LAB 1:

Q1)

Server:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

struct message {

int choice, key, res, num[50], even[50], odd[50], numSize, evenSize, oddSize;

};

int main() {

int serverSocket = socket(AF\_INET, SOCK\_STREAM, 0), connSocket;

struct sockaddr\_in server, client;

server.sin\_family = AF\_INET;

server.sin\_port = htons(8001);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

if (bind(serverSocket, (struct sockaddr\*) &server, sizeof(server)) < 0) {

printf("Bind failed");

close(serverSocket); exit(0);

} else {

printf("Bind successful\n");

}

if (listen(serverSocket, 5) < 0) {

printf("Listening error");

close(serverSocket); exit(0);

} else {

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

}

socklen\_t addrLen = sizeof(client);

connSocket = accept(serverSocket, (struct sockaddr\*) &client, &addrLen);

if (connSocket < 0) {

printf("Can't accept connection");

close(serverSocket); exit(0);

} else {

printf("Connection accepted!\n");

}

struct message data;

while(1) {

int readBytes = recv(connSocket, (struct message \*) &data, sizeof(data), 0);

if(readBytes < 0) {

printf("Error in data reception");

continue;

}

int i;

printf("Received following data: ");

for(i=0; i<data.numSize; i++) {

printf("%d ", data.num[i]);

} printf("\n");

int c = data.choice;

printf("Choice: %d", c);

if (c == 1) {

printf("\nSearching!");

int found = 0;

for (i = 0; i < data.numSize; i++) {

if (data.num[i] == data.key) {

found = 1;

break;

}

}

if(found) data.res = 1;

else data.res = 0;

} else if (c == 2) {

printf("\nSorting the array in ascending order!");

for (i=0; i<data.numSize-1; i++) {

int j;

for (j = 0; j < data.numSize - i -1; j++) {

if (data.num[j] > data.num[j + 1]) {

int temp = data.num[j];

data.num[j] = data.num[j + 1];

data.num[j+1] = temp;

}

}

}

} else if (c == 3) {

printf("\nSorting the array in descending order!");

for (i=0; i<data.numSize-1; i++) {

int j;

for (j=0; j<data.numSize-i-1; j++) {

if (data.num[j] < data.num[j+1]) {

int temp = data.num[j];

data.num[j] = data.num[j+1];

data.num[j+1] = temp;

}

}

}

} else if (c == 4) {

printf("\nSpliting into odd even sets!");

data.evenSize = 0;

data.oddSize = 0;

for (i = 0; i < data.numSize; i++) {

if (data.num[i] % 2 == 0)

data.even[data.evenSize++] = data.num[i];

else

data.odd[data.oddSize++] = data.num[i];

}

} else if(c == 5) {

printf("Closing connection");

break;

}

int sentBytes = send(connSocket, (struct message \*) &data, sizeof(data), 0);

if(sentBytes<0) {

printf("Error in data sending!\n");

continue;

}

printf("Data Sent\n");

}

close(connSocket);

close(serverSocket);

}

Client:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

struct message {

int choice, key, res, num[50], even[50], odd[50], numSize, evenSize, oddSize;

};

int main() {

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

struct sockaddr\_in server;

server.sin\_family = AF\_INET;

server.sin\_port = htons(8001);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

if (connect(clientSocket, (struct sockaddr\*) &server, sizeof(server)) < 0) {

printf("Can't connect to server");

exit(0); close(clientSocket);

} else {

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

}

struct message data;

printf("Size of array: ");

scanf("%d", &data.numSize);

printf("Enter integers: ");

int i;

for(i=0; i<data.numSize; i++) {

scanf("%d", &data.num[i]);

}

while(1) {

int exitFlag = 0;

printf("Enter choice: \n1. Search \n2. Sort ascending \n3. Sort descending \n4. Split into odd even \n5. Exit\n");

scanf("%d", &data.choice);

if(data.choice == 1) {

printf("Enter key to be searched: ");

scanf("%d", &data.key);

} else if(data.choice == 5) {

exitFlag = 1;

}

if(exitFlag == 1) {

break;

}

int sentBytes = send(clientSocket, (struct array \*) &data, sizeof(data), 0);

if(sentBytes<0) {

printf("Error in data sending!\n");

continue;

}

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

int i;

int readBytes = recv(clientSocket, (struct message \*) &data, sizeof(data), 0);

if(readBytes < 0) {

printf("Error in data reception");

continue;

}

printf("Data received from server!\n");

int c = data.choice;

if (c == 1) {

if(data.res == 1) printf("Found!\n");

else printf("Not found :(\n");

} else if (c == 2) {

printf("Sorted array in ascending order:\n");

for (i=0; i<data.numSize; i++) {

printf("%d ", data.num[i]);

}

} else if (c == 3) {

printf("Sorted array in descending order:\n");

for (i=0; i<data.numSize; i++) {

printf("%d ", data.num[i]);

}

} else if (c == 4) {

printf("Split into odd even sets:\n");

printf("Even set: \n");

for(i=0; i<data.evenSize; i++) {

printf("%d ", data.odd[i]);

} printf("\n");

printf("Odd set: \n");

for(i=0; i<data.oddSize; i++) {

printf("%d ", data.even[i]);

}

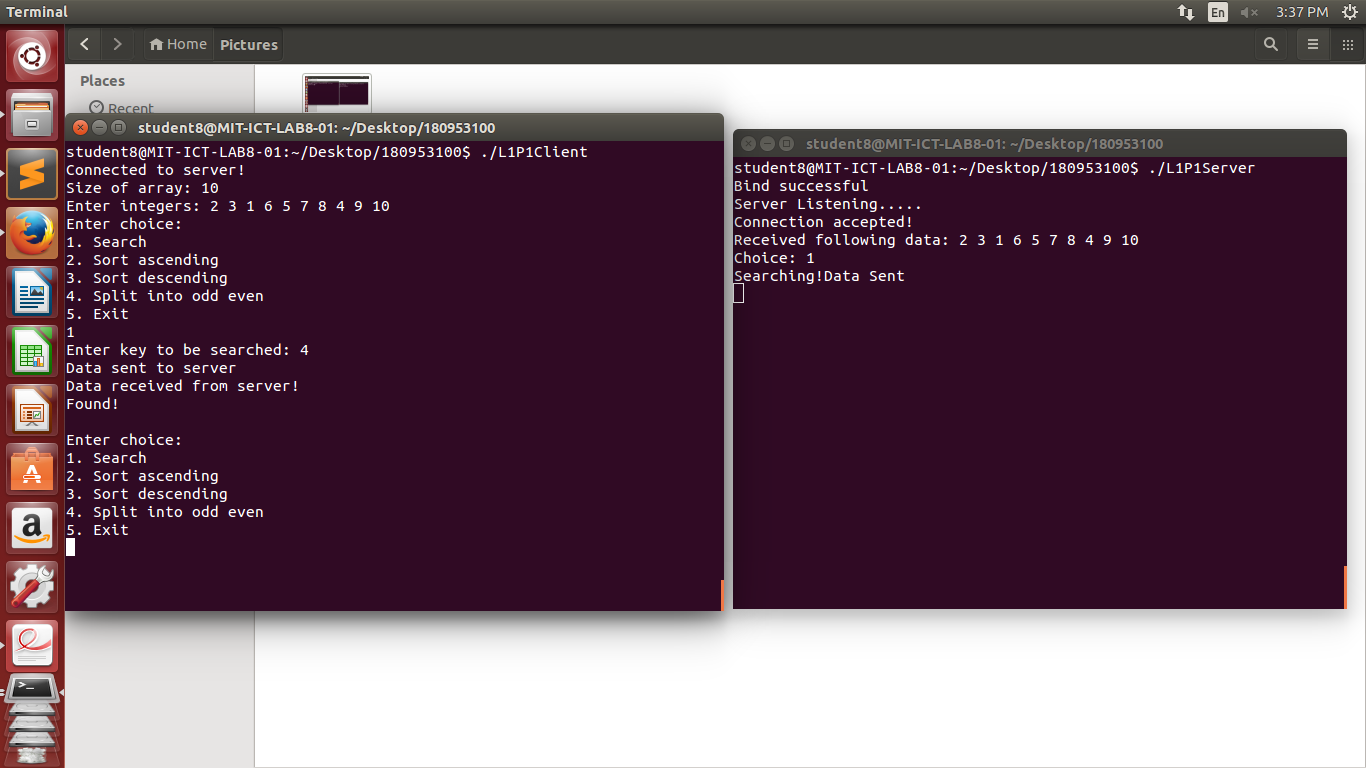
}

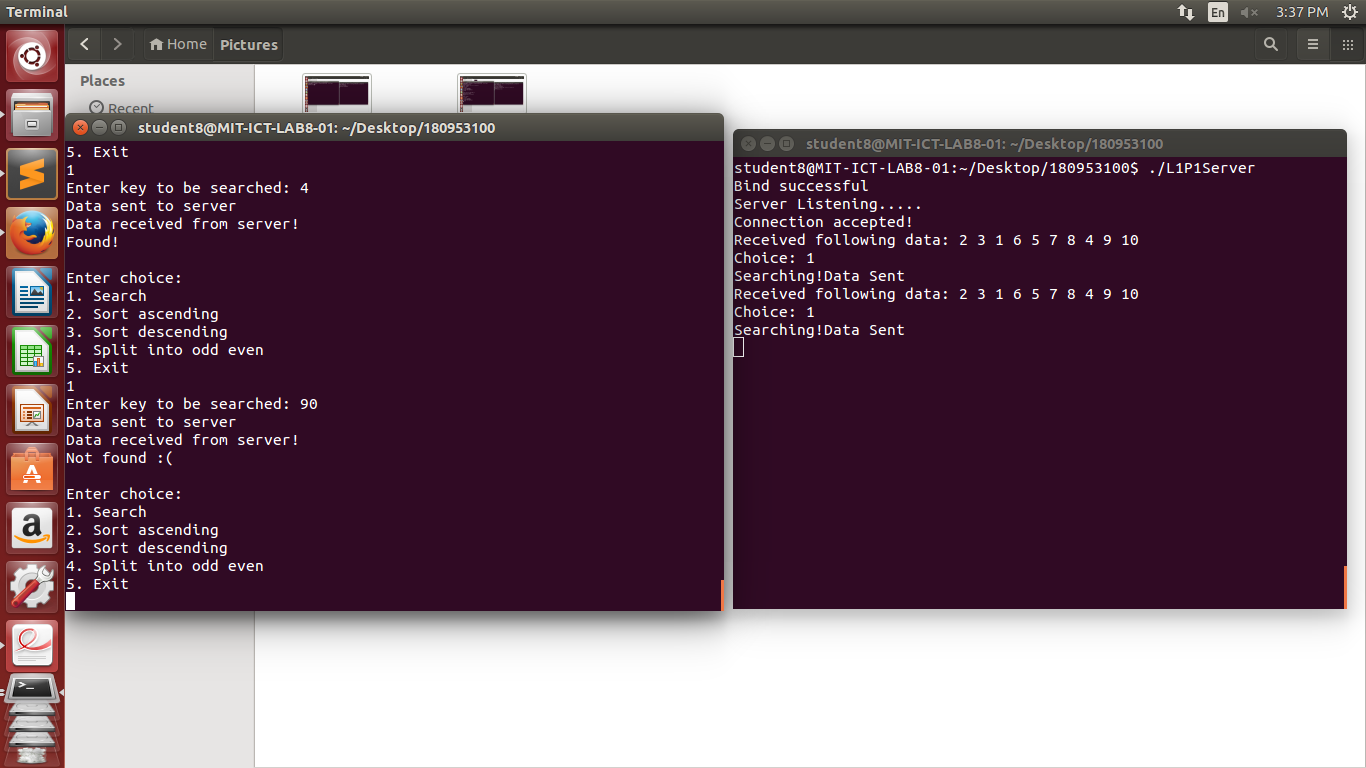
printf("\n");

