**Socket programming Assignment**

**Likhita Gonchikara**

**RedID: 815376062**

**1. Introduction**

* 1. **Aim:**

The aim of this project is to create three programs – two client programs and one server program. Client1 will send a C struct comprising of an integer value, a character value and a float value to the server process.

The server changes each of the value in the following way – it doubles the integer value (a 2 becomes a 4), adds 1 to the float value (a 2.3 becomes a 3.3) and increases the character value to the next higher character with a roll over (i.e a ‘z’ becomes ‘a’, ‘b’ becomes a ‘c’). The server then prints the original values that it received and the modified values as well (label these values as “Before” and “After” values). The server then transmits these values to Client 2.

Client 2 prints the values that it received from the server and terminates the connection. All other processes are terminated as well.

The jason.sdsu.edu and rohan.sdsu.edu severs are sun Solaris machines.

* 1. **Client Program :**

Source code of client.c program :

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include "structandhelperfunctions.h"

#define BUF\_SIZE 4096

int main(int argc, char \*argv[]){

printf("This is the client program\n");

int sockfd,port;

char buf[BUF\_SIZE];

struct hostent \*host;

struct sockaddr\_in address;

struct information my\_info;

// check for 3 command line arguments

if (argc < 3) {

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

exit(0);

}

//atoi() converts ASCII to integer and gets the port number

port = atoi(argv[2]);

//Create socket for client.

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

if (sockfd == -1) {

perror("Socket create failed.\n") ;

return -1 ;

}

printf("Client socket created..\n");

//gethostbyname() gets the IP address given the host name

host=gethostbyname(argv[1]);

if(!host){

perror("gethostbyname failed !");

exit(-1);

}

printf("Fetching the server's IP address..\n");

memset(&address,0,sizeof(address));

address.sin\_family = AF\_INET;

memcpy(&address.sin\_addr.s\_addr,host->h\_addr,host->h\_length);

address.sin\_port = htons(port);

//connecting to the server

printf("Connecting to the server, %s on port, %d\n", argv[1], port);

if(connect(sockfd,(struct sockaddr \*)&address,sizeof(address)) == -1)

{

perror(" Connect error has occurred");

exit(-1);

}

printf("Client is now connected to the server \n");

//getting the data from the user and converting the integer and

// float values from host to network format using htonl() and

// convert\_float()

printf("Please enter any integer value:");

scanf("%d", &my\_info.number);

my\_info.number = htonl(my\_info.number);

printf("\nPlease enter any char value:");

scanf("%s", &my\_info.letter);

printf("\nPlease enter any float value:");

scanf("%f", &my\_info.f);

my\_info.f = convert\_float(my\_info.f);

//writing data to the socket file descriptor sockfd

if(write(sockfd, &my\_info, sizeof(struct information)) == -1){

perror(" Write error has occurred");

exit(-1);

}

printf("Data sent to the server!\n");

//read data sent back from server

read(sockfd, buf, BUF\_SIZE);

printf("Server sent : %s\n", buf);

//closing the socket connection

close(sockfd);

printf("Client closes the connection.\n");

exit(0);

}

**1.3 Server Program:**

Source code of server.c program:

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <netdb.h>

#include "structandhelperfunctions.h"

#define BUF\_SIZE 4096

#define QUEUE\_LENGTH 10

int main(int argc, char \*argv[]){

int server\_sockfd, client\_sockfd,port,server\_port;

int sockfd;

char option;

char server\_name[20];

char buf[BUF\_SIZE];

struct sockaddr\_in server\_address;

struct sockaddr\_in address;

struct hostent \*host;

struct information received\_info,modified\_info;

// check for 2 command line arguments

if (argc < 2) {

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

exit(0);

}

//atoi() converts ASCII to integer and gets the port number

port = atoi(argv[1]);

memset(&server\_address,0,sizeof(server\_address));

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

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

server\_address.sin\_port = htons(port) ;

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

if (server\_sockfd == -1) {

perror("Server socket create failed.\n") ;

return -1 ;

