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);
```

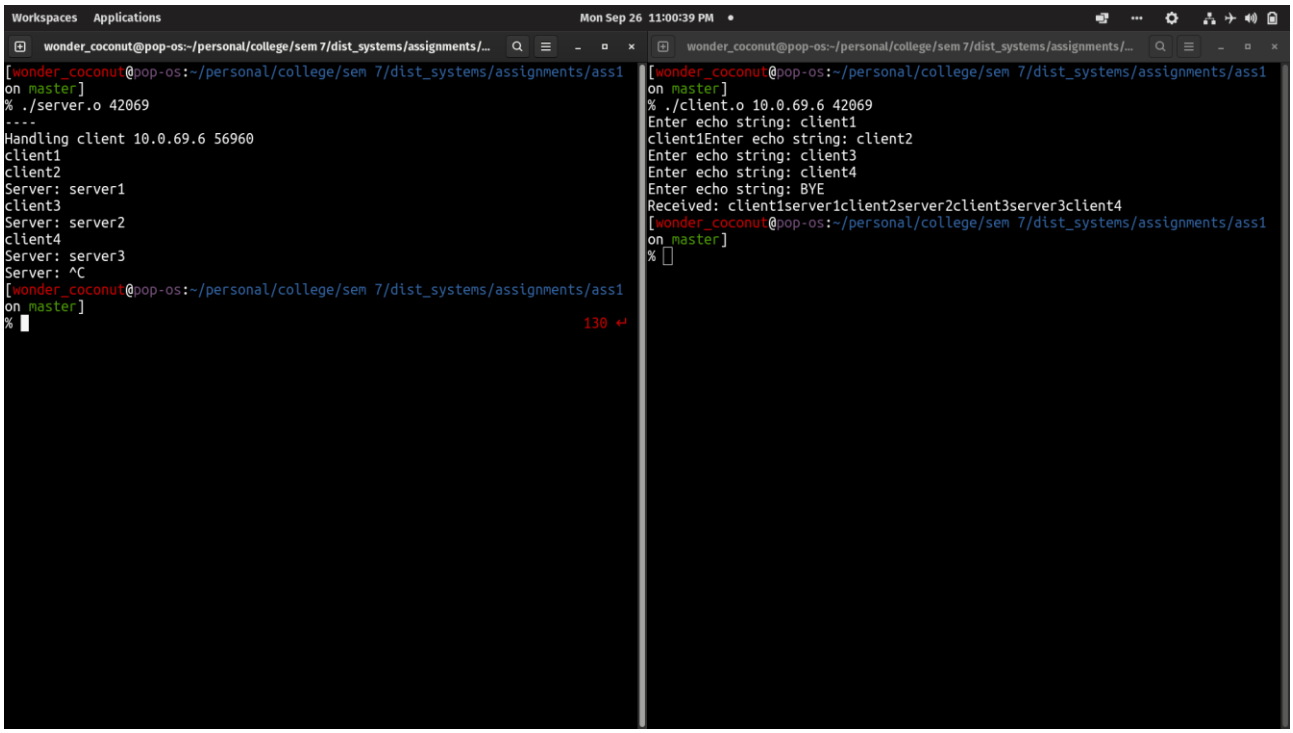
```
//print server side typing
if(strcmp(servermsg,"BYE") != 0)
send(clntSock, servermsg, strlen(servermsg), 0);
else
break;
// printf("End of Client Loop\n");
}

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:


Workspaces Applications
Mon Sep 26 11:00:39 PM
wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignments/...
[wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignments/ass1
on master]
% ./server.o 42069

Handling client 10.0.69.6 56960
client1
client2
Server: server1
client3
Server: server2
client4
Server: server3
Server: ^C
[wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignments/ass1
on master]
% 130 ↵
wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignments/...
[wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignments/ass1
on master]
% ./client.o 10.0.69.6 42069
Enter echo string: client1
client1Enter echo string: client2
Enter echo string: client3
Enter echo string: client4
Enter echo string: BYE
Received: client1server1client2server2client3server3client4
[wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignments/ass1
on master]
%