CS 457 –Project 1

CHAT Client and CHAT Server

Instructor: Francisco R. Ortega

Due Date: See Canvas.

Additional Files and Links: See Canvas

Discussion Platform: Piazza .

1. Introduction

This project consists in building a chat client and chat server. To obtain full grade, it is important that you complied with the requirements. However, the instructor will provide latitude to students that may performed certain tasks differently. Please make sure you document everything properly.

Please note that this document outlines a general chat. However, the specific minimum requirements for Project 1 in Section X.

1.1. Environments

We will test and support the following:

Operating System: Latest Ubuntu (18.10) or current linux used at CS CSU.
Language: C++. Please update to gcc 8 and g++8. Use g++-8

You may use any editor but I'm providing files for Visual Studio Code.

While working in your local computer, you can use any port (use above 1024). However, if you work here at school, you may try to use a port than another student may be using. For this reason, in Canvas, you will find a link to ports assign to you. You will find your initials and ports that you can use.

Finally, remember that the local loopback in any computer is 127.0.0.1.

1.2. PORTS

When you deliver the final submission, they must be using the ports assigned to you. The default ports must be found in a configuration file (for the server). However, you will notice in the requirements, that you can override this via the command line. Please note that in the school's server, you may not use port 1024 or below. Please use the assigned ports.

1.3. About Chat Clients and Servers

The following section provides generic notes about CHAT clients and servers.

1.3.1. Chat Clients

We will not follow the RFC for the IRC protocol, as it requires more time to develop properly. However, we do want to emulate the important behaviors. I recommend that you download an IRC client. For example, in Windows, mIRC is very popular. Linux and MacOS X also have a variety of IRC clients.There are many IRC clients for you to download. This will help you to understand what is expected. You will need to read the RFC for the IRC protocol: https://tools.ietf.org/html/rfc1459.

Please follow the instructions in this guide and instructions that were provided in class. While we are not complying a 100% with the RFC, we are using the commands used in the IRC client and server.

1.3.2. About Chat Servers.

There are multiple IRC servers. Their operation is beyond this project. Nevertheless, you may want to see a very popular IRC that is deployed in linux servers: https://www.unrealircd.org.

1.4. Chat Commands that must be supported

All commands below must be passed with slash and case insensitive. For example, /list or /List or /LIST. The following commands must be supported. Additional commands are shown for extra credit and/or completeness. Please note that I will be flexible how you implement the comments. In addition, I do ask you to implement the critical ones (sending messages, and so forth) and not to worry about all the comments.

AWAY
CONNECT
DIE
HELP
INFO
INVITE
ISON
JOIN
KICK
KILL
KNOCK
LIST
MODE
NICK
NOTICE
PART
OPER
PASS
PING
PONG
PRIVMSG
QUIT
RESTART
RULES
SETNAME
SILENCE

TIME
TOPIC
USER
USERHOST
USERIP
USERS
VERSION
WALLOPS
WHO
WHOIS

Extra Credit

ADMIN
CNOTICE
CPRIVMSG
ERROR
LUSERS
MOTD
NAMES
SERVER
SERVICE
SERVLIST
SQUIT
SQUERY
SUMMON
TRACE
WATCH
WHOWAS

The following commands complete the list. However, they may be extremely difficult to implement in some cases.

ENCAP
LINKS
NAMESX
REHASH

The definitions of the commands can be found in the RFC. Additional information (must read) is located at: https://en.wikipedia.org/wiki/List_of_Internet_Relay_Chat_commands

1.5. Languages and Environment.

You must use C++. Remember that you may leave processes running on the background (even if you logout) by putting the ampersand (&) next to it. You may also try (this may vary if the server allows you to do) to run it as a daemon (goo.gl/Y4jcdF).

2. CHAT Client

A wrapper for C++ is provided for you for your convenience. It is your responsibility to review it and modify it if needed. Of course, we will assist and provide updates. However, you are not required to use our updates or use or wrapper. While the example is provided for the server, you can easily wrap the C++ client. This is provided for your convenience but you can write the code from scratch. Note that if you use the code provided – it is provided as is. This is true with any software. It is good to be sure that what you have works and works the way you expect. Therefore you are responsible to make sure that it works and that it conforms to the requirements. Follow the instructions from class to complete the Chat client and the additional information provided in this document. You must read RFC's specifications (but we are not complying with the RFC – more about this in class) to implement the commands in §1.3 (and read the link provided). It is important that you understand the behavior of the CHAT client by using one. Matching the behavior will provide the highest change of getting full grade on your assignment. This is part of the assignment.

An additional important requirement is that while you implement the chat server commands, you must use the common commands already described in order to make it operational. Finally, it is important that the client is responsive, therefore, making it multi-threaded is required.

It is expected that you allow users to be in rooms and send private messages to individual users.

2.1. Chat Client Usage

While this will be a graphical user interface (GUI) application at later stage, you will be able to pass parameters to program. There are multiple ways to parse arguments, with the most basic parsing done by creating your own parsing function to more sophisticated argument parses (including boost library). There are many in the web and you may free to use any of them (please reference it). One interesting one is found in github (but I haven't tried it): https://github.com/Taywee/args

Other links below:
https://github.com/CLIUtils/CLI11

https://stackoverflow.com/questions/865668/how-to-parse-command-line-arguments-in-c

The following are the parameters that the client must accept.

-h hostname
-u username
-p server port
-c configuarion file

-t run test file; -t test2.txt
-L log_file_name (for log messages)

2.2. Chat Client's Configuration File

The ftp client configuration file will be text based (chatclient.conf), with the following style.

```
#this is a comment
last_server_used        127.0.0.1
port                    50000
default_debug_mode      False
log                     True
default_log_file        chat.log
```

You may add any additional configuration parameter that you may need.

2.3. Chat Client's Test File

The first line of the test file will allow the same parameters as provided by the usage (above). In other words, any configuration needed

Each additional line will contain a valid chat client command. This will help you to run a system test using multiple clients if needed. You can use # for comments.

3. Chat Server

The chat sever is critical for this assignment. It must be multi-threaded (this has been explained in class). The assignment requires that your server support the command listed (shown in §1.4.) While in theory this may mean that any IRC client should work with your server, this will not be the case in most cases, as we are working in a simplified version. The following list provides the required components:

- Server must run either interactive with a shell or completely in the background without one (or both for extra credit).
    - If running interactive, you should be able to have a shell (>) where you can type basic commands to know the state of the server. For example, how many connections, user list, and so forth. While this is interactive, you could still run it in the background and comeback to it when needed (see §1.5).
    - If running in background without a shell, then you must provide a way to interact with the server.
    - An **extra credit** option is that your server can do both.
    - Commands that require support from the server point of view are provided in §3.1.
- Your Server must have a configuration file with basic information (chatserver.conf). This must include:
    - port 50000

    - dbpath db/
    - additional_ports 50001, 50002, 50003
        - In case you want to run more than one instance (however, they share all the resources – extra credit).
- Your server must include a series of files to support the functioning of the application
    - banusers.txt: contains users who are banned.
    - users.txt: contains list of users with the following properties
        - username password level banned
        - if password is empty, it must be represented with @
        - password symbol @ is not allowed.
        - Examples:
            - fortega cnt4713 admin false
        - levels: user, channelop, sysop, admin
            - user: regular user
            - channelop: has control of channel kicking or so (shown with + in the userlist)
            - sysop: has control of entire chat system except admin privileges.
            - Admin: has all the access plus can start, stop, restart the server.
        - Users must not be registered (this is optional)
    - channels.txt: contains list of registered channels.
        - name description password [list of channelops – if any – otherwise sysop controls and can promote if needed]
        - empty password is represented with @
    - extra credit: channel_user.txt
        - user created channels, only active during session of the user.
    - banner.txt: Contains basic information that server displays during login.
    -
- Chat Server must be able to respond to the commands in §1.4
- Chat Server must handle each connection in a separate thread.
- The Chat Server usage should include (at least)
    - -port Port Number where this chat server will run. Default will be found in configuration file (-port 1000)
    - -configuration configuration file path
    - –db user file path
- The default locations for configurations for the chat server are located in the same place that the chat server is running under a /conf folder. Finally, the default location for log files should be /logs.

4. Tests

It is recommended that you test. This will help your grade.

5. Documentation

You are required to provide a readme.me file containing all-important information including how to run it and all the instructions required. In addition, a well-written report including details about this assignment will be required at the end of the project (final project). Please see §5.1 for specific information.

5.1. Report Sections and Information (for final project only)

- Report Name
- Complete name and CSU EID
- Introduction: Describe the problem that you are trying to solve and additional study you needed to do. For example, the requirements may be that you needed to read another book or a web site. Provide an insight on the final product of the lab in this section as well.
- Problem Statement: Describe what is the problem you needed to solve
- Methodology: Described how you solved the problem. This must include important information such as socket information and so forth. This may also include the configuration and anything else you created.
- Results: Provide some sample output and provide discussion if needed.
- Analysis: Describe your learning experience, problems you encountered, and anything else that may be useful.
- References: Provide references including sites, repos, books, and any other information you have used for this lab and this report. Use correct citation when doing it.
- Additional Notes:
    - Please be sure to have a well formatter report. Always use "Justify Text" when writing and also create the headings with numbering of them, such as 1. Introduction. You may need to create sub-sections at times.
    - Be clear and concise. Don't add code that you didn't write unless extremely important and don't add your entire code.
- You must hand this report physically in class (unless you are in an online class) and uploaded as PDF with your code. Word documents, text files, or other formats will be ignored.
- This report is equally important as your code.

5.2. Video

You are required to create a video demonstrating your application.

6. Grading Criteria

The following grading criteria will be used:

- FTP Client: 40%
- FTP Server: 40%
- Overall Project: 10%(Here, I measure the quality and so forth).
- Report (if applicable), readme file, and Video: 10%
- Extra Credit: At discretion of the instructor

7. Project 1 Requirements

This document provides a very detailed system. Given this is part of a larger project (Part 1, Part2+Part3), it is important to clarify what is needed:

1) 80% of the commands implemented by the end of Project 1.
    a. With basic commands, such as the ones that provide the ability to connect and send messages.
    b. This means a functional system
2) Configuration files and command-line arguments.
3) I do suggest testing files implemented. However, if they are not, I will have my own. This will be required to be implemented later.
4) Readme and Video.
5) Multi-threaded system
6) Binary files are not required (optional).

8. Future Requirements

This document does not provide detailed information of the requirements for Project 2+3. However, here are some indication of what it will be expected if you continue working on this project:

1) Complete missing requirements (from Project 1).
2) Add Graphical User Interface (Qt 5).
3) Use UDP connections in addition to TCP connections.
4) Send binary files.
5) Test Files

If you decide to work on a different project for Project 2+3, you must submit a proposal (via online canvas) specifying your project for approval. The lack of submission means that you will continue working in the Chat program.