Sahil Khose

180953218

CCE-B 39

Batch C4

LAB 1)

1)

SERVER

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

int main(){

int sockfd, newsockfd, retval;

struct sockaddr\_in serveraddr, clientaddr;

char buff[50];

int recvdbytes, sentbytes;

int addrlen;

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

if(sockfd == -1){

printf("Error in creating socket\n");

exit(0);

}

printf("Socket Created\n");

serveraddr.sin\_family = AF\_INET;

serveraddr.sin\_port = htons(3212);

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

retval = bind(sockfd, (struct sockaddr \*) &serveraddr, sizeof(serveraddr));

if(retval == -1){

printf("Error in binding\n");

exit(0);

}

printf("Bind Successful\n");

retval = listen(sockfd, 5);

if(retval == -1){

printf("Error in Listening\n");

exit(0);

}

printf("Listen Successful\n");

addrlen = sizeof(clientaddr);

newsockfd = accept(sockfd, (struct sockaddr \*) &clientaddr, &addrlen);

if(newsockfd == -1){

printf("Error in Accepting\n");

close(sockfd);

exit(0);

}

printf("Accept Successful\n");

char a[50];

recvdbytes = recv(newsockfd, a, sizeof(a), 0);

if(recvdbytes == -1){

printf("Error in Receiving\n");

close(newsockfd);

close(sockfd);

exit(0);

}

printf("Recieved Successful\n");

int z = 1, n, search;

while(z!=5){

recvdbytes = recv(newsockfd, buff, sizeof(buff), 0);

if(recvdbytes == -1){

printf("Receive Error\n");

close(sockfd);

close(newsockfd);

exit(0);

}

n = buff[0];

z = buff[1];

switch(z){

case 1: search = buff[2];

int l;

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

if(a[l] == search)

break;

}

if(l==n)

printf("Element not found\n");

else

printf("Element exists at %d\n",l+1 );

break;

case 2: for(int i=0;i<n-1;i++)

{

for(int j=0;j<n-i-1;j++)

{

if(a[j]>a[j+1])

{

int temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

printf("\nSorted array is: \n");

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

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

printf("\n\n");

break;

case 3: for(int i=0;i<n-1;i++)

{

for(int j=0;j<n-i-1;j++)

{

if(a[j]<a[j+1])

{

int temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

printf("\nSorted array is: \n");

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

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

printf("\n\n");

break;

case 4:

printf("\nEven aay is: \n");

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

{

if(a[i]%2==0)

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

}

printf("\n\nOdd aay is: \n");

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

{

if(a[i]%2!=0)

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

}

printf("\n\n");

break;

case 5: break;

default: break;

}

}

close(newsockfd);

close(sockfd);

exit(0);

}

CLIENT

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

int main(){

int sockfd, retval;

struct sockaddr\_in serveraddr;

char buff[50];

int recvbytes, sendbytes;

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

if(sockfd == -1){

printf("Socket Creation Error\n");

exit(0);

}

printf("Socket Created\n");

serveraddr.sin\_family = AF\_INET;

serveraddr.sin\_port = htons(3212);

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

retval = connect(sockfd, (struct sockaddr \*) &serveraddr, sizeof(serveraddr));

if(retval == -1){

printf("connection failed\n");

exit(0);

}

printf("connected successfully\n");

int n;

int a[50];

char ca[50];

printf("Number of Integers \n");

scanf("%d", &n);

printf("Enter Integers\n");

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

scanf("%d", &a[i]);

}

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

ca[i] = a[i];

}

sendbytes = send(sockfd, ca, sizeof(ca), 0);

if(sendbytes == -1){

close(sockfd);

printf("Send Error\n");

exit(0);

}

int z = 1;

while(z!=5){

printf("Choose an option\n1) Search a number\n2) Ascending sort\n3) Descending sort\n4) Split into even and odd\n5) Exit \n");

scanf("%d", &z);

switch(z){

case 1: printf("Enter number to search");

int search;

scanf("%d", &search);

buff[0] = n;

buff[1] = 1;

buff[2] = search;

sendbytes = send(sockfd, buff, sizeof(buff), 0);

if(sendbytes == -1){

close(sockfd);

printf("Send Error\n");

exit(0);

