Abubakar Omar

CSCI 413 – Computer Networks

Programming Project 1

02/11/2022

Text

Description automatically generated

**//SocketsServer.cpp**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <strings.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

using namespace std;

void error (char \*msg)

{

perror(msg);

exit(1);

}

int main(int argc, char \*\*argv)

{

//DEFINING VARIABLES

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

int sockfd, newsockfd, portno, clilen;

char buffer[256];

int n;

bool close\_connection = false;

//CHECKS THE ARGUMENT COUNT TO MAKE SURE THE PORT NUMBER IS ENTERED

if (argc < 2)

{

fprintf(stderr, "ERROR, no port provided\n");

exit(1);

}

//CREATES A SOCKET FILE DESCRIPTOR

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if(sockfd < 0)

{

error("ERROR opening socket");

}

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

//CONVERTS THE PORT FROM STRING TO INTEGER

portno = atoi(argv[1]);

//DEFINING THE SERVER ADDRESS

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

serv\_addr.sin\_port = htons(portno);

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

//BINDING THE SOCKET TO THE SPECIFIED IP AND PORT

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

{

error("ERROR on binding");

}

//LISTENS TO THE NETWORK FOR ANY CONNECTIONS

listen(sockfd, 5); //ALLOWS AT MOST 5 CONNECTIONS

clilen = sizeof(cli\_addr);

//ACCEPTS CLIENT SOCKET

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

if(newsockfd < 0)

{

error("ERROR on accept");

}

//LOOP ALLOWS MULTIPLE MESSAGES TO BE TRANSMITTED BACK AND FORTH

while(!close\_connection)

{

bzero(buffer, 256);

//READS INPUT FROM THE SOCKET FILE DESCRIPTOR INTO THE BUFFER

n = read(newsockfd, &buffer, 255);

if(n < 0)

{

error("ERROR reading from socket");

}

//COMPARES THE BUFFER TO THE STRING EXIT

if(strcmp(buffer, "exit\n") == 0) //IF BUFFER == EXIT

{

//CLOSES THE CONNECTION AND TERMINATES THE PROGRAM

close\_connection = true;

n = write(newsockfd, "exit", 4);

if(n < 0)

{

error("ERROR writing to socket");

}

}

//ECHOS THE CLIENT INPUT BACK TO THE CLIENT

else

{

printf("Here is the message: %s\n", buffer);

n = write(newsockfd, &buffer, 255);

if(n < 0)

{

error("ERROR writing to socket");

}

}

}

return 0;

}

**//SocketsClient.cpp**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <strings.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

using namespace std;

void error(char \*msg)

{

perror(msg);

exit(0);

}

int main(int argc, char \*\*argv)

{

//DEFINING VARIABLES

int sockfd, portno, n;

struct sockaddr\_in serv\_addr;

struct hostent \*server;

char buffer[256];

bool close\_connection = false;

//CHECKS THE ARGUMENT COUNT TO MAKE SURE THE HOSTNAME AND PORT NUMBER IS ENTERED

if(argc < 3)

{

fprintf(stderr, "usage %s hostname port\n", argv[0]);

exit(0);

}

//CONVERTS THE PORT FROM STRING TO INTEGER

portno = atoi(argv[2]);

//CREATES A SOCKET FILE DESCRIPTOR

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if(sockfd < 0)

{

error("ERROR opening socket");

}

//FINDS HOST USING THE HOSTNAME IN THE ARGUMENT

server = gethostbyname(argv[1]);

if(server == NULL)

{

fprintf(stderr, "ERROR, no such host");

exit(0);

}

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

//DEFINING THE SERVER ADDRESS

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

bcopy((char \*) server->h\_addr, (char \*) &serv\_addr.sin\_addr.s\_addr, server->h\_length);

serv\_addr.sin\_port = htons(portno);

//CONNECTING THE CLIENT TO THE SERVER

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

{

error("ERROR connecting");

}

//LOOP ALLOWS MULTIPLE MESSAGES TO BE TRANSMITTED BACK AND FORTH

while(!close\_connection)

{

printf("Please enter the message (or type exit to terminate connection): ");

bzero(buffer, 256);

fgets(buffer, 255, stdin);

//WRITES THE USER INPUT TO THE SOCKET DESCRIPTOR

n = write(sockfd, &buffer, strlen(buffer));

if(n < 0)

{

error("ERROR writing to socket");

}

bzero(buffer, 256); //CLEAR THE BUFFER

//READS THE SOCKET DESCRIPTER TO THE BUFFER

n = read(sockfd, &buffer, 255);

if(n < 0)

{

error("ERROR reading from socket");

}

//COMPARES THE BUFFER TO THE STRING EXIT

if(strcmp(buffer, "exit") == 0) // IF BUFFER CONTENT == EXIT

{

//CLOSE CONNECTION AND TERMINATE THE PROGRAM

close\_connection = true;

printf("Connection Terminated");

}

//ELSE WRITE THE CONTENTS SENT FROM THE SERVER

else

{

printf("%s\n", buffer);

}

}

return 0;

}