail
3. Send mail
4. Forward mail
5. Display the mail
6. Delete mail
7. Create mail account
8. Log out

Used Data Structure: String(character type array) this data structure is built in Unix.

Data types are used: integer(int), character(char), constant character(const char).

Now we shall discuss about the phases-->

1. **Login**: First, we ask username & password from user. We, then check validity of given username-password combination. If they are valid, we give access to that account to the user.
2. **Compose mail**: If user choose the option to compose mail, then we invoke a text-editor. User can write his/her mail there and send it to the desired destination (any person’s or organization’s e-mail address),and then we save that mail file to sent-mail folder.
3. **Send mail**: If user chooses the option for sending mail, we ask for the user account if given the mail is sent to the corresponding account located in server machine.
4. **Forward mail**: If there is a mail on user’s account and if he/she choose the option ‘forward mail’ then we ask for the mail number of the mail to be forwarded and receiver’s user account, if given then the mail is sent to the corresponding account located in the server computer.
5. **Display the mail**: To display mails users have to choose the option ‘inbox’ after logged in. Here we ask for the corresponding mail number to display a particular mail, if given, we print it on screen.
6. **Delete mail**: If user chooses the option for deleting mail, then we ask for the mail number, if given, the corresponding mail file is deleted (removed) from the folder inbox.
7. **Create mail account**: If user choose ‘create mail’ option, then we ask for username. If given username is valid then we ask for a password and given username-password combination is stored in log file and corresponding user account is created in server computer.
8. **Log out**: If a logged in user choose to ‘log out’ option then we save the corresponding transaction to server machine and we send the user to the home or first screen.

# 2. Description of the software of Mail transfer protocol

|  |  |  |
| --- | --- | --- |
| **Procedure** | **Function** | **Used system calls** |
| Login | login(), fopen(), fprintf(), fclose(), system() | socket(),bind(),listen(),accept() |
| Compose & send | login(),fprintf(),fopen(),fclose() | Socket(),bind(),listen(),accept() |
| Display mail | login(),fopen(),fprintf(),fclose(),system() | Socket(),bind(),listen(),accept() |
| Delete mail | login(),fopen(),fprintf(),fclose(),system() | Socket(),bind(),listen(),accept() |
| Create mail account | login(),fopen(),fprintf(),fclose(),system() | Socket(),bind(),listen(),accept() |
| Forward mail | login(),fopen(),fprintf(),fclose(),system() | Socket(),bind(),listen(),accept() |
| Log out | System() |  |

# 3. Log in:

**Data structure used here**- String.

**System calls**- socket ()

listen ()

bind ()

accept ()

**Files are used**: tempfile.txt

logfile.txt

**Built in functions**: fprintf()

system()

fopen() & fclose()

This phase has been implemented in following steps(page no.- 11)-

1. A server program is running, the function name is ‘runningserver ()’. (page no.- 7) That connects the client machine by system calls socket, bind, listen and accept.
2. We get a file from client computer where username & password is given and on the top of the file a ‘1’ tag (which indicates that user wants to login on that particular account) is also given. Using ‘write2file()’ function(page no.- 8) that file is written from the socket and is kept to new file ‘tempfile.txt’ by using ‘fputs()’ function.
3. Now we call ‘processfile()’ (page no.- 8) to calculate what to do next.
4. In ‘processfile()’ function we put the tag value in ‘code’ variable. we put the ‘username ’ in ‘usrid’ string and password in ‘pwd’ variable.
5. Now if ‘code=0’ then do procedure for creating a mail account.
6. We check ‘usrid’ whether it is already in ‘logfile.txt’ or not. If not then we create a folder according to that ‘usrid’ and append ‘logfile.txt’ with new ‘usrid’ and ‘pwd’ .We also create inbox and sent mail folder inside that account and ‘count.txt’,’inbox.txt’,’sentmail.txt’ file is created inside the folder corresponding to that account.
7. After log in this we create a feedback file name ‘feedback.txt’ and that file is printed on client desktop monitor, whether the account is created or not.

# 4. Compose mail:

**Data structure used**-string.

**System calls**-socket, listen, bind, accept.

**Files are used**-tempfile.txt,logfile.txt, count.txt, inbox.txt, sentmail.txt

**Used built in function**-fprintf(), System(), fopen(), fclose()

This phase has been implemented in the following steps(page no.- 12)-

On processfile() function if we get ‘code=4’,then if ‘feedback=0’

**a>**Open a temporary file ‘temp.c’ .This file is created dynamically (on runtime).

**b>**The written mail created by user, collected from socket stream and written on a file ‘ftemp.txt’.

**c>**We create a text file on the corresponding user folder in server machine.

**d>**We take the value from ‘count.txt’ that is a integer number. Then we create a text file with the integer value inside the sent mail folder.

**e>**Create another text file name with that integer value inside the account (to whom the user want to send that file) and inside the inbox folder.

**f>**Then open sentmail file of user accout and append the new entry.

**g>**Then open inbox.txt file of the receiver account and append the new entry.

# 5. Send mail:

Implementation mechanism of send mail phase is same as compose mail.