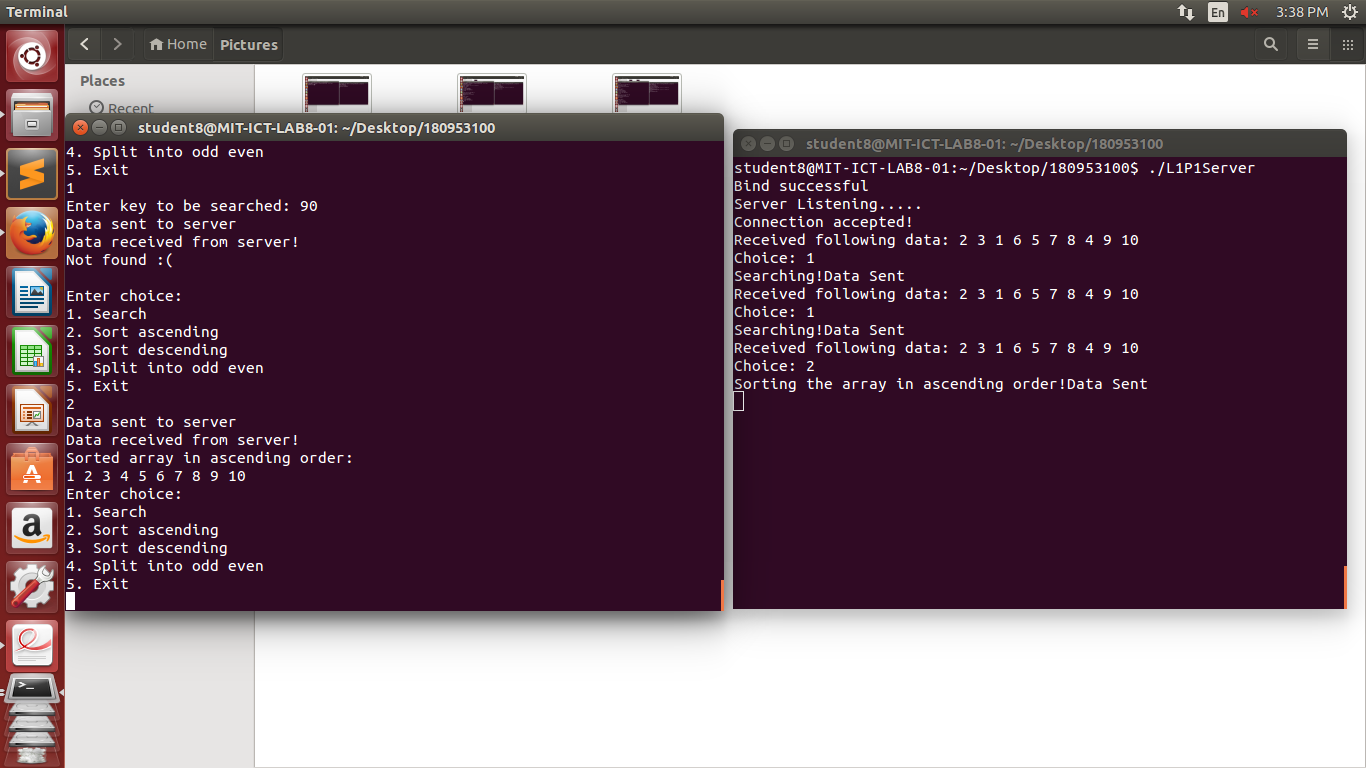
}

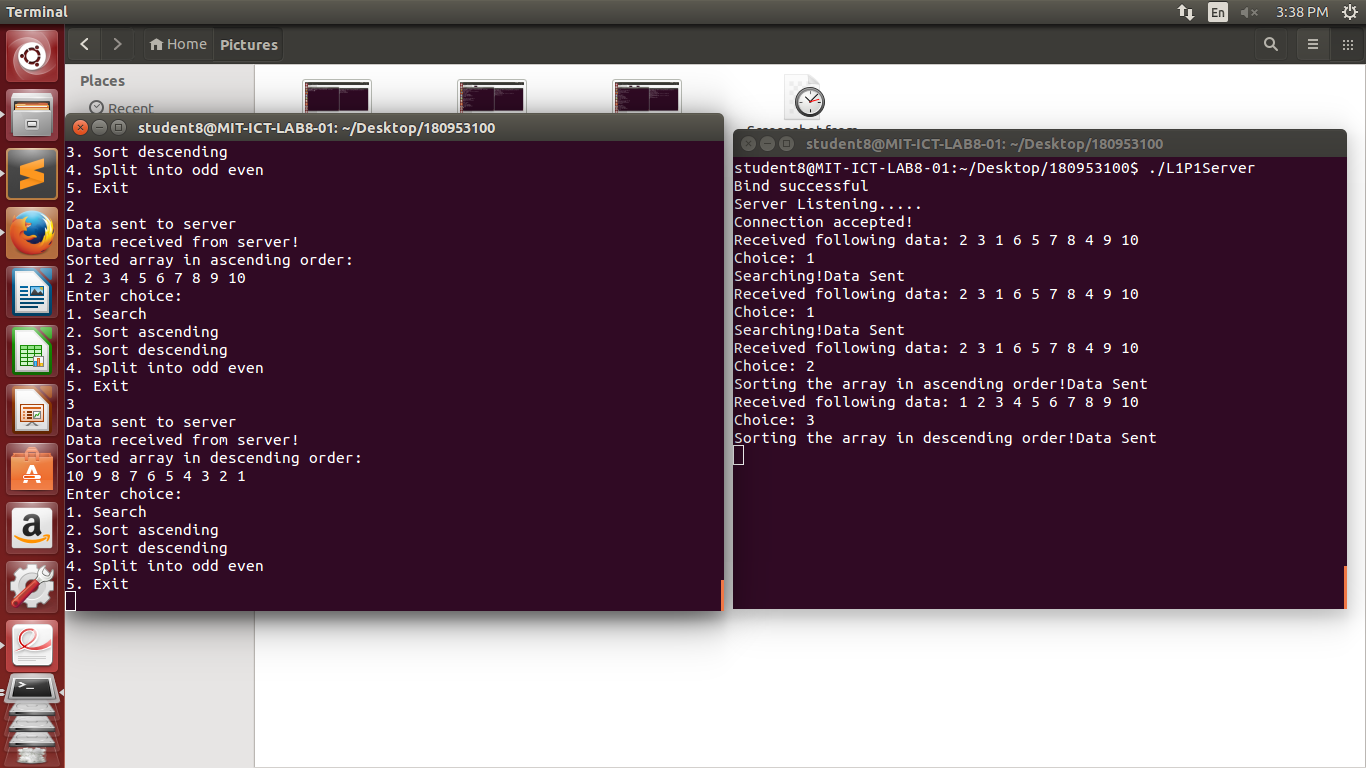
close(clientSocket);

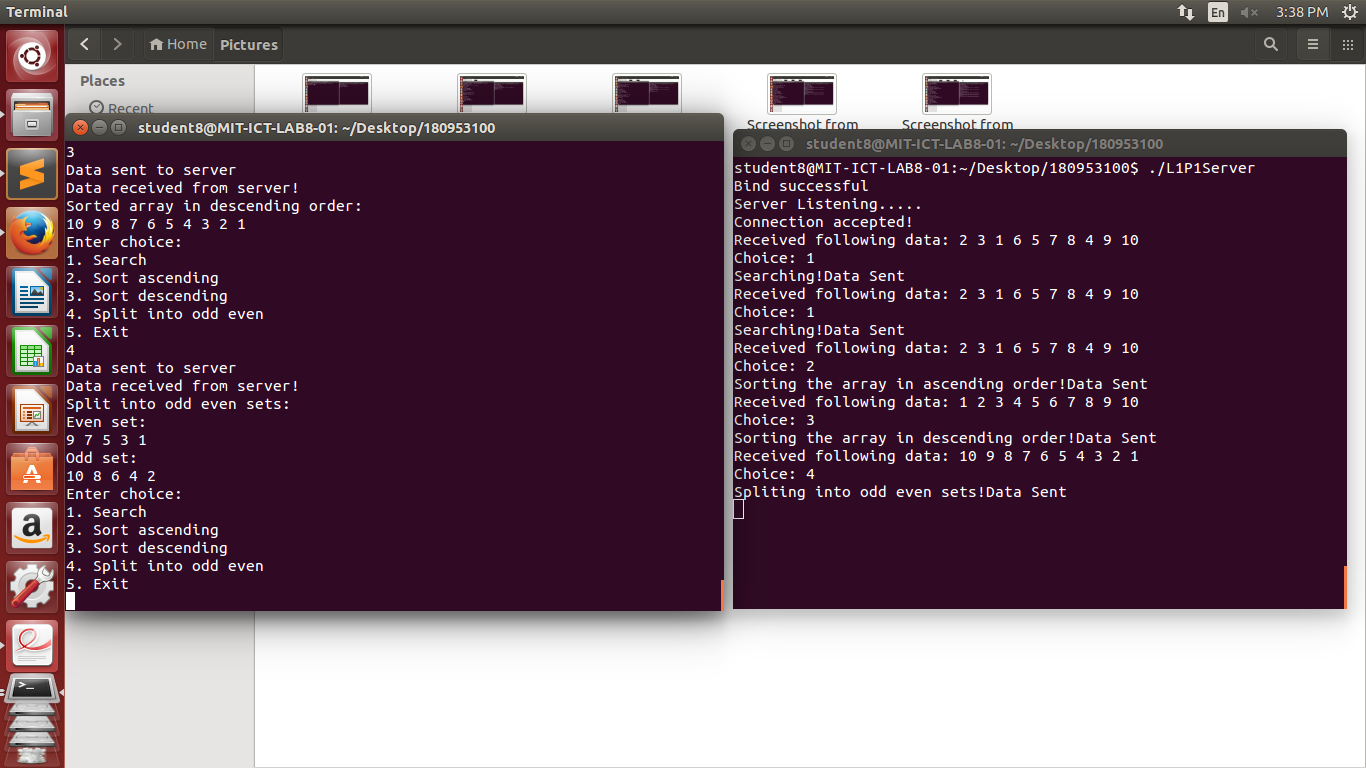
}











Q2): (TCP)

Server:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

struct message {

int size, palindrome;

char string[50];

int a, e, i, o, u;

};

int main() {

int serverSocket = socket(AF\_INET, SOCK\_STREAM, 0), connSocket;

struct sockaddr\_in server, client;

server.sin\_family = AF\_INET;

server.sin\_port = htons(8002);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

if (bind(serverSocket, (struct sockaddr\*) &server, sizeof(server)) < 0) {

printf("Bind failed");

close(serverSocket); exit(0);

} else {

printf("Bind successful\n");

}

if (listen(serverSocket, 5) < 0) {

printf("Listening error");

close(serverSocket); exit(0);