}

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

perror(" Bind error has occurred");

exit(-1);

}

printf("Server Bind successful..\n");

//creates a connection queue and wait for clients

if(listen(server\_sockfd, QUEUE\_LENGTH) == -1){

perror(" Listen error has occurred");

exit(-1);

}

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

client\_sockfd = accept(server\_sockfd, 0, 0);

if (client\_sockfd < 0) {

perror(" Accept error has occurred");

exit(-1);

}

printf("Server accepted the client connection..\n");

printf("Server waiting for data..\n");

if(read(client\_sockfd,&received\_info,sizeof(struct information))== -1){

perror(" Read error has occurred");

exit(-1);

}

received\_info.number = ntohl(received\_info.number);

received\_info.f = convert\_float(received\_info.f);

printf("Data received from client:\n");

printf("Integer value: %d\n", received\_info.number) ;

printf("Char value: %c\n", received\_info.letter) ;

printf("Float value: %f\n", received\_info.f) ;

write(client\_sockfd, "Thanks, received your data", 27);

if (close(client\_sockfd) == -1) {

perror("Error occurred while closing connection");

exit(-1);

}

printf("Connection with client has been closed.\n");

if (close(server\_sockfd) == -1) {

perror("Error occurred while closing connection");

exit(-1);

}

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

//modifing the received data as required

modified\_info.number=received\_info.number\*2;

modified\_info.f=received\_info.f+1.0;

if(received\_info.letter == 'z')

modified\_info.letter = 'a';

else if(received\_info.letter == 'Z')

modified\_info.letter = 'A';

else

modified\_info.letter = received\_info.letter+1;

printf("------------------------------\n");

printf("The values after modification:\n");

printf("Integer value: %d\n", modified\_info.number) ;

printf("Char value: %c\n", modified\_info.letter) ;

printf("Float value: %f\n", modified\_info.f) ;

printf("------------------------------\n");

printf("Do you wish to send the modified data to another server? (Y/N):");

scanf("%s", &option);

if(option =='Y'){

printf("\nPlease enter the name of the server:");

scanf("%s", &server\_name);

printf("\nPlease enter the port on which the server is running:");

scanf("%d", &server\_port);

//Create socket for client.

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

if (sockfd == -1) {

perror("Socket create failed.\n") ;

return -1 ;

}

printf("Client socket created\n");

host=gethostbyname(server\_name);

if(!host){

perror("gethostbyname failed");

exit(-1);

}

printf("Fetching the server's IP address..\n");

memset(&address,0,sizeof(address));

address.sin\_family = AF\_INET;

memcpy(&address.sin\_addr.s\_addr,host->h\_addr,host->h\_length);

address.sin\_port = htons(server\_port);

printf("Connecting to the server, %s on port, %d\n", server\_name, server\_port);

if(connect(sockfd, (struct sockaddr \*)&address, sizeof(address)) == -1)

{

perror("Error has occurred");

exit(-1);

}

printf("Connected to the server!\n");

//converting the data to the required before sending.

modified\_info.number = htonl(modified\_info.number);

modified\_info.f = convert\_float(modified\_info.f);

if(write(sockfd, &modified\_info, sizeof(struct information)) == -1)

{

perror(" Write error has occurred");

exit(-1);

}

printf("Data sent to the server..\n");

read(sockfd, buf, BUF\_SIZE);

printf("Server says:%s\n", buf);

close(sockfd);

printf("Client closes the connection.\n");

return 0;

}

printf("thank you!\n");

}

**1.4 Server1.c program**

Source code of server1.c program:

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <netdb.h>

#include "structandhelperfunctions.h"

#define BUF\_SIZE 4096

#define QUEUE\_LENGTH 10

int main(int argc, char \*argv[]){

int server\_sockfd, client\_sockfd,port;

int server\_len ;

int rc ;

unsigned client\_len;

struct sockaddr\_in server\_address;

struct sockaddr\_in client\_address;

struct information received\_info;