# 6. Forward mail:

**Data structure used**-string.

**System calls**-socket, listen, bind, accept.

**Files are used**-tempfile.txt

logfile.txt

count.txt

inbox.txt

sentmail.txt

This phase has been implemented in the following steps(page no.-12)-

1. User have to choose the mail from inbox (which file he/she want to send to the receiver). Client side program store the corresponding file name and send it to the server.
2. Server reads the file name from socket and store it in a string.
3. Now from receiver account folder we open count.txt, take the value for inbox and increment it.
4. Create a text file by that integer value inside the inbox and copy the copy the content of the corresponding file here.
5. Now from user account folder we open count.txt and take the value according to sent mail and increment it.
6. Create a text file by that integer value inside the sent mail folder and copy the content of the corresponding file here.

# 7. Display mail:

**Data structure used**-string.

**System calls**-socket, listen, bind, accept

**Files are used**-tempfile.txt

Inbox.txt

Sentmail.txt

This phase has been implemented as follows(page no.-13)-

1. First, client choose the option inbox or sent mail. According to that option server sends the corresponding file to client socket.
2. Client machine writes that file in tempfile.txt.
3. Now we display the tempfile.txt on screen.
4. User choose the sequence number of a file in inbox or sentmail.
5. Client send the value to server machine.
6. According to that value server sends the corresponding text file from the corresponding folder.
7. Client machine reads the file from socket and write it on the terminal.

# 8. Delete mail:

Data structure used- string

System calls- socket, listen, bind ,accept.

Files used- inbox.txt

This phase has been implemented as follows (page no.-14)–

1. First, client choose the option inbox.
2. Then inbox.txt file is sent to the client socket by server.
3. The inbox.txt file is displayed at the terminal.
4. User select serial number of the file he wants to delete.
5. Selected serial number is sent to the server.
6. According to the serial number the corresponding text file is removed from the inbox folder.
7. According to the serial number the corresponding line is removed from ‘inbox.txt’.

# 9. Create a mail account:

This phase has been implemented as the following steps(Page no.- 9)-

1. First, ‘create mail’ option is chosen. According to that option ‘0’ is saved on a temporary file.
2. Then user is asked for username and password. Given username and password is saved in the temporary file, where two separate variables hold username and password. This file is sent to the server machine.
3. Server reads that file and save them in a temporary file.
4. Then server checks whether the given username is in log file or not.

**a>**If no match is found then create the account (create a folder according to that username and enter the details of that account in the logfile.txt).

**b>**Else write on feedback.txt file that, this username already exists and send it to the client trying to create an account.

# 10. Log out:

This phase has been implemented as the following steps(page no.-21)-

1. If client choose the option ‘log out’ then put ‘username’ in the variable which holds username and put ‘password’ in the variable that holds password.
2. Remove the temporary file from client.

# 11. Coding:

Server Code:

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<string.h>

#include "inet.h"

#define MAXLINE 512

#define SERV\_TCP\_PORT 6000

#define SERV\_HOST\_ADDR "192.43.235.6"

char \*pname;

void main()

{

const char \*command1="mkdir pmail";

const char \*command2="mkdir pmail/log";

const char \*logfile="touch pmail/log/logfile.txt";

system(command1);

system(command2);

system(logfile);

runingserver();

}

runingserver()

{

int sockfd,newsockfd,clilen,childpid;

struct sockaddr\_in cli\_addr,serv\_addr;

/\* open a tcp socket(an internet stream socket)\*/

if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))<0) /\*SOCKFD IS SOCKET DESCRIPTOR\*/

err\_dump("server:can't open stream socket");

/\* bind our local addresses so that client can send to us \*/

bzero((char \*)&serv\_addr,sizeof(serv\_addr));

serv\_addr.sin\_family=AF\_INET;

serv\_addr.sin\_addr.s\_addr=htonl(INADDR\_ANY);

serv\_addr.sin\_port=htons(SERV\_TCP\_PORT);

if(bind(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0)

err\_dump("server:can't bind local addressing");

listen(sockfd,5); /\* 5 is the max value of backlog\*/

for(;;)

{

/\*wait for a connection from a client process\*/

clilen=sizeof(cli\_addr);

newsockfd=accept(sockfd,(struct sockaddr \*)&cli\_addr,&clilen);

if(newsockfd<0)

err\_dump("server:accept error");

if((childpid=fork())<0)

err\_dump("server:fork error");

else if(childpid==0)

{ /\*child process\*/

close(sockfd); /\*close original socket\*/

write2file(); /\*process the request\*/

processfile();

}

close(newsockfd);

}

}

write2file()

{

FILE \*fp,\*fw;

int sockfd;

struct sockaddr \*pserv\_addr; /\* ptr to appropriate sockaddr\_XX structure \*/

int servlen; /\* actual sizeof{\*pserv\_addr) \*/

int n;

char sendline[MAXLINE], recvline[MAXLINE + 1];

while (fgets(sendline, MAXLINE, fp) !=NULL)

{

n = strlen(sendline);

if (sendto(sockfd, sendline, n, 0, pserv\_addr, servlen) != n)

err\_dump("dg\_cli: sendto error on socket");

/\*

\*Now read a message from the socket and write it to

\*our standard output.

\*/

n = recvfrom(sockfd, recvline, MAXLINE, 0,(struct sockaddr \*) 0, (int \*) 0);

if (n < 0)

err\_dump("dg\_cli: recvfrom error"); recvline[n] = 0; /\* null terminate \*/

fw=fopen("pmail/log/tempfile.txt","w");

fputs(recvline, fw);

fclose(fw);

}

if(ferror(fp))

err\_dump("dg\_cli: error reading file");

}

processfile()

{

FILE \*f,\*tf,\*fr=fopen("pmail/log/tempfile.txt","r");

char ch,code,usrid[50],pwd[50],check,checkusrid[50],sendusr[50],coma='"',persent='%';

int i,j,idcheck,encript,feedback,count,imc,smc;

//copy usr details from incoming file to variables start

code=fgetc(fr);

ch=fgetc(fr);

i=0;

while((ch=fgetc(fr))!='\n')

{

usrid[i]=ch;

i++;

}

usrid[i]='\0';

ch=fgetc(fr);

i=0;

while((ch=fgetc(fr))!='\n')

{

pwd[i]=ch;

i++;

}

pwd[i]='\0';

//copy usr details from incoming file to variables end

if(code=='0')// 0 for creat mail account ####################

{

check=usrid[i];

while(check!='\0')//check validity of userid

{

if(check==' ' || check=='@' || check=='!' || check=='#' || check=='$' || check=='%' || check=='^' || check=='&')

{ break; }

check=usrid[i++];

}

if(check=='\0')

{

f=fopen("pmail/log/logfile.txt","r");

ch=fgetc(f);

while(ch!=EOF)

{

i=0;

while(ch!=' ')

{

checkusrid[i]=ch;

ch=fgetc(f);

i++;

}

checkusrid[i]='\0';

idcheck=strcmp(checkusrid,usrid);

if(idcheck==0)

{

feedback=1;

break;

}

while(ch!='\n')

ch=fgetc(f);

ch=fgetc(f);

}

if(ch==EOF)

{

fclose(f);

i=0;

while(pwd[i]!='\0')//encripting the password

{

encript=pwd[i];

encript+=100;

pwd[i]=encript;

i++;

}

f=fopen("pmail/log/logfile.txt","a");//saving usrid & encripted password to log file

fprintf(f,"%s %s\n",usrid,pwd);

fclose(f);

f=fopen("tempo.c","w");//a temporary file for creating account

fprintf(f,"#include<stdio.h>\n#include<stdio.h>\n#include<stdlib.h>\n#include<sys/types.h>\n#include<sys/stat.h>\n#include<fcntl.h>\nvoid main()\n{\nFILE \*fp;");

fprintf(f,"\nconst char \*foldername=%cmkdir pmail/%s%c;",coma,usrid,coma);

fprintf(f,"\nconst char \*folderinbox=%cmkdir pmail/%s/inbox%c;",coma,usrid,coma);

fprintf(f,"\nconst char \*fileinbox=%ctouch pmail/%s/inbox.txt%c;",coma,usrid,coma);

fprintf(f,"\nconst char \*foldersentmail=%cmkdir pmail/%s/sentmail%c;",coma,usrid,coma);

fprintf(f,"\nconst char \*filesentmail=%ctouch pmail/%s/sentmail.txt%c;",coma,usrid,coma);

fprintf(f,"\nconst char \*filecount=%ctouch pmail/%s/count.txt%c;",coma,usrid,coma);

fprintf(f,"\nsystem(foldername);\nsystem(fileinbox);\nsystem(filesentmail);\nsystem(filecount);\nsystem(folderinbox);\nsystem(foldersentmail);");

fprintf(f,"\nfp=fopen(%cpmail/%s/count.txt%c,%ca%c);",coma,usrid,coma,coma,coma);

fprintf(f,"\nfprintf(fp,%c0 0%c);",coma,coma);

fprintf(f,"\nfclose(fp);\n}");

fclose(f);

system("gcc tempo.c");

system("./a.out");

system("rm tempo.c");//remove the temporary file

}

}

else

{

feedback=1;

}

f=fopen("feedback.txt","w");

if(feedback==0)//feedback chooses what will be the message for the client machine

fprintf(f,"Your account created successfully.");

if(feedback==1)

fprintf(f,"This name allready exists, try another one...");

fclose(f);

}

if(code=='1')// 1 for login in mail account #################

{

feedback=login(usrid,pwd);

f=fopen("feedback.txt","w");

if(feedback==1)

fprintf(f,"bad username or password");

if(feedbach==0)

fprintf(f,"loggedin successfully.");

fclose(f);