} else {

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

}

socklen\_t addrLen = sizeof(client);

connSocket = accept(serverSocket, (struct sockaddr\*) &client, &addrLen);

if (connSocket < 0) {

printf("Can't accept connection");

close(serverSocket); exit(0);

} else {

printf("Connection Accepted!\n");

}

struct message data;

char recvBuff[50];

while(1) {

int readBytes = recv(connSocket, recvBuff, sizeof(recvBuff), 0);

recvBuff[readBytes] = '\0';

printf("Received from client: %s", recvBuff);

if(strcmp(recvBuff, "Halt") == 0) {

break;

}

int l=0;

int r=strlen(recvBuff) - 1;

data.palindrome = 1;

while (r>l) {

if (recvBuff[l++] != recvBuff[r--]) {

data.palindrome = 0;

}

}

int i;

data.a = 0; data.e = 0; data.i = 0; data.o = 0; data.u = 0;

for(i=0; i<strlen(recvBuff); i++) {

if(recvBuff[i] == 'a') data.a++;

if(recvBuff[i] == 'e') data.e++;

if(recvBuff[i] == 'i') data.i++;

if(recvBuff[i] == 'o') data.o++;

if(recvBuff[i] == 'u') data.u++;

data.string[i] = recvBuff[i];

}

data.string[strlen(recvBuff)] = '\0';

data.size = strlen(recvBuff);

strcpy(recvBuff, "");

printf("\n");

send(connSocket, (struct message \*) &data, sizeof(data), 0);

}

close(connSocket);

close(serverSocket);

}

Client:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

struct message {

int size, palindrome;

char string[50];

int a, e, i, o, u;

};

int main() {

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

struct sockaddr\_in server;

server.sin\_family = AF\_INET;

server.sin\_port = htons(8002);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

if (connect(clientSocket, (struct sockaddr\*) &server, sizeof(server)) < 0) {

printf("Can't connect to server");

exit(0); close(clientSocket);

} else {

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

}

char sendBuff[50];

struct message data;

while(1) {

printf("Enter string: ");

scanf("%s", sendBuff);

send(clientSocket, sendBuff, sizeof(sendBuff), 0);

if(strcmp(sendBuff, "Halt") == 0) {

break;

}

strcpy(sendBuff, "");

int readBytes = recv(clientSocket, (struct message \*) &data, sizeof(data), 0);

printf("Received string: %s \n", data.string);

if(data.palindrome == 1) {

printf("It is a palindrome\n");

} else {

printf("It is not a palindrome\n");

}

printf("Size of string: %d \n", data.size);

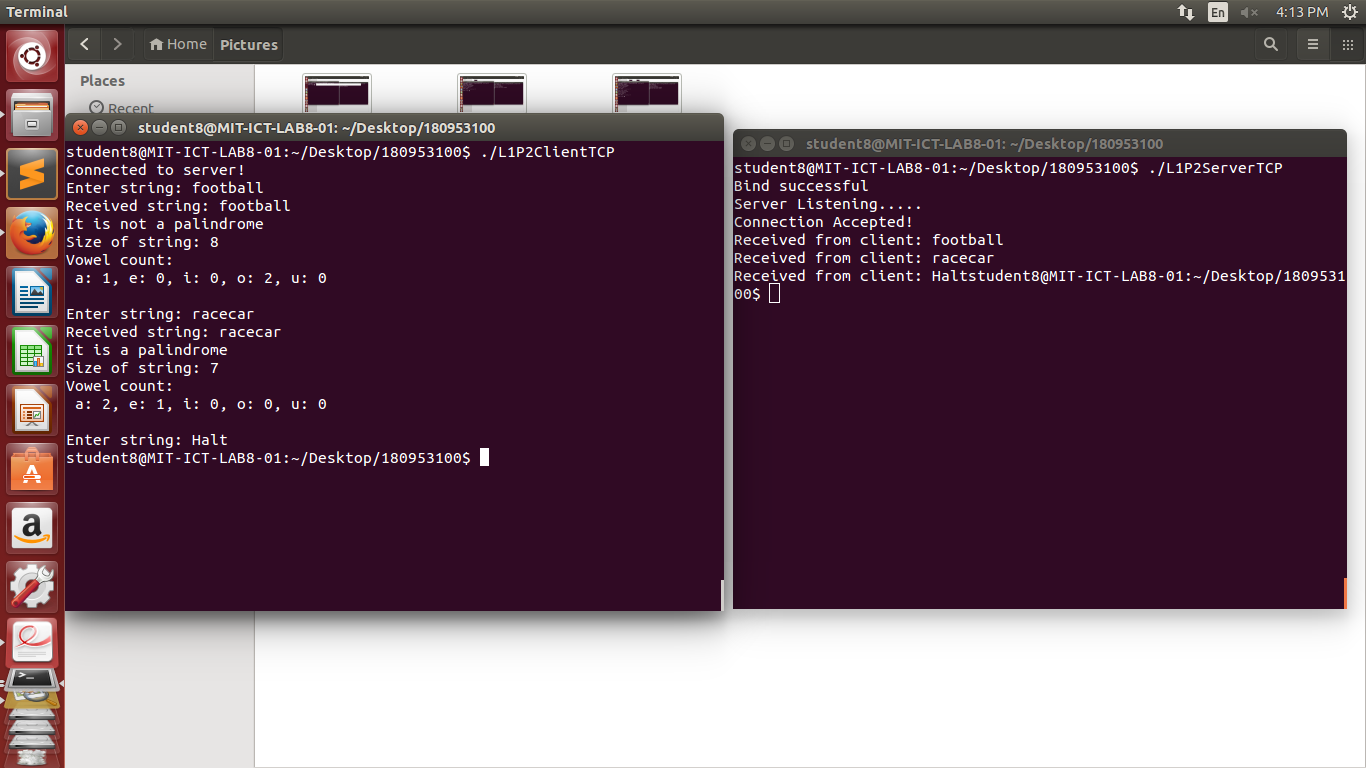
printf("Vowel count: \n a: %d, e: %d, i: %d, o: %d, u: %d \n", data.a, data.e, data.i, data.o, data.u);

printf("\n");

}

close(clientSocket);

}



Q2: (UDP)

Server:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

struct message {

int size, palindrome;

char string[50];

int a, e, i, o, u;

};

int main() {

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

struct sockaddr\_in server, client;

server.sin\_family = AF\_INET;

server.sin\_port = htons(8002);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

if (bind(serverSocket, (struct sockaddr\*) &server, sizeof(server)) < 0) {

printf("Bind failed");

close(serverSocket); exit(0);

} else {

printf("Bind successful\n");

}

struct message data;

char recvBuff[50];

while(1) {

socklen\_t len = sizeof(client);

int readBytes = recvfrom(serverSocket, recvBuff, sizeof(recvBuff), 0, (struct sockaddr \*) &client, &len);

printf("Received from client: %s", recvBuff);

if(strcmp(recvBuff, "Halt") == 0) {

break;

}

int l=0;

int r=strlen(recvBuff) - 1;

data.palindrome = 1;

while (r>l) {

if (recvBuff[l++] != recvBuff[r--]) {

data.palindrome = 0;

}

}

int i;

data.a = 0; data.e = 0; data.i = 0; data.o = 0; data.u = 0;

for(i=0; i<strlen(recvBuff); i++) {

if(recvBuff[i] == 'a') data.a++;

if(recvBuff[i] == 'e') data.e++;

if(recvBuff[i] == 'i') data.i++;

if(recvBuff[i] == 'o') data.o++;

if(recvBuff[i] == 'u') data.u++;

data.string[i] = recvBuff[i];

}

data.string[strlen(recvBuff)] = '\0';

data.size = strlen(recvBuff);

strcpy(recvBuff, "");

printf("\n");

sendto(serverSocket, (struct message \*) &data, sizeof(data), 0, (struct sockaddr \*) &client, sizeof(client));

}

close(serverSocket);

}

Client:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

struct message {

int size, palindrome;

char string[50];

int a, e, i, o, u;

};

int main() {

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

struct sockaddr\_in server;

server.sin\_family = AF\_INET;

server.sin\_port = htons(8002);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

char sendBuff[50];

struct message data;

while(1) {

printf("Enter string: ");

scanf("%s", sendBuff);

sendto(clientSocket, sendBuff, sizeof(sendBuff), 0, (struct sockaddr \*) &server, sizeof(server));

if(strcmp(sendBuff, "Halt") == 0) {

break;

}

strcpy(sendBuff, "");

socklen\_t len = sizeof(server);

int readBytes = recvfrom(clientSocket, (struct message \*) &data, sizeof(data), 0, (struct sockaddr \*) &server, &len);

printf("Received string: %s \n", data.string);

if(data.palindrome == 1) {

printf("It is a palindrome\n");

} else {

printf("It is not a palindrome\n");

}

printf("Size of string: %d \n", data.size);

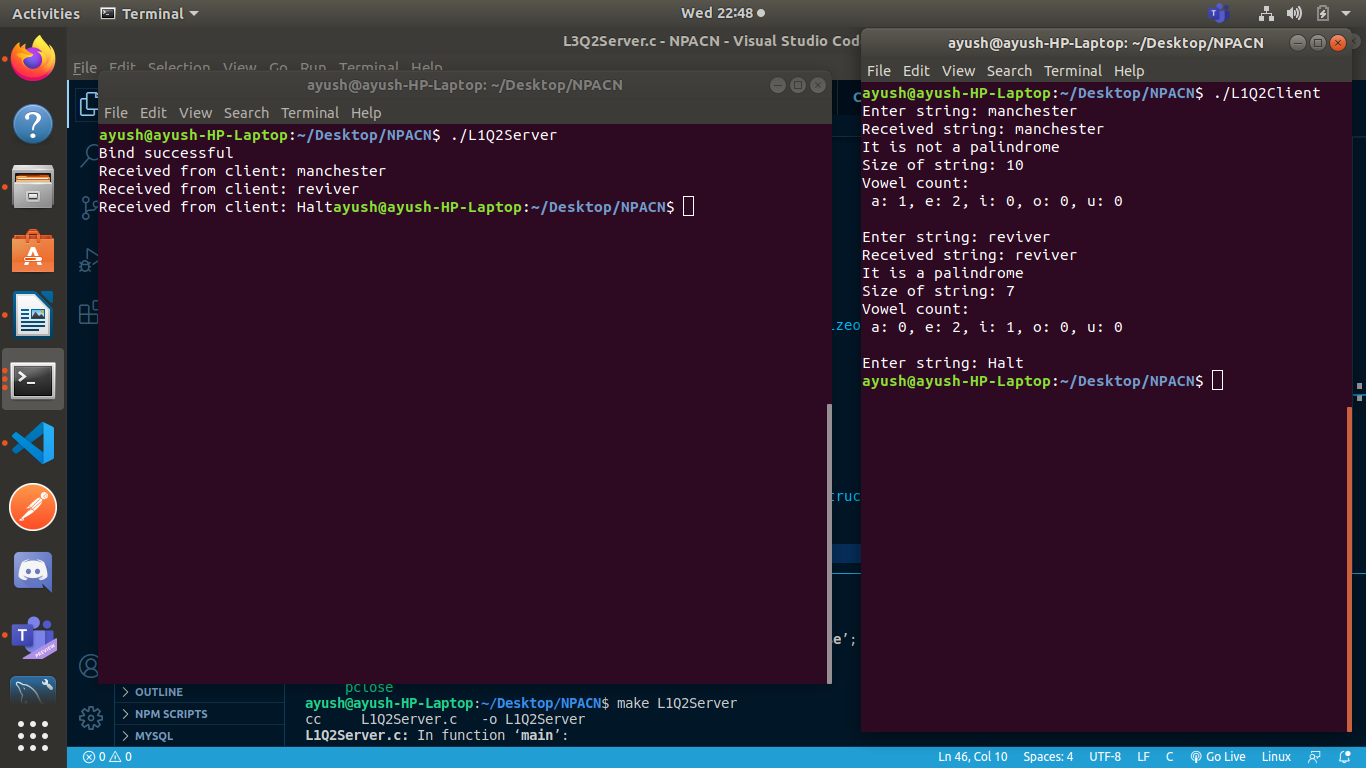
printf("Vowel count: \n a: %d, e: %d, i: %d, o: %d, u: %d \n", data.a, data.e, data.i, data.o, data.u);

printf("\n");

}

close(clientSocket);

}



LAB 2:

Q1: (TCP)

Server:

#include<string.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdlib.h>

#include<stdio.h>

#define MAX\_LEN 100

void replaceAll (char \*str, const char \*oldWord, const char \*newWord) {

char \*pos = NULL, temp[1000];

int index = 0;

int owlen;

owlen = strlen(oldWord);

while((pos = strstr(str,oldWord)) != NULL) {

strcpy(temp, str);

index = pos-str;

str[index] = '\0';

strcat(str, newWord);

strcat(str, temp + index + owlen);

}

}

int main()

{

int s,r,recb,sntb,x,ns,a=0;

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50];

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

if(s==-1) {

printf("\nSocket creation error.");

exit(0);

}

server.sin\_family=AF\_INET;

server.sin\_port=htons(7503);

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

r=bind(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1) {

printf("\nBinding error."); exit(0);

} else {

printf("Bind Successful!");

}

r=listen(s,5);

if(r==-1) {

close(s); exit(0);

}

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

len = sizeof(client);

ns = accept(s, (struct sockaddr\*) &client, &len);

if(ns==-1) {

close(s);

exit(0);

}

recb=recv(ns,buff,sizeof(buff),0);

printf("\nFile Name Received!\n");

char fil[50];

if( access(buff, F\_OK) != -1 ) {

strcpy(fil,buff);

strcpy(buff,"File exists");

}

else {

strcpy(buff,"File does not exist!");

}

sntb=send(ns,buff,sizeof(buff),0);

if(strcmp(buff,"File does not exist!")==0) {

close(s);

close(ns);

exit(0);

}

int ch=0;

while(ch!=4)

{

recb=recv(ns,buff,sizeof(buff),0);

ch = buff[0];

int i,n,n1,n2,j;

char str[50],str1[50],str2[50];

char strTempData[MAX\_LEN];

char \*\*strData = NULL;

int noOfLines = 0;

switch(ch)

{

case 1:

//Searching for file

printf("\nSearching..\n");

n=buff[1];

for(i=0;i<n;i++)

str[i]=buff[i+2];

str[n]='\0';

FILE \*fp;

int line\_num = 1;

int find\_result = 0;

char temp[512];

if((fp = fopen(fil, "r")) == NULL) {

printf("\nFile not found");

close(s);

close(ns);

exit(0);

}

while(fgets(temp, 512, fp) != NULL) {

if((strstr(temp, str)) != NULL) {

find\_result++;

}

line\_num++;

}

if(fp) {

fclose(fp);

}

buff[0]=find\_result;

sntb=send(ns,buff,sizeof(buff),0);

break;

case 2:

//replacing word

n1=buff[1];

i=2;

for(j=0;j<n1;j++) {

str1[j]=buff[i];

i++;

}

str1[j]='\0';

recb=recv(ns,buff,sizeof(buff),0);

n=buff[1]; i=2;

for(j=0;j<n;j++) {

str2[j]=buff[i];

i++;

}

str2[j]='\0';

printf("\nReplacing %s with %s..\n",str1,str2);

FILE \* fPtr;

FILE \* fTemp;

char buffer[1000];

fPtr = fopen(fil, "r");

fTemp = fopen("replace.tmp", "w");

if (fPtr == NULL || fTemp == NULL) {

printf("\nUnable to open file.\n");

exit(0);

}

while ((fgets(buffer, 1000, fPtr)) != NULL) {

replaceAll(buffer, str1, str2);

fputs(buffer, fTemp);

}

fclose(fPtr);

fclose(fTemp);

remove(fil);

rename("replace.tmp", fil);

strcpy(buff,"String replaced");

sntb=send(ns,buff,sizeof(buff),0);

break;

case 3:

printf("\nOrdering file..\n");

FILE \* ptrFileLog = NULL;

FILE \* ptrSummary = NULL;

if ( (ptrFileLog = fopen(fil, "r")) == NULL ) {

fprintf(stderr,"Error: Could not open %s\n",fil);

return 1;

}

if ((ptrSummary = fopen("temp.txt", "a")) == NULL ) {

fprintf(stderr,"Error: Could not open temp.txt\n");

return 1;

}

while(fgets(strTempData, MAX\_LEN, ptrFileLog) != NULL) {

if(strchr(strTempData,'\n'))

strTempData[strlen(strTempData)-1] = '\0';

strData = (char\*\*)realloc(strData, sizeof(char\*\*)\*(noOfLines+1));

strData[noOfLines] = (char\*)calloc(MAX\_LEN,sizeof(char));

strcpy(strData[noOfLines], strTempData);

noOfLines++;

}

for(i= 0; i < (noOfLines - 1); ++i) {

for(j = 0; j < ( noOfLines - i - 1); ++j) {

if(strcmp(strData[j], strData[j+1]) > 0) {

strcpy(strTempData, strData[j]);

strcpy(strData[j], strData[j+1]);

strcpy(strData[j+1], strTempData);

}

}

}

for(i = 0; i < noOfLines; i++)

fprintf(ptrSummary,"%s\n",strData[i]);

for(i = 0; i < noOfLines; i++)

free(strData[i]);

free(strData);

remove(fil);

rename("temp.txt",fil);

fclose(ptrFileLog);

fclose(ptrSummary);

strcpy(buff,"Ordering done!");

sntb=send(ns,buff,sizeof(buff),0);

break;

case 4:

ch=4;

break;

}

}

close(ns);

close(s);

}

Client:

#include<string.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<stdio.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<fcntl.h>

#include<sys/stat.h>

int main()

{

int s,r,recb,sntb,x;

struct sockaddr\_in server;

char buff[50];

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

server.sin\_family=AF\_INET;

server.sin\_port=htons(7503);

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

r=connect(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1) {

printf("\nConnection error.");

exit(0);

}

printf("Enter File Name: ");

scanf("%s", buff);

sntb=send(s,buff,sizeof(buff),0);

recb=recv(s,buff,sizeof(buff),0);

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

if(strcmp(buff,"File does not exist!")==0) {

close(s);

exit(0);

}

int ch=0;

while(ch!=4)

{

printf("\n1.Search\n2.Replace\n3.Reorder\n4.Exit\nEnter your choice: ");

scanf("%d",&ch);

buff[0]=ch;

char str1[50],str2[50];

int n,i,j;

switch(ch)

{

case 1:

printf("\nEnter string to be searched: ");

scanf("%s",str1);

n=strlen(str1);

buff[1]=n;

for(i=0;i<n;i++)

buff[i+2]=str1[i];

buff[i+2]='\0';

sntb=send(s,buff,sizeof(buff),0);

recb=recv(s,buff,sizeof(buff),0);

n=buff[0];

printf("\nWord Count: %d\n",n);

break;

case 2:

printf("\nEnter string to be searched: ");

scanf("%s",str1);

n=strlen(str1);

buff[1]=n;

for(i=0;i<n;i++)

buff[i+2]=str1[i];

buff[i+2]='\0';

sntb=send(s,buff,sizeof(buff),0);

printf("\nEnter replacement string: ");

scanf("%s",str2);

n=strlen(str2);

buff[1]=n;

i=2;

for(j=0;j<n;j++) {

buff[i]=str2[j];

i++;

}

buff[i]='\0';

sntb=send(s,buff,sizeof(buff),0);

recb=recv(s,buff,sizeof(buff),0);

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

break;

case 3:

sntb=send(s,buff,sizeof(buff),0);

recb=recv(s,buff,sizeof(buff),0);

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

break;

case 4:

sntb=send(s,buff,sizeof(buff),0);

break;

default:

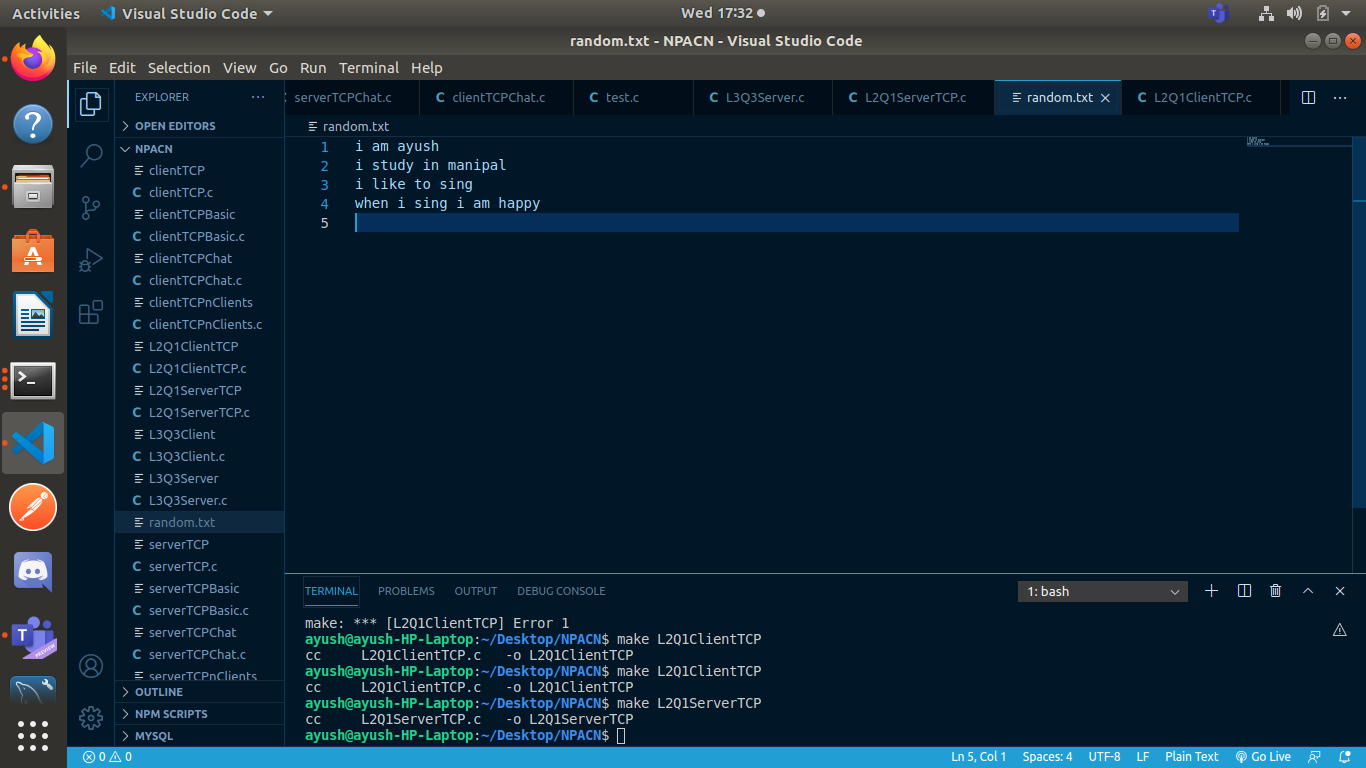
printf("\n Try Again!\n");

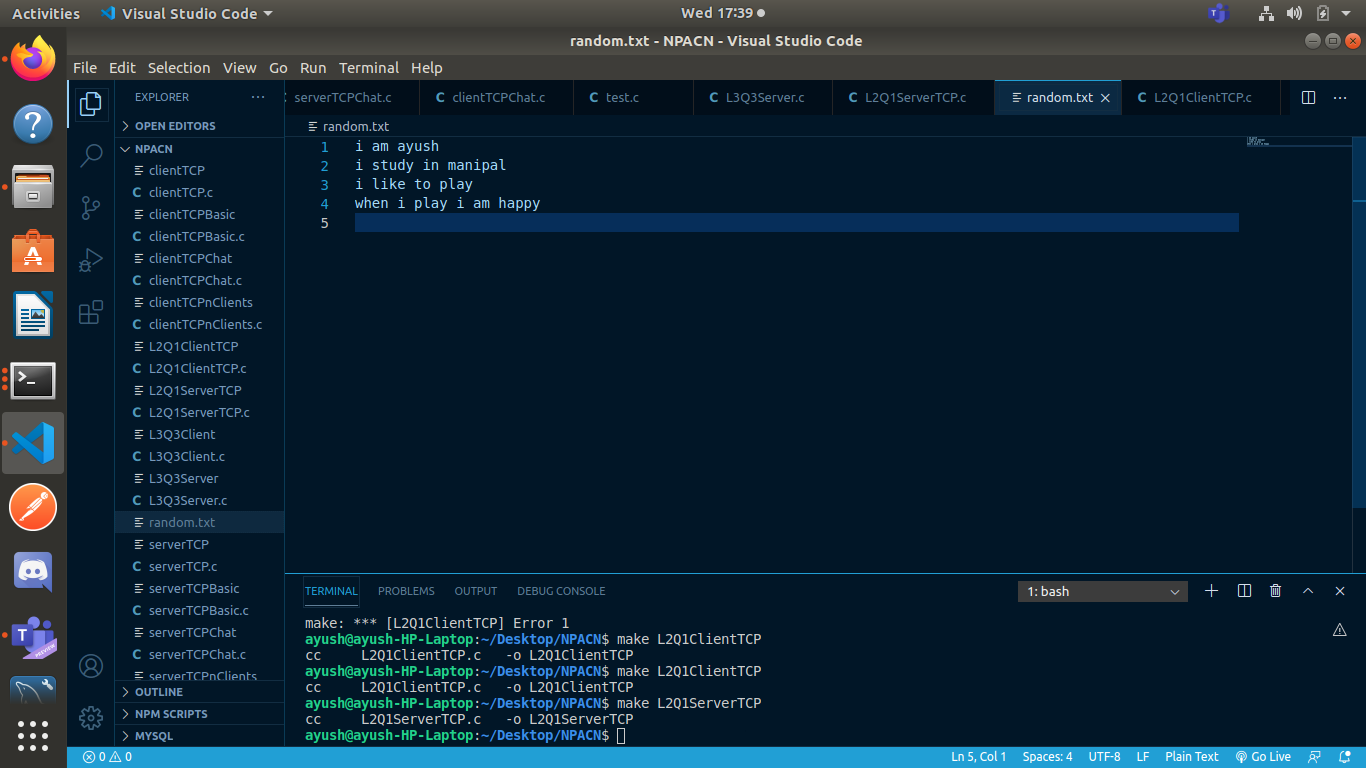
}

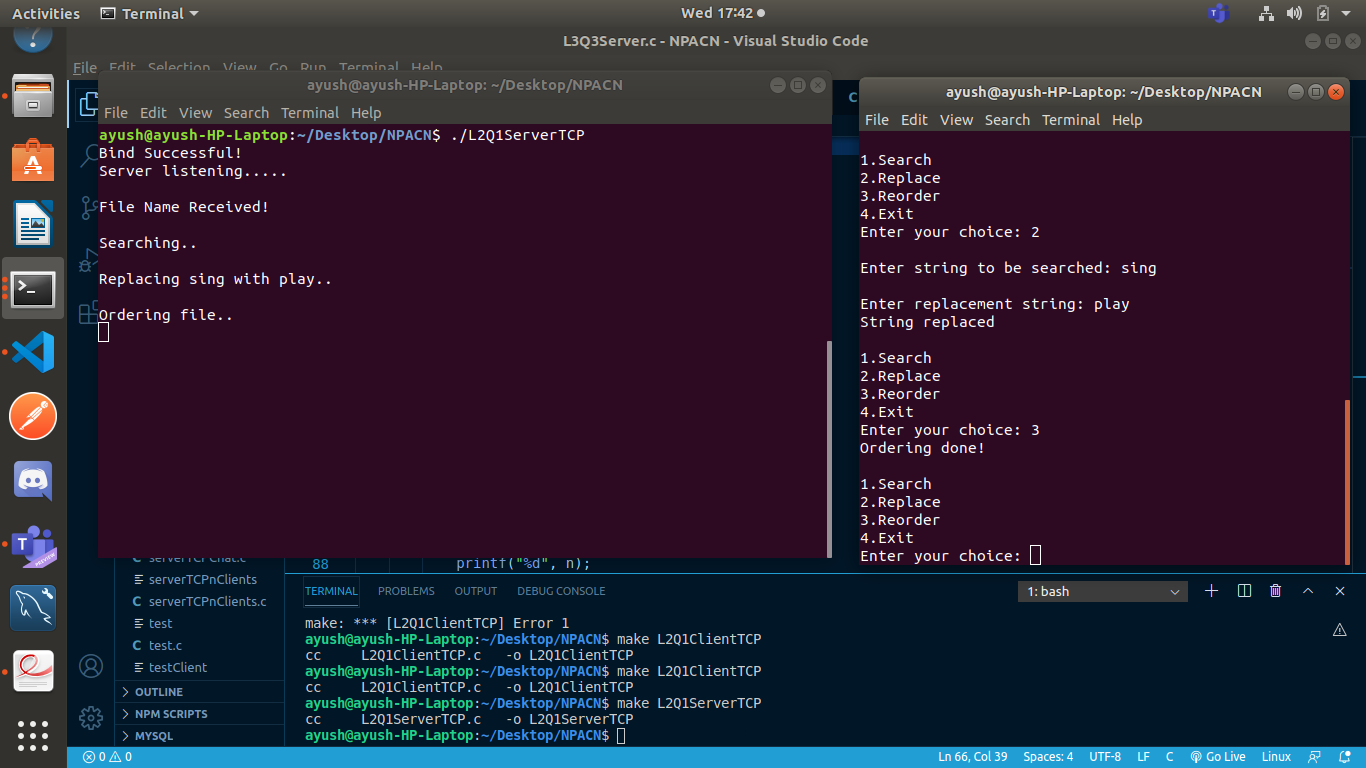
}

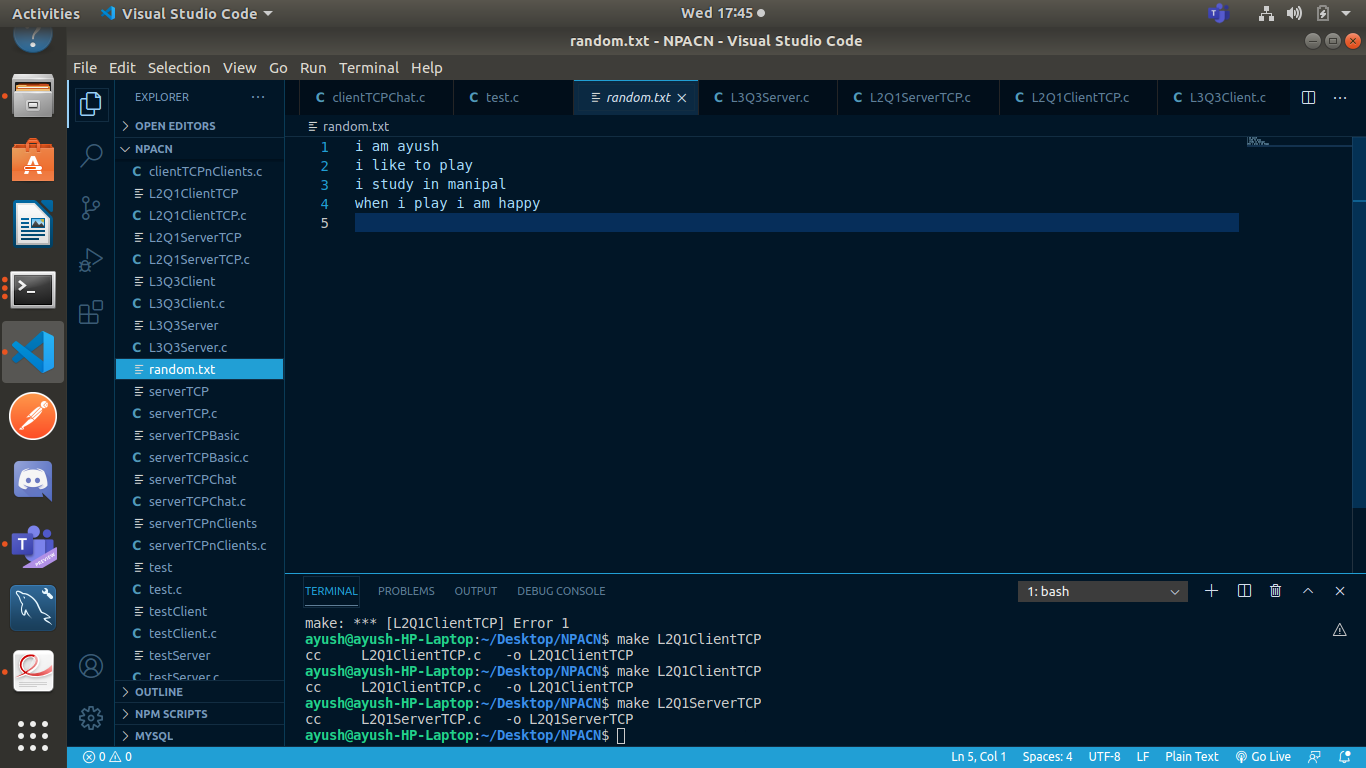
close(s);

}









Q1) UDP:

Server:

#include<string.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdlib.h>

#include<stdio.h>

#define MAX\_LEN 100

void replaceAll(char \*str, const char \*oldWord, const char \*newWord)

{

char \*pos, temp[1000];

int index = 0;

int owlen;

owlen = strlen(oldWord);

while ((pos = strstr(str, oldWord)) != NULL)

{

strcpy(temp, str);

index = pos - str;

str[index] = '\0';

strcat(str, newWord);

strcat(str, temp + index + owlen);

}

}

int main()

{

int s,r,recb,sntb,x;

int ca;

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50];

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

server.sin\_family=AF\_INET;

server.sin\_port=htons(7504);

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

len=sizeof(client);

ca=sizeof(client);

r=bind(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1) {

printf("\nBinding error.");

exit(0);

