#### COEN 146: Computer Networks – Fall 2018

**Lab assignment 4: Programming TCP/IP socket**

**Objectives**

##### To build client/ server applications using TCP/IP socket

##### **Part 1: Client – Server with TCP/IP [26 points]**

The following program gives the source code of a TCP client

#include<stdio.h>

#include<stdlib.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<netdb.h>

#include<string.h>

#include<unistd.h>

#define SORT\_PORT 1719

#define LINESIZE 80

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

struct sockaddr\_in server;

struct hostent \*host\_info;

int sock, count;

char inlin[LINESIZE]; /\*buffer to copy from user to server\*/

char outline[LINESIZE];/\*buffer to copy from server to user\*/

char \*server\_name;

/\*get server name from command line. if none, use'local host'\*/

server\_name =(argc>1) ? argv[1] :"localhost";

/\*create the socket\*/

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

if(sock<0) {

perror("creating stream socket");

exit(1);

}

host\_info=gethostbyname(server\_name);

if(host\_info==NULL){

fprintf(stderr,"%s:unknown host: %s\n",argv[0],server\_name);

exit(2);

}

/\*set up the server's socket address, then connect\*/

server.sin\_family=host\_info->h\_addrtype;

memcpy((char \*)&server.sin\_addr,host\_info->h\_addr,host\_info->h\_length);

server.sin\_port=htons(SORT\_PORT);

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

perror("connecting to server");

exit(3);

}

int wordIndex = 0;

printf("\nClient: connect to server %s\n",server\_name);

printf("Client: Write word %d to sort at server:\n", ++wordIndex);

while((count=read(0, outline,LINESIZE))>0){

//printf("Client: count = %d\n",count);

outline[count]='\0';

if(write(sock, outline,count)<0)

printf("writing to server failed\n");

if(outline[0]=='0')

break;

printf("Client: sending word to server: %s",outline);

printf("\nClient: Write word %d to sort at server:\n", ++wordIndex);

}

printf("Client: results are...\n");

while ((count=read(sock,inlin,LINESIZE)) >0) {

//printf("Client:");

write(1,inlin,count);

printf("\n");

}

return (0);

}

1. Write a TCP server that would respond to continuous client requests. You may use the following code snippet if needed. Run, compile and test the client/ server communication and write down your observation. Your file code for both client and server need to be commented in details, explaining the purpose of each line(s) of code.

Demonstrate client/server communication to the TA and upload your source code to Camino.

struct sockaddr\_in server,client;

if((s=socket(AF\_INET,SOCK\_STREAM,0))<0) {

perror("creating stream socket error!\n");

exit(1);

}

server.sin\_family=AF\_INET;

server.sin\_port=htons(SORT\_PORT);

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

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

perror("error in binding!");

exit(1);

}

if(listen(s,3)<0){

perror("error in listening!");

exit(1);

}

while(1){

client\_len=sizeof(client);

if((sd=accept(s,(struct sockaddr\*)&client,&client\_len))<0){

perror("error in binding!");

exit(1);

}

write(sd,"Server: connection is OK..",18);

     ….

1. Assign your TCP client to read from src.dat file a series of words, send to TCP server, and then the TCP server sorts words alphabetically and copies to them to the dest.dat file.

You may use the follow sort function:

void sort(char \*str[],int n){

int i,j;

char temp[20];

for(i=n;i>=0;i--)

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

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

strcpy(temp,str[j]);

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

strcpy(str[j+1],temp);

}

}

}

Demonstrate client/server communication and sorting to the TA and upload your source code to Camino.

1. Now modify your TCP server program to accept multiple client connections. This can be easily achieved by assigning connections to multiple threads.

Demonstrate multiple client/server communication and sorting to the TA and upload your source code to Camino. Coordinate with your classmate (s) for the demonstration on multiple machines.

**Part 2: Quiz [4 points]**

Complete the quiz on Part 2 of Google Lab3\_Turn-in Form through Camino.

**Requirements to complete the lab**

1. Show the TA correct execution of the programs you wrote for Part 1 and upload source code to Camino.
2. Complete the quiz on Google Lab4\_Turn-in Form through Camino.

Be sure to retain copies (machine and/or printed) of your source code. You will want these for study purposes and to resolve any grading questions (should they arise)

Please start each program with a descriptive block that includes minimally the following information:

/\*

\* Name: <your name>

\* Date:

\* Title: Lab4 - Part ….

\* Description: This program … <you should

\* complete an appropriate description here.>

\*/