}

if(code=='2')//2 for display inbox ############################

{

feedback=login(usrid,pwd);

if(feedback==0)

{

tf=fopen("tempo.c","w");

fprintf(f,"#include<stdio.h>\n#include<stdio.h>\n#include<stdlib.h>\n#include<sys/types.h>\n#include<sys/stat.h>\n#include<fcntl.h>\nvoid main()\n{\nFILE \*fp,\*fb;");

fprintf(f,"fp=fopen(%cpmail/%s/inbox.txt%c,%cr%c);\n",coma,usrid,coma,coma,coma);

fprintf(f,"fb=fopen(%cfeedback.txt%c,%cw%c);",coma,coma,coma);

fprintf(f,"while((ch=fgetc(fp))!=EOF){fprintf(fb,%c%c%c,coma,ch,coma);}\nch=EOF;\nfprintf(fb,%c%c%c,coma,ch,coma);\nfclose(fp);\nfclose(fb);\n}");

fclose(tf);

system("gcc tempo.c");

system("./a.out");

}

}

if(code=='3')//3 for display sentmail ###########################

{

feedback=login(usrid,pwd);

if(feedback==0)

{

tf=fopen("tempo.c","w");

fprintf(f,"#include<stdio.h>\n#include<stdio.h>\n#include<stdlib.h>\n#include<sys/types.h>\n#include<sys/stat.h>\n#include<fcntl.h>\nvoid main()\n{\nFILE \*fp,\*fb;");

fprintf(f,"fp=fopen(%cpmail/%s/sentmail.txt%c,%cr%c);\n",coma,usrid,coma,coma,coma);

fprintf(f,"fb=fopen(%cfeedback.txt%c,%cw%c);",coma,coma,coma);

fprintf(f,"while((ch=fgetc(fp))!=EOF){fprintf(fb,%c%c%c,coma,ch,coma);}\nch=EOF;\nfprintf(fb,%c%c%c,coma,ch,coma);\nfclose(fp);\nfclose(fb);\n}");

fclose(tf);

system("gcc tempo.c");

system("./a.out");

system("rm tempo.c");

}

}

if(code=='4')// 4 for compose mail ###########################

{

feedback=login(usrid,pwd);

count=0;

if(feedback==0)

{

i=0;

while((ch=fgetc(fr))!='\n')

{

sendusr[i]=ch;

i++;

}

fclose(fr);

tf=fopen("tempo.c","w");

fprintf(tf,"#include<stdio.h>\n#include<stdio.h>\n#include<stdlib.h>\n#include<sys/types.h>\n#include<sys/stat.h>\n#include<fcntl.h>\nvoid main()\n{\nFILE \*fi,\*fs,\*ftemp;");

fprintf(tf,"ftemp=fopen(%cpmail/log/tempfile.txt%c,%cr%c);\n",coma,coma,coma,coma);

fprintf(tf,"for(count=0;count<=4; )\n{if((ch=fgetc(ftemp))=='\n')\ncount++;}\n");

fprintf(tf,"fi=fopen(%cpmail/%s/inbox/%d.txt%c,%cw%c);",coma,sendusr,imc,coma,coma,coma);

fprintf(tf,"fs=fopen(%cpmail/%s/sentmail/%d.txt%c,%cw%c);",coma,usrid,smc,coma,coma,coma);

fprintf(tf,"while((ch=fgetc(ftemp))!=EOF){\nfprintf(fi,%c%c%c,coma,ch,coma);\nfprintf(fs,%c%c%c,coma,ch,coma);\n}");

fprintf("fclose(fi);\nfclose(fs);");

fclose(tf);

}

}

if(code=='5')// 5 for forword mail ################################

{

feedback=login(usrid,pwd);

if(feedback==0)

{

i=0;

while((ch=fgetc(fr))!='\n')

{

sendusr[i]=ch;

i++;

}

i=0;

while((ch=fgetc(fr))!=EOF)

{

sendfile[i]=ch;

i++;

}

fclose(fr);

tf=fopen("tempo.c","w");

fprintf(tf,"#include<stdio.h>\n#include<stdio.h>\n#include<stdlib.h>\n#include<sys/types.h>\n#include<sys/stat.h>\n#include<fcntl.h>\nvoid main()\n{\nFILE \*old,\*new1,\*new2,fr;");

fprintf(tf,"int i,j,mult,smc,imc;\n");

fprintf(tf,"char inboxc[6],sentmailc[6],ch;\n");

fprintf(tf,"old=fopen(%cpmail/%s/inbox/%s.txt%c,%cr%c);\n",coma,usrid,sendfile,coma,coma,coma);

fprintf(tf,"fr=fopen(%cpmail/%s/count.txt%c,%cr%c);\n",coma,usrid,coma,coma,coma);

fprintf(tf,"i=0;\nwhile((ch=fgetc(fr))!=' '){ inboxc[i]=ch; i++; }\n inboxc[i]='\0';");

fprintf(tf,"i=0;\nwhile((ch=fgetc(fr))!='\n'){ sentmailc[i]=ch; i++; }\n sentmailc[i]='\0';");

