Users Manual

Reversi Server

for Reversi Server version 0.1 and later

Bai Aijun
<baj@mail.ustc.edu.cn,
baj_forever@126.com>

Xu Luping
<xuluping87@gmail.com,
kelp@mail.ustc.edu.cn>

2006-12-01

Contents

3. GPL Documents

1. Introduction

1.1. Backgrounds

This is a production of *Algorithm and it's Highly Implementations* which is an optional course of us, this course is as great as our teacher of it, but we do this not only for this, we want to enjoy the FREEDOM of programming under GNU/Linux, we also appreciate you to join us.

Reversi (also known as Othello), is a board game for two players, played on a grid of 64 squares. The game begins with each player having two stones. The players alternate turns, each adding an additional stone to the board. A valid move must capture at least one of the opponent's stones. This is done by surrounding it, either vertically, horizontally, or diagonally (or a combination of all three). When this occurs, the opponent's stones that you have surround become yours. The turn then passes to your opponent, who tries to take as many of your stones as he can. Play continues until either one player loses all their stones, or until the entire board if filled. Whoever ends up owning the most stones at the end of the game wins. < from web site >

Our purpose is to let two programs can play this game in a contest, we developed this Server as well as the Monitor. We use Client/Server framework, the Server provides a platform of the contest, the Clients connected to the server conclude the front end. There may be two kinds of Clients, Agents and Monitors. Agents included Agent-machine and Agent-human really take part in a contest. The Monitor mainly shows the contest process, but it also supports mouse events to send action message when it is took as an Agent-human. With this method, we can realize a contest with mode of machine-machine, machine-human or human-human. But we just provide a contest platform, we need you to join us to develop some Agents, thanks very much if you do so.

1.2. Reader's Guide to the Manual

With this manual we hope you will understand the method of run a contest at least, we also hope you can develop your own Agent-machine which must be a great present of us. In this manual we will do our best to tell you about this as much as possible, but the most recommended way is to READ THE FUCK SOURCE CODE <by Linus Towards>.

2. Overview

2.1. The Server

The Server mainly deal with the CONNECT from all it's Clients both Agents and Monitors< to know the details, see source file>, NOTE that the counts of Clients is indefinite we use POSIX threads to realize this. There is a model of chess board in the Server's memory too, which must be all the Client's standards. We use TCP protocol to connect with Clients as well as some information agreement <see the rules>, so we can easily run an online contest upside the web.

