User Manual of Chinese Dark Chess Client

Jr-Chang Chen, Gang-Yu Fan and Yao-Rong Yang

This document includes three parts. We provide the client program to connect to the Chinese dark chess server. Section 1 is the environment settings in **Ubuntu** for the client package. Section 2 describes the codes you have to write in your game-playing program to fit the client. Section 3 is our contact information.

1. ENVIRONMENT

The demo game-playing program is a random-play program, called **myai** in the file ”search”. You can run two copies and connect to the chess server to test the connection.

1.1 Environment Setting

- The Chinese dark chess interface (**CDC_interface.zip**) includes an execution file ”**DarkChess_linux**”, a library ”**GameDLL.so**” and a folder ”**Search**”, as shown in Figure 1.

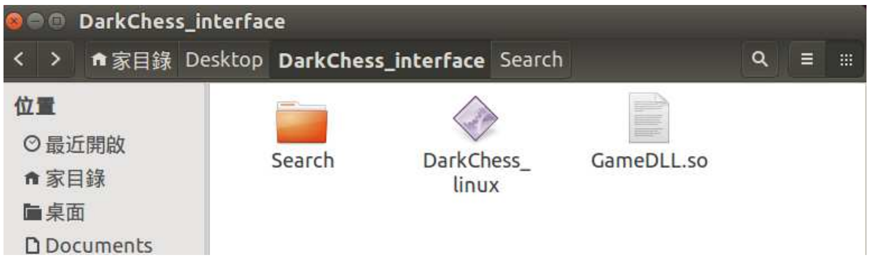


Figure 1:

- Your game-playing program name has to be renamed as ”**search**”, and then replace the ”**search**” execution file in the ”**Search**” folder by your program, as shown in Figure 2.

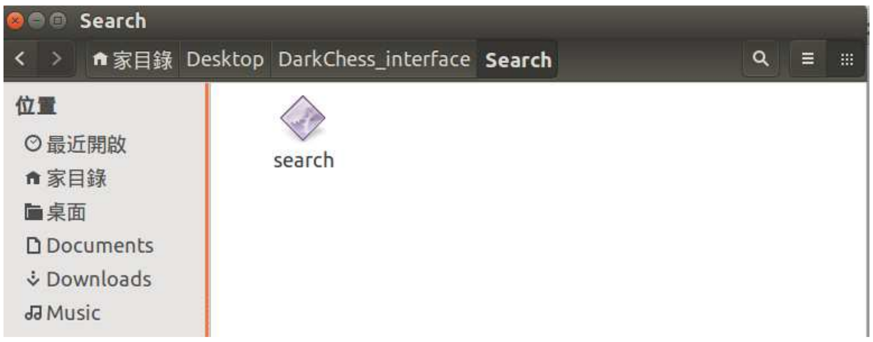
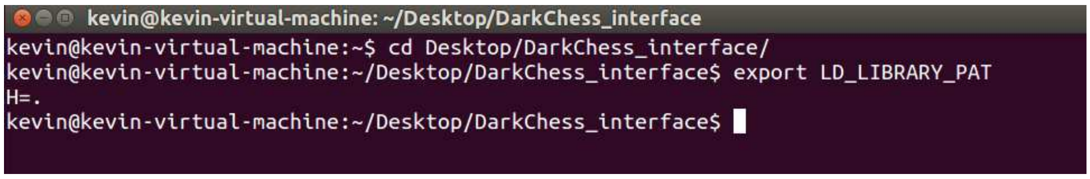


Figure 2:

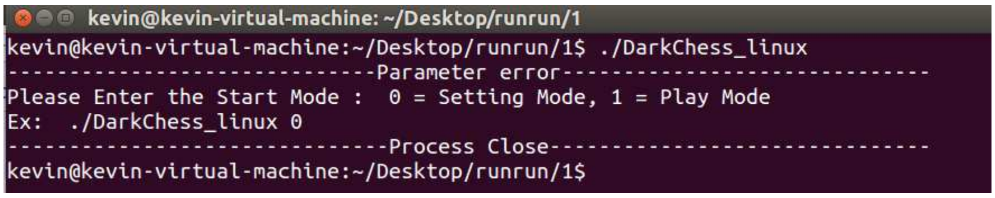
- Export the library path to the folder that includes ”**GameDLL.so**”. For example, in the following figure, ”**GameDLL.so**” is in the current folder, so we type the command ”**export LD_LIBRARY_PATH=.**”, as shown in Figure 3.



```
kevin@kevin-virtual-machine: ~/Desktop/DarkChess_interface
kevin@kevin-virtual-machine:~$ cd Desktop/DarkChess_interface/
kevin@kevin-virtual-machine:~/Desktop/DarkChess_interface$ export LD_LIBRARY_PAT
H=.
kevin@kevin-virtual-machine:~/Desktop/DarkChess_interface$ 
```

Figure 3:

- To run your game-playing program and connect to the server, you have to execute "**DarkChess_linux**". There are two modes by typing the command "**DarkChess_linux mode**". If **mode = 0**, we enter the **Setting Mode** (see subsection 1.2); and if **mode = 1**, we enter the **Play Mode** (see subsection 1.3), as shown in Figure 4.

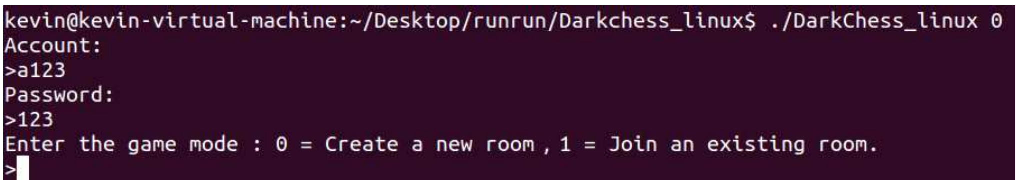


```
kevin@kevin-virtual-machine: ~/Desktop/runrun/1
kevin@kevin-virtual-machine:~/Desktop/runrun/1$ ./DarkChess_linux
------------------------------Parameter error------------------------------
Please Enter the Start Mode :  0 = Setting Mode, 1 = Play Mode
Ex:  ./DarkChess_linux 0
-------------------------------Process Close-------------------------------
kevin@kevin-virtual-machine:~/Desktop/runrun/1$
```

Figure 4:

1.2 Setting Mode

Input the command "**DarkChess_linux 0**" to start the Setting Mode, as shown in Figure 5.



```
kevin@kevin-virtual-machine:~/Desktop/runrun/Darkchess_linux$ ./DarkChess_linux 0
Account:
>a123
Password:
>123
Enter the game mode : 0 = Create a new room , 1 = Join an existing room.
>
```

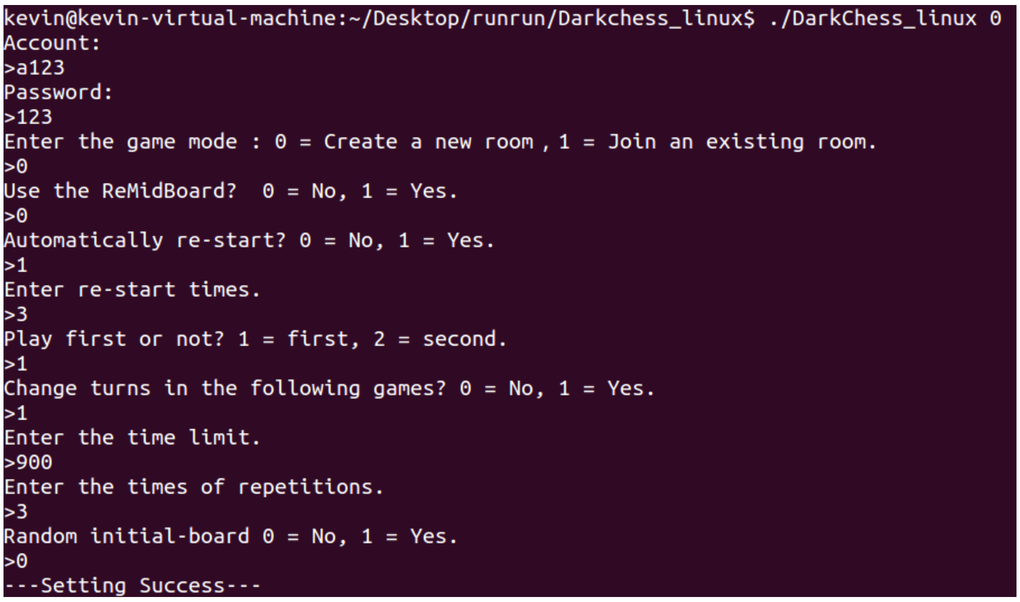
Figure 5:

Create a Room

Input the game mode "**0**" to create a room. Then, you set the room information by the following message as shown in Figure 6.

- Testing accounts are "a0", "a1", ..., and "a999".
- The password for all accounts is "123".
- **ReMidBoard** must be "0".
- If you want to play multiple games and re-start automatically, set **automatically re-start** to "1"; and "0" otherwise.
- Set **re-start times** if **automatically re-start** is "1". For example, "3" if three games will be played.
- **Play first or not?** Input "1" to be the first player and "2" to be the second player.
- **Change turns in the following games?** Input "1" to mean playing first and playing second in turns if multiple games are played. Input "0" to mean that you always play first or second you select in "**play first or not?**" in all games.

- **Time limit** is the total time you have in a game. For example, ”900” means you have to finish a game within 900 seconds.
- **Times of repetitions**. Input 3 to mean the game is judged as a draw if the same board is repeated three times.
- **Random initial-board** must be ”0”.



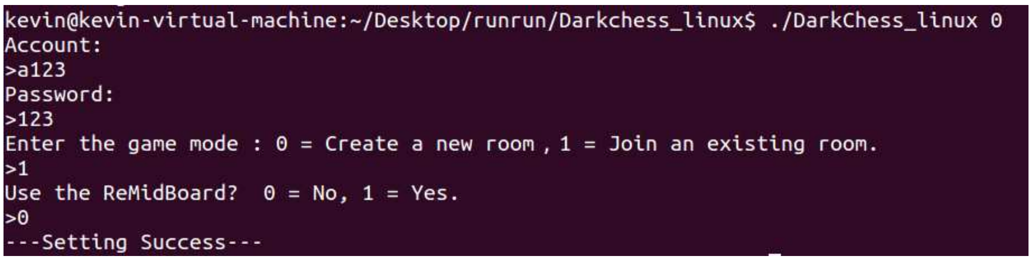
```
kevin@kevin-virtual-machine:~/Desktop/runrun/Darkchess_linux$ ./DarkChess_linux 0
Account:
>a123
Password:
>123
Enter the game mode : 0 = Create a new room , 1 = Join an existing room.
>0
Use the ReMidBoard?  0 = No, 1 = Yes.
>0
Automatically re-start? 0 = No, 1 = Yes.
>1
Enter re-start times.
>3
Play first or not? 1 = first, 2 = second.
>1
Change turns in the following games? 0 = No, 1 = Yes.
>1
Enter the time limit.
>900
Enter the times of repetitions.
>3
Random initial-board 0 = No, 1 = Yes.
>0
---Setting Success---
```

Figure 6: Create a room.

Join a Room

Input the game mode ”**1**” to join a room as shown in Figure 7. The room information is set by the room owner as described above.

- **ReMidBoard** must be ”0”.



```
kevin@kevin-virtual-machine:~/Desktop/runrun/Darkchess_linux$ ./DarkChess_linux 0
Account:
>a123
Password:
>123
Enter the game mode : 0 = Create a new room , 1 = Join an existing room.
>1
Use the ReMidBoard?  0 = No, 1 = Yes.
>0
---Setting Success---
```

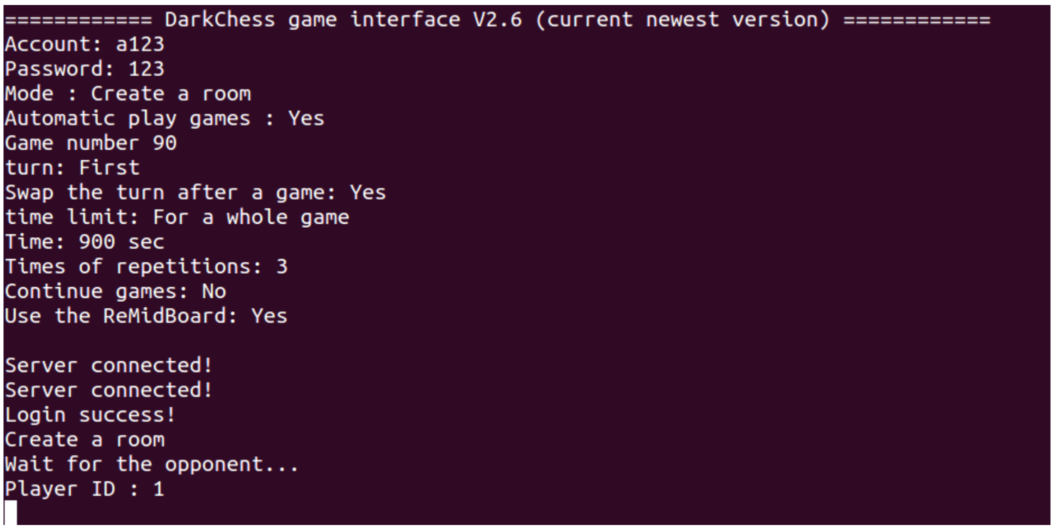
Figure 7: Join a room.

1.3 Play Mode

Input the command ”**DarkChess_linux 1**” to start the Play Mode.

Wait for the Opponent

If you select the ”**create a room**” in the setting mode, you have to wait for an opponent to join the room, as shown in Figure 8.

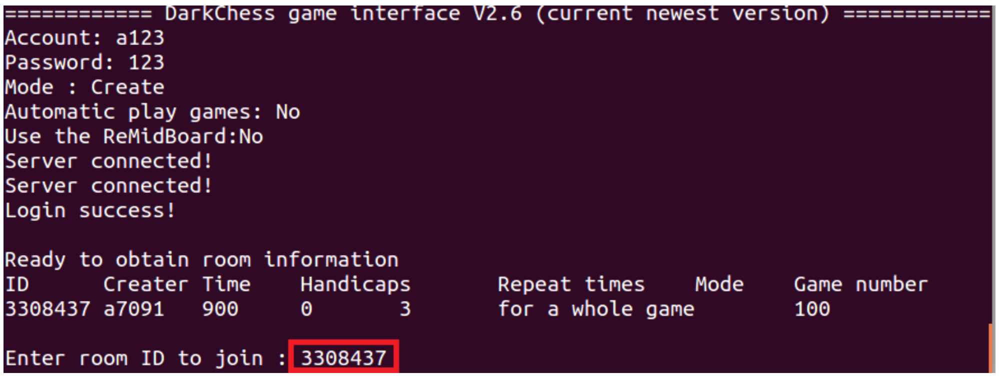


```
============ DarkChess game interface V2.6 (current newest version) ============
Account: a123
Password: 123
Mode : Create a room
Automatic play games : Yes
Game number 90
turn: First
Swap the turn after a game: Yes
time limit: For a whole game
Time: 900 sec
Times of repetitions: 3
Continue games: No
Use the ReMidBoard: Yes

Server connected!
Server connected!
Login success!
Create a room
Wait for the opponent...
Player ID : 1
```

Figure 8: Wait for the opponent.

Join a Room

If you select the "**join a room**" in the setting mode, you have to select the "**room ID**" of the opponent, as shown in Figure 9.

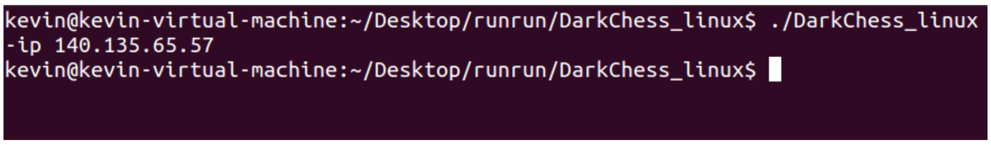


```
============ DarkChess game interface V2.6 (current newest version) ============
Account: a123
Password: 123
Mode : Create
Automatic play games: No
Use the ReMidBoard:No
Server connected!
Server connected!
Login success!

Ready to obtain room information
ID      Creater Time    Handicaps       Repeat times    Mode    Game number
3308437 a7091   900     0       3       for a whole game        100

Enter room ID to join : 3308437
```

Figure 9: Join a room.

1.4 Set Server IP

Input the command "**DarkChess_linux -ip** <*ip_address*>" to start to set the server IP, as shown in Figure 10.

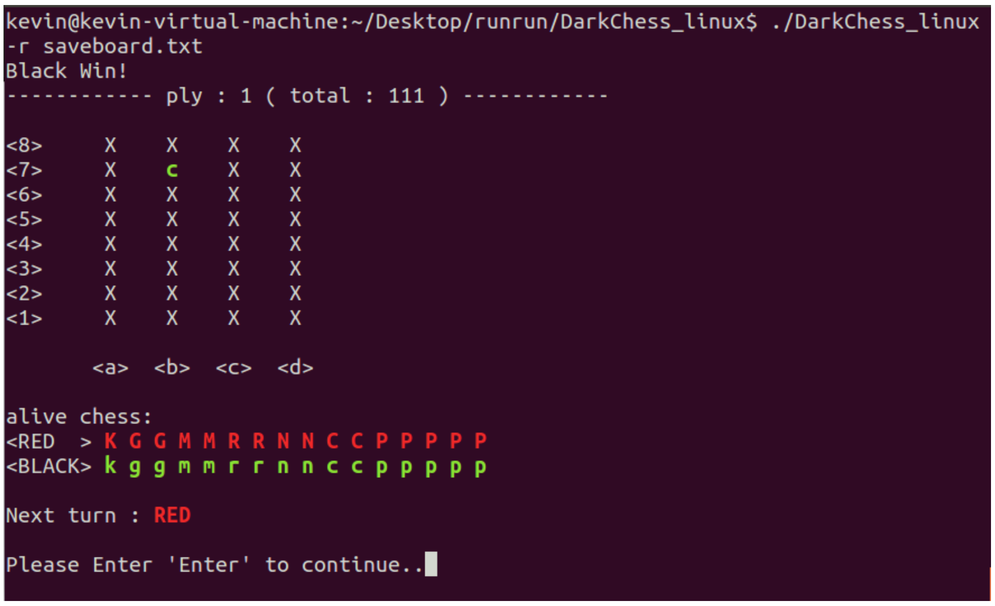


```
kevin@kevin-virtual-machine:~/Desktop/runrun/DarkChess_linux$ ./DarkChess_linux
-ip 140.135.65.57
kevin@kevin-virtual-machine:~/Desktop/runrun/DarkChess_linux$ 
```

Figure 10: Server IP setting.

1.5 Read Game Record Mode

Input the command "**DarkChess_linux -r** <*game_record_file*>" to start to read the history of the game saved in *game_record_file*, as shown in Figure 11.

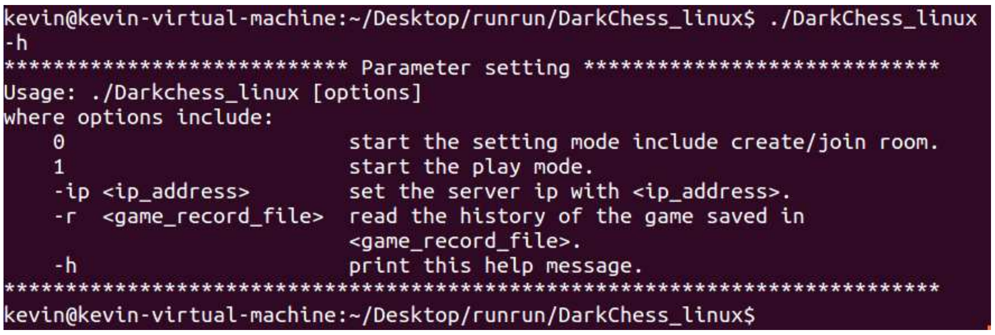


```
kevin@kevin-virtual-machine:~/Desktop/runrun/DarkChess_linux$ ./DarkChess_linux
-r saveboard.txt
Black Win!
------------ ply : 1 ( total : 111 ) ------------

<8>     X    X    X    X
<7>     X    c    X    X
<6>     X    X    X    X
<5>     X    X    X    X
<4>     X    X    X    X
<3>     X    X    X    X
<2>     X    X    X    X
<1>     X    X    X    X

       <a>  <b>  <c>  <d>

alive chess:
<RED  > K G G M M R R N N C C P P P P P
<BLACK> k g g m m r r n n c c p p p p p

Next turn : RED

Please Enter 'Enter' to continue..
```

Figure 11: Read a game record.

1.6 Help Mode

Input the command "**DarkChess_linux -h**" to list the descriptions of all commands, as shown in Figure 12.



```
kevin@kevin-virtual-machine:~/Desktop/runrun/DarkChess_linux$ ./DarkChess_linux
-h
**************************** Parameter setting *****************************
Usage: ./Darkchess_linux [options]
where options include:
    0                        start the setting mode include create/join room.
    1                        start the play mode.
    -ip <ip_address>         set the server ip with <ip_address>.
    -r  <game_record_file>   read the history of the game saved in
                             <game_record_file>.
    -h                       print this help message.
*****************************************************************************
kevin@kevin-virtual-machine:~/Desktop/runrun/DarkChess_linux$
```

Figure 12: Help mode.

2. PROTOCOL

The package **CDC_client.zip** includes:

- ClientSocket.h, ClientSocket.cpp, Protocol.h and Protocol.cpp - the client protocol
- myai.h, myai.cpp - the random-playing program in Section 1
- main.cpp, main_clear.cpp - to connect myai and the client

To connect your game-playing program (assume called YourAI.cpp) to the client, you only have to modify a few lines in **main.cpp**. These lines are marked by ”**// todo:**” in **main_clear.cpp**. Then, compile with the command

```
    g++ -o search main.cpp Protocol.cpp ClientSocket.cpp YourAI.cpp
```

The newly compiled file, **search**, should overwrite the **search** file in the **Search** folder. Thus, you can connect to the server with your program by execute **DarkChess_linux**, as described in Subsection 1.1.

The class and functions used are described in the following subsections.

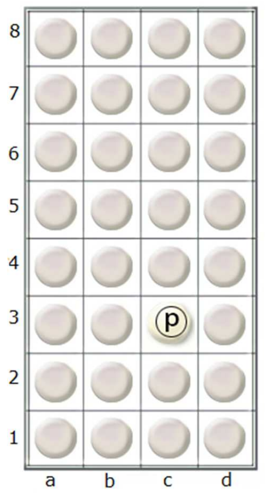
```
enum PROTO_CLR {PCLR_RED, PCLR_BLACK, PCLR_UNKNOWN};

class Protocol
{
public:
    Protocol();
    ˜Protocol();
    void init_protocol(const char *ip, const int port);
    void init_board(int piece_count[14], char current_position[32], struct History &history,
                    int &time);
    void get_turn(bool &turn, PROTO_CLR &color);
    void send(const char src[3], const char dst[3]);
    void send(const char move[6])
    void recv(char move[6], int &time);
    PROTO_CLR get_color(const char move[6]);
};
```

2.1 init_protocol

```
void init_protocol(const char *ip, const int port);
```

Connect to the server by inputting the **ip** and **port** of the server via command line or GUI interface. **init_protocol** must be called in the beginning.

```
#include "protocol.h"
int main(int argc, char **argv)
{
    Protocol protocol;
    switch (argv) {
    case 3:
        if (!protocol.init_protocol(argv[1], atoi(argv[2]))) return 0;
        break;
    }
    ...
    return 0;
}
```


(a)

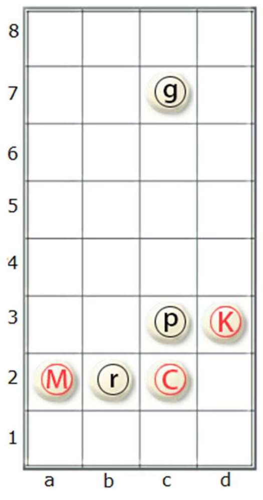

(b)

Figure 13: Notations of piece kinds and the board.

2.2 Notations

Piece kinds

- The letters ’**K**’, ’**G**’, ’**M**’, ’**R**’, ’**N**’, ’**C**’, ’**P**’ represent the king, guard, minister, rook, knight, cannon, pawn of red pieces.
- The letters ’**k**’, ’**g**’, ’**m**’, ’**r**’, ’**n**’, ’**c**’, ’**p**’ represent those of the black pieces.
- The letter ’**X**’ represents an unrevealed/hidden piece.
- The letter ’**-**’ represents an empty square.

Board configuration

- Letters **a** to **d** from left to right for columns
- Numbers **1** to **8** bottom up for rows

For example, in Figure 13(a), a black pawn (labelled by **p**) is on square **c3**, and an unrevealed piece (labelled by **X**) is on square **b4**. In Figure 13(b), square **a4** is empty and is labelled by **-**, and the red king (labelled by **K**) is on square **d3**.

2.3 struct History

```
struct History{
    char** move;
    int number_of_moves;
};
```

The meanings of **move** and **number_of_moves** are as follows.

- **move**:
  If the ply is a move or capture (e.g., the 2nd ply is moving a piece from a3 to a4), then **move[2]** = ”a3-a4”. If the ply is a flip (e.g., the 2nd ply is flipping a red king on c2), then **move[2]** = ”c2(K)”.

- **number_of_moves**:
  The total number of ply.
  For example, if **number_of_moves** = 3, we use move[0], move[1] and move[2].

If the program resumes to play, you have to restore the history as follows (in the TODO part).

```
struct History history;
protocol->init_board(piece_count, current_position, &history);
for (int i = 0; i < history.number_of_moves; i++) {
    // TODO: restore the history to your program.
}
```

2.4 init_board

```
void init_board(int piece_count[14], char current_position[32], struct History &history,
                int &time);
```

After calling **init_board**, you get the initial board settings as follows.

- **piece_count[14]**: The number of pieces that are alive of 14 piece kinds.
- **current_position[32]**: The value of each element represents the piece kind on the board. Notations are described in Subsection 2.2.
- **history**: The history of the game.
- **time**: The remaining time of my turn. (millisecond)

For example, in Figure14, two red ministers (M), a black knight (n), and a black king (k) are revealed. And a red pawn and a black cannon are captured. The values of **piece_count[14]** and **current_position[32]** are:

```
    piece_count[14] = {1, 2, 2, 2, 2, 2, 4, 1, 2, 2, 2, 2, 1, 5}
    current_position[32] = "XnXXX-XXXXXXXXXXXM-XXXkXMXXXXXXX"
```

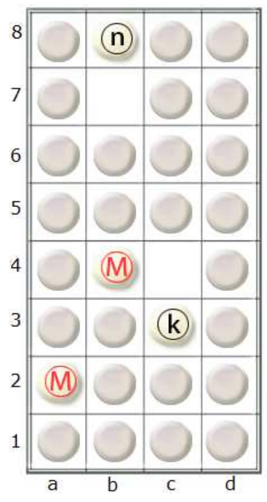


Figure 14: current_position[32].

2.5 get_turn

```
enum PROTO_CLR {PCLR_RED, PCLR_BLACK, PCLR_UNKNOWN};
void get_turn(bool &turn, PROTO_CLR &color);
```

turn = true means the first player, and **turn = false** means the second player.
The value of **color** are **PCLR_RED** as red, **PCLR_BLACK** as black, and **PCLR_UNKNOWN** as unknown.

If the game is played when all pieces are unrevealed/hidden, you get your turn and color is set to **PCLR_UNKNOWN**.
If the game is played from midgame, you get your turn and color that may be set to **PCLR_RED** or **PCLR_BLACK**.

2.6 send

You may choose one of the following two functions to send your ply.

```
void send(const char src[3], const char dst[3]);
```

If the ply is a move or capture (e.g., move a piece from d5 to c5), then **src** = ”d5” and **dst** = ”c5”.
If the ply is a flip (e.g., flip a piece on d5), then **src** = ”d5” and **dst** = ”d5”.

```
void send(const char move[6]);
```

If the ply is a move or capture (e.g., move a piece from d5 to c5), then **move** = ”d5-c5”.
If the ply is a flip (e.g., flip a piece on d5), then **move** = ”d5-d5”.

2.7 recv

```
void recv(char move[6], int &time);
```

move is the ply that the opponent played and sent to you by the server.
If the ply is a move or capture (e.g., move a piece from a3 to a4), then **move** = ”a3-a4”.
If the ply is a flip (e.g., flip a red king on c2), then **move** = ”c2(K)”.

time is the remaining time of my turn. (millisecond)

2.8 get_color

```
enum PROTO_CLR {PCLR_RED, PCLR_BLACK, PCLR_UNKNOWN};
PROTO_CLR get_color(const char move[6]);
```

This function returns the color of the flipped piece. For example,

```
PROTO_CLR color;
char move[6] = "a8(G)";
color = get_color(move);     /* color == PCLR_RED */
move[6] = "d6(p)";
color = get_color(move);     /* color == PCLR_BLACK */
```

3. CONTACT INFORMATION

If there are any unclear description about the protocol, please contact:

- Jr-Chang Chen, email: jcchen@cycu.edu.tw
- Gang-Yu Fan, email: imloed10000@gmail.com
- Yao-Rong Yang, email: kevin12345621@gmail.com

The rules and notations of Chinese dark chess are mentioned in the following articles.

- Chen, B.N., Shen, B.J., and Hsu, T.s., ”Chinese Dark Chess,” *ICGA Journal*, vol. 33, no. 2, pp. 93-106, 2010.
- Chen, J.C., Lin, T.Y., Hsu, T.s., ”Equivalence Classes in Dark Chess Endgames,” accepted by *IEEE Transactions on Computational Intelligence and AI in Games (IEEE TCIAIG)* (DOI: 10.1109/TCIAIG.2014.2317832).
- Yen, S.J, Chou, C.W., Chen, J.C., Wu, I.C., Kao, K.Y., ”Design and Implementation of Chinese Dark Chess Programs,” accepted by *IEEE TCIAIG* (DOI: 10.1109/TCIAIG.2014.2329034).