fprintf(tf,"i=strlen(sentmailc);\nfor(mult=1,smc=0;i>0;i--){ j=sentmailc[i]; j=j-48; smc=smc+(j\*mult); mult=mult\*10; }");

fprintf(tf,"smc++;");

fprintf(tf,"i=strlen(inboxc);\nfor(mult=1,imc=0;i>0;i--){ j=inboxc[i]; j=j-48; imc=imc+(j\*mult); mult=mult\*10; }");

fprintf(tf,"fclose(fr);fr=fopen(%cpmail/%s/count.txt%c,%cw%c);\n",coma,usrid,coma,coma,coma);

fprintf(tf,"fprintf(fr,%d %d,imc,smc);fclose(fr);");

fprintf(tf,"new1=fopen(%cpmail/%s/sentmail/%d.txt%c,%cw%c);\n",coma,usrid,imc,coma,coma,coma);

fprintf(tf,"new2=fopen(%cpmail/%s/inbox/%d.txt%c,%cw%c);\n",coma,sendusr,smc,coma,coma,coma);

fprintf(tf,"while((ch=fgetc(old))!=EOF){fprintf(new1,%c%cc%c,ch);}\n}",coma,persent,coma);

fclose(tf);

system("gcc tempo.c");

system("./a.out");

system("rm tempo.c");

}

}

if(code=='6')//6 for display a mail #############################

{

feedback=login(usrid,pwd);

if(feedback==0)

{

i=0;

while((ch=fgetc(fr))!=EOF)

{inboxc[i]=ch; i++;}

inboxc[i]='\0'

tf=fopen("tempo.c","w");

fprintf(f,"#include<stdio.h>\n#include<stdio.h>\n#include<stdlib.h>\n#include<sys/types.h>\n#include<sys/stat.h>\n#include<fcntl.h>\nvoid main()\n{\nFILE \*fp,\*fb;");

fprintf(f,"fp=fopen(%cpmail/%s/inbox/%s.txt%c,%cr%c);\n",coma,usrid,inboxc,coma,coma,coma);

fprintf(f,"fb=fopen(%cfeedback.txt%c,%cw%c);",coma,coma,coma);

fprintf(f,"while((ch=fgetc(fp))!=EOF){fprintf(fb,%c%c%c,coma,ch,coma);}\nch=EOF;\nfprintf(fb,%c%c%c,coma,ch,coma);\nfclose(fp);\nfclose(fb);\n}");

fclose(tf);

system("gcc tempo.c");

system("./a.out");

system("rm tempo.c");

}

}

if(code=='7')// 7 for delete a mail ########################

{

feedback=login(usrid,pwd);

if(feedback==0)

{

while((ch=fgetc(fr))!=EOF)

{inboxc[i]=ch; i++;}

inboxc[i]='\0'

tf=fopen("tempo.c","w");

fprintf(tf,"#include<stdio.h>\n#include<stdio.h>\n#include<stdlib.h>\n#include<sys/types.h>\n#include<sys/stat.h>\n#include<fcntl.h>\nvoid main()\n{\nFILE \*fp,\*fb;\nchar ch;");

fprintf(tf,"fp=fopen(%cpmail/%s/inbox.txt%c,%cr%c);\n",coma,usrid,coma,coma,coma);

fprintf(tf,"fb=fopen(%ctempinbox.txt%c,%cw%c);",coma,coma,coma);

fprintf(tf,"while((ch=fgetc(fp))!=EOF){fprintf(fb,%c%cc%c,ch)} fclose(fp);fclose(fb);\n",coma,persent,coma);

fprintf(tf,"system(rm pmail/%s/inbox.txt);",usrid);

fprintf(tf,"fp=fopen(%cpmail/%s/inbox.txt%c,%cw%c);\n",coma,usrid,coma,coma,coma);

fprintf(tf,"while((ch=fgetc(fb))!=EOF){");

fprintf(tf,"i=0;while((ch=fgetc(fb))!=' '){temps[i]=ch;i++} temps[i]='\0';");

fprintf(tf,"i=strcmp(temps,%s)",inboxc);

fprintf(tf,"if(i==0){while((ch=fgetc(fb))!='\n')}else{fprintf(fp,%c%cs%c,temps); }",coma,persent,coma);

fclose(tf);

system("gcc tempo.c");

system("./a.out");

system("rm tempo.c");

}

}

}

void login(char \*usrid, char \*pwd)

{

FILE \*f;

int encript,idcheck,i,check,flag=0;

char checkusrid[50],checkpwd[50],ch;

f=fopen("pmail/log/logfile.txt","r");

ch=fgetc(f);

while(ch!=EOF)

{

i=0;

while(ch!=' ' && ch!=EOF)

{

checkusrid[i]=ch;

ch=fgetc(f);

i++;

}

checkusrid[i]='\0';

if(ch==' ')

ch=fgetc(f);

idcheck=strcmp(checkusrid,usrid);

if(idcheck==0)

{

i=0;

while(ch!='\n' && ch!=EOF)

{

encript=ch;

encript-=100;

checkpwd[i]=encript;

ch=fgetc(f);

i++;

}

checkpwd[i]='\0';

idcheck=strcmp(checkpwd,pwd);

if(idcheck==0)

flag=1;

}

else

{

while(ch!='\n' && ch!=EOF)

ch=fgetc(f);

}

if(ch!=EOF)

ch=fgetc(f);

}

fclose(f);

if(flag==1)

{

return(0);

}

else

return(1);

}

/\*ESTABLISHMENT OF CONNECTION AND TRANSFER OF CONTENT OF THE FILE FROM SERVER TO CLIENT(INTERNET DOMAIN)\*/

/\*A filename is taken by the client program in the command line argument.client process sends this filename to the server

process through the socket.server process reads the filename from the socket and opens that file,and sends the content

of the file to the client process through socket.client reads it from the socket and prints it to the output.\*/

/\*server code\*/

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/un.h>

#include "unix.h"

#define NAME "SOCKET"

main()

{

int sock,msgsock,rval,fd,n;

struct sockaddr\_in serv\_addr;

char buff[1024];

/\*open a internet domain stream socket\*/

if((sock=socket(AF\_INET,SOCK\_STREAM,0))<0)

err\_dump("server:cant open stream socket");

serv\_addr.sin\_family=AF\_INET;

serv\_addr.sin\_addr.s\_addr=htonl(INADDR\_ANY); /\*32 bit netid/hosted\*/

serv\_addr.sin\_port=htons(SERV\_TCP\_PORT); /\* 16 bit port id\*/

if(bind(sockfd,(struct sockaddr\*)&serv\_addr,sizeof(struct sockaddr\_in))<0)

err\_dump("server:cant bind local address");

listen(sockfd,5);

for(;;)

{

msgsock=accept(sock,0,0);

if(msgsock==-1)

err\_dump("accept error");

else

do

{

bzero(buf,sizeof(buf));

if((rval=read(msgsock,buf,1024))<0)

err\_dump("error in reading message");

else if(rval==0)

print("ending connection");

else

{

print("-->%s\n".buf);

fd=open(buf,0);

bzero(buf,sizeof(buf));

while((n=read(fd,buf,1024))>0)

{

write(msgsock,buf,n);

}

print("ending connection\n");

if(n<0)

err\_dump("error");

}

}while(rval>0);

close(msgsock);

}

close(sock);

unlink(NAME);

}

/\*TRANSFER A FILE FROM CLIENT TO SERVER IN INTERNET DOMAIN\*/

/\*server using tcp protocol\*/

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include "inet.h"

#define SERV\_TCP\_PORT 6000

#define SERV\_HOST\_ADDR "192.43.235.6"

char \*pname;

main(argc,argv)

int argc;

char \*argv[];

{

int sockfd,newsockfd,clilen,childpid;

struct sockaddr\_in cli\_addr,serv\_addr;

char \*pname=argv[0];

/\* open a tcp socket(an internet stream socket)\*/

if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))<0) /\*SOCKFD IS SOCKET DESCRIPTOR\*/

err\_dump("server:can't open stream socket");

/\* bind our local addresses so that client can send to us \*/

bzero((char \*)&serv\_addr,sizeof(serv\_addr));

serv\_addr.sin\_family=AF\_INET;

serv\_addr.sin\_addr.s\_addr=htonl(INADDR\_ANY);

serv\_addr.sin\_port=htons(SERV\_TCP\_PORT);

if(bind(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0)

err\_dump("server:can't bind local addressing");

listen(sockfd,5); /\* 5 is the max value of backlog\*/

for(;;)

{

/\*wait for a connection from a client process\*/

clilen=sizeof(cli\_addr);

newsockfd=accept(sockfd,(struct sockaddr \*)&cli\_addr,&clilen);

if(newsockfd<0)

err\_dump("server:accept error");

if((childpid=fork())<0)

err\_dump("server:fork error");

else if(childpid==0)

{ /\*child process\*/

close(sockfd); /\*close original socket\*/

str\_req(newsockfd); /\*process the request\*/

exit(0);

}

close(newsockfd);

}

}

/\*Read a stream socket one line at a time,return when the connection is terminated\*/

#define MAXLINE 512

str\_req (sockfd)

int sockfd;

{

int n;

char line[MAXLINE];

for(;;)

{

n=readline(sockfd,line,MAXLINE);

if(n==0)

return;

else if(n<0)

err\_dump("str\_echo:readline error");

}

}

/\*Read a line from a descriptor.read the line one byte at a time,

looking for the new line.We store the newline in the buffer,

then follow it with a null.

