**16) Write a program to implement HTTP.**

**Server program:**

import java.io.\*;

import java.net.\*;

import java.lang.\*;

public class https

{

public static void main(String args[]) throws Exception

{

ServerSocket ssoc=new ServerSocket(1111);

System.out.println("Server waits for client:\n");

Socket so=ssoc.accept();

System.out.println("client connected to Server :\n");

BufferedReader br=new BufferedReader(new InputStreamReader(so.getInputStream()));

PrintWriter pw=new PrintWriter(so.getOutputStream(),true);

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

int ch;

do

{

ch=Integer.parseInt(br.readLine());

String file;

byte line[]=null;

File f;

switch(ch)

{

case 1: System.out.println("1.head");

file=br.readLine();

f=new File(file);

int index=file.lastIndexOf(".");

String type=file.substring(index+1);

pw.println(type);

long length=f.length();

pw.println(length);

break;

case 2: System.out.println("2.post");

file=br.readLine();

System.out.println("message from client:\n");

System.out.println(file);

break;

case 3: System.out.println("3.get");

file=br.readLine();

FileInputStream fs=new FileInputStream(file);

while(fs.available()!=0)

{

if(fs.available()<1024)

line=new byte[fs.available()];

else

line=new byte[1024];

fs.read(line);

file=new String(line);

pw.println(file);

}

pw.println("\*\*\*");

fs.close();

break;

case 4: System.out.println("4.delete");

file=br.readLine();

f=new File(file);

f.delete();

break;

default: System.out.println("5.exit");

System.exit(0);

}

}

while(ch<=4);

so.close();

ssoc.close();

}

}

**Client program:**

import java.io.\*;

import java.net.\*;

import java.lang.\*;

public class httpc

{

public static void main(String args[]) throws Exception

{

Socket soc=new Socket("localhost",1111);

BufferedReader br=new BufferedReader(new InputStreamReader(soc.getInputStream()));

PrintWriter pw=new PrintWriter(soc.getOutputStream(),true);

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

System.out.println("server is connected:\n");

int ch;

do

{

System.out.println("COMMANDS");

System.out.println("\n 1.head \n 2.post \n 3.get \n4.delete\n 5.exit");

System.out.println("ENTER UR CHOICE:");

ch=Integer.parseInt(in.readLine());

byte line[]=null;

String file;

switch(ch)

{

case 1:pw.println("1");

file=in.readLine();

pw.println(file);

String type=br.readLine();

String length=br.readLine();

System.out.println("FILE:"+file+"\nTYPE:"+type+"\nLEN GTH:"+length);

break;

case 2: pw.println("2");

System.out.println("enter text to post");

file=in.readLine();

pw.println(file);

System.out.println("text is posted at server");

break;

case 3:pw.println("3");

System.out.println("enter file name to get");

file=in.readLine();

pw.println(file);

System.out.println("enter file name to save:");

file=in.readLine();

FileOutputStream fs=new FileOutputStream(file);

while(true)

{

String s=br.readLine();

if(s.equals("\*\*\*"))

break;

int count=s.length();

if(count<1024)

line=new byte[1024];

line=s.getBytes();

fs.write(line);

}

fs.close();

System.out.println("\n file successfully tranfered:");

break;

case 4: pw.println("4");

System.out.println("enter the file to delete:");

file=in.readLine();

pw.println(file);

System.out.println("given file deleted:");

break;

default: pw.println("5");

System.exit(0);

}

}

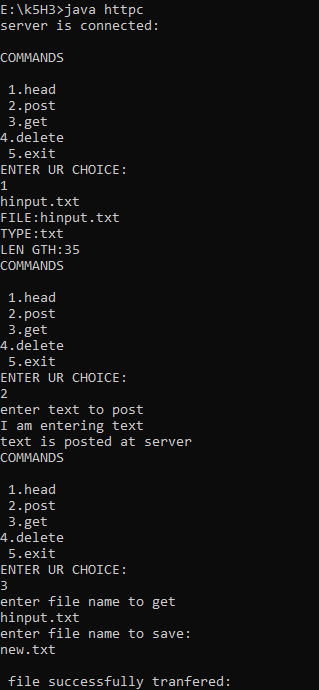
while(ch<=4);

