Chess Chat Program
Version 0.0014: May 31, 2020

Andres Cervantes, Bradley Chu,
James Guise, Taylor Togami,
& Trenton Nguyen
(Team 16)

University of California, Irvine ▪ Samueli School of Engineering
5200 Engineering Hall Irvine, CA 926297-2700▪+1-949-824-4333▪ugengr@uci.edu▪ +1-949-824-4334

Table of Contents

Glossary **3**

1. Client Overview **5**
1.1 Main Data Types and Structures 5
1.2 Major Software Components 5
1.3 Module Interfaces 5
1.4 Overall Program Control Flow 5

2. Server Overview **7**
2.1 Main Data Types and Structures 7
2.2 Major Software Components 7
2.3 Module Interfaces 7
2.4 Overall Program Control Flow 7

3. Installation **8**
3.1 System Requirements and Compatibility 8
3.2 Setup and Configuration 8
3.3 Building, Compiling, and Installing 8

4. Documentation of Packages, Modules, and Interfaces **9**
4.1 Detailed Description of Data Structures 9
4.2 Detailed Description of Functions and Parameters 9
4.3 Detailed Description of Communication Protocol 9

5. Development Plan and Timeline **11**
5.1 Partitioning of Tasks 11
5.2 Team Member Responsibilities 11

Glossary

Account - Account data (Username/Password/Friends) is stored in the server. This data is then accessed by the individual clients.

Chat - Chat is used in conjunction with ChatWin.c/h which displays the string of characters one account sends to another account. Chat consists of strings.

ChatWin.c/h - GTK is used to provide a GUI interface of the chat window which contains the list of the messages that *have been* sent and a textbox with a “Send” button to write new messages to another account/user. The chat window also includes the chess game built in at the same time

Chess.c/h - Chess.c/h contains the functions and variables needed to play Chess in a terminal based environment. Chess.c/h is used in order to interact with other users/accounts in a chess match.

ClientWin.c/h - A GTK GUI based client which serves as the main client with displaying online friends, account settings, and other general stuff. Basically is the lobby window to set up a game of chess

Contacts - Contacts are to be stored in the GTK GUI for the friends list.

GTK.c/h - An open-source cross-platform widget toolkit that allows for graphical user interfaces (GUI)

LoginWin.c/h - In this GTK GUI window, users are able to login into their account or create an account with the specified information entered into the *Username* and *Password* textboxes.

Main.c/h - Initializes and runs required functions from server, client, chat functionality, and chess functionality.

netinent/in.h - Contains definitions used for Internet Protocol

Server.c/h - Account username, password, and friend list is stored in a server. This server is also where the clients send messages to and receive messages from. The server is also central to communication with different clients to be able to message and play chess against them

Register - Creates a new username and password stored in the server.

sys/socket.h - Contains sockets definitions and functions used for communication between server and client

sys/types.h - Used for defining certain types of data.

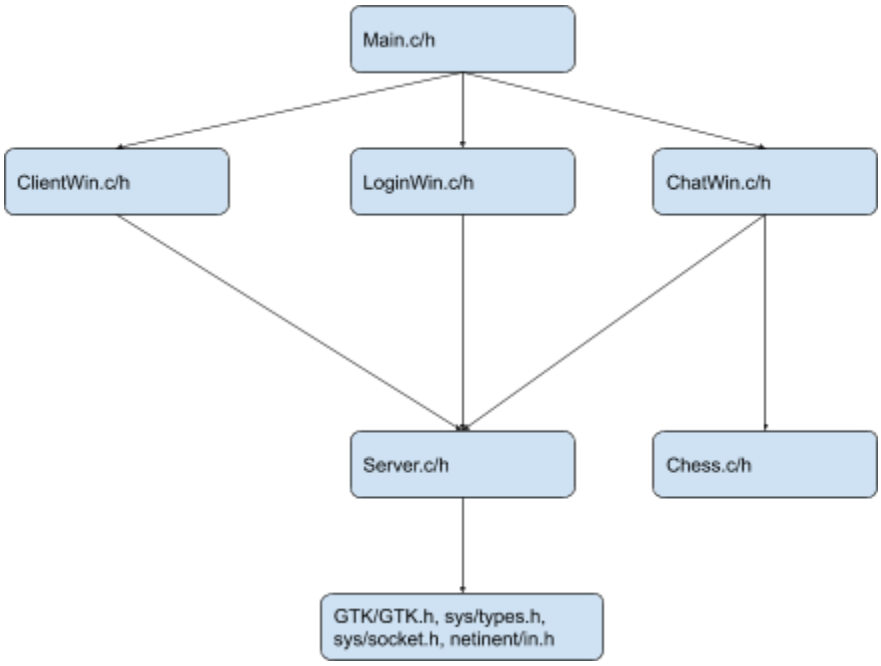
X11/Xming/Xlaunch - Required software for being able to run ChessChat. This software allows for the user to use the GUI widgets from the gtk library.

