User Manual

HeroChess Version 2.0

June 7, 2021

Paul John Lee, Irania Ruby Mazariegos,
Rachel Villamor, Keane Wong
EECS 22L, University of California Irvine, Irvine, CA 92697

Contents

Glossary of Networking Terms

Client: any hardware or software that connects, uses, and/or communicates with a server
Connection: session by which information is transmitted and received throughout a network
Host: any hardware that permits service and communication to other hardware or networks
IP address: a unique string of characters that identifies each computer using the Internet Protocol to communicate over a network
Network: two or more computers connected together for electronic communication
Port: virtual endpoint that is associated with a host's IP address and serves as a means of communication between a server and application
Protocol: a set of rules that describe the how data is communicated and processed between devices within a network
Server: a computer or computer program which manages access to a centralized resource or service in a network.
Session: a temporary period of communication between hardware or users
Socket: A term used to refer to a single end of a communication channel used to manage data transfer
Error Messages: Check the "Error Messages" section for any networking error during the game.

1 Installation

1.1 System requirements

To use this version of the HeroChess Multiplayer functionality, the user(s) must have access to the EECS Linux Servers, or a suitable host with a stable release of the program. Users are recommended to have the Linux version CentOS 6.10 and perhaps even Xming if the graphical user interface is included. The amount of memory required for the program to run is still to be determined. All users that use the multiplayer functionality must have access to an internet connection for the duration of their playtime.

1.2 Setup and Configuration

Download a SSH client with terminal support.

1. After unpacking the User/Customer Package, type on the terminal 'cd `Chess_V2.0`' and press enter.
2. Finally, type 'make' and press enter. Installation is now complete.
3. To run the game server, type './bin/`HeroChessV2_Server` 11919' and press enter.
4. To run the game client, the clients must type './bin/`HeroChessV2_Client` zuma 11919' and press enter.

   NOTE: Other servers apart from zuma can be used, but the server and clients must all be using the same server name and port.

   - bondi.eecs.uci.edu
   - crystalcove.eecs.uci.edu
   - laguna.eecs.uci.edu

1.3 Uninstalling

Users will be able to uninstall the Chess game executable files by being in the directory called `Chess_V2.0_src` and using the 'make clean' command.

2 Chess V2 - Client

2.1 Overview of Features

ChessV2 is the extended version of Chess V1.0. It will support client system and multiplayer features on the UCI servers along with all these features from the previous version. Players can create an account with a custom username and password which will remain encrypted and confidential permanently. Account will contain individual information in the game, such as player name, win/losses ratio, and the player rating (ELO).

2.2 Logging In/Registering

When the client first opens ChessV2, they will see a menu asking if they are a new user or a returning user:

```
[rvillam1@zuma Chess_V2.0_src]$ ./bin/HeroChessV2_Client zuma 11920
Welcome! Lets begin by logging you in.
1. New User
2. Returning User
3. Exit
```

Figure 1: Login/Registration Menu

The registration process for new users involves entering a username and password. The username cannot contain spaces or newline characters, neither can it be a username that already exists. If the username already exists, the system will notify the user and they will have to enter a new one. After the username is entered into the database, the user enters their desired credentials, also known as their password.
Once the account is created, the client will be taken to the main login menu again to log in as a returning user.

```
[team19@zuma Chess_V2.0_src]$ ./bin/HeroChessV2_Client zuma 11920
Welcome! Lets begin by logging you in.
1. New User
2. Returning User
3. Exit
1
Welcome, new agent, to the Avengineers initiative
Enter your designated S.H.I.E.L.D ID (Username):
(No Spaces, newline characters, or a preexisting name)
CorporalAmerica48
Enter your desired Credentials:
mj0lnir
User now registered. Redirecting to main menu now:
1. New User
2. Returning User
3. Exit
2
Welcome back.
Designated S.H.I.E.L.D ID (Username):
CorporalAmerica48
Hello User: CorporalAmerica48
Designated S.H.I.E.L.D CREDENTIALS (Password):
mj0lnir
```

Figure 2: Registration to Login Process Example

If the username entered does not match a registered user, the client will be directed to the main login/registration menu again.
If the password entered is incorrect, the client will be given the option to try again.

