

The University of Queensland
School of Information Technology and Electrical
Engineering
Semester One, 2011
COMP2303 / COMP7306 - Assignment 4
Due: 11pm 3rd June, 2011
Marks: 50
Weighting: 25% of your overall assignment mark
(COMP2303)
Version 1.1

Introduction

Your task is to write a client/server networked game of **flip**. The server component will be multithreaded. The client can be multithreaded if you wish. The server will record statistics about the performance of each player.

You will need to submit the source for both your client and server. As with prior assignments, submission will be via svn. **Do not commit any code which is not either your own work or supplied by teaching staff.**

Your assignment must comply with the C style guide available on the course website.

This is an *individual assignment*. You should feel free

to discuss aspects of C programming and the assignment specification with fellow students. You should not actively help (or seek help from) other students with the actual coding of your assignment solution. It is cheating to look at another student's code and it is cheating to allow your code to be seen or shared in printed or electronic form. You should note that all submitted code may be subject to automated checks for plagiarism and collusion. If we detect plagiarism or collusion, formal misconduct proceedings will be initiated against you. A likely penalty for a first offence would be a mark of 0 for the assignment. Don't risk it! If you're having trouble, seek help from a member of the teaching staff — don't be tempted to copy another student's code. You should read and understand the statements on student misconduct in the course profile and on the school website: <http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism>

Server

The server (called **flipserv**) will be started in the following way.

```
./flipserv dim maxgames port logfile
```

Once started, the server will wait for connections from clients on the specified port. Clients will be assigned to

games in the order they connect. That is, the first two players will be placed in the first game. The first player to connect will play **O**.

If a player disconnects before the start of their game, they are not replaced. For example, Player 1 connects and then disconnects. Player 2 and 3 connect. Player 2 will be placed in the first game, while Player 3 will be placed in the second game to wait for another player.

The second parameter controls how many games can be running simultaneously. If a client connects but the maximum number of games are already running that client will be disconnected and will not occupy a space in a game.

If the logfile exists, then any new text should be written to the end of the file.

Client

The client (called **netflip**) will be run in the following way.

```
./netflip name port [host]
```

Where *name* is a string to refer to this player and *port* and *host* give the location of the server. Each client deals with the input and output for one player. To play a game you will need to run two clients and a server.

If the host is not specified connect to the machine the

client is running on.

Except where described in this specification, the I/O to and from the user will be the same as in Assignment 3. The client will not support saving games or the automatic players from previous assignments.

Relationship between clients and server

This specification describes the inputs and outputs for the client and server. It does not regulate where (client or server) particular functionality is to be implemented

All communication between clients must be done via the server. The clients should not communicate with each other directly.

Functionality

Early disconnection

It is possible that both clients will disconnect before the game is over. The client to “blame” is the first client for which input is not available when it is **required** by the server.

For example, it is **X**'s turn. **O** disconnects, then **X** disconnects. The game over will be blamed on **X** because input will be required from **X** before **O**.

When a client detects end of input on **stdin** it can immediately close down or it can display all messages up to and including the end of input message (from Assignment 3). See marking scheme.

Statistics

When the server receives a SIGHUP signal it should print information about the players who have connected since the server started (in the order in which they first connected) to the logfile. Each line of output will consist of the player's name followed by the number of games they have connected to, the number they have won and the number they have disconnected from early. These items will be separated by single spaces.

If both players remain connected until the end of the game, the winner is the player with the most tiles. A draw is counted as a win for both players.

The list should be followed by a blank line.

Client errors

In the event of a normal game over the client should return 0.

Condition	Output	Exit	Message
Incorrect number of parameters	stderr	1	Usage: netflip name port [host]
Invalid port (< 1 > 65535)	stderr	2	Invalid port.
Unable to connect to server	stderr	3	Unable to connect to server.
Server disconnected unexpectedly	stdout	4	Server disconnected.
Server is full	stdout	5	Server is full at the moment.

Extra output from the client

When the first client in a game connects, it should display:

Waiting for another player.

If the connection to the server is lost before the client is informed that the game is over, then the client should display (to stdout):

Server disconnected.

This message should not be displayed if the server was full.

