Distributed Systems Assignment 1

Rishab Ramanathan

19XJ1A0558

1) Server:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

// ./server.o <server port>

#define BUFSIZE 1024

static const int MAXPENDING = 5; // Maximum outstanding connection requests

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

if (argc != 2) {

perror("<server port>");

exit(EXIT\_FAILURE);

}

in\_port\_t servPort = atoi(argv[1]); // Local port

// create socket for incoming connections

int servSock;

if ((servSock = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP)) < 0) {

perror("socket() failed");

exit(-1);

}

// Set local parameters

struct sockaddr\_in servAddr;

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

servAddr.sin\_family = AF\_INET;

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

servAddr.sin\_port = htons(servPort);

// Bind to the local address

if (bind(servSock, (struct sockaddr \*) &servAddr, sizeof(servAddr)) < 0) {

perror("bind() failed");

exit(-1);

}

// Listen to the client

if (listen(servSock, MAXPENDING) < 0) {

perror("listen() failed");

exit(-1);

}

// Server Loop

for (;;) {

struct sockaddr\_in clntAddr;

socklen\_t clntAddrLen = sizeof(clntAddr);

// Wait for a client to connect

int clntSock =

accept(servSock, (struct sockaddr \*) &clntAddr, &clntAddrLen);

if (clntSock < 0) {

perror("accept() failed");

exit(-1);

}

char clntIpAddr[INET\_ADDRSTRLEN];

if (inet\_ntop(AF\_INET, &clntAddr.sin\_addr.s\_addr,

clntIpAddr, sizeof(clntIpAddr)) != NULL) {

printf("----\nHandling client %s %d\n", clntIpAddr, ntohs(clntAddr.sin\_port));

} else {

puts("----\nUnable to get client IP Address");

}

// Receive data

char buffer[BUFSIZE];

memset(buffer, 0, BUFSIZE);

ssize\_t recvLen = recv(clntSock, buffer, BUFSIZE - 1, 0);

if (recvLen < 0) {

perror("recv() failed");

exit(-1);

}

buffer[recvLen] = '\n';

fputs(buffer, stdout);

char servermsg[1024];

while (recvLen > 0) {

// printf("Begining of Client Loop\n");

// Send the received data back to client

ssize\_t sentLen = send(clntSock, buffer, recvLen, 0);

if (sentLen < 0) {

perror("send() failed");

exit(-1);

} else if (sentLen != recvLen) {

perror("send() sent unexpected number of bytes");

exit(-1);

}

// See if there is more data to receive

memset(buffer, 0, BUFSIZE);

recvLen = recv(clntSock, buffer, BUFSIZE, 0);

if (recvLen < 0) {

perror("recv() failed");

exit(-1);

} else if (recvLen > 0) { // some data was remaining

buffer[recvLen] = '\n';

fputs(buffer, stdout);

}

printf("Server: ");

scanf("%s", servermsg);

//print server side typing

if(strcmp(servermsg,"BYE") != 0)

send(clntSock, servermsg, strlen(servermsg), 0);

else

break;

// printf("End of Client Loop\n");

}

close(clntSock);

// printf("End of Server Loop\n");

}

printf("End of Program\n");

}

2) Client

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

// ./client.o <Server Address> <Server Port>

#define BUFSIZE 1024

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

if (argc != 3) {

perror("<Server Address> <Server Port> <Echo Word>");

exit(-1);

}

char \*servIP = argv[1];

char \*echoString;

char tempstring[100];

printf("Enter echo string: ");

scanf("%s",tempstring);

echoString = tempstring;

printf("%s", echoString);

// Set port number as given by user or as default 12345

// in\_port\_t servPort = (argc == 3) ? atoi(argv[2]) : 12345;

// Set port number as user specifies

in\_port\_t servPort = atoi(argv[2]);

//Creat a socket

int sockfd = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (sockfd < 0) {

perror("socket() failed");

exit(-1);

}

// Set the server address

struct sockaddr\_in servAddr;

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

servAddr.sin\_family = AF\_INET;

int err = inet\_pton(AF\_INET, servIP, &servAddr.sin\_addr.s\_addr);

if (err <= 0) {

perror("inet\_pton() failed");

exit(-1);

}

servAddr.sin\_port = htons(servPort);

// Connect to server

if (connect(sockfd, (struct sockaddr \*) &servAddr, sizeof(servAddr)) < 0) {

perror("connect() failed");

exit(-1);

}

size\_t echoStringLen = strlen(echoString);

// Send string to server

do

{

echoStringLen = strlen(echoString);

ssize\_t sentLen = send(sockfd, echoString, echoStringLen, 0);

if (sentLen < 0) {

perror("send() failed");

exit(-1);

} else if (sentLen != echoStringLen) {

perror("send(): sent unexpected number of bytes");

exit(-1);

}

printf("Enter echo string: ");

scanf("%s",echoString);

} while(strcmp(echoString,"BYE") != 0);

// Receive string from server

unsigned int totalRecvLen = 0;

fputs("Received: ", stdout);

while (totalRecvLen < echoStringLen) {

char buffer[BUFSIZE];

memset(buffer, 0, BUFSIZE);

ssize\_t recvLen = recv(sockfd, buffer, BUFSIZE - 1, 0);

if (recvLen < 0) {

perror("recv() failed");

exit(-1);

} else if (recvLen == 0) {

perror("recv() connection closed prematurely");

exit(-1);

}

totalRecvLen += recvLen;

buffer[recvLen] = '\n';

fputs(buffer, stdout);

}

close(sockfd);

exit(0);

}

3) Output:

