Unix System

my_irc

Table des matières

Administrative details

- Your sources shall be turned-in on the `PSU_year_myirc` directory
  ex : PSU_2013_myirc for the 2013-2014 scolar year
- Your binaries shall be compiled by one (and only one!) Makefile.
- The server binary shall be named `server`
- The client binary shall be named `client`
- The Makefile must have a `server` and a `client` rule to build eponymous binaries.
- This project must be done in groups of two.
- Your report should contain a `auteur` file with the logins to each group member, separated by ';'
- Official informations about the project can be find in the `PSU promo` Yammer group

Subject

- The aim of this project is to realise a IRC client / server.

Indices For those who do not know IRC, it is a kind of CHAT, or a chat system in real time, which manages discussion groups called "channel", and also allows file exchanges.

- the communication network will be through TCP sockets.
- Your server will accept multiple simultaneous connections.

Attention, the use of fork is prohibited. So you should imperatively use **select**

- Your server must not be blocking

Only one **select** is allowed for each binary

Indices This has nothing to do with non-blocking sockets, which are prohibited (so do not use fcntl(s, O_NONBLOCK))

- Your server will provide several channels.
- Your server must be RFC 1459 (and updates) compliant (Internet Relay Chat Protocol).
- Synopsis :

```
1   Usage : ./server port
```

- Your client will manage the following command :
  - `/server _host_[:_port_]` : connects to a server
  - `/nick _nickname_` : defines the nickname of the user on the server
  - `/list [string]` : list the channels available on the server. Displays only the channels containing the string "string" if it is specified.
  - `/join _channel_` : joins a channel on server
  - `/part _channel_` : leave the channel
  - `/users` : display the users connected to the server (display the nicknames of course)
  - `/names _channel_` : display the users connected to the given channel (display the nicknames of course)

- `_message_` : sends a message to all users connected to the channel.
- `/msg _nickname_ _message_` : sends a message to a specific user
- `/msg _channel_ _message_` : sends a message to a specific channel
- