B4 - Unix System Programming

B-PSU-403

my IRC

Clients and Servers

2.2

my IRC

binary name:	server, client
group size:	2
repository name:	PSU_myirc_$ACADEMICYEAR
repository rights:	ramassage-tek
language:	C
compilation:	via Makefile, including re, clean and fclean rules

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Your Makefile must contain a **server** and a **client** rules to compile the eponymous binaries.

The whole standard C library is allowed, except **fork**.

This project consists of creating an IRC client and server.

For those of you who don't know what an IRC is: it's a kind of CHAT, or real-time dicussion system, that handles discussion groups called *channels*.
It also allows you to exchange files.
The network communication will be done via TCP sockets.

Server

Your server must :

- accept several simulataneous connections,
- be non-blocking and must absolutely implement the concept of a channel,
- comply with the RFC 1459 (Internet Relay Chat Protocol), its updates and dependencies.

Using a fork is unauthorized: you must use **poll/e-poll** or **select**.
One sole **select** per executable is authorized throughout your project.
This has nothing to do with non-blocking sockets, which are unauthorized.
So no *fcntl(s, O_NONBLOCK)*...

Terminal

```
~/B-PSU-403> ./server -help
USAGE: ./server port
```

Client

Your client must, at the very least, handle the following commands:

- **/server $host[:$port]**: connects to a server
- **/nick $nickname**: defines the user's nickname in the server
- **/list [$string]**: lists the server's available channels.
  Only display the channels that contain the *$string* string (when specified)
- **/join $channel**: joins a server's channel
- **/part $channel**: leaves a channel
- **/users**: lists the nicknames of the users that are connected to the server
- **/names $channel**: lists the nicknames of the users connected to the channel
- **$message**: sends a message to all of the users that are connected to the channel
- **/msg $nickname $message**: sends a message to a specific user
- **/msg $channel $nickname $file**: sends a file to a user
- **/accept_file $nickname**: accepts the reception of a file coming from one of the channel's users

Constraints

Your code must not only be non-blocking, but it must also use **circular buffers** in order to secure and optimize the various commands and responses that are being sent and received.

It is your responsibility to produce a "clean" code and to verify absolutely all errors and instances, which can lead to problems.
Otherwise, we will have no problem making your server inoperational (and therefore, non-functional).

man nc + Ctrl -D

Your client can be graphical.
Therefore, you have the option to use a graphic library (Gtk, SDL etc.) (as long as it's a C or even C++ library).
However, the network treatment section must be created with lib C functions (not QtNetwork, for example).