1. Client Overview

1.1 Main Data Types and Structures

- The main data types and structures used for this project are:
    - The login window: gtk
    - The client and server .c files
    - The chess game function (to be implemented): structure
    - Chat window: gtk

1.2 Major Software Components



1.3 Module Interfaces

- Chess.c
- LoginWin.c
- Game client
- Game server

1.4 Overall Program Control Flow

- Setup:
    - Get user input for login credentials or create a new account for input
- Chat Function:
    - User picks button to either chat or challenge a game of chess
    - If challenge picked -> repeat:
        - User types in input for chess game: MOVE:
        - Input sends to server
        - Server interprets information
        - Chess game takes input from server
        - Game of chess displayed on same window

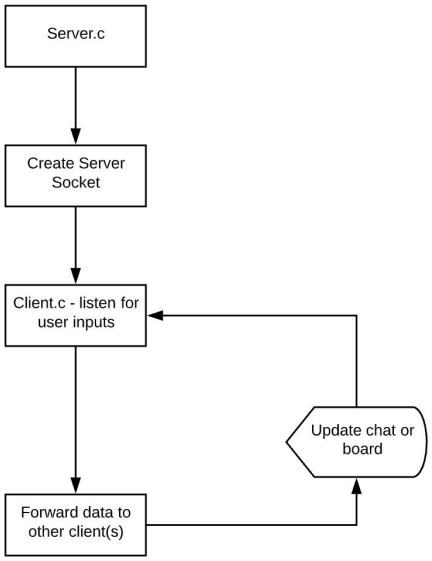
- If chat picked -> repeat:
    - User types input: MESSAGE “asdfghjkl;”
    - Input sent to server
    - Server receives message
    - Server sends message to other client user

2. Server Overview

2.1 Main Data Types and Structures

- The main data type and structure used for this project is:
    - Sock_addr_in (STRUCT): contains various structs and other variables that are connected to the server port to hold all of the information from various clients

2.2 Major Software Components



2.3 Module Interfaces

- Server.c
- ServerResponse(int sock): takes the input of the client and determines what to return back

2.4 Overall Program Control Flow

- Setup:
    - Create the server and allows clients to connect
- Chat Function:
    - Waits for a response (input) from client(s)
    - If challenge picked -> repeat:
        - Server waits for client to pick a chess move (input)
        - Server sends chess move to other client user (output)
    - If chat picked -> repeat:
        - Server waits for client to send a message (input)
        - Server sends message to other client user (output)

3. Installation

3.1 System Requirements and Compatibility

- Red Hat Linux - Linux OS (RHEL-6-x86_64)
- Available disk space: 100 MB
- X11 Server Client(Xming or XLaunch)

3.2 Setup and Configuration

- Download the latest version of the Chat and Chess file
- Type tar -zxvf Chat_and_Chess_Alpha.tar.gz to untar the file
- Type make to create the server and client executable files in bin/

3.3 Building, Compiling, and Installing

- Type ./bin/Server <port number> to create the server on the port number (Example: ./bin/Server 16420)
- Type ./bin/ChatAndChess <host name> <port number> to connect to the server (Example: ./bin/ChatAndChess localhost 16420)

4. Documentation of Packages, Modules, and Interfaces

4.1 Detailed Description of Data Structures

- Login Struct (From Login.h)

```
typedef struct loginstruct{
    GtkWidget *Box;
    GtkWidget *TopTextLabel;
    GtkWidget *LoginButton;
    GtkWidget *CreateAccountButton;
    GtkWidget *UsernameEntry;
    GtkWidget *PasswordEntry;
    GtkWidget *PasswordRe_EntryLabel;
    GtkWidget *PasswordRe_Entry;
} LOGIN;
```

  - This struct is essentially a global variable struct that houses GTK widgets that need to access other widgets
  - By malloc()'ing for this struct and assigning each widget, one can pass in the struct to access any other widget inside the struct

