Name: Kaustav Ghosh

Section: CSE C

Roll no: 29

Reg no:180905188

Date: 7/12/2020

Q1

// lab01\_q1\_client.c

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include <stdlib.h>

#define PORT 5000

#define MAXLINE 1000

// Driver code

int main()

{

char buffer[100];

// char \*message = "Hello Server";

char \*rows[3];

rows[0] = "1 2 3";

rows[1] = "4 5 6";

rows[2] = "7 8 9";

int sockfd, n, len;

struct sockaddr\_in servaddr, cliaddr;

// clear servaddr

bzero(&servaddr, sizeof(servaddr));

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

servaddr.sin\_port = htons(PORT);

servaddr.sin\_family = AF\_INET;

// create datagram socket

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

for (int i = 0; i < 3; i++)

{

sendto(sockfd, rows[i], MAXLINE, 0, (struct sockaddr \*)&servaddr, sizeof(servaddr));

len = sizeof(cliaddr);

}

// // waiting for response

// n = recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr \*)&cliaddr, &len);

// buffer[n] = '\0';

// printf("message fromser is %s \n", buffer);

// getchar();

// close the descriptor

close(sockfd);

}

// lab01\_q1\_server.c

// server program for udp connection

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#define PORT 5000

#define MAXLINE 1000

// Server code

int main()

{

char buffer[100];

int servsockfd, len, n;

struct sockaddr\_in servaddr, cliaddr;

bzero(&servaddr, sizeof(servaddr));

// Create a UDP Socket

servsockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

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

servaddr.sin\_port = htons(PORT);

servaddr.sin\_family = AF\_INET;

// bind server address to socket descriptor

bind(servsockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr));

while (1)

{

//receive the datagram

len = sizeof(cliaddr);

n = recvfrom(servsockfd, buffer, sizeof(buffer), 0, (struct sockaddr \*)&cliaddr, &len);

buffer[n] = '\0';

puts(buffer);

//Echoing back to the client

sendto(servsockfd, buffer, n, 0, (struct sockaddr \*)&cliaddr, sizeof(cliaddr));

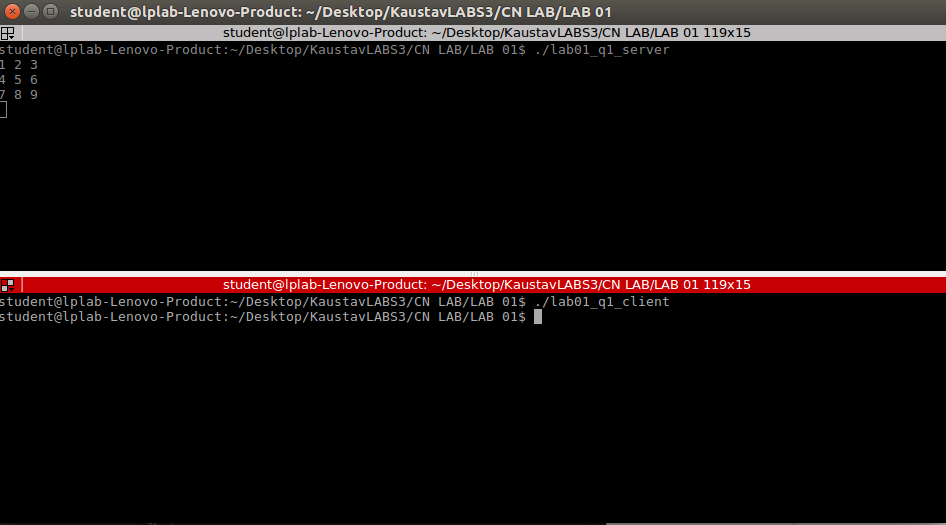
}

getchar();

// close the descriptor

close(servsockfd);

}



// lab01\_q2\_client.c

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

int sockfd;

int len;

struct sockaddr\_in address;

int result;

char ch[100];

/\* Create a socket for the client. \*/

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

/\* Name the socket, as agreed with the server. \*/

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

address.sin\_port = 9735;

len = sizeof(address);

/\* Now connect our socket to the server's socket. \*/

result = connect(sockfd, (struct sockaddr \*)&address, len);

if (result == -1)

{

perror("oops: client2");

exit(1);

}

/\* We can now read/write via sockfd. \*/

printf(" Please Type the String to send:");

scanf("%s", ch);

write(sockfd, ch, 100);

read(sockfd, ch, 100);

printf("char from server = %s\n", ch);

close(sockfd);

exit(0);

}

// lab01\_q2\_server.c

// Will practice again

/\* Make the necessary includes and set up the variables. \*/

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

int server\_sockfd, client\_sockfd;

int server\_len, client\_len;

struct sockaddr\_in server\_address;

struct sockaddr\_in client\_address;

/\* Create an unnamed socket for the server. \*/

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

/\* Name the socket. \*/

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

server\_address.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

server\_address.sin\_port = 9735;

server\_len = sizeof(server\_address);

bind(server\_sockfd, (struct sockaddr \*)&server\_address, server\_len);

/\* Create a connection queue and wait for clients. \*/

listen(server\_sockfd, 5);

while (1)

{

char ch[100];

printf("server waiting\n");

/\* Accept a connection. \*/

client\_len = sizeof(client\_address);

client\_sockfd = accept(server\_sockfd,

(struct sockaddr \*)&client\_address, &client\_len);

/\* We can now read/write to client on client\_sockfd. \*/

//char \*inet\_ntoa(client\_addr.sin\_addr);

char \*ip\_add = inet\_ntoa(client\_address.sin\_addr);

int port = client\_address.sin\_port;

printf("IP:%s PORT:%d\n", ip\_add, port);

read(client\_sockfd, ch, 100);

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

for (int i = 0; ch[i] != '\0'; i++)

{

if (ch[i] >= 'a' && ch[i] <= 'z')

{

ch[i] = ch[i] - 32;

}

}

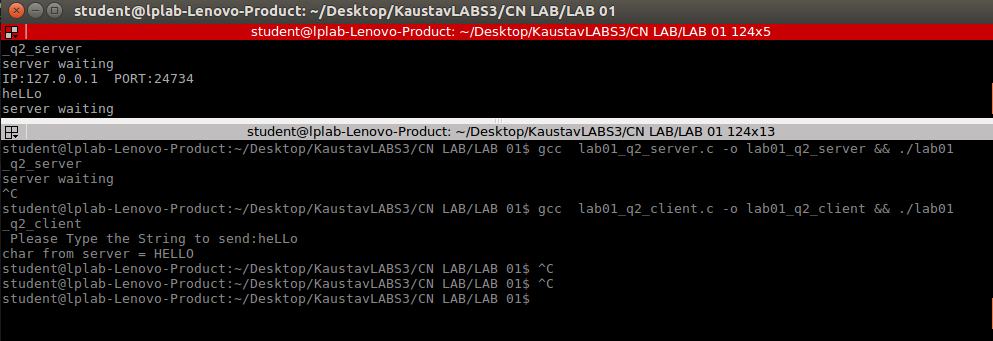
// ch.upper();

write(client\_sockfd, ch, 100);

close(client\_sockfd);

}

}



// lab01\_q3\_client.c

//TCP Client program

#include<unistd.h>

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include<arpa/inet.h>

#include <string.h>

#include <sys/socket.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void clifunc(int sockfd)

{

char buff[MAX];

int n;

for (;;) {

bzero(buff, sizeof(buff));

printf("Enter the string : ");

n = 0;

while ((buff[n++] = getchar()) != '\n')

;

write(sockfd, buff, sizeof(buff));

bzero(buff, sizeof(buff));

read(sockfd, buff, sizeof(buff));

printf("From Server : %s\n", buff);

if ((strncmp(buff, "quit", 4)) == 0) {

printf("Client Exit...\n");

break;

}

}

}

int main()

{

int sockfd, connfd;

struct sockaddr\_in servaddr, cli;

// socket create and verification

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

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

servaddr.sin\_port = htons(PORT);

// connect the client socket to server socket

if (connect(sockfd, (SA\*)&servaddr, sizeof(servaddr)) != 0) {

printf("connection with the server failed...\n");

exit(0);

}

else

printf("connected to the server..\n");

// function for client

clifunc(sockfd);

// close the socket

close(sockfd);

}

// lab01\_q3\_server.c

#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include<unistd.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

// Function designed for chat between client and server.

void servfunc(int sockfd)

{

char buff[MAX];

int n;

// infinite loop for chat

for (;;) {

bzero(buff, MAX);

// read the message from client and copy it in buffer

read(sockfd, buff, sizeof(buff));

// print buffer which contains the client contents

printf("From client: %s\t To client : ", buff);

// Read server message from keyboard in the buffer

n=0;

int op1=0;

int op2=0;

int res;

char sep[10];

int i=0;

while(buff[n]!='\0')

{

if(buff[n]!=' ' && buff[n]!='\n')

{

sep[i++]=buff[n];