} else {

printf("Bind successful!\n");

}

recb=recvfrom(s,buff,sizeof(buff),0,(struct sockaddr\*)&client,&ca);

printf("\nFile Name Received!\n");char fil[50];

if( access( buff, F\_OK ) != -1 ) {

strcpy(fil,buff);

strcpy(buff,"File exists");

} else {

strcpy(buff,"File does not exist!");

}

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr\*)&client,len);

if(strcmp(buff,"File does not exist!")==0) {

close(s); exit(0);

}

int ch=0;

while(ch!=4) {

recb=recvfrom(s,buff,sizeof(buff),0,(struct sockaddr\*)&client,&ca);

ch = buff[0];

int i,n,n1,n2,j;

char str[50],str1[50],str2[50];

char strTempData[MAX\_LEN];

char \*\*strData = NULL;

int noOfLines = 0;

switch(ch) {

case 1:

printf("\nSearching..\n");

n=buff[1];

for(i=0;i<n;i++)

str[i]=buff[i+2];

str[n]='\0';

FILE \*fp;

int line\_num = 1;

int find\_result = 0;

char temp[512];

if((fp = fopen(fil, "r")) == NULL) {

printf("\nFile not found");

close(s);

exit(0);

}

while(fgets(temp, 512, fp) != NULL) {

if((strstr(temp, str)) != NULL)

find\_result++;

line\_num++;

}

if(fp) fclose(fp);

buff[0]=find\_result;

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr\*)&client,len);

break;

case 2:

//replacing word

n1=buff[1];

i=2;

for(j=0;j<n1;j++) {

str1[j]=buff[i];

i++;

}

str1[j]='\0';

recb=recvfrom(s,buff,sizeof(buff),0,(struct sockaddr\*)&client,&ca);

n=buff[1];

i=2;

for(j=0;j<n;j++) {

str2[j]=buff[i];

i++;

}

str2[j]='\0';

printf("\nReplacing %s with %s..\n",str1,str2);

FILE \* fPtr;

FILE \* fTemp;

char buffer[1000];

fPtr = fopen(fil, "r");

fTemp = fopen("replace.tmp", "w");

if (fPtr == NULL || fTemp == NULL) {

printf("\nUnable to open file.\n");

exit(0);

}

while ((fgets(buffer, 1000, fPtr)) != NULL) {

replaceAll(buffer, str1, str2);

fputs(buffer, fTemp);

}

fclose(fPtr);

fclose(fTemp);

remove(fil);

rename("replace.tmp", fil);

strcpy(buff,"String replaced");

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr\*)&client,len);

break;

case 3:

printf("\nOrdering file..\n");

FILE \* ptrFileLog = NULL;

FILE \* ptrSummary = NULL;

if ( (ptrFileLog = fopen(fil, "r")) == NULL ) {

fprintf(stderr,"Error: Could not open %s\n",fil);

return 1;

}

if ( (ptrSummary = fopen("temp.txt", "a")) == NULL ) {

fprintf(stderr,"Error: Could not open temp.txt\n");

return 1;

}

while(fgets(strTempData, MAX\_LEN, ptrFileLog) != NULL) {

if(strchr(strTempData,'\n'))

strTempData[strlen(strTempData)-1] = '\0';

strData = (char\*\*)realloc(strData, sizeof(char\*\*)\*(noOfLines+1));strData[noOfLines] = (char\*)calloc(MAX\_LEN,sizeof(char));

strcpy(strData[noOfLines], strTempData);

noOfLines++;

}

for(i= 0; i < (noOfLines - 1); ++i) {

for(j = 0; j < ( noOfLines - i - 1); ++j) {

if(strcmp(strData[j], strData[j+1]) > 0) {

strcpy(strTempData, strData[j]);

strcpy(strData[j], strData[j+1]);

strcpy(strData[j+1], strTempData);

}

}

}

for(i = 0; i < noOfLines; i++)

fprintf(ptrSummary,"%s\n",strData[i]);

for(i = 0; i < noOfLines; i++)

free(strData[i]);

free(strData);

remove(fil);rename("temp.txt",fil);

fclose(ptrFileLog);

fclose(ptrSummary);

strcpy(buff,"Ordering done!");

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr\*)&client,len);

break;

case 4:

ch=4;

break;

}

}

close(s);

}

Client:

#include<string.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<stdio.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<fcntl.h>

#include<sys/stat.h>

int main()

{

int s,r,recb,sntb,x;

int sa;

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50];

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

server.sin\_family=AF\_INET;

server.sin\_port=htons(7504);

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

sa=sizeof(server);

len=sizeof(server);

printf("\nEnter File Name: ");

scanf("%s", buff);

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr \*)&server, len);

recb=recvfrom(s,buff,sizeof(buff),0,(struct sockaddr \*)&server,&sa);

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

if(strcmp(buff,"File does not exist!")==0) {

close(s); exit(0);

}

int ch=0;

while(ch!=4)

{

printf("\n1.Search\n2.Replace\n3.Reorder\n4.Exit\nEnter your choice: ");

scanf("%d",&ch);

buff[0]=ch;

char str1[50],str2[50];

int n,i,j;

switch(ch)

{

case 1:

printf("\nEnter string to be searched: ");

scanf("%s",str1);

n=strlen(str1);

buff[1]=n;

for(i=0;i<n;i++)

buff[i+2]=str1[i];

buff[i+2]='\0';

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr \*)&server, len);

recb=recvfrom(s,buff,sizeof(buff),0,(struct sockaddr \*)&server,&sa);

n=buff[0];

printf("\nWord Count: %d\n",n);

break;

case 2:

printf("\nEnter string to be searched: ");

scanf("%s",str1);

n=strlen(str1);

buff[1]=n;

for(i=0;i<n;i++)

buff[i+2]=str1[i];

buff[i+2]='\0';

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr \*)&server, len);

printf("\nEnter replacement string: ");

scanf("%s",str2);

n=strlen(str2);

buff[1]=n;

i=2;

for(j=0;j<n;j++) {

buff[i]=str2[j];

i++;

}

buff[i]='\0';

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr \*)&server, len);

recb=recvfrom(s,buff,sizeof(buff),0,(struct sockaddr \*)&server,&sa);

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

break;

case 3:

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr \*)&server, len);

recb=recvfrom(s,buff,sizeof(buff),0,(struct sockaddr \*)&server,&sa);

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

break;

case 4:

sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr \*)&server, len);

break;

default:

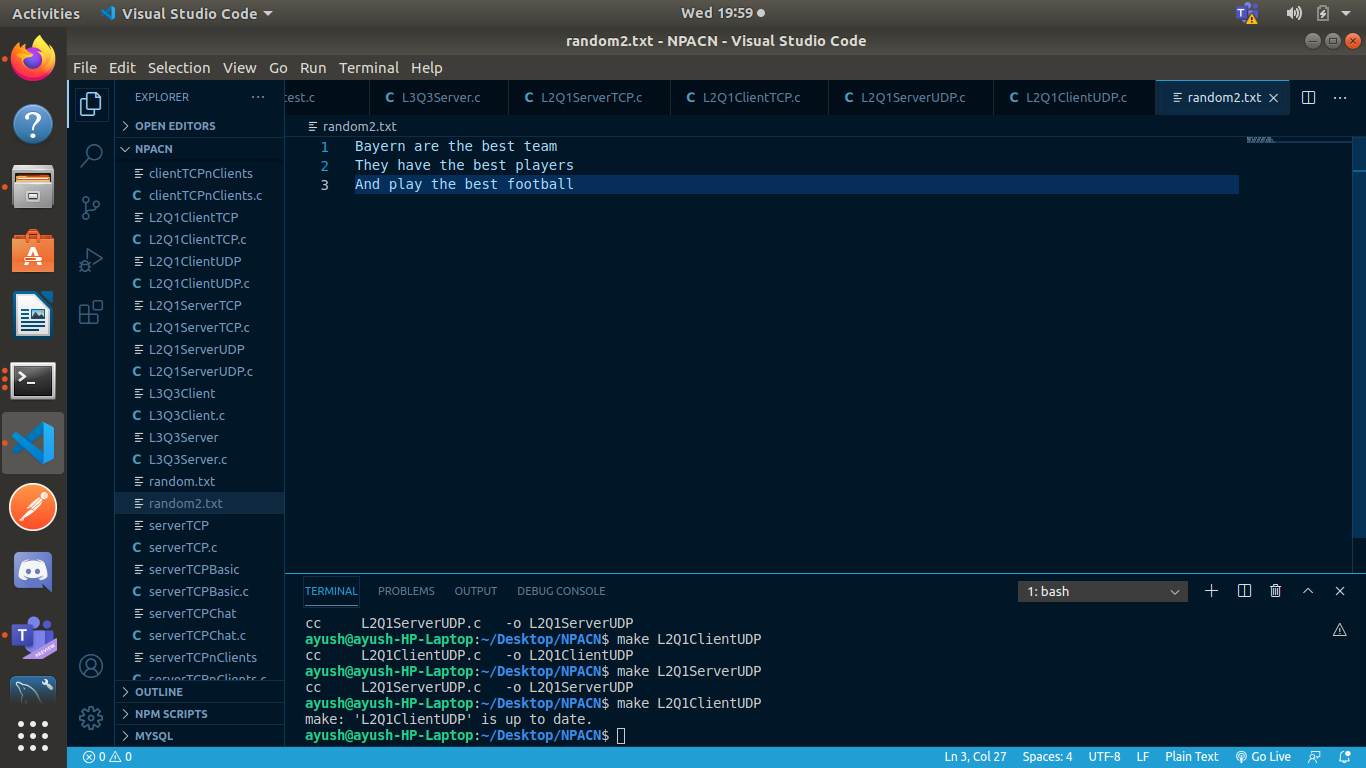
printf("\n Try Again!\n");

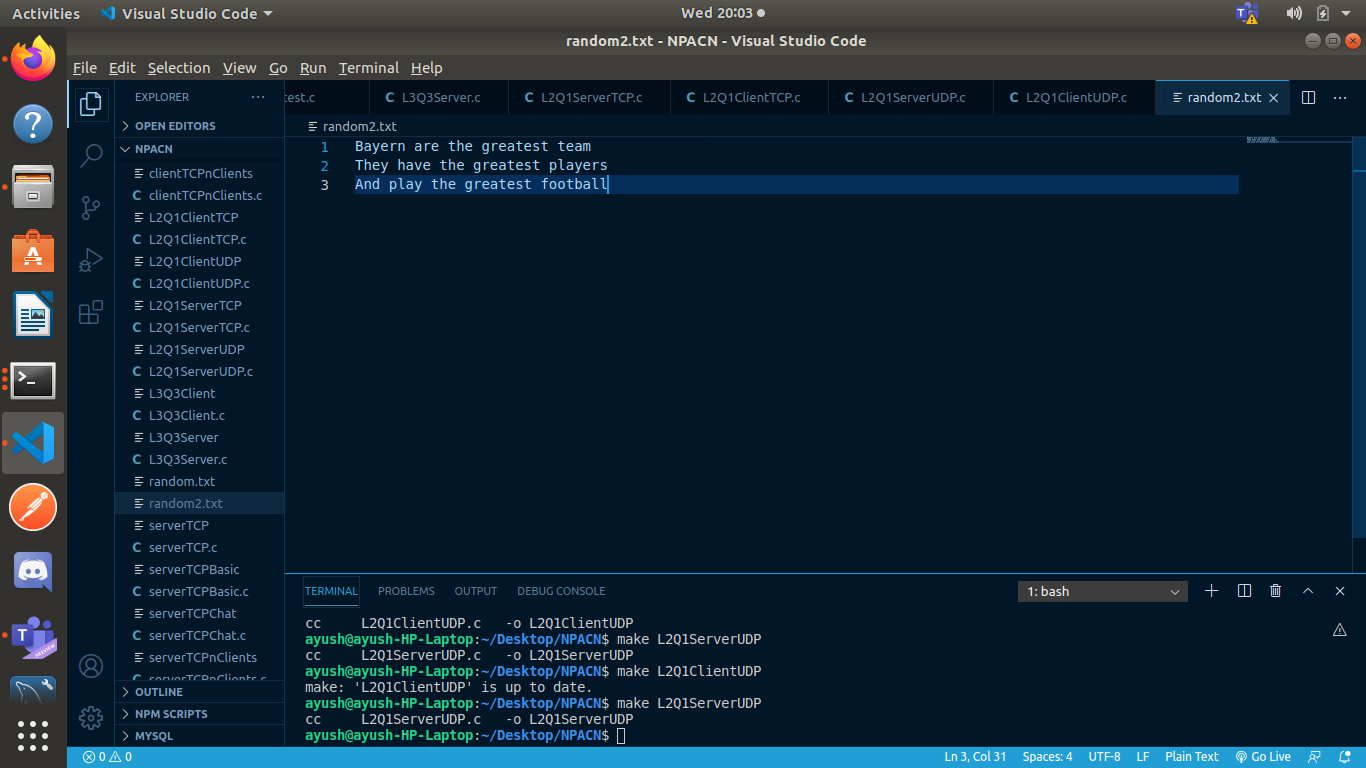
}

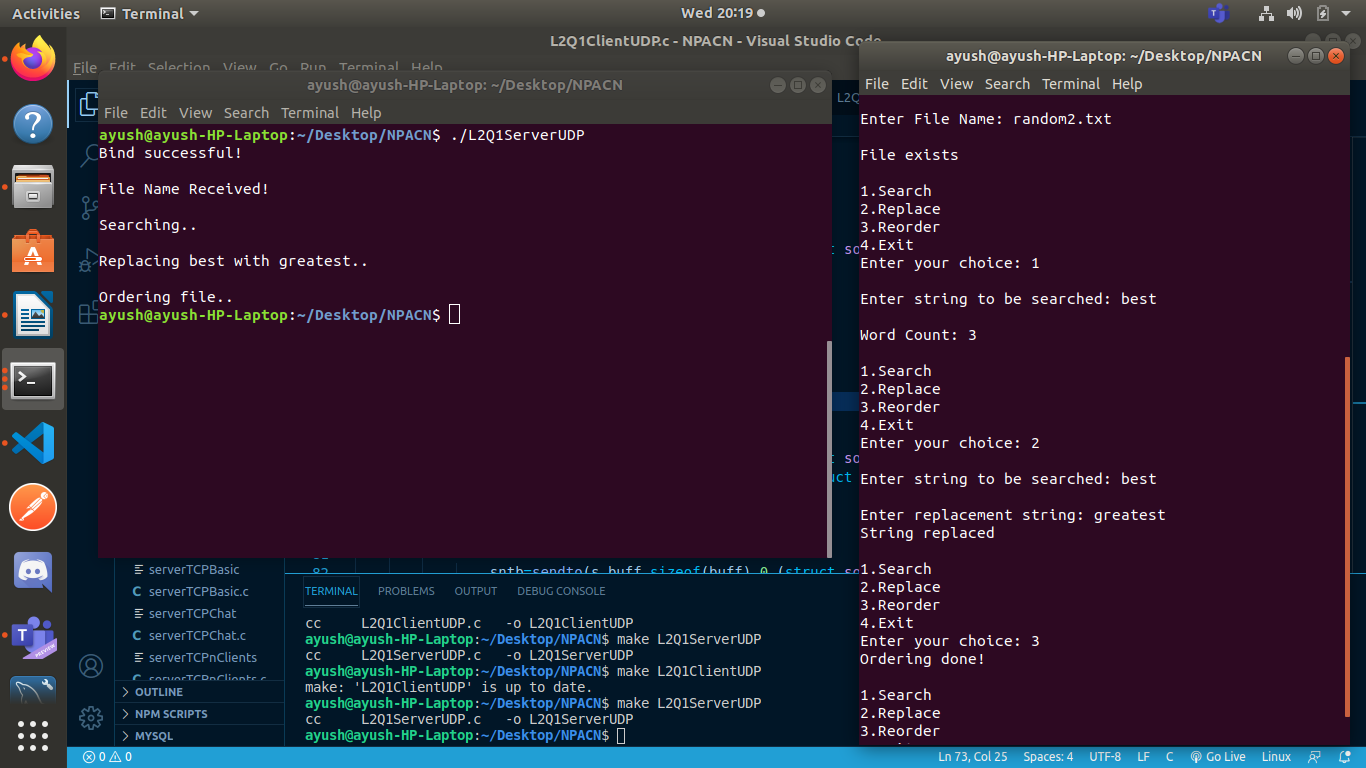
}

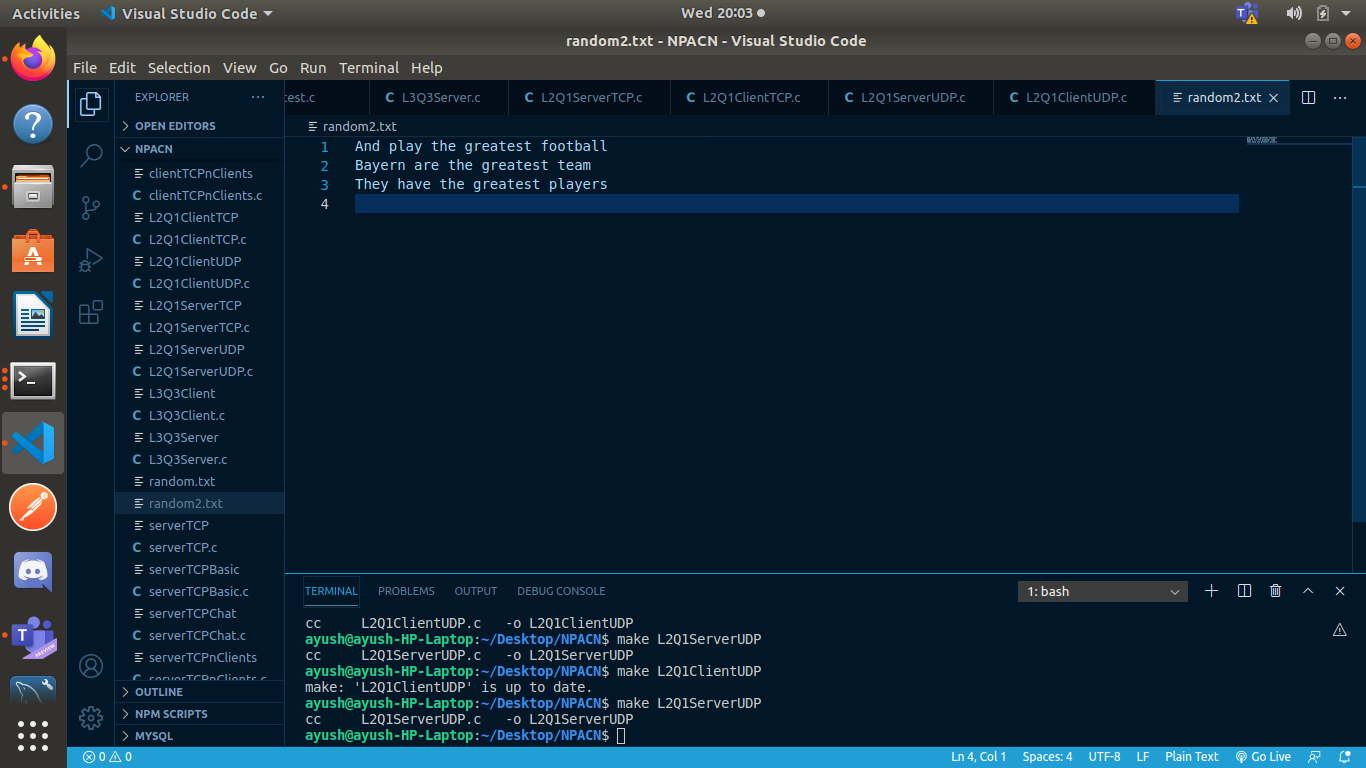
close(s);

}









LAB 3:

Q1)

Server:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

int main() {

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

struct sockaddr\_in server, client;

server.sin\_family = AF\_INET;

server.sin\_port = htons(7500);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

if (bind(serverSocket, (struct sockaddr\*) &server, sizeof(server)) < 0) {

printf("Bind failed");

close(serverSocket); exit(0);

} else {

printf("Bind successful\n");

}

if (listen(serverSocket, 5) < 0) {

printf("Listening error");

close(serverSocket); exit(0);

} else {

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

}

socklen\_t addrLen = sizeof(client);

int connSocket = accept(serverSocket, (struct sockaddr\*) &client, &addrLen);

if (connSocket < 0) {

printf("Can't accept connection");

close(serverSocket); exit(0);

} else {

printf("Connection Accepted! Start chatting\n");

}

int pid = fork();

char buff[50];

while(1) {

if(pid == 0) { //child reads

int bytes = recv(connSocket, buff, sizeof(buff), 0);

buff[bytes] = '\0';

printf("Client: %s", buff);

printf("Received by child process: %d\n", getpid());

if(strcmp(buff, "bye\n") == 0) {

printf("Receiving process stopped\n");

break;

}

strcpy(buff, "");

} else { //parent sends

fgets(buff, 50, stdin);

send(connSocket, buff, sizeof(buff), 0);

printf("Sent by parent process: %d\n", getpid());

if(strcmp(buff, "bye\n") == 0) {

printf("Sending process stopped \n");

break;

}

strcpy(buff, "");

}

}

close(connSocket);

close(serverSocket);

}

Client:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

int main() {

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

struct sockaddr\_in server;

server.sin\_family = AF\_INET;

server.sin\_port = htons(7500);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

if (connect(clientSocket, (struct sockaddr\*) &server, sizeof(server)) < 0) {

printf("Can't connect to server");

exit(0); close(clientSocket);

} else {

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

}

int pid = fork();

char buff[50];

while(1) {

if(pid == 0) { //child sends

fgets(buff, 50, stdin);

send(clientSocket, buff, sizeof(buff), 0);

printf("Sent by child process: %d\n", getpid());

if(strcmp("bye\n", buff) == 0) {

printf("Sending process stopped\n");

break;

}

strcpy(buff, "");

} else { //parent reads

int bytes = recv(clientSocket, buff, sizeof(buff), 0);

buff[bytes] = '\0';

printf("Server: %s", buff);

printf("Received by parent process: %d\n", getpid());

if(strcmp("bye\n", buff) == 0) {

printf("Receiving process stopped \n");

break;