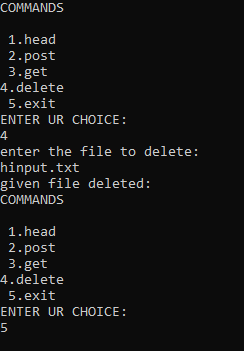
soc.close();

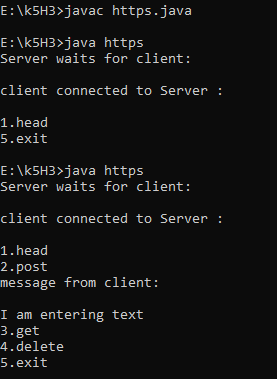
}

}

**Output:**







**11) AIM:** Write a program to implement FTP.

**PROGRAM:**

**// server.c**

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<netdb.h>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<errno.h>

#include<string.h>

#include<strings.h>

int main() {

int sock, connected;

char b[500];

FILE \*fp;

char fname[500], op[1000];

struct sockaddr\_in server\_addr, client\_addr;

int sin\_size;

if((sock=socket(AF\_INET, SOCK\_STREAM, 0))==-1) {

perror("socket");

exit(1);

}

bzero(&server\_addr, sizeof(server\_addr));

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

server\_addr.sin\_port=htons(16961);

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

if(bind(sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr))==-1) {

perror("unable to bind");

exit(1);

}

if(listen(sock, 4)==-1) {

perror("listen");

exit(1);

}

printf("\nTcp server waiting for client");

fflush(stdout);

while(1) {

sin\_size=sizeof(client\_addr);

connected=accept(sock, (struct sockaddr \*)&client\_addr, &sin\_size);

printf("\n I got a connection from client");

recv(connected, fname, 1500, 0);

fp=fopen(fname, "r");

fread(op, 1000, 1, fp);

printf("\nData in the file is \n %s\n", op);

//send(connected, fname, strlen(fname), 0);

fclose(fp);

break;

}

}

**SERVER OUTPUT:**

**[17A91A05H3@Linux CN]$ gcc server.c**

**[17A91A05H3@Linux CN]$ ./a.out**

**Tcp server waiting for  client**

**I got a connection from client**

**Data in the file is**

**Welcome To Computer Science & Engineering.**

**PROGRAM:**

**// client.c**

#include <strings.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <arpa/inet.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

int main() {

int sock;

char fname[500], op[1000];

FILE \*fp;

struct sockaddr\_in server\_addr;

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("socket");

exit(1);

}

bzero(&server\_addr, sizeof(server\_addr));

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

server\_addr.sin\_port = htons(16961);

server\_addr.sin\_addr.s\_addr = inet\_addr("172.7.7.12");

if (connect(sock, (struct sockaddr \*)&server\_addr, sizeof(struct sockaddr)) == -1) {

perror("connect");

exit(1);

}

while (1) {

printf("\nenter the file name\n");

scanf("%s", fname);

send(sock, fname, strlen(fname), 0);

recv(sock, op, 1000, 0);

//printf("\nThe file from server is%s\n",op);

fp = fopen("cse.txt", "wr");

int l = strlen(op);

fwrite(op, l, 500, fp);

printf("\nfile sent successfully\n");

fclose(fp);

break;

}

close(sock);

return 0;

}

**CLIENT OUTPUT:**

**[17A91A05H3@Linux CN]$ cat cse.txt**

**Welcome To Computer Science & Engineering.**

**[17A91A0592@Linux CN]$ gcc client.c**

**[17A91A0592@Linux CN]$ ./a.out**

**enter the file name**

**cse.txt**

**file sent successfully**

**RESULT:** Hence, successfully implemented FTP.