**Operating Systems**

**University at Albany**

**Department of Computer Science**

**CSI 500**

**Assignment-1**

**Assigned: Wednesday, April 1st, 2020**

**Due: Wednesday, April 8th, 2020 by 11:59 PM**

**Student Name:**

## OBJECTIVES

## To develop a client/server application using TCP sockets and the C programming language. Your solution will respond to service requests by clients. Such requests may be by either providing the IP address of the server or the name of the host where the server is executing.

## PROBLEM

You are to use both the client and the server programs included in this document. You are also to modify the server program to respond to client requests that include either the name of the host where the server is executing or its IP address. Note that the successful completion of this exercise requires the use of two or more computers to test your solution. It is assumed that all computers used for testing your solution are connected to the Internet. Note that different clients may request either the same service or a totally different one. The services supported by your server include:

(1) changing all characters of a text file to uppercase, and

(2) counting the number of a supplied character in a text file.

Your client program will forward service requests to the server stating

(1) the kind of service is needed, and

(2) all required data necessary for the successful completion of the request.

Your client program must use the following syntax for any request to the server where *toUpper* is the request to change characters to uppercase and *count* to return how many times the supplied character is present in the file. The following syntax illustrates this requirement:

*toUpper < file.txt >*

*count < char, file.txt >*

In the above two examples *file.txt* is to indicate that a text file is to be supplied as an input parameter. The *char* string is to inform that a character is to be supplied. The information returned to the client, by your server, must be stored in text files with the names *fileUpper.txt* and *fileChar.txt*.

**HOW TO TEST YOUR SOLUTION**

You are to use the *intext.txt* file to test your solution. You are to use the following steps to test your solution:

1. Open a command line window for the server and type the following command where *port-number* is the port number you have selected for your solution.

Your-Prompt> *./server port-number*

1. Open a second command line window for your client and type the following command where *server-IP-address* is the IP address of your computer in the network and *server-host-name* is the name given to the computer where the server is running.

Your-Prompt> *./*client *server-IP-address port-number*

or

Your-Prompt> *./*client *server-host-name port-number*

Your server then should start parsing and executing client requests according to the syntax defined in the **PROBLEM** section.

**INPUT TEST FILE (*intext.txt*)**

You are to name your input file as *intext.txt* and populate it with the following contents:

*source code*

*represents the part of*

*process that*

*contains the programming*

*language itself. you may*

*use a text editor to*

*write your source code file.*

**SAMPLE CODE**

/\* Program: server.c

\* A simple server TCP sockets.

\* Server is executed before Client.

\* Port number is to be passed as an argument.

\*

\* To test: Open a terminal window.

\* At the prompt ($ is my prompt symbol) you may

\* type the following as a test:

\*

\* $ ./server 54554

\* Run client by providing host and port

\*

\*

\*/

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

void error(const char \*msg)

{

perror(msg);

exit(1);

}

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

{

int sockfd, newsockfd, portno;

socklen\_t clilen;

char buffer[256];

struct sockaddr\_in serv\_addr, cli\_addr;

int n;

if (argc < 2) {

fprintf(stderr,"ERROR, no port provided\n");

exit(1);

}

fprintf(stdout, "Run client by providing host and port\n");

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

if (sockfd < 0)

error("ERROR opening socket");

bzero((char \*) &serv\_addr, sizeof(serv\_addr));

portno = atoi(argv[1]);

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

serv\_addr.sin\_port = htons(portno);

if (bind(sockfd, (struct sockaddr \*) &serv\_addr,

sizeof(serv\_addr)) < 0)

error("ERROR on binding");

listen(sockfd,5);

clilen = sizeof(cli\_addr);

newsockfd = accept(sockfd,

(struct sockaddr \*) &cli\_addr,

&clilen);

if (newsockfd < 0)

error("ERROR on accept");

bzero(buffer,256);

n = read(newsockfd,buffer,255);

if (n < 0)

error("ERROR reading from socket");

printf("Here is the message: %s\n",buffer);

n = write(newsockfd,"I got your message",18);

if (n < 0)

error("ERROR writing to socket");

close(newsockfd);

close(sockfd);

return 0;

}

/\*

simple client to work with server.c program.

\* Host name and port used by server is to be

\* passed as arguments.

\*

\* To test: Open a terminal window.

\* At prompt ($ is my prompt symbol) you may

\* type the following as a test:

\*

\* $./client 127.0.0.1 54554

\* Please enter the message: Operating Systems is fun!

\* I got your message

\* $

\*

\*/

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

void error(const char \*msg)

{

perror(msg);

exit(0);

}

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

{

int sockfd, portno, n;

struct sockaddr\_in serv\_addr;

struct hostent \*server;

char buffer[256];

if (argc < 3) {

fprintf(stderr,"usage %s hostname port\n", argv[0]);

exit(0);

}

portno = atoi(argv[2]);

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

if (sockfd < 0)

error("ERROR opening socket");

server = gethostbyname(argv[1]);

if (server == NULL) {

fprintf(stderr,"ERROR, no such host\n");

exit(0);

}

bzero((char \*) &serv\_addr, sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

bcopy((char \*)server->h\_addr,

(char \*)&serv\_addr.sin\_addr.s\_addr,

server->h\_length);

serv\_addr.sin\_port = htons(portno);

if (connect(sockfd,(struct sockaddr \*) &serv\_addr,sizeof(serv\_addr)) < 0)

error("ERROR connecting");

printf("Please enter the message: ");

bzero(buffer,256);

fgets(buffer,255,stdin);

n = write(sockfd,buffer,strlen(buffer));

if (n < 0)

error("ERROR writing to socket");

bzero(buffer,256);

n = read(sockfd,buffer,255);

if (n < 0)

error("ERROR reading from socket");

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

close(sockfd);

return 0;

}

## WHAT TO SUBMIT

The following are to be submitted through Blackboard:

1. The source code for both your client.c and server.c solutions as well as
2. The two output files, *fileUpper.txt* and *fileChar.txt,* according the specifications included in this exercise.