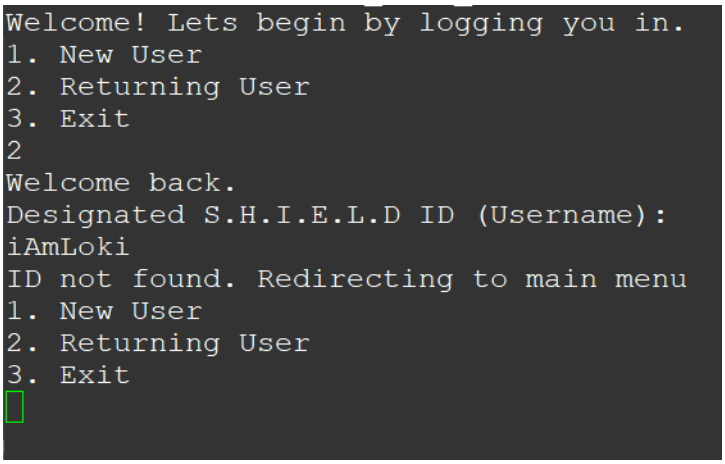

Figure 3: Incorrect Username Login Example

2.3 Getting Started

Players will first see the login screen where they either create an account or login to the existing account. Refer to the previous section, **2.2 Logging In/Registering**, for further details.
After logging in, players will be asked what player they want to be (black or white). This prompt will appear to both clients, but the final decision for who gets to be the first-turn player (white) is decided by the server. This is to handle issues for when both clients choose the same color.

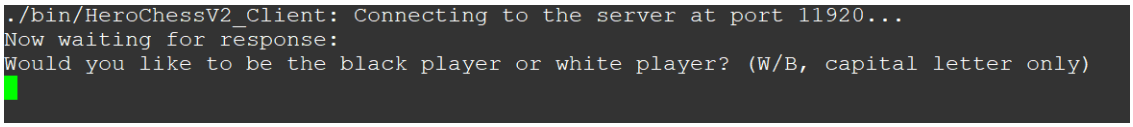

Figure 4: Client Prompt for Color Assignments

2.4 Usage Scenario (Playing a Game - Client View for Both Players

Upon logging in as a user, both clients will see the main menu to the chess game. Here, they will have the option to start a game, change game settings (colors, etc.), or go over the rules of chess. Refer to the previous sections **2.2** and **2.3** for more details.

Throughout the game, prompts will appear on the screen asking the player to enter the positions of pieces they want to move and the positions of the locations they want the piece to move to. Inputs will be taken as keyboard inputs and must follow the format of letter-number, e.g. a4 or A4.

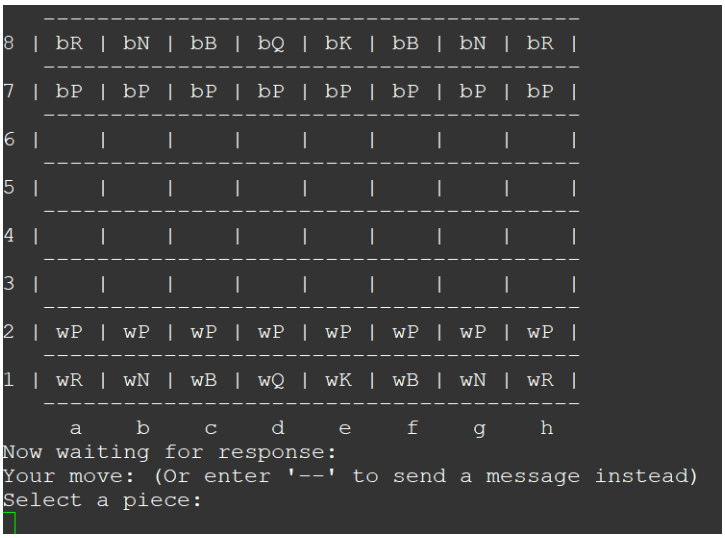

Figure 5: Blank Board

Players will also be given the option to send a message to their opponent. They can do so by typing and entering '–' and then providing the message they want to send. The message will show up on the opponent's side after the sender enters their chess move.

