Project Report

Adam Vandolder

Contents

1 Introduction

For my Computer Networks project, I decided to create a client-server chat application, called termchat, in Python. termchat is so-called because both the server and client are entirely terminal-based applications, utilizing a text-based user interface made with Curses. It uses the TCP protocol in order to communicate between the clients and server.

2 Features of Application

termchat is a fully functional terminal-based chat server and client, with commands inspired by the IRC protocol. The client uses a text-based user interface, while the server isn't interactive. Many clients can connect to a given server by its domain and port number, and can then communicate with each other. Servers provide a series of different channels that clients can join in order to chat about more specific topics, and clients can make new channels on-the-fly. Clients are free to set non-conflicting nick names for themselves, and request lists of current users connected to the server. Clients can also direct message any other client connected to the same server, regardless of which channels those clients are currently on. The client's text window can be cleared of current messages as well.

3 Tools and Techniques

termchat relies primarily upon Python for its operation. More specifically, I made use of Python 3.6 and some of its libraries: sys, threading, typing, socket, and string, all of which come with Python by default. I also made use of Python's curses library, which only comes with Python on Linux and Mac OS X, so on Windows the 'windows-curses' library must be installed as well. When writing the project, I mainly used Vim. In order to create the report, I used

LaTex and the texlive package. In order to generate the Windows binaries, I used the PyInstaller program.

4 Implementation Details

termchat is based around a client-server architecture, using the TCP protocol via the socket library to provide communication between the processes. The server is multi-threaded, spinning off a new thread to handle each new client that connects. The client is single-threaded, and uses curses in order to maintain a Text-User-Interface (TUI) that shows both the messages from the server as well as the users input at the same time.

5 Execution Instructions

If you are going to be running termchat on Windows, I have included pre-built binaries for the server and client programs. These are accessible as the dist/server/server.exe and dist/client/client.exe respectively. You will still need to pass the domain and port number as command-line arguments to the server program as shown below.

In order to run termchat, you will need Python installed, and at it will need to be at least version 3.6. In order to run the client, you will also need the Curses library installed. If you are running Python on Linux or Mac, then Curses will be installed by default. However, on Windows, you will need to install the *windows-curses* library. This can be installed with the following command:

python -m pip install windows-curses

Then, in order to launch the server, you will have to run the following command from the project directory:

python src/server.py domain portnumber

If you want to run a server instance that interacts with clients on the same

machine, set *domain* to localhost and *portnumber* to a high number that is likely unused, such as 9999.

In order to launch a client, run the following command:

python src/client.py

You can then enter the */help* command in order to get a list of the supported commands. Enter */server* followed by the domain and port number you gave the server instance in order to join your local server.

6 Conclusion

In the process of creating this project, I learned a substantial amount about how to work with TCP sockets as well as how servers work, especially in regards to handling many clients at once with multi-threading. All in all, this project was a successful learning experience.

7 References

When making this project I relied heavily upon the main Python documentation, especially:

sockets docs: `https://docs.python.org/3.6/library/socket.html`

threading docs: `https://docs.python.org/3.6/library/threading.html`

curses docs: `https://docs.python.org/3.6/library/curses.html`

Curses Programming with Python: `https://docs.python.org/3.6/howto/curses.html`

I also utilized Curses Programming in Python on Dev Dungeon: `https://www.devdungeon.com/content/curses-programming-python`