}

break;

case 2: buff[0] = n;

buff[1] = 2;

sendbytes = send(sockfd, buff, sizeof(buff), 0);

if(sendbytes == -1){

close(sockfd);

printf("Send Error\n");

exit(0);

}

break;

case 3: buff[0] = n;

buff[1] = 3;

sendbytes = send(sockfd, buff, sizeof(buff), 0);

if(sendbytes == -1){

close(sockfd);

printf("Send Error\n");

exit(0);

}

break;

case 4: buff[0] = n;

buff[1] = 4;

sendbytes = send(sockfd, buff, sizeof(buff), 0);

if(sendbytes == -1){

close(sockfd);

printf("Send Error\n");

exit(0);

}

break;

case 5: buff[0] = n;

buff[1] = 5;

sendbytes = send(sockfd, buff, sizeof(buff), 0);

if(sendbytes == -1){

close(sockfd);

printf("Send Error\n");

exit(0);

}

break;

default: printf("Enter valid option\n");

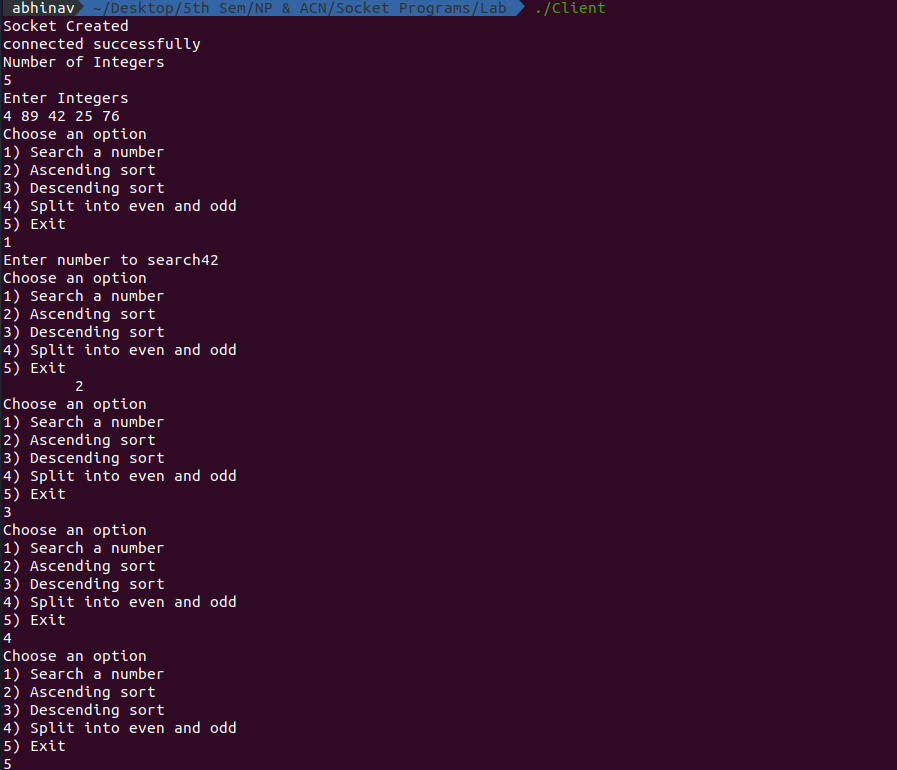
}

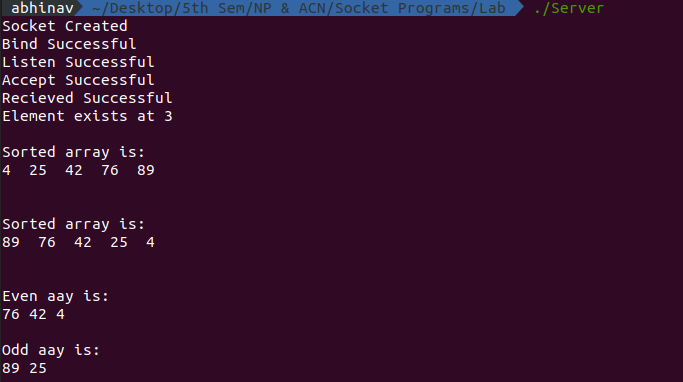
}

close(sockfd);

exit(0);

}





2)

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>

int main()

