CS 353: Programming Assignment 1

Revision History:
1/22 Release
1/23 Fixed Typo 67 → 167
1/26 –p path → -d path in part III

Introduction:

The main goal of this assignment is to give you hands-on experience in developing classical client-server networked applications. In this assignment you will learn how to program with both TCP and UDP sockets, understand network- and host-byte ordering, and provide hands-on experience on how to support higher level application requirements such as transmitting and receiving image files and supporting multiple client connections.

This assignment must be developed in C or C++ and is organized in three parts.

THE PARTS

I. Develop a server and client application using UDP sockets to exchange information across the network. The server listens on a port specified at the command line. The client connects to the server and sends the usc id and a name. The server the generates a random 100-250 character string and sends the string to the client.

Command Line Options:

> server –u –p portno –l logfile
where

-u	indicates using udp sockets
-p portno	the port number the server is listening on
-l loggile	name of the logfile

> client –u –s serverIP –p portno –l logfile
where

-u	indicates using udp sockets
-s <serverIP>	indicates the serverIP address
-p <portno>	indicates the server port number
-l <logfile>	name of the logfile

If nothing is specified on the command line the server and client programs should print the usage instructions and quit.

Message format:

The client messages to the server MUST be formatted as follows:

1. USCID <10digit number>
2. Name <STRING>
3. StringLength <IntergerNumber>

The server messages to the client MUST be formatted as follows:
1. RSTRING <STRING>

Required output:

The following lines of output MUST be present in the logfiles. You can also print additional debugging information in the logfiles, but prepend debugging information with the keyword "DEBUG" so we can ignore it during grading. The text in the angled brackets below should be replaced with the results from the communication of the client and server.

Server Logfile

server started on <1.2.3.4> at port <12345>…
received client connection from hostname <cs.usc.edu> port <port>
received USCID <0123456789>
received Name <John Doe>
sending random string length <167>…
received string length <167>
terminating server…

Client Logfile

connecting to the server <1.2.3.4> at port <12345>
connected to server hostname <cs.usc.edu>
sending USCID <0123456789>
sending Name <John Doe>
received string <STRING>
sending string length <167>
terminating client…

II. Extend the above server and client programs to support TCP sockets to exchange information across the network. The server listens on a port specified at the command line. The client connects to the server and sends the usc id and name. The server then sends an image file over to the client.

Command Line Options:

> server –t –p portno –l logfile –I imagefile
where

-t	indicates using TCP sockets
-p portno	the port number the server is listening on
-l loggile	name of the logfile
-i imagefile	name of the image file

> client –t –s serverIP –p portno –l logfile –I imagefile
where

-t	indicates using TCP sockets
-s <serverIP>	indicates the serverIP address
-p <portno>	indicates the server port number
-l <logfile>	name of the logfile
-i imagefile	name of the imagefile

If nothing is specified on the command line the server and client program should print the usage instructions and quit.

Message format:

The client messages to the server MUST be formatted as follows:

1. USCID <10digit number>
2. Name <STRING>

Required output:

The following lines of output MUST be present in the logfiles. You can also print additional output in the logfiles, but prepend debugging information with the keyword "DEBUG" so we can ignore it during grading. The text in the angled brackets below should be replaced with the results from the communication of the client and server.

Server Logfile

server started on <1.2.3.4> at port <12345>…
received client connection from hostname <cs.usc.edu> port <port>
received USCID <0123456789>
received Name <John Doe>
sending image file <imagefile>...
terminating server…

Client Logfile

connecting to the server <1.2.3.4> at port <12345>
connected to server hostname <cs.usc.edu>
sending USCID <0123456789>
sending Name <John Doe>
received image and saved in <imagefile>
terminating client…

III. Extend the above TCP server and client programs to support multiple clients at the same time. You will need to fork a process for each client connection that is received so that you can handle them concurrently. The server should also assign each connection a unique ID, UID, and then prepend it to all the

relevant entries in the logfile so that the client and server logfile can be correctly coupled. Just as before, the client sends a USCID and name. The server then selects an image file from a collection and sends the image file over to the client. In this case, the server in this case does not terminate after a client interaction unless a CTRL-C is pressed at the command line.

Command Line Options:

> server –t –p portno –l logfile –d <path>
where

-t	indicates using TCP sockets
-p portno	the port number the server is listening on
-l loggile	name of the logfile
-d <path>	path to a collection of image files

> client –t –s serverIP –p portno –l logfile –I imagefile
where

-t	indicates using TCP sockets
-s <serverIP>	indicates the serverIP address
-p <portno>	indicates the server port number
-l <logfile>	name of the logfile
-i imagefile	name of the imagefile

If nothing is specified on the command line the server and client program should print the usage instructions and quit.

Message format:

The client messages to the server MUST be formatted as follows:
1. USCID <10digit number>
2. Name <STRING>

The server message to the client MUST be formatted as follows:
1. UID <Alphanumeric>

Required output:

The following lines of output MUST be present in the logfiles. You can also print additional output in the logfiles, but prepend debugging information with the keyword “DEBUG” so we can ignore it during grading. The text in the angled brackets below should be replaced with the results from the communication of the client and server.

Server Logfile

NOTE: the output below will be interleaved with several client connections. Below we just show a connection from one client to make the logfile clear

server started on <1.2.3.4> at port <12345>…
received client connection <UID> from hostname <cs.usc.edu>
port <port>
<UID> received USCID <0123456789>
<UID> received Name <John Doe>
<UID> sending image file <imagefile>...
<UID> terminating client connection…

Client Logfile
connecting to the server <1.2.3.4> at port <12345>
connected to server hostname <cs.usc.edu> received <UID>
sending USCID <0123456789>
sending Name <John Doe>
received image and saved in <imagefile>
terminating client…

Code and Collaboration Policy

You are encouraged to refer to the socket programming tutorials. You can also discuss the assignment and coding strategies with your classmates. However, your solution **must** be coded and written by yourself. Please refer to the plagiarism policy in the course syllabus.

The submissions will be run through code similarity tests. Any flagged submissions will result in a **failing score**. Keeping your code private is your responsibility.

You are strongly encouraged to pair and test your client and server implementations with your peers in class. Use Piazza to publish the serverIP and port number for your server implementations so that other students can connect to them.

Online Resources:

The UNIX Network Programming Guide
https://scoecomp.files.wordpress.com/2014/02/2003-unix-network-programming-vol-1-3rd-ed.pdf

Common Port Numbers http://ciscotut.com/pdf/common_ports.pdf

Submission Instructions

You can develop and test your code on aludra or using your own machines. Create a compressed tar file which includes a README, Makefile, and the source code. The code should compile and execute on aludra.
To submit, create a folder called assign1 with the code, Makefile, and README. Tar the folder assign1. Submit the tar file on blackboard.

The README must contain: USCID, compiling instructions, additional notes on usage if needed.

To evaluate your project we will untar and type
% make

It is a project requirement that your code build without any warnings (when com- piled with -Wall flag). Structure the Makefile such that it creates the client and server executable (not a.out). For more information please read the make(1) man page.

We will then run your programs using a suite of test inputs. After running the program, we will grade your work based on the output. It is recommended that your implementation be somewhat modular. This means that you should follow good programming practices—keep functions relatively short, use descriptive variable names.

Deadlines

Due on Feb 5th by 6pm