}

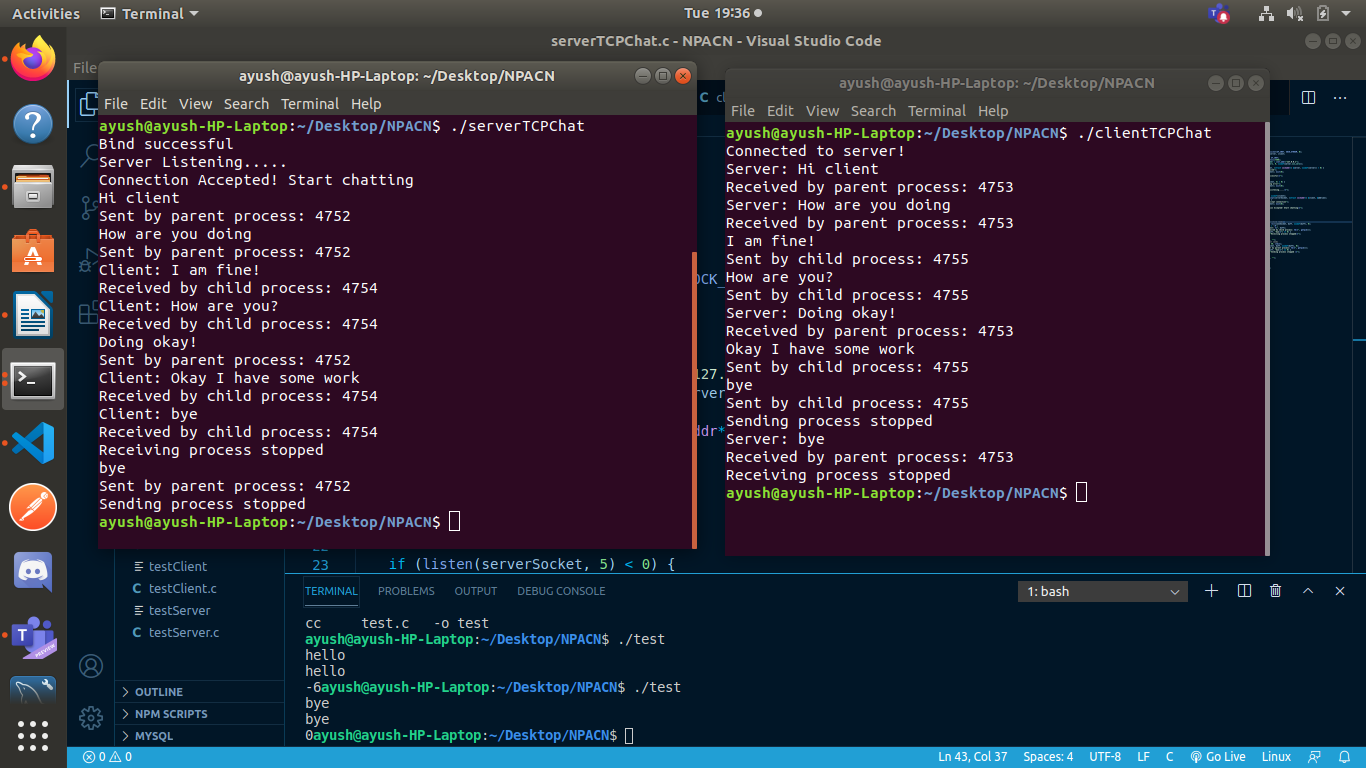
strcpy(buff, "");

}

}

close(clientSocket);

}



Q2)

Server:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

void swap(char \*x, char \*y) {

char temp; temp = \*x; \*x = \*y; \*y = temp;

}

void permute(char \*a, int l, int r) {

int i;

if (l == r)

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

else {

for (i=l; i<=r; i++) {

swap((a+l), (a+i));

permute(a, l+1, r);

swap((a+l), (a+i));

}

}

}

int main() {

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

struct sockaddr\_in server, client;

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

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

server.sin\_family = AF\_INET;

server.sin\_port = htons(7505);

if(bind(serverSocket, (struct sockaddr \*) &server, sizeof(server)) < 0) {

printf("Bind failed\n");

} else {

printf("Bind Successful\n");

}

char buff[50];

while(1) {

socklen\_t len = sizeof(client);

recvfrom(serverSocket, buff, sizeof(buff), 0, (struct sockaddr \*) &client, &len);

if(strcmp("bye", buff) == 0) {

break;

}

permute(buff, 0, strlen(buff)-1);

strcpy(buff,"Permutations printed!");

sendto(serverSocket, buff, sizeof(buff), 0, (struct sockaddr \*) &client, sizeof(client));

strcpy(buff, "");

}

close(serverSocket);

}

Client:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

int main() {

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

struct sockaddr\_in server;

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

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

server.sin\_family = AF\_INET;

server.sin\_port = htons(7505);

char buff[50];

while(1) {

printf("Enter string: ");

scanf("%s", buff);

sendto(clientSocket, buff, sizeof(buff), 0, (struct sockaddr \*) &server, sizeof(server));

if(strcmp("bye", buff) == 0) {

break;

}

socklen\_t len = sizeof(server);

recvfrom(clientSocket, buff, sizeof(buff), 0, (struct sockaddr \*) &server, &len);

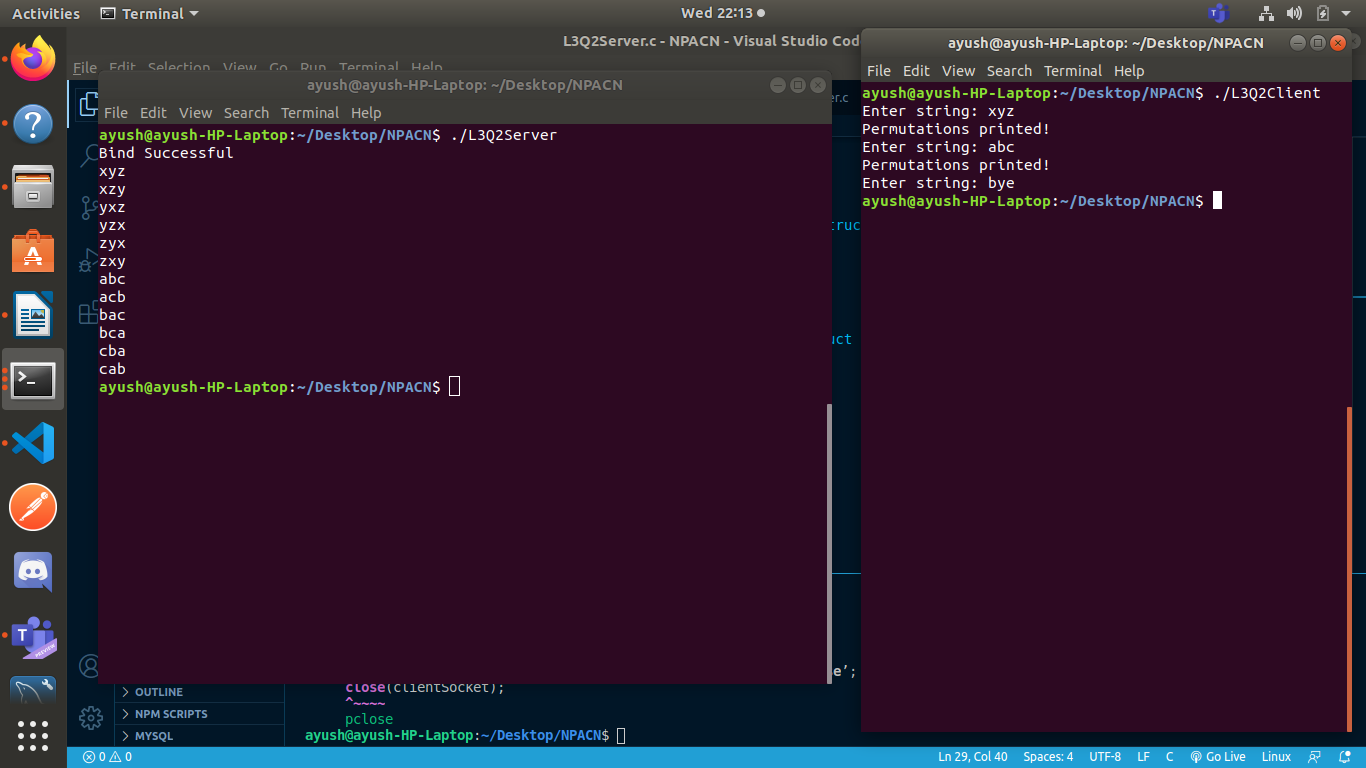
printf("%s\n", buff);

strcpy(buff, "");

}

close(clientSocket);

}



Q3)

Server:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

int compare (const void \* a, const void \* b) {

return ( \*(int\*)a - \*(int\*)b );

}

int compare1 (const void \* a, const void \* b) {

return ( \*(int\*)b - \*(int\*)a );

}

int main() {

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

struct sockaddr\_in server, client;

server.sin\_family = AF\_INET;

server.sin\_port = htons(7501);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

if (bind(serverSocket, (struct sockaddr\*) &server, sizeof(server)) < 0) {

printf("Bind failed");

close(serverSocket); exit(0);

} else {

printf("Bind successful\n");

}

if (listen(serverSocket, 5) < 0) {

printf("Listening error");

close(serverSocket); exit(0);

} else {

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

}

socklen\_t addrLen = sizeof(client);

int connSocket = accept(serverSocket, (struct sockaddr\*) &client, &addrLen);

if (connSocket < 0) {

printf("Can't accept connection");

close(serverSocket); exit(0);

} else {

printf("Connection Accepted!\n");

}

char buff[50];

int bytes = recv(connSocket, buff, sizeof(buff), 0);

buff[bytes] = '\0';

printf("Received String: %s\n", buff);

int pid = fork();

int t = 1;

while(t--) {

if(pid == 0) { //child

int num[50], n=1, i;

for(i=0; i<strlen(buff); i++) {

if(buff[i]>='0' && buff[i]<='9') {

num[n++] = buff[i] - 48;

}

}

qsort(num, n, sizeof(int), compare);

num[0] = n;

printf("Sorted numbers: ");

for(i=1; i<n; i++) {

printf("%d ", num[i]);

} printf("\n");

num[n] = getpid();

send(connSocket, num, sizeof(num), 0);

close(connSocket);

close(serverSocket);

} else { //parent

int chars[50], n=1, i;

for(i=0; i<strlen(buff); i++) {

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

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

}

}

chars[0] = n;

qsort(chars, n, sizeof(int), compare1);

printf("Sorted characters: ");

for(i=0; i<=n; i++) {

printf("%c ", (char)chars[i]);

} printf("\n");

chars[0] = n;

chars[n] = getpid();

send(connSocket, chars, sizeof(chars), 0);

break;

}

}

close(connSocket);

close(serverSocket);

}

Client:

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

int main() {

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

struct sockaddr\_in server;

server.sin\_family = AF\_INET;

server.sin\_port = htons(7501);

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

memset(server.sin\_zero, 0, sizeof(server.sin\_zero));

if (connect(clientSocket, (struct sockaddr\*) &server, sizeof(server)) < 0) {

printf("Can't connect to server");

exit(0); close(clientSocket);

} else {

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

}

char buff[50];

int buff1[50], buff2[50];

printf("Enter alphanumeric string: ");

scanf("%s", buff);

send(clientSocket, buff, sizeof(buff), 0);

int bytes = recv(clientSocket, buff1, sizeof(buff1), 0);

printf("Received from process: %d\n", buff1[buff1[0]]);

bytes = recv(clientSocket, buff2, sizeof(buff2), 0);

printf("Received from process: %d\n", buff2[buff2[0]]);

close(clientSocket);

}