{

int s,r,recvbytes,sentbytes,x;

socklen\_t ca;

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50];

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

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(3212);

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);

}

printf("\nSocket binded.");

while(1){

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

if(recvbytes==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("\nMessage Recieved: ");

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

if(!strcmp(buff,"Halt"))

break;

char buff2[50];

strcpy(buff2,buff);

buff[1]=strlen(buff2);

int n=0;

for(int i=0;i<buff[1];i++)

if(buff2[i]=='a'||buff2[i]=='e'||buff2[i]=='o'||buff2[i]=='i'||buff2[i]=='u')

n++;

buff[2]=n;

buff[0]=1;

for(int i=0;i<buff[1]/2;i++)

{

if(buff2[i]!=buff2[buff[1]-i-1])

{

buff[0]=0;

break;

}

}

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

if(sentbytes==-1)

{

printf("\nMessage Sending Failed");

close(s);

exit(0);

}

if(!strcmp(buff,"halt"))

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,recvbytes,sentbytes;

socklen\_t sa;

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50];

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

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(3212);

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

sa=sizeof(server);

len=sizeof(server);

while(1){

printf("\n\n");

printf("Enter new string: ");

scanf("%s", buff);

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

if(!strcmp(buff,"Halt"))

break;

if(sentbytes==-1)

{

close(s);

printf("\nSend Error");

exit(0);

}

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

if(recvbytes==-1)

{

printf("\n Receive Error");

close(s);

exit(0);

}

if(buff[0]==1)

printf("\nPalindrome String \nString Length: %d\nNo of vowels: %d ",buff[1],buff[2]);

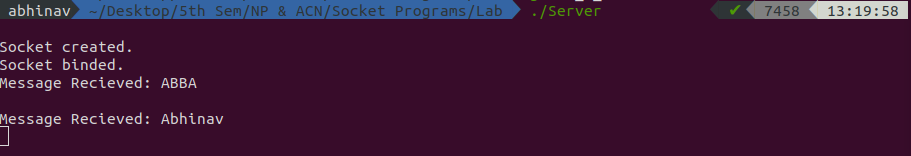
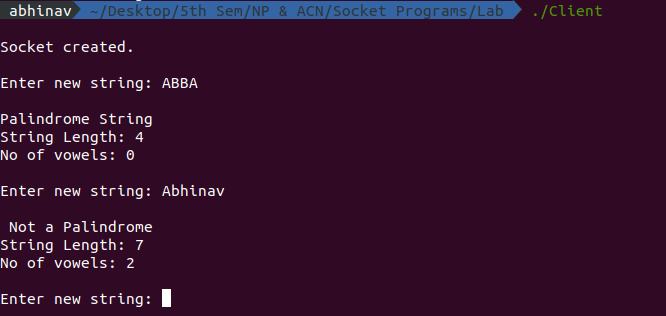
else

printf("\n Not a Palindrome \nString Length: %d\nNo of vowels: %d ",buff[1],buff[2]);

}

close(s);

}



LAB 2)

1. 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, temp[1000];

int index = 0;

int owlen;

owlen = strlen(oldWord);

/\*

\* Repeat till all occurrences are replaced.

\*/

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

{

// Bakup current line

strcpy(temp, str);

// Index of current found word

index = pos - str;

// Terminate str after word found index

str[index] = '\0';

// Concatenate str with new word

strcat(str, newWord);

// Concatenate str with remaining words after

// oldword found index.

strcat(str, temp + index + owlen);

}

}

int main()

{

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

printf("INPUT port number: ");

scanf("%d", &x);

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);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

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

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

if(r==-1)

{

printf("\nBinding error.");

exit(0);

}

printf("\nSocket binded.");

r=listen(s,1);

if(r==-1)

{

close(s);

exit(0);

}

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

len=sizeof(client);

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

if(ns==-1)

{

close(s);

exit(0);

}

printf("\nSocket accepting.\n");

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

close(ns);

exit(0);

}

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

/\*printf("%s", buff);

printf("\n\n");

scanf("%s", buff);\*/

char fil[50];

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

// file exists

strcpy(fil,buff);

strcpy(buff,"File exists");

} else {

// file doesn't exist

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

}

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

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(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);

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

close(ns);

exit(0);

}

ch = buff[0];

int i,n,n1,n2,j;