// check for 2 command line arguments

if (argc < 2) {

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

exit(0);

}

//atoi() converts ASCII to integer and gets the port number

port = atoi(argv[1]);

memset(&server\_address,0,sizeof(server\_address));

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

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

server\_address.sin\_port = htons(port) ;

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

if (server\_sockfd == -1) {

perror("Server socket create failed.\n") ;

return -1 ;

}

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

perror(" Bind error has occurred");

exit(-1);

}

printf("Server Bind successful..\n");

//creates a connection queue and wait for clients

if(listen(server\_sockfd, QUEUE\_LENGTH) == -1){

perror(" Listen error has occurred");

exit(-1);

}

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

client\_sockfd = accept(server\_sockfd, 0, 0);

if (client\_sockfd < 0) {

perror(" Accept error has occurred");

exit(-1);

}

printf("Server accepted the client connection..\n");

printf("Server waiting for data..\n");

if(read(client\_sockfd,&received\_info, sizeof(struct information)) == -1)

{

perror(" Read error has occurred");

exit(-1);

}

received\_info.number = ntohl(received\_info.number);

received\_info.f = convert\_float(received\_info.f);

printf("------------------------------\n");

printf("Data received from client:\n");

printf("------------------------------\n");

printf("Integer value: %d\n", received\_info.number) ;

printf("Char value: %c\n", received\_info.letter) ;

printf("Float value: %f\n", received\_info.f) ;

printf("------------------------------\n");

write(client\_sockfd, "Thanks, received your data", 27);

if (close(client\_sockfd) == -1) {

perror("Error occurred while closing connection");

exit(-1);

}

printf("Connection with client closed.\n");

if (close(server\_sockfd) == -1) {

perror("Error occurred while closing connection");

exit(-1);

}

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

return 0;

}

**1.5. Helper functions**

Source code of structandhelperfunctions.h

struct information {

int number;

char letter;

float f;

};

int is\_big\_endian(){

int num = 1;

return (\*(char \*)&num != 1);

}

float convert\_float(float value) {

float endian\_value= value;

char\* oldValptr = (char \*)&value;

char\* endianValptr = (char \*)&endian\_value;

if (is\_big\_endian()) {

return value;

}

endianValptr [0] = oldValptr [3];

endianValptr [1] = oldValptr [2];

endianValptr [2] = oldValptr [1];

endianValptr [3] = oldValptr [0];

return endian\_value;

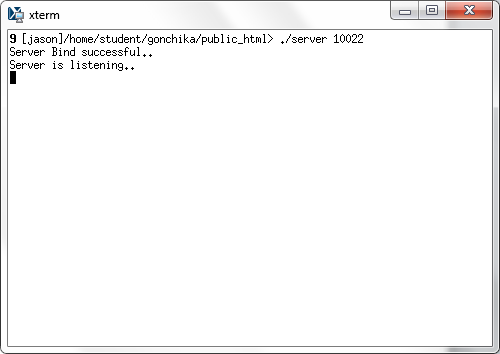
}

**1.6. OUTPUT :**

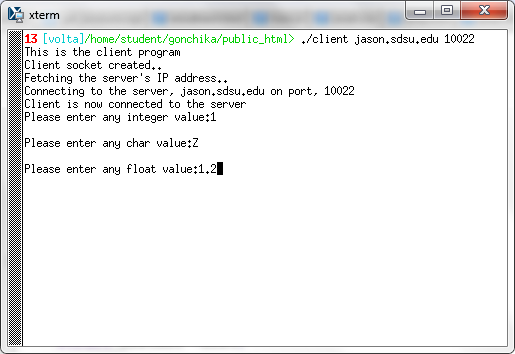
The client.c program is run on the server volta.sdsu.edu. The server.c program is run on the Jason.sdsu.edu. on port number 10022.