We return the number of characters up to,but not including,the null.\*/

int

readline(fd,ptr,maxlen)

register int fd;

register char \*ptr;

register int maxlen;

{

int n,rc;

char c;

for(n=1;n<maxlen;n++)

{

if((rc=read(fd,&c,1))==1)

{

\*ptr++=c;

if(c=='\n')

break;

}

else if(rc==0)

{

if(n==1)

return(0); /\*EOF,no data read\*/

else

break;

}

else

return(-1);

}

\*ptr=0;

return(n);

}

Client Code:

#include<stdio.h>

#include<string.h>

void writemail()

{

FILE \*f;

char mail[3];

f=fopen("temp.txt","a");

printf("Enter ur mail here: \n");

while(1)

{

gets(mail);

if(mail[0]=='.' && mail[1]=='\n')

break;

fprintf(f,"%s\n",mail);

}

fclose(f);

}

void create()

{

FILE \*f;

int ;

char usrid[30],pwd[30];

printf("\nChoosse user id: ");

scanf("%s",&usrid);

printf("\nChoosse password: ");

scanf("%s",&pwd);

f=fopen("temp.txt","w");

fprintf(f,"0\n%s\n%s\n",usrid,pwd);

fclose(f);

system("gcc c2sc.c");

system("./a.out temp.txt");

system("gcc s2cc.c");

system("./a.out feedback.txt");

}

void login()

{

FILE \*f;

int choice,flag=0,msgnum;

char usrid[30],pwd[30],sendusr[30];

printf("\nChoosse user id: ");

scanf("%s",&usrid);

printf("\nChoosse password: ");

scanf("%s",&pwd);

f=fopen("temp.txt","w");

fprintf(f,"1\n%s\n%s\n",usrid,pwd);

fclose(f);

system("gcc c2sc.c");

system("./a.out temp.txt");

system("gcc s2cc.c");

system("./a.out feedback.txt");

printf("If login success press 1, else 0: ");

scanf("%d",&choice);

if(choice==1)

{

while(1)

{

printf("1 : see inbox\n2 : see sent mail\n3 : compose mail\n4 : forword mail\n5 : display mail\n6 : delete mail\n7 : logout.\n You choose: ");

scanf("%d",&choice);

switch(choice)

{

case 1: printf("\n INBOX \n");f=fopen("temp.txt","w");fprintf(f,"2\n%s\n%s\n",usrid,pwd);fclose(f);system("gcc c2sc.c"); system("./a.out temp.txt");system("gcc s2cc.c");system("./a.out feedback.txt");break;

case 2: printf("\n SENT MAIL \n");f=fopen("temp.txt","w");fprintf(f,"3\n%s\n%s\n",usrid,pwd);fclose(f);system("gcc c2sc.c"); system("./a.out temp.txt");system("gcc s2cc.c");system("./a.out feedback.txt");break;

case 3: f=fopen("temp.txt","w");printf("Enter reciever id: ");scanf("%s",&sendusr);fprintf(f,"4\n%s\n%s\n%s\n",usrid,pwd,sendusr);fclose(f);writemail();printf("\nmail sent\n");break;

case 4: f=fopen("temp.txt","w");printf("Enter reciever id: ");scanf("%s",&sendusr);printf("enter the no. of msg you want to forword: ");scanf("%d",&msgnum);fprintf(f,"5\n%s\n%s\n%s\n%d\n",usrid,pwd,sendusr,msgnum);printf("\nmail forworded.\n");break;

case 5: printf("enter the no. of mail you want to see from inbox: ");scanf("%d",&msgnum);f=fopen("temp.txt","w");fprintf(f,"6\n%s\n%s\n",usrid,pwd);fclose(f);system("gcc c2sc.c"); system("./a.out temp.txt");system("gcc s2cc.c");system("./a.out feedback.txt");break;

case 6: printf("enter the no. of mail you want to delete from inbox: ");scanf("%d",&msgnum);f=fopen("temp.txt","w");fprintf(f,"7\n%s\n%s\n",usrid,pwd);fclose(f);system("gcc c2sc.c"); system("./a.out temp.txt");system("gcc s2cc.c");system("./a.out feedback.txt");break;

case 7: system("rm temp.txt");flag=1;break;

}

if(flag==1)

break;

}

}

}

void main()

{

int choice;

while(1)

{

printf("\n\nTo create an account: 1\nTo login an account: 2\nTo exit: 0\nEnter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 0: exit(0);break;

case 1: create();break;

case 2: login();break;

}

}

}

/\*ESTABLISHMENT OF CONNECTION AND TRANSFER OF CONTENT OF THE FILE FROM SERVER TO CLIENT(INTERNET DOMAIN)\*/

/\*A filename is taken by the client program in the command line argument.client process sends this filename to the server

process through the socket.server process reads the filename from the socket and opens that file,and sends the content

of the file to the client process through socket.client reads it from the socket and prints it to the output.\*/

/\* client code \*/

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/un.h>

#include "unix.h"

main(argv)

char \*argv[];

{

int sock,c;

struct sockaddr\_in serv\_addr;

char buff[1024];

if((sock=socket(AF\_INET,SOCK\_STREAM,0))<0) /\*open internet domain stream socket\*/

err\_dump("server:cant open stream socket");

serv\_addr.sun\_family=AF\_INET;

serv\_addr.sin\_addr.s\_addr=inet\_addr(SERV\_HOST\_ADDR);

serv\_addr.sin\_port=htons(SERV\_TCP\_PORT);

if(connect(sock,(struct sockaddr\*)&serv\_addr,sizeof(struct sockaddr\_in)),0)