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

char strTempData[MAX\_LEN];

char \*\*strData = NULL; // String List

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);

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);

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(0);

}

break;

case 2:

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);

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

close(ns);

exit(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)

{

/\* Unable to open file hence exit \*/

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

printf("Please check whether file exists and you have read/write privilege.\n");

exit(0);

}

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

{

// Replace all occurrence of word from current line

replaceAll(buffer, str1, str2);

// After replacing write it to temp file.

fputs(buffer, fTemp);

}

/\* Close all files to release resource \*/

fclose(fPtr);

fclose(fTemp);

/\* Delete original source file \*/

remove(fil);

/\* Rename temp file as original file \*/

rename("replace.tmp", fil);

strcpy(buff,"Replace finished!");

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

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(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;

}

// Read and store in a string list.

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

// Remove the trailing newline character

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++;

}

// Sort the array.

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);

}

}

}

// Write it to outfile. file.

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

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

// free each string

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

free(strData[i]);

// free string list.

free(strData);

remove(fil);

rename("temp.txt",fil);

fclose(ptrFileLog);

fclose(ptrSummary);

strcpy(buff,"Ordering done!");

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

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(0);

}

break;

case 4: ch=4;

break;

}

}

close(ns);

close(s);

}

TCP 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;

printf("INPUT port number: ");

scanf("%d", &x);

struct sockaddr\_in server;

char buff[50];

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

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

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("\nSocket connected.");

printf("\n\n");

printf("Type File Name: ");

scanf("%s", buff);

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

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("\n");

printf("%s", buff);

printf("\n\n");

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);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

n=buff[0];

printf("\nWord found %d number of times!\n",n);

break;

case 2:printf("\nEnter string to be searched and replaced: ");

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);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

printf("\nEnter new 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);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

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

break;

case 3:sntb=send(s,buff,sizeof(buff),0);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

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

break;

case 4:sntb=send(s,buff,sizeof(buff),0);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

break;

default:

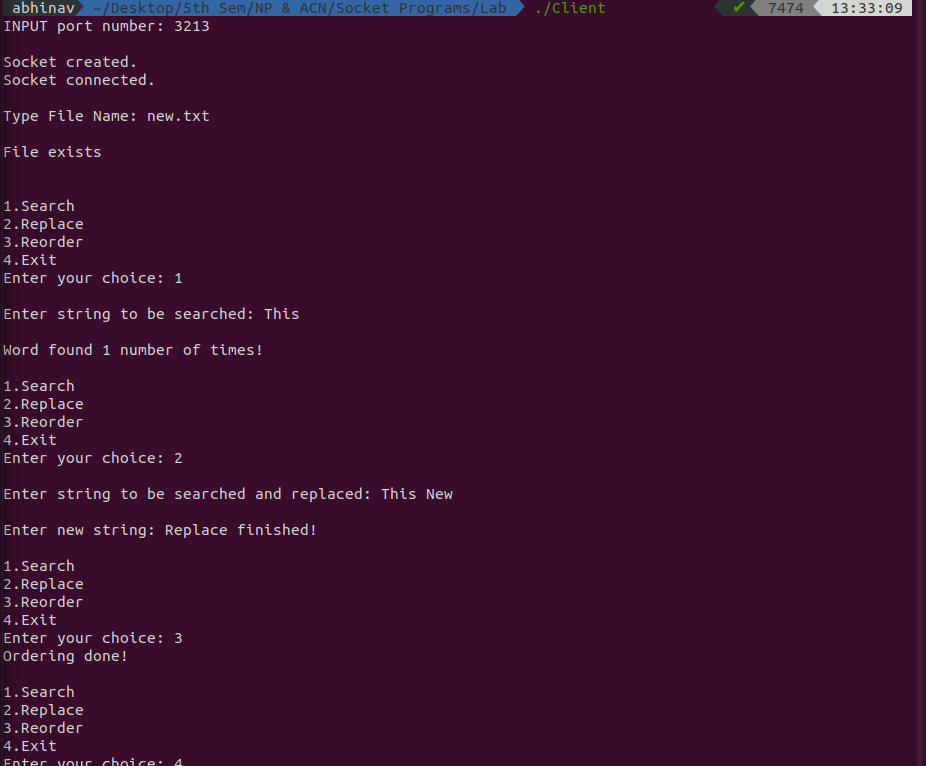
printf("\n Try Again!\n");

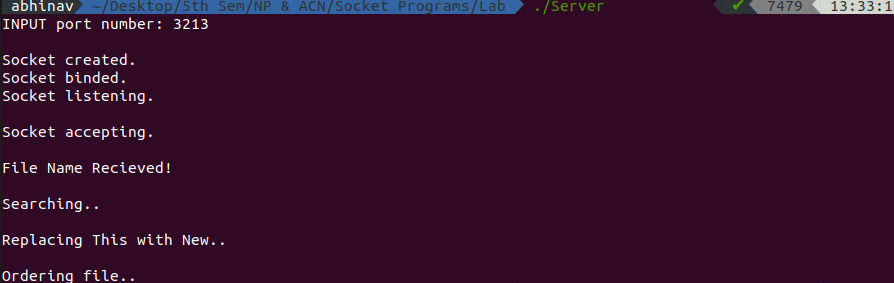
}

}

close(s);

}





2) 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);

/\*

\* Repeat till all occurrences are replaced.

\*/

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

{

// Bakup current line

strcpy(temp, str);

// Index of current found word

index = pos - str;

// Terminate str after word found index

str[index] = '\0';

// Concatenate str with new word

strcat(str, newWord);

// Concatenate str with remaining words after

// oldword found index.

strcat(str, temp + index + owlen);

}

}

int main()

{

int s,r,recb,sntb,x;

int ca;

printf("INPUT port number: ");

scanf("%d", &x);

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50];

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

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

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);

}

printf("\nSocket binded.");

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

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

char fil[50];

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

// file exists

strcpy(fil,buff);

strcpy(buff,"File exists");

} else {

// file doesn't exist

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

}

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

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

exit(0);

}

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

{

close(s);

//close(ns);

exit(0);

}

int ch=0;

while(ch!=4){

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

ch = buff[0];

int i,n,n1,n2,j;

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

char strTempData[MAX\_LEN];

char \*\*strData = NULL; // String List

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);

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

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

exit(0);

}

break;

case 2:

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);

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(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)

{

/\* Unable to open file hence exit \*/

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

printf("Please check whether file exists and you have read/write privilege.\n");

exit(0);

}

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

{

// Replace all occurrence of word from current line

replaceAll(buffer, str1, str2);

// After replacing write it to temp file.

fputs(buffer, fTemp);

}

/\* Close all files to release resource \*/

fclose(fPtr);

fclose(fTemp);

/\* Delete original source file \*/

remove(fil);

/\* Rename temp file as original file \*/

rename("replace.tmp", fil);

strcpy(buff,"Replace finished!");

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

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

exit(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;

}

// Read and store in a string list.

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

// Remove the trailing newline character

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++;

}

// Sort the array.

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);

}

}

}

// Write it to outfile. file.

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

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

// free each string

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

free(strData[i]);

// free string list.

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);

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

exit(0);

}

break;

case 4: ch=4;

break;

}

}

close(s);

}

UDP 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;

printf("INPUT port number: ");

scanf("%d", &x);

struct sockaddr\_in server,client;

char buff[50];

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

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

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

sa=sizeof(server);

len=sizeof(server);

printf("\n\n");

printf("Type File Name: ");

scanf("%s", buff);

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

if(sntb==-1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("\n");

printf("%s", buff);

printf("\n\n");

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);

if(sntb==-1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

n=buff[0];

printf("\nWord found %d number of times!\n",n);

break;

case 2:printf("\nEnter string to be searched and replaced: ");

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);

if(sntb==-1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

printf("\nEnter new 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);

if(sntb==-1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

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

break;

case 3:sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr \*)&server, len);

if(sntb==-1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

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

break;

case 4:sntb=sendto(s,buff,sizeof(buff),0,(struct sockaddr \*)&server, len);

if(sntb==-1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

break;

default:

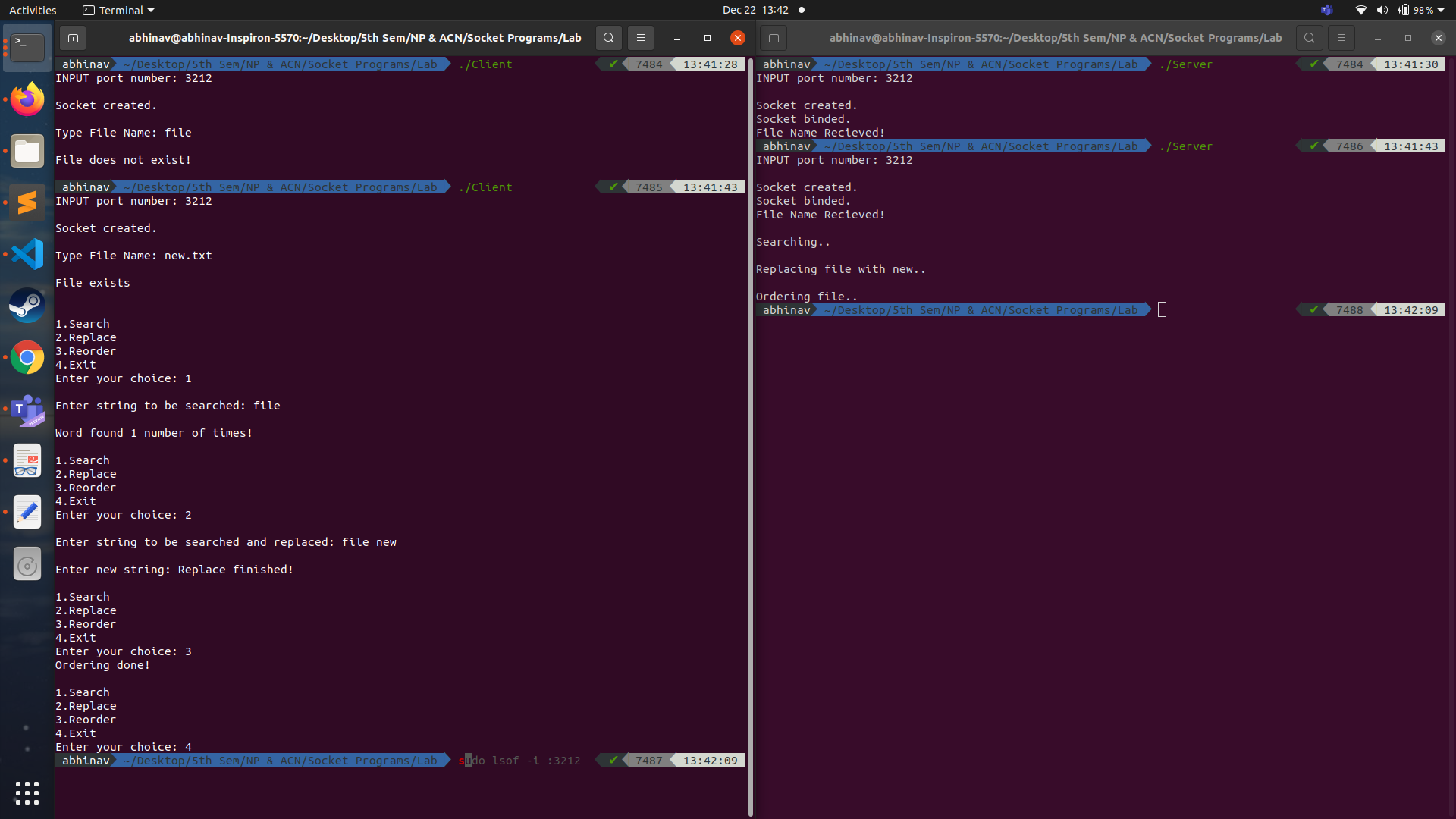
printf("\n Try Again!\n");

}

}

close(s);

}



LAB 3)

1)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>

int main()

{

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

printf("INPUT port number: ");

scanf("%d", &x);

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50],buff2[50];

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

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

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

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

if(r==-1)

{

printf("\nBinding error.");

exit(0);

}

printf("\nSocket binded.");

r=listen(s,1);

if(r==-1)

{

close(s);

exit(0);

}

printf("\nSocket listening.");

len=sizeof(client);

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

if(ns==-1)

{

close(s);

exit(0);