The screen shots of the client communicating with the server are shown below:

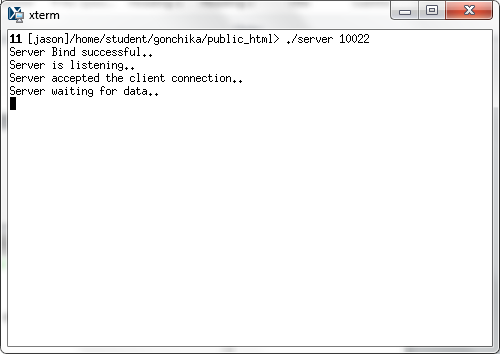
1. Server is listening on jason.sdsu.edu at port 10022.

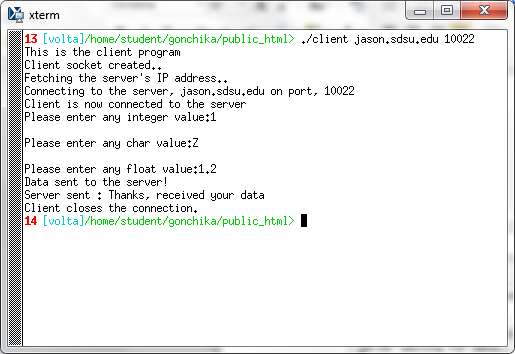


2. Client program is run on volta.sdsu.edu .This client connects to server on Jason.sdsu.edu at port 10022.



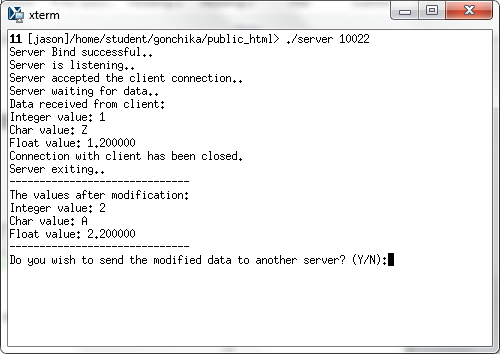
4. Server on jason.sdsu.edu waiting for data.



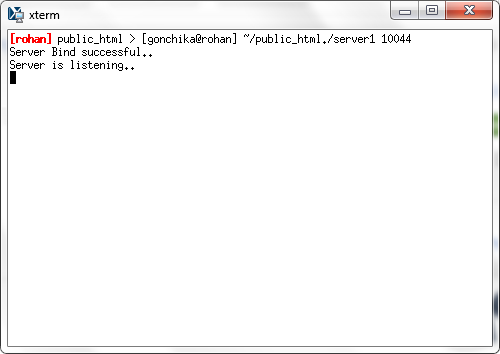


1. Server received the data sent from the client and modifies the data

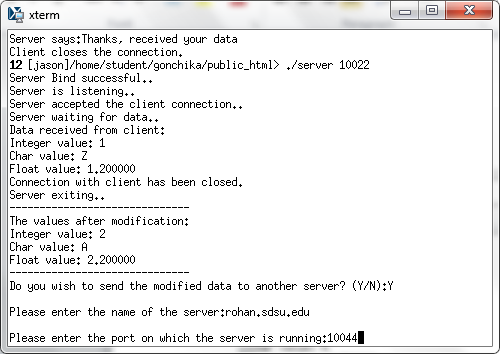
as required.



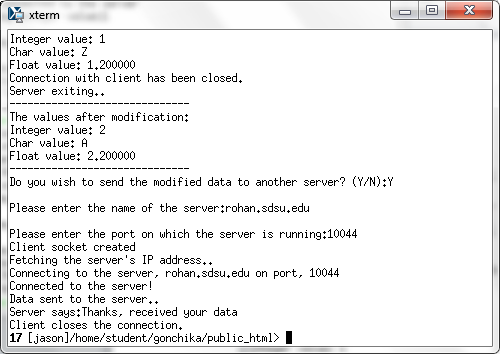
1. Another Server server1.c is run on rohan.sdsu.edu at port 10044.



1. Connecting to server on rohan.sdsu.edu.



1. Modified data sent to server on rohan.sdsu.edu



1. The modified data finally received by the server on rohan.sdsu.edu.