4.2 Detailed Description of Functions and Parameters

- Static void CreateAccount(GtkWidget *widget, gpointer data)
  - Callback function that sends in the username and password to server to create an account for the user so they can log in
- Static void Login (GtkWidget *widget, gpointer data)
  - Callback function that sends whatever is in the username and password fields to the server to check if it's a valid login. If valid then logs the user into the game client, else it keeps the user on the login screen
- LOGIN * CreateLogin()
  - Creates the login struct, basically malloc()'s for all of the widgets so they can be accessed by other widgets
- Void DeleteLogin(LOGIN * login)
  - Deletes the login struct by free()'ing it
- Static void destroy(GtkWidget *widget, gpointer data)
  - Callback function that closes the window upon clicking the red 'X' on the top right or quit

4.3 Detailed Description of Communication Protocol

- This program uses TCP/IP protocol in which a client requests from the server and the server responds
- The client sends specific messages so that the server knows what to do when it receives them

Protocols(*Partially Implemented):

- LOGREQ <username> <password>: Login request protocol, sent from client to server with username and password attached
    - Ex. LOGREQ Jacob hunter2
    - It is planned to encrypt the username and password to avoid revealing sensitive data to the server runner
    - The server will check the database of user files and see if the username and password match
    - The server will return a single message out of two possible ones, SUCCESS or FAILURE
    - The client will read the message from the server and if it's a SUCCESS then the client will proceed to the Client Window screen, else the client will stay in the login screen
- CREACC <username> <password>: Create account protocol, sent from client to server with username and password attached
    - Ex. CREACC PogChamp ultrasecretpassword
    - Just like in LOGREQ, it is planned to encrypt the username and password
    - The server will first check if the username already exists, if it does then returns a FAILURE to the user
    - If successful the server creates a user file in the database that stores the username, password, IP address, and port number
    - The user will then be able to login to their account

Planned Protocols:

- GAMECH <user1> <user2>: Challenge request from user1 to user2
- GHSMVE <initialposition> <finalposition>: Move piece from initial to final position
- MESSAG <user1> <messagecontents> <user2>: send messagecontents from user1 to user2
- ADDFRI <user1> <user2>: Send friend request from user1 to user2
- DELFRI <user1> <user2>: Delete user2 off of user1's friendlist

5. Development Plan and Timeline

5.1 Partitioning of Tasks

- Andres Cervantes
    a. User Manual: Section 1
    b. Software Architecture: Glossary
    c. Code: Server.c & ClientWin.c/h

- Bradley Chu
    a. User Manual: Section 3 (Chat Functions and Features)
    b. Software Architecture: Section 4 (Documentation of Packages, Modules, and Interfaces)
    c. Code: LoginWin.c/h, ClientWin.c/h, server2.c

- James Guise
    a. User Manual: Installation
    b. Software Architecture: Section 2
    c. Code: Server.c

- Taylor Togami
    a. User Manual: Title, back matter
    b. Software Architecture: Title, 1.1-1.4, back matter
    c. Code: LoginWin.c/h, Server.c

- Trenton Nguyen
    a. User manual: Table of Contents, Glossary
    b. Software Architecture: Table of Contents, 5.1-5.2
    c. Code: ChatWin.c/h

5.2 Team Member Responsibilities

- Manager: Andres Cervantes
    - Leads meetings and keeps meetings productive by moving from one subject to the next
    - Assigns deadlines and responsibilities with team approval
    - Turns in team assignments
- Presenter: Taylor Togami
    - Presents progress updates and effectively communicates with TAs when needed
    - Makes sure all members are clear on development plans and project infrastructure
- Recorder: James Guise
    - Documents and keeps track of important points during meetings
    - Brings up important information from previous meetings to go over or review
- Reflector: Bradley Chu and Trenton Nguyen

- Ensures team are all agreeing on same points before moving onto next topic
- Pass around the group’s ideas to better refine and come up with the final implementation.

References

- User Manual
- Network Communication & GUI Practice Videos
- <sys/types.h> File

Index

Account, 3, 5, 10
Array, 10
Button, 3, 5
Challenge, 5, 7
Chat, 3, 4, 5, 7, 8, 11
Check, 3, 9, 10
Client, 3, 5, 6, 7, 8, 9, 10, 11
Fail, 10
Login, 3, 5, 9, 10, 11
Message, 3, 6, 7, 9, 10
Password, 3, 9, 10
Port, 7, 8
Protocol, 9
Request, 7, 9, 10
Server,3, 4, 5, 6, 7, 8, 9, 10, 11
Terminal, 3
Username, 3, 9, 10
Window, 3, 5, 9