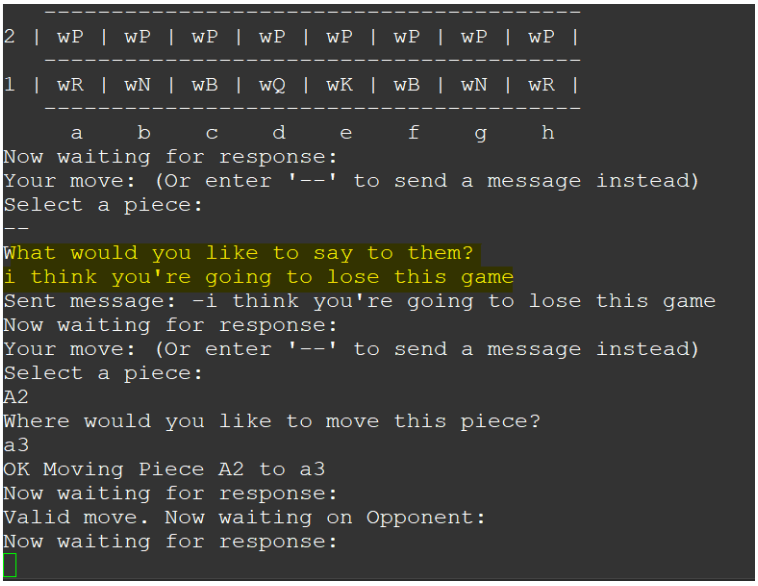

Figure 6: Example of a Client Sending a Message to Opponent

To quit at any time, a client must hold the Ctrl key while pressing the letter 'C'.

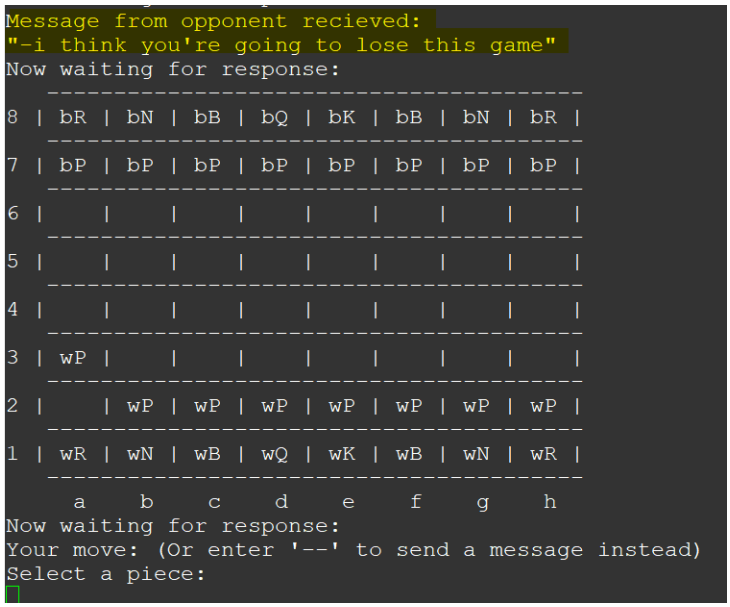

Figure 7: Example of a Client Receiving a Message

3 Chess V2 - Server

3.1 Overview of Features

For this project, in addition to adding features that allow users to better interact with one another, this project will include a server provider in the form of a simple active program that listens for and responds to client requests and stores relevant data necessary for the code's functionality. Relevant data includes information such as account information, and associated verification data such as passwords, usernames, IP addresses, and port numbers. All of these will be used in conjunction with one another to collectively create a cohesive network of user identities, all of which will be generally handled server side. Data stored is kept in a combination of binary data and persistent text file data that is accessible through privately handled functions that accesses and displays it as needed without any unnecessary input from the user.

The user side of the program requires a chat functionality that, besides needing to ensure the correct users are talking to one another, that their messages are being actively listened for in a timely manner, and like most chat user interfaces, accurately displays the correct users in chat and allows for challenges between friendly users.. The user side allows for data updates as necessary, and of course the server side supports any changes accurately.

All of this will be handled in some way through a program executable hosted on a separate machine that, over the internet, connects multiple players while each player, once installed the correct files, can play online.

3.2 Usage scenario (general, starting server)

The program itself has an executable command that can be run like any program, and runs off the local machine to become a hosting server. Once booted up, the server will begin creating a number of sockets in which users can communicate with it, and binding them to ports on the machine, each of which then go on to listen to requests and communicate data. This is what listens for client side instigation as the program then goes on to verify users or register them, then play games amongst one another.

3.3 Logging in/registering

Upon connecting to the server successfully, the program will prompt the user to either log in or sign up, at which point the server will then be actively engaging with the user through an established port. From there, the server will have then opened a socket to respond to and process data with. The inputted user data will then be run through a database of established users, or appended to it, as appropriate, and then further checked against their associated password, kept in the same file and synchronously updated.

From the server side, any administrators managing the provider will have access to the ports and open sockets that stay open to queries sent from potential players. Any changes to it are reflected by messages that mark client-server communication, as well as errors. For example, a typical server communication is detailed below:

```
[team19@zuma Chess_V2.0_src]$ ./bin/HeroChessV2_Server 11920
./bin/HeroChessV2_Server: Waiting for connections at port 11920..
.
Clock: Sun Jun  6 18:21:23 2021 -Waiting on REQUESTING BOARD requ
est
Requesting board request receieved: REQUESTING_BOARD
Player 1 fd: 4
Player 2 fd: 5
Sending turn decision to player 2. Code: BLACK_WHITE
Receiving responses:
Got W from User 2
Sending turn decision to player 1. Code: BLACK_WHITE
Receiving other player's response:
Got B from User 1
Sending respective colors:
Player 1:B
Player 2:W
Sending color decision to Player 2. Code: YOU_ARE_W
Recieved '' instead of OK4
Sending color decision to Player 1. Code: YOU_ARE_B
Recieved '' instead of OK4
Found a player!
Player 1 fd: 4
Player 2 fd: 5
```

Figure 8: Sample Server Log

3.4 Detailed description of chess game integration

The Chess Game runs primarily on the server but has checks as well as client side. Some of the appropriate game checks, especially for legality and simple checks like bounds, run on the client side to help reduce runtime and stress on the server end. However, the game is ultimately run on the server to ensure a synchronous multiplayer experience.

On the client ends of the program, there are minimal running processes, mostly focusing on taking in valid inputs and sending them to the server. On the server end of it, the program looks to be playing an ordinary game of chess, much like with the single player mode, but the moves are fed in directly to the code as pre-processed data from two different machines. The example below demonstrates a typical view of the program from server side:

```
Sending message to user: -i think you're going to lose this game
Sending string: 'PRINT_BOARD'
Now sending Board data: wR  wP      bPbRwNwP        bPbNwBwP
  bPbBwQwP        bPbQwKwP        bPbKwBwP        bPbBwNwP
bPbNwRwP        bPbR
Waiting for ok 2...
Now requesting move
Now waiting for move:
Receieved move: +C7C5
Sending to client: VALID_MOVE
Found a player!
Player 1 fd: 4
Player 2 fd: 5
   ---------------------------------------
8 | bR | bN | bB | bQ | bK | bB | bN | bR |
   ---------------------------------------
7 | bP | bP |    | bP | bP | bP | bP | bP |
   ---------------------------------------
6 |    |    |    |    |    |    |    |    |
   ---------------------------------------
5 |    |    | bP |    |    |    |    |    |
   ---------------------------------------
4 |    |    |    |    |    |    |    |    |
   ---------------------------------------
3 | wP |    |    |    |    |    |    |    |
   ---------------------------------------
2 |    | wP | wP | wP | wP | wP | wP | wP |
   ---------------------------------------
1 | wR | wN | wB | wQ | wK | wB | wN | wR |
   ---------------------------------------
     a    b    c    d    e    f    g    h
Sending string: 'PRINT_BOARD'
Now sending Board data: wR  wP      bPbRwNwP        bPbNwBwP    b
P    bBwQwP        bPbQwKwP        bPbKwBwP        bPbBwNwP
  bPbNwRwP        bPbR
Waiting for ok 2...
Now requesting move
Now waiting for move:
```

Figure 9: Sample Server Log

Error Messages

The error messages of this program are labeled by a 3 digit error code, each digit denoting a different aspect of the error. The first digit denotes where the error occurred, the second digit typically corresponds to a class of errors that occurred in that location, and the last digit denotes the specific error in that class. Below are detailed some common denominations for this error labeling.

List of common 4XX errors
400 Bad Request - The server could not understand the command due to invalid syntax.
401 Unauthorized - the current user lacks valid authentication credentials for the target resource.
403 Forbidden - The clients does not have access rights to the content
404 Not Found - The server cannot find the requested resource. Links that lead to a 404 page are often called broken or dead links and can be subject to link rot.
408 Request Timeout - This response is sent on an idle connection by some servers, even without any previous request by the client. It means that the server would like to shut down this unused connection.
List of common 5XX errors
502 Bad Gateway - This error response means that the server, while working as a gateway to get a response needed to handle the request, got an invalid response.
503 Service Unavailable - The server is not ready to handle the request. Common causes are a server that is down for maintenance or that is overloaded.
504 Gateway Timeout - The server is acting as a gateway but cannot get a response in time.
511 Network Authentication Required - This status code indicates that the client needs to authenticate to gain network access.
In General, the first digit 4 corresponds to a client side error, meaning some issue from the user end, and the first digit 5 corresponds to a server side error, meaning some issue from the administrative end.
Other status codes include 1XX, 2XX, and 3XX, however these are typically not relevant to errors and are not supposed to be seen by users.

Copyright

Index