ROSE-HULMAN INSTITUTE OF TECHNOLOGY
Department of Electrical and Computer Engineering

ECE 312 Communication Networks Winter 2016-2017

Due Friday, December 16

Project 1 – Socket Chat Program

Read Chapter 2.7 of William Stallings on sockets programming. Look at the examples in Figures 2.13 and 2.14, and read the description that follows on each program to gain an understanding of how they work. You can also look at the thorough yet still very approachable *Beej's Guide to Network Programming* for a more complete understanding of socket programming: https://beej.us/guide/bgnet/.

Modify the example programs to create a 2-person chat application. You will need two programs, as in the examples: ChatServer.c and ChatClient.c. The application should set up a TCP (SOCK_STREAM) connection between the two hosts, as in the example, but after the connection is established it should allow both hosts to continue sending messages to the other host until the application is exited. At the start of the program, the application should prompt the user for a username to be used in the chat program. This username should be shared between the hosts following connection establishment. The program should indicate the IP of the connected host once the connection is established, and messages should be labeled with the message sender, such as:

Provide user name: student
Waiting for connection . . .
Connection established with 137.112.48.21 (OtherStudent)
<you> hello
<OtherStudent> how are you?
<you> better now that this chat program is working

Your program should have a fixed exit signal, which will cause both hosts to exit the program. For instance, entering 'exit' on either host will cause each to terminate. Instead of providing the port number as a command prompt, you may use a fixed port # in your program. You can use the command 'ipconfig' from a shell prompt to obtain the IP address of the server machine (along with a bunch of other details about your network interface). The example programs are for a UNIX environment, so you will need to run them on a UNIX system, such as Linux, or use Cygwin to run them on Windows. Details on installing Cygwin will follow. If you are familiar with WinSocks, feel free to modify the program to use Windows Sockets instead.

This project is due Friday, December 16. You may work in teams of two. Please submit the source code for your client and server application, along with screen captures from each hosts that successfully demonstrate each of the requirements stated above. Your submission should have a cover page with each team member's name. With your submission, please also provide a response for the following: Consider how you would implement the application to allow more than two people to chat at once. In this group chat, any message from one user should go to all users. You do not need to implement the group chat, but discuss what changes you could make or would be needed to accomplish this goal.

Cygwin install instructions

Cygwin is a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows. Go to https://www.cygwin.com/install.html and download the *setup-x86_64.exe* install program. Choose the default options while installing until you reach the window for selecting packages. Choose the *Default* option next to *All* to select a range of basic packages. Then expand *Devel* and choose the following packages: binutils, gcc-core, gcc-g++, and gdb. These should be sufficient for what you need in this project. Feel free to add additional packages if you like, but you probably don't want to choose *Install All* packages, since this would be many gigabytes of applications. Choose Next, and continue to choose default options until installation is complete.

After installation, you should have a shortcut to Cygwin on your homescreen or Start menu. When you open Cygwin, it will open a bash shell with the current directory initially set to ~. ~ is the home directory for the current user in the UNIX environment. You can find this directory in your file explorer by going to C:\cygwin64\home\username. You will want to copy any files you will be working with into this directory. You may add additional folders in this directory as needed. A Linux Bash Shell cheat sheet can be found at http://cli.learncodethehardway.org/bash_cheat_sheet.pdf if you need assistance with unix commands.

In order to compile the program to run in Cygwin you can use gcc, which is a Gnu C compiler. Use the following command syntax at the command prompt:

```
gcc <c_filename> -o <executable_name> [optional list of libraries to link]
```

As an example, it may look like:

```
gcc ChatServer.c -o Server –pthread
```

This will compile the program ChatServer.c and generate an executable named Server.exe. This example links a thread library.

You can then run the executable by entering:

```
./Server
```