Server outputs

Error conditions should be checked in the order in which they appear in the following table.

Condition	Output	Exit	Message
Incorrect number of parameters	stderr	1	Usage: flipserv dim maxgames port logfile
Invalid board dimension	stderr	2	Invalid board dimension.
Invalid maximum games	stderr	3	Invalid maxgames.
Invalid port	stderr	4	Invalid port number.
Error setting up listening socket	stderr	5	Error listening on port.
Problem setting up the logfile	stderr	6	Error accessing log file.
Error handling an incoming connection	stderr	7	Error accepting connection.
Server starts successfully	logfile	-	

If the server starts successfully, it should output **Server started on port %d.** to the log file (substitute the real

port number).

Submission

Your submission must be to `/ass4/trunk` in your repository on `svn.eait`. It must include a **Makefile** which compiles both your client and server as its default action.

You must compile with the flags

```
--std=gnu99 -pedantic -Wall
```

The usual rules about non-standard headers/libraries apply and suppression of warnings apply.

The due date for this assignment is 11pm, 3rd June.

Late penalties will apply as outlined in the course profile.

Style

You must follow *version 1.5* of the COMP2303/COMP7306 C programming style guide found at http://www.itee.uq.edu.au/~comp2303/resources/c_resources.html.

Marks

Marks will be awarded for both style and functionality.

Functionality (42 marks)

Provided that your code compiles (see above), you will earn functionality marks based on the number of features your program correctly implements, as outlined below. Partial marks will be awarded for partially meeting the functionality requirements. Not all features are of equal difficulty. If your program does not allow a feature to be tested then you will receive 0 marks for that feature, even if you claim to have implemented it. The markers will make no alterations to your code (other than to remove code without academic merit). Your programs should not crash or lock up/loop indefinitely.

- Client parameters and messages (5 marks)
- Server
 - Parameters and messages (10 marks)
 - Set up log file (2 marks)
 - Player statistics (5 marks)
- Placing players in games properly (3 marks)
- Handling client disconnection (3 marks)
- Displaying extra game messages after end of input on client [tricky] (1 mark)
- Handling full server (2 marks)

- Various combinations of games (11 marks)

Style (8 marks)

If g is the number of style guide violations and w is the number of compilation warnings, your style mark will be the minimum of your functionality mark and:

$$8 \times 0.9^{g+w}$$

The number of compilation warnings will be the total number of distinct warning lines reported during the compilation process described above. The number of style guide violations refers to the number of violations of version 1.5 of the COMP2303/COMP7306 C Programming Style Guide. A maximum of 5 violations will be penalised for each broad guideline area. The broad guideline areas are Naming, Comments, Braces, Whitespace, Indentation, Line Length and Overall. For naming violations, the penalty will be one violation per offending name (not per use of the name) up to the maximum of five. You should pay particular attention to commenting so that others can understand your code. The marker's decision with respect to commenting violations is final — it is the marker who has to understand your code. To satisfy layout related guidelines, you may wish to consider the **indent(1)** tool. Your style mark

can never be more than your functionality mark — this prevents the submission of well styled programs which don't meet at least a minimum level of required functionality.

Specification Updates

It is possible that this specification contains errors or inconsistencies or missing information. It is possible that clarifications will be issued via the course website. Any such clarifications posted 5 days (120 hours) or more before the due date will form part of the assignment specification. If you find any inconsistencies or omissions, please notify the teaching staff.

Test Data

Test data and scripts for this assignment will be made available. The purpose of these are to help clarify some areas of the specification and to provide a basic sanity check of code which you have committed. *These are not guaranteed to check all possible problems nor are they guaranteed to resemble the tests which will be used to mark your assignments.* Testing that your assignment complies with this specification is still *your* responsibility.

Notes

1. Do not use the `select()` function.
2. A failure writing to a network connection should not produce an error message. To avoid problems with shutdowns due to broken pipe, you should suppress the SIGPIPE signal.
3. If the client is waiting for user input, then it is ok if it does not notice a server disconnect until after that input has been given.
4. Use the command `kill` to send signals to processes from the shell.
5. (v1.1) Your client and server must not execute any other processes.
6. (v1.1) Client message for server disconnect should go to stdout.