}

printf("\nSocket accepting.\n");

int pid=fork();

while(1){

if(pid>0){

//parent

printf("\n\nParent - Type message: ");

scanf("%s", buff2);

sntb=send(ns,buff2,sizeof(buff2),0);

if(sntb==-1)

{

printf("\nMessage Sending Failed");

close(s);

close(ns);

exit(0);

}

if(strcmp(buff2,"BYE")==0)

break;

}

else{

//child

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

close(ns);

exit(0);

}

if(strcmp(buff,"BYE")==0)

break;

printf("\nChild - Message Recieved: ");

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

}

}

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;

printf("INPUT port number: ");

scanf("%d", &x);

struct sockaddr\_in server;

char buff[50],buff2[50];

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

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

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("\nSocket connected.");

printf("\n\n");

int pid;

pid=fork();

while(1){

if(pid>0)

{

//parent

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

if(strcmp(buff,"BYE")==0)

break;

printf("\nParent - Message Recieved: ");

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

}

else

{

//child

printf("\nChild - Type Message: ");

scanf("%s", buff2);

sntb=send(s,buff2,sizeof(buff2),0);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

if(strcmp(buff2,"BYE")==0)

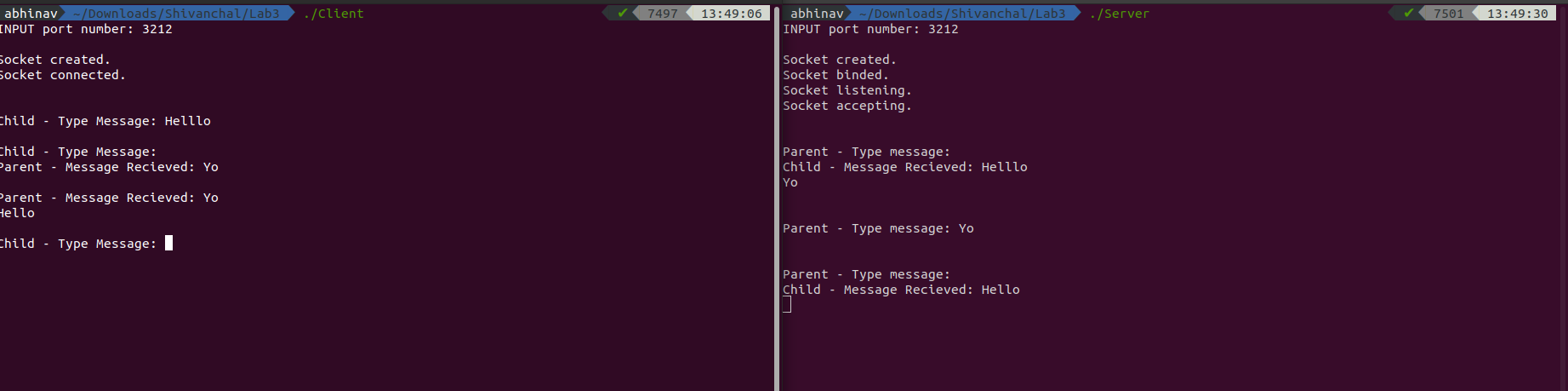
break;

}

}

close(s);

}



2) 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>

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)); //backtrack

}

}

}

int main()

{

int s,r,recb,sntb,x;

int ca;

printf("INPUT port number: ");

scanf("%d", &x);

socklen\_t len;

struct sockaddr\_in server,client;

char buff[50];

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

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.\n");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

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);

}

printf("\nSocket binded.\n");

while(1){

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

if(recb==-1)

{

printf("\nMessage Recieving Failed");

close(s);

exit(0);

}

printf("\nMessage Recieved: ");

printf("%s", buff);

if(!strcmp(buff,"stop"))

break;

printf("\nPermutations of the string are: \n");

int n=strlen(buff);

permute(buff, 0, n-1);

}

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;

printf("INPUT port number: ");

scanf("%d", &x);

struct sockaddr\_in server,client;

char buff[50];

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

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

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

sa=sizeof(server);

len=sizeof(server);

while(1){

printf("\n\n");

printf("Type Message (enter 'stop' to exit): ");

scanf("%s", buff);

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

if(sntb==-1)

{

close(s);

printf("\nMessage sending Failed");

exit(0);

}

if(!strcmp(buff,"stop"))

break;

}

close(s);}