}

else

{

if(op1==0)

{

sep[i]='\0';

op1=atoi(sep);

i=0;

}

else if(op2==0)

{

sep[i]='\0';

op2=atoi(sep);

i=0;

}

else

{

char op=sep[0];

if(op=='+')

{

res=op1+op2;

}

}

}

n++;

}

sprintf(buff, "%d", res);

// and send that buffer to client

write(sockfd, buff, sizeof(buff));

// if msg contains "Exit" then server exit and session ended.

if (strncmp("quit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}

}

}

// Driver function

int main()

{

int sockfd, connfd, len;

struct sockaddr\_in servaddr, cli;

// socket create and verification

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

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

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

servaddr.sin\_port = htons(PORT);

// Binding newly created socket to given IP and verification

if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0) {

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

// Now server is ready to listen and verification

if ((listen(sockfd, 5)) != 0) {

printf("Listen failed...\n");

exit(0);

}

else

printf("Server listening..\n");

len = sizeof(cli);

// Accept the data packet from client and verification

connfd = accept(sockfd, (SA\*)&cli, &len);

if (connfd < 0) {

printf("server acccept failed...\n");

exit(0);

}

else

printf("server acccept the client...\n");

int prt=ntohs (cli.sin\_port);

printf("%d\n",prt);

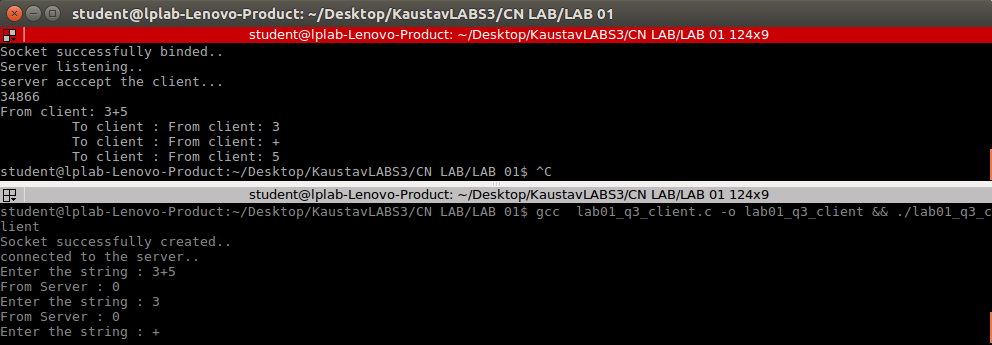
// Function for chatting between client and server

servfunc(connfd);

// After sending exit message close the socket

close(sockfd);

}



// lab01\_q4\_client.c

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include <stdlib.h>

#define PORT 5000

#define MAXLINE 1000

// Driver code

int main()

{

char buffer[100];

char \*message = "What is time?";

int sockfd, n, len;

struct sockaddr\_in servaddr, cliaddr;

// clear servaddr

bzero(&servaddr, sizeof(servaddr));

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

servaddr.sin\_port = htons(PORT);

servaddr.sin\_family = AF\_INET;

// create datagram socket

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

sendto(sockfd, message, MAXLINE, 0, (struct sockaddr \*)&servaddr, sizeof(servaddr));

len = sizeof(cliaddr);

// waiting for response

n = recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr \*)&cliaddr, &len);

buffer[n] = '\0';

printf("Time is : %s \n", buffer);

getchar();

// close the descriptor

close(sockfd);

}

// lab01\_q4\_server.c

// Will practice again

#include <stdio.h>

#include <strings.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <time.h>

#include <unistd.h> // close function

#define PORT 5000

#define MAXLINE 1000

int main()

{

char buffer[100];

int servsockfd, len, n;

struct sockaddr\_in servaddr, cliaddr;

bzero(&servaddr, sizeof(servaddr));

// Create a UDP Socket

servsockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

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

servaddr.sin\_port = htons(PORT);

servaddr.sin\_family = AF\_INET;

// bind server address to socket descriptor

bind(servsockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr));

//receive the datagram

len = sizeof(cliaddr);

n = recvfrom(servsockfd, buffer, sizeof(buffer), 0, (struct sockaddr \*)&cliaddr, &len);

buffer[n] = '\0';

printf("Request: \n");

puts(buffer);

//Echoing back to the client

time\_t t; // not a primitive datatype

time(&t);

sendto(servsockfd, ctime(&t), n, 0, (struct sockaddr \*)&cliaddr, sizeof(cliaddr));

getchar();

// close the descriptor

close(servsockfd);

}

