

CS 5200
Assignment 4
Due September 26, 2008
20 Points

The License Plate Game

Overview

This is the second part in our multi-part programming assignment associated with the *license plate game*, which is an educational way to pass the time during long car trips. In this assignment, you'll develop a more sophisticated game that will reuse portions of your first solution. In addition, you'll get to use Linux sockets in implementing a client-server application.

Program Description

The initial assignment involved writing a simple C++ program that automated the license plate game by supporting the following commands:

- **Dictionary** – Allow players to designate a dictionary of valid words.
- **Set** – Set the current letter set to the desired three letter string of letters from a license plate.
- **Guess** – Allow players to guess whether a particular word contains the current letter set in order in the word and also exists in the dictionary
- **List** – List all the words in the dictionary that match the current letter set.
- **Score** – Show the current score for the player.
- **Quit** – Stop playing the game and exit the program.

In the second part of the assignment, you enhance your license plate game program to use a client/server architecture. As before, the client reads commands from standard input and displays the results on its standard output. Instead of running the commands in the client process, however, the commands are sent to the server process where the commands are run and their results returned to the client for display.

Below, the general behavior of the client and the server processes are discussed.

Client Application

The license plate game client program should perform the following activities:

1. Read the server machine's port number (a number greater than 7000) and hostname from the command line.
2. Create a socket object that is connected to the server machine on the designated port.
3. Read commands from the user's standard input and convert these into the appropriate type of encoded messages.

4. Send the encoded messages to the server and wait for the reply.
5. Read the reply from the server and display it to the standard output of the client.

Server Application

The license plate game server program should perform the following activities:

1. Read the server machine's port number (a number greater than 7000 but the same as the server) from the command line.
2. Create an Internet domain "passive-mode" stream socket on the port number specified on the command line and wait for connections to arrive from license plate game clients.
3. When a client request arrives and is accepted use the connected socket object to receive the client request from the socket and convert the message into the appropriate type of command.
4. Perform the appropriate operation designated by the type of command received from the client.
5. Assuming the operation succeeds, encode and transfer the reply back to the client.

The server should not exit unless you explicitly kill it via a signal (e.g., SIGINT). You should add a signal handler that performs any termination code necessary to gracefully shutdown the server upon receipt of a signal. It is hard to come up with a completely portable scheme for handling signals and shutting down tasks in multi-threaded programs. For this assignment, you can simply exit the server and close down all the open sockets.

Concluding Remarks

I strongly encourage you to use, develop, and apply reusable C components based on the code you write. You will reuse these components throughout the course since the assignments build off each other.