{

close(sock);

err\_dump("error in connecting");

}

while((c=read(sock,buff,sizeof(buff)))>0)

write(1,buff,c);

close(sock);

}

/\*TRANSFER A FILE FROM CLIENT TO SERVER IN INTERNET DOMAIN\*/

/\*client using tcp protocol\*/

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include "inet.h"

#define SERV\_TCP\_PORT 6000

#define SERV\_HOST\_ADDR "192.43.235.6"

main(argv)

char \*argv[];

{

int sockfd;

struct sockaddr\_in serv\_addr;

char \*pname=argv[0];

/\*fill in the structure "serv\_addr" with the address of the server\*/

bzero((char \*)&serv\_addr,sizeof(serv\_addr));

serv\_addr.sin\_family=AF\_INET;

serv\_addr.sin\_addr.s\_addr=inet\_addr(SERV\_HOST\_ADDR);

serv\_addr.sin\_port=htons(SERV\_TCP\_PORT);

if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))<0) /\*OPEN A TCP SOCKET\*/

err\_sys("CLIENT:CAN'T OPEN STREAM SOCKET");

/\*connection to the server\*/

if(connect(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(serv\_addr))<0)

err\_sys("client:cant connect to the server");

str\_cli(stdin,sockfd); /\*do it all\*/

close(sockfd);

exit(0);

}

/\*Read the contents of the FILE \*fp,write each line to the stream socket(to the server process)\*/

#include <stdio.h>

#define MAXLINE 512

str\_cli(fp,sockfd)

register FILE \*fp;

register int sockfd;

{

int n;

char sendline[MAXLINE],recvline[MAXLINE+1];

while(fgets(sendline,MAXLINE,fp)!= NULL)

{

n=strlen(sendline);

if(written(sockfd,sendline,n)!= n)

err\_sys("str\_cli:written error on socket");

}

if(ferror(fp))

err\_sys("str\_cli:error reading file");

}

/\*Write "n" bytes to a descriptor\*/

int

written(fd,ptr,nbytes)

register int fd;

register char \*ptr;

register int nbytes;

{

int nleft,nwritten;

nleft=nbytes;

while(nleft>0)

{

nwritten=write(fd,ptr,nleft);

if(nwritten<=0)

return(nwritten); /\*error\*/

nleft-=nwritten;

ptr+=nwritten;

}

return(nbytes-nleft);

}

# 12. Conclusion:

**Some notes on SMTP** (The Simple Mail Transfer Protocol):

To transmit mail through internet this protocol is widely used.It has many drawbacks and as remedy to these drawbacks modified versions are replacing previous versions day-to-day.

SMTP works as follows-

Within the internet , e-mail is delivered by having the source machine established a TCP connection to port 25 of the destination machine.Listening to this port is an e-mail daemon that speaks SMTP.This daemon accepts incoming connections and copies messages from them into the appropriate mailboxes.If a message cannot be delivered , an error containing the first part of the undelivered message is returned to the sender.AS SMTP use TCP connection, so it performs reliable mail transfer.

SMTP is a simple ASCII protocol.

**Comparison of our implementation to SMTP**:

If on a UNIX system, in a shell, we type-

telnet mail.isp.com 25 (here, mail.isp.com is DNS name of the user’s ISP’s mail server)

This command will establish a telnet (i.e. TCP) connection to port 25 on that machine. We probably get following response-

Trying 192.30.200.66….

Connected to mail.isp.com

escape character is ‘^]’

220 mail.isp.com Smail #74 ready at Thu, 20 June 2013 20:26 +0200

The first three lines are from telnet telling user what it is doing. The last line is from the server on the remote machine announcing its willingness to talk to user and accept e-mail.

Thus SMTP and most internet protocol use ASCII text for commands.

Using ASCII text makes the protocol easy to test and debug.

They can be tested by sending commands manually, and dumps of the messages are easy to read.

But mail transfer system implemented by us has limitation on that aspect. In our system ASCII text for commands cannot be used.

As we saw, e-mail is delivered by having the sender establish a TCP connection to the receiver and then ship the e-mail over it. This model worked fine for decades when all ARPANET (and later internet) hosts were, in fact, on-line all the time to accept TCP connections.

However, with the advent of people who access the internet by calling their ISP over a modem, it breaks down. The problem is that, when both sender and receiver are not on-line, a TCP connection cannot be established and thus cannot run SMTP protocol. This is a limitation of SMTP protocol.

while we send mail to receiver mail account, not necessary that the corresponding client machine is online. The mail is stored in a folder named ‘inbox’ corresponding to that mail account in server machine.

**Limitations of our implementation**:

In addition to the above limitation, our implementation has no interface to user, so it is usable only by us (designer) in terminal.