MULTIPLAYER CHESS

Software Specification

Version 1.1

By:

Electric Lizards

UC Irvine

Bokor, Jacob	jbokor@uci.edu
Lam, Rayi	rayil@uci.edu
Nghi Nguyen	nghihn2@uci.edu
Shum, Ryan	rwshum@uci.edu
Wang,Lichen	lichew1@uci.edu

Development Glossary **1**

Software Architecture Overview **3**

Main Data Types and Structures **3**

Major Software Components **3**

Module Interfaces **3**

Overall Program Control Flow **4**

Installation **5**

System Requirements, Compatibility **5**

System Requirements **5**
- Compatibility 5
- Setup and Configuration 5
- Building, Compilation, Installation 5
- Detailed description of data structures 6
  - Critical snippets of source code 6
- Detailed description of functions and parameters 7
- Detailed description of input and output formats 8
  - Messaging Format (in game while playing) 8
  - Move Format 8
  - Login Format 8
  - Start Game Format 8

Development plan and timeline **9**
- Partitioning of tasks 9
- Team member responsibilities 9

Back Matter **10**
- Copyright 10
- References 10
- Index 10

Development Glossary

AI: Artificial intelligence; an in-game entity that uses computer code to make decisions.

Bug: An error in a computer program.

Datatype: An attribute of data which tells the compiler or interpreter how the programmer intends to use the data.

Struct: In C programming, a struct (or structure) is a collection of variables (can be of different types) under a single name.

String: An array of characters as a variable.

Pointer: a variable whose value is the address of another variable; a link.

Socket: A socket is one endpoint of a two-way communication link between two programs running on the network.

Host: A host (also known as "network host") is a computer or other device that communicates with other hosts on a network. Hosts on a network include clients and servers -- that send or receive data, services or applications.

1. Software Architecture Overview

1.1. Main Data Types and Structures

- Board - 2D string array
- Username - char pointer
    - Usernames are stored in a text file for the server to read from
- User's password - char pointer
    - Passwords are stored in a text file for the server to read from
- Message - char pointer

1.2. Major Software Components

- main.c
- server.c
- client.c
- networking.h, networking.c
- board.h, board.c

Refer to the Chess version 1.0 software specifications for the remaining software components.

1.3. Module Interfaces

1.3.1. main.c

1.3.1.1. server

1.3.2. account.c

1.3.2.1. Handle accounts

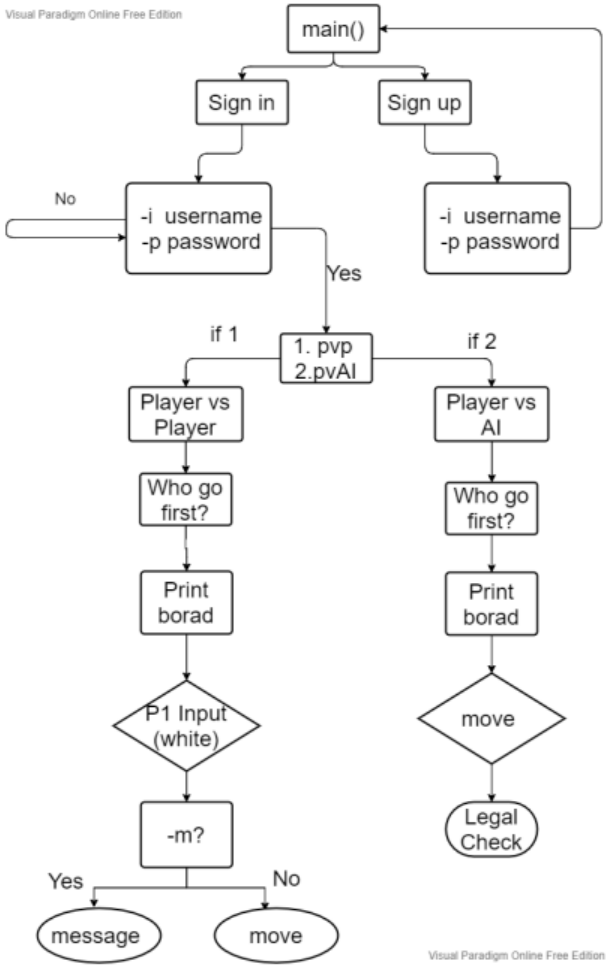
1.3.3. testclient.c

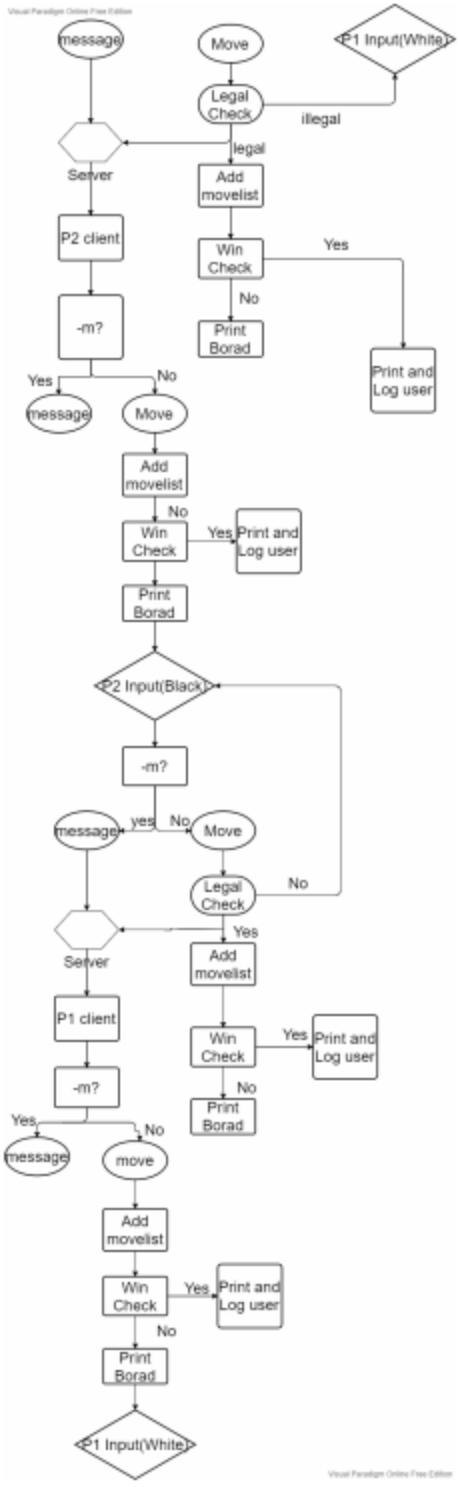
1.3.3.1. client

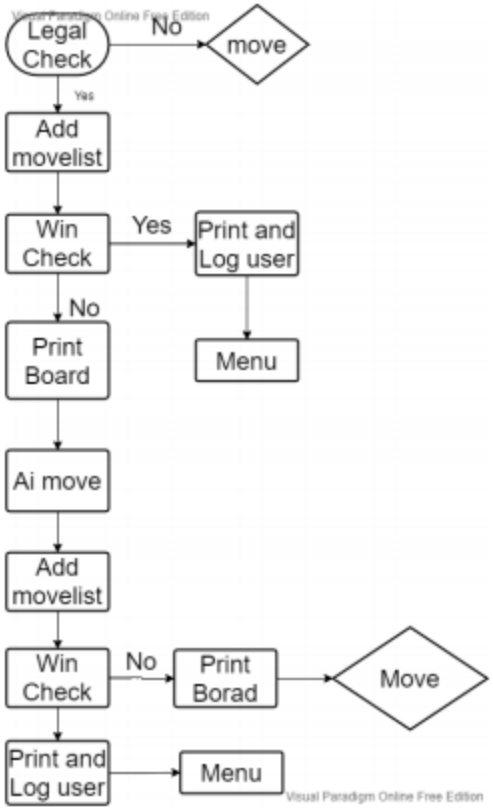
Refer to the Chess version 1.0 software specifications for the remaining module interfaces.

1.4. Overall Program Control Flow

1. Display login menu for users to log in
    a. Login
    b. Register
2. Display the main game menu for users to play multiplayer chess with the first option
    a. Player vs. Player
    b. Player vs. AI
    c. Exit
3. Determine which player goes first or second
4. Print the board
5. White player inputs move or message
6. Check for input error
7. Check if white side has a check or checkmate
8. Print the board
9. Black player inputs move or message
10. Check for input error
11. Check if black side has a check or checkmate
12. Repeat steps 4 through 11
13. Print out the winner
14. Save chess game replay to a text file
15. Exit back to main game menu (Step 2)




message
Move
P1 Input(White)
Legal Check
illegal
legal
Server
Add movelist
P2 client
Win Check
Yes
No
Print Borad
-m?
Print and Log user
Yes
No
message
Move
Add movelist
No
Win Check
Yes
Print and Log user
Print Borad
P2 Input(Black)
-m?
message
yes
No
Move
Legal Check
No
Yes
Server
Add movelist
P1 client
Win Check
Yes
Print and Log user
-m?
No
Print Borad
Yes
No
message
move
Add movelist
Win Check
Yes
Print and Log user
No
Print Borad
P1 Input(White)


Legal Check
No
move
Yes
Add movelist
Win Check
Yes
Print and Log user
No
Print Board
Menu
Ai move
Add movelist
Win Check
No
Print Borad
Move
Print and Log user
Menu

2. Installation

2.1. System Requirements, Compatibility

2.1.1. System Requirements

Operating System: Linux
Storage Requirements: 10mb

2.1.2. Compatibility

Linux EECS Server

2.2. Setup and Configuration

The entirety of the program will be within a tarball file (.tar.gz). Once extracted, the program will be in a folder called ./chessgame/ located at the location where the tarball was extracted. Navigate to the root directory of the program and proceed to the following step.

2.3. Building, Compilation, Installation

Once in the root directory, running the “make” command will compile and combine the modules files into one executable file. The makefile will compile the program in compliance with the ANSI C99 Standard for all of the modules.

After compiling, the generated executable can be run with command “ ./chessgame “.

3. Documentation of Packages, Modules and Interfaces

3.1. Detailed description of data structures

3.1.1. Critical snippets of source code

A text file will be created or does exist holding all users' account information in the server. The text file "serverAccounts.txt" will contain user account information on one line separated by a space. The format is (userID, username, password, friends[0], friends[1], ...). Each username and password will be a char pointer. A new line will separate each user's information. The text file is not developed. For example, serverAccounts.txt may look like
"

0 username password 1 2

1 username2 password2 0 2

2 username3 password3 0 1
"

For each account that is created, the account will hold the userID, the username, password, socket number, opponent's socket number, and a list of the user's friends. The existing accounts will be held in a linked list.

```
typedef struct accountEntry {
	int userID;
	char user[20];  // username
	char pass[20];  // password
	int socket;  // 0 if not connected/online
	int opponentSocket; // 0 if not playing against an opponent
	int friends[10];  // storing another user's userID
	account* next;
	account* prev;
} account;

typedef struct accountList {
	account* first;
	account* last;
	int length;
} accountList;
```

For remaining data structures, refer to the chess version 1.0 software specifications.

3.2. Detailed description of functions and parameters

Client Side Functionality

- testclient.c
    - int sendData(char *data)
        - Parameter: string message of the client.
        - Return: 1 or 0
        - Process: Send data to Server
    - int receiveData(char *data)
        - Parameter: string message of the server.
        - Return: 1 or 0
        - Process: Receive data from server.
    - int createAccount(char *user, char *pass)
        - Parameter: string username and password
        - Return: 1 or 0 for whether an account is created successfully.
        - Process: Writes a new username and password for a user into a text file holding all the usernames and passwords of all the users on the server.
    - void login()
        - Parameter: None
        - Return: None
        - Process: The login function uses the checkAccount() function to check if a user has entered in the correct account user information.
    - void register()
        - Parameter: None
        - Return: None
        - Process: The register function uses the createAccount() function to create a new account for a user in the server.

- board.h, board.c
    - int handleOnlineMove(char board[8][8][2])
        - Parameter: gameBoard array. It holds the current state of the board in where all the chess pieces are.
        - Return: 1 or 0
        - Process: calls receiveData(), handles messages and online player's move. Determines if the string is a message or a move. If it is a move, the function will convert the string into the respective coordinates and update the board as well.

Server Side Functionality

- server.h, server.c
    - int checkAccount (char* user, char* pass)
        - Parameter: string username and password

        - Return: 1 or 0 for whether there is an existing account or not
        - Process: Checks for the account against known accounts and returns 1 if successful, else 0.
    - Int serverInit(short port, int backlog)
        - Parameter: port number, and backlog
        - Return: server socket
        - Process: initialize server, setting up
    - int acceptConnection(int serverSocket)
        - Parameter: server socket number
        - Return: client socket
        - Process: accepts all connection of the server
    - int doStuff(int clientSocket, accountList *list)
        - Parameter: client socket number and the list of accounts
        - Return: 0
        - Process: Handle parsing of data sent and sends data to other client
- main.h, main.c
    - int main()
        - Parameter: None
        - Return: 0
        - Process: The main will start the server and handle the client inputs and returns them.
- account.h, account.c
    - accountList *createAccountList()
        - Parameter:none
        - Return: a new account linked list
        - Process: create new account linked list
    - void deleteAccountList(accountList *list)
        - Parameter: account list
        - Return: none
        - Process: delete an account list
    - account *createAccount()
        - Parameter: none
        - Return: a new account
        - Process: Create a new account linked list
    - void deleteAccount(account *acct)
        - Parameter: account
        - Return: none
        - Process: Delete the account passed in
    - void appendAccount(accountList *list, account *acct)
        - Parameter: Account linked list and account
        - Return: none
        - Process: Add an account to the end of the account list
    - account *getAccount(accountList *list, int currentPointer)

- Parameter: Account linked list and integer current pointer
- Return: an account
- Process: Acquire an account with a number pointing to the position

For remaining functions and parameters, refer to the chess version 1.0 software specifications.

3.3. Detailed description of input and output formats

3.3.1. All data sent to the server will be either a valid move or a valid message, all error handling will be client side.

3.3.2. Messaging Format (in game while playing)

3.3.2.1. -m <message text>

3.3.3. Move Format

3.3.3.1. Same as original format: ie: A2A3

3.3.4. Login Format

3.3.4.1. -l <username> <password>

3.3.5. Start Game Format

3.3.5.1. -g <username of the other player>

4. Development plan and timeline

4.1. Partitioning of tasks

- Client and server
- Login and Registration
- Send and Receive data
    - Messages and game input moves

4.2. Team member responsibilities

5/23-5/30	**Jacob-** Login System **Lichen -** menu update **Ryan -** Login system **Nghi -** Send and Receive inputs **Rayi -** Write test client and server

Back Matter

Copyright

References

Contributors: Jacob Bokor
Rayi Lam
Ryan Shum
Nghi Nguyen
Lichen Wang

Affiliation: Henry Samueli School of Engineering
University of California, Irvine

Index

board.h, board.c 3, 6
Char pointer 3
client.h, client.c 3, 6
Linux 5
Login 3, 4
main.h, main.c 3, 7
Networking.h, networking.c 3, 6
Register 3, 4
server.h, server.c 3, 7
Text File 3, 6