Computer Networks HW1 Report

B03901078 蔡承佑

1 Overview – Data Structure

I use C++ to implement this program, with socket programming in C. So basically this is a program mixing C and C++, with pretty much OOP in it.

1.1 The top level – `bot`

On the top level, there is a class `bot`, managing all variables and functions that should be maintained in this program. And it has a member object `mySocket` which provide C-style socket programming, providing a more convinient interface for IO of the bot program. Besides, there are still two little class, `UserInfo` and `Line`, storing data for function passing convinience.

Following are the prototype of these classes:

```
class bot
{
 public:
    bot(const char* hostname="irc.freenode.net", const char* port="6667",
        const char* nick="MikeTsai", const char* user="Mike");
    void setInfo(const string& nick, const string& user);
    bool getConfig(ifstream& fin);
    void reply(const string& str) const;
    bool handleMsg();
    void display() const;
 private:
    // Helper functions for handling messages:
    // Reading _buf to make decisions.
    bool isPING() const;
    void extractMsg();
    void initHelpMsg();
    // Member variables:
    mySocket                _socket;
    UserInfo                _userinfo;
    Line                    _line;
    string                  _intUser;
    mutable char            _buf[1024];
    GuessNum                _guessNum;
    Calculator              _calculator;
    map<string,string>      _help;
};
class mySocket
{
  public:
    mySocket();
    mySocket(const char*, const char*);
    ~mySocket();
    int connect(UserInfo&);
    int read(char*,size_t, double timeout=1.2) const;
    int write(const char*, size_t) const;
    void close() { ::close(_sockFd); }
    void joinChannel(UserInfo&);
  private:
    void initSocket();
    // Socket:
    struct sockaddr_in      _serverAddress;
    string                  _hostname;
    unsigned                _port;
    int                     _sockFd;
    mutable fd_set          _rset;
    mutable fd_set          _wset;
};
```

1.2 `GuessNum` and `Calculator`

This bot provides several commands. The complex ones such as `GuessNum` and `Calculator` are encapsulated in a class. This method provides a simplier interface for the top program. We can write and test these functions independently without altering the code structure of the top-level program. And for further optimization or to add more function, we can simply work on the lower-level codes, what the top-level codes should do is make sure that the interface is correctly connected. Another thing worth mentioning is that the class `rnGen` in the class `GuessNum`, which stands for random number generator. Although this may not be a real random, we can further optimize this generator without changing the codes in `GuessNum`, this also shows the advantage of encapsulation.

1.3 help

The data of the command ”help” is stored in a `map<string,string>` and initiated by the function `bot::initHelpMsg()`. To extend more help message, simply add them in that function.

1.4 Other details

Though as more level we divided, codes can be worked and tested independently, communication between these functions such as exception handling and argument passing becomes more important. There are two points that should be careful.

1. The arguments passing for large object should be passed in *reference*.
2. Using exception (`stdexcept` library) throwing and catching to handle the exceptions.

1.5 Summary – all self-defined classes

`bot`	The top level structure include socket connecting and message handling.
`mySocket`	Encapsulate the C-style socket programming.
`UserInfo`	Encapsulate variables that is needed to connect to channel.
`Line`	The message received will be divided into this structure.
`GuessNum`	A class manages all variables that are needed in a guess-number machine.
`Calculator`	A class manages all variables and functions that are needed for calculating a string.

2 Challenges and Solution

2.1 Connection to the channel

At the beginning, it takes very much time on realizing the connection of IRC and socket programming. Although TAs and the professor may assume that the students have learned socket programming in System Programming, but this is a whole new field for me. And there are many rules that we must follow to send messages to the socket; otherwise, the socket simply replies the command not found.

Also, timing for input or output is also important in connection. Sometimes we need to get messages from the socket, but sometimes we need to send messages to it. When to read and when to write becomes very important. That is the reason why I use `select()` in `mySocket` class. This function can provide a blocking read but waits for only assigned time. In connection, we only have to read finite messages from the socket, and basically other time is writing. This functions avoids infinite waiting for reading and do not miss any.

2.2 ”Realizing” the messages

Parsing the messages received from the socket is a very important problem that must be carefully handled. That’s what the function `bot::extractMsg()` does. It reads message from the buffer and parse it into a better structure so that we can simply handle the `Line::msg` part which stands for the messages sent by users.

2.3 Calculator

Mainly three problem to handle in implementing a command-line calculator:

1. Recognize and parse operators and operands.
2. Operator precedance
3. Exception handling

Each of them takes considerably much time. I use the concept of FSM to handling parsing and operator precedance problem. If a step does not follow the FSM flow, there must be a exception should be thrown.

3 Reflection

1. Data structure must be planned thoroughly at the very beginning. Changing the data structure of a existing program is very much time-consuming.
2. Self-learning is very important in doing this homework. Many details and information are not provided in the tutorial.
3. A better code structure saves a lot of time. I’m glad that the function `GuessNum` and `Calculator` can be tested in a smaller program.