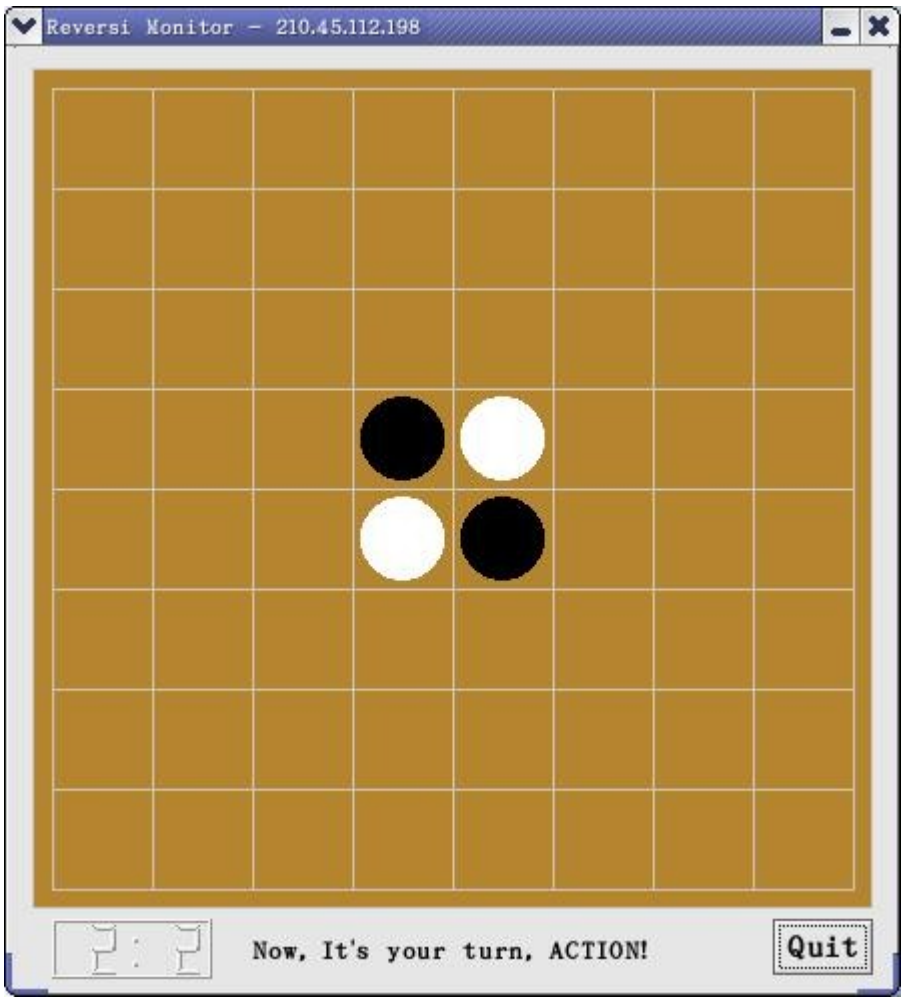
The Server is order to be existed with this manual, if not please contact to us (see the front cover). We release the Server with the source, also providing a prebuilt binary file. Complination on GNU/Linux just type make in the existed directory, by now it just support GNU/Linux platform.

2.2. The Monitor

The Monitor displays the chess board, and if it is running as a Agent-human it also deals with mouse event and message sending to the Server, actually a Monitor never knows whether he is a pure monitor or a human-controlled chess board, the Server will do this using simple method.

We develop the Monitor using Qt develop kit, supporting GNU/Linux only. If you do not know about QPL and do something illegal, we will take no responsibility.

Following is the screen shot of our Monitor, COOL!



2.3. The Rules

OK, let's talk about the rules, w hitch is of course the key part of the whole contest. TCP is used for communication of Server and Clients. We assumed that a Client never sends message to Server actively, it only REPLYs the message of Server if necessary, NOTE that some message from Server dose not need to be replied, we will talk this later. We now introduce the whole process of communication.

First of all, there is some WORDS say ed by Server and Clients, we give these words in format of C macros with some comments:

```
/** to tell Client (only be an Agent) to take an action to place chess */
#define ACTION  (char)70

/** to tell Client to wait for one time when there's no way to place for him*/
#define WAIT    (char)71
```

```
/** a flag, after it is the chess board data (see later) */
#define BOARD_DESCRIPTION (char)69

/** we don't permit continual mistakes */
#define ONE_MORE_TIME    (char)72
#define TWO_TIMES_LEFT   (char)73
#define THREE_TIMES_LEFT (char)76

/** the contest is finished, we need to tell you the result */
#define YOU_WIN (char)74
#define YOU_LOSE (char)75
#define YOU_TIE  (char)77

/** following two will only be sended to a pure monitor */
#define BLACK_WIN (char)81
#define WHITE_WIN (char)80
```

NOTE that every WORD is only one byte, we use char to realize this.

When a Client is connected with the Server, the Server will send a random character to the Client, the Client must reply with the same character, we use this simple procedure as authentication. Now if the Server thinks one Client as a legal client, then the Server will create an independent thread to communicate with it. All the thing will be done next is to maintain the good communication order. We assumed that a Client is always inactive as we sayed before. He is always waiting for message sended by Server, only if he dose need to reply to the Server he then gives a reply. For an example, the message from the Server is ACTION, then the Client will reply his action(for details, see later), but if the message is BOARD_DESCREPTION the Client will not give any reply(the Server don't wait for a reply also), we have the following table:

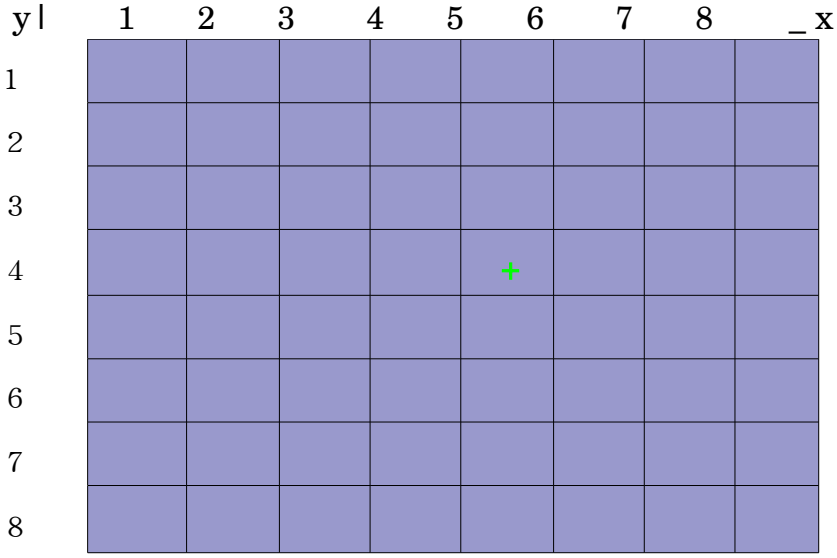
```
ACTION ..................................... NEED_TO_SEND
WAIT    ..................................... NO_NEED_TO_SEND
BOARD_DESCRIPTION  ..... NO_NEED_TO_SEND

ONE_MORE_TIME ..... NEED_TO_SEND
TWO_TIMES_LEFT ..... NEED_TO_SEND
```

THREE_TIMES_LEFT ..... NEED_TO_SEND

YOU_WIN ..... FINISHED
YOU_LOSE ..... FINISHED
YOU_TIE ..... FINISHED

NOTE that we use FINISHED to mean not only NO_NEED_TO_SEND but also the contest is finished.

In Server a chess board is a string of 66 characters as declaration of char layout[66] , layout[0] is filled with BOARD_DESCRIPTION and layout[65] the NULL, so we can easily use strcpy to translate message.
following is the chess board:



The position '+' in the board is signed as x = 5, y = 4 or pos = x + 8 * (y - 1) = 29, please remember that pos = x + 8 * (y - 1), x = (pos%8 == 0)? 8: pews%8 and
y = (pos%8 == 0)? pos/8: pos/8 + 1, because in order to reduce the length of the message taking the Client's action, we hope the Client use pos instead of x, y to description his action.

2.3. How to run a contest

To run a contest, a Server is necessary, for this you should run the Server first of all. Then Clients can be connected to the running Server. The Server will take the first and second connected Client as Agent both Agent-machine and Agent-human, the other Clients are all took as Monitors. By now we can

only run a single contest at the same time, but it is easy to changed if necessary.

3. GPL Documents

```
/**************************************************************************
 *  Copyright (C) 2006 by Bai Aijun, Xu Luping   *
   *  baj@mail.ustc.edu.cn,baj_forever@126.com *
•  *   xuluping87@gmail.com, kelp@mail.ustc.edu.cn *
 *  This program is free software; you can redistribute it and/or modify  *
 *  it under the terms of the GNU General Public License as published by  *
 *  the Free Software Foundation; either version 2 of the License, or     *
 *  (at your option) any later version.                                   *
 *                                                                        *
 *  This program is distributed in the hope that it will be useful,       *
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
*
 *  GNU General Public License for more details.                          *
 *                                                                        *
 *  You should have received a copy of the GNU General Public License     *
 *  along with this program; if not, write to the                         *
 *  Free Software Foundation, Inc.,                                       *
 *  59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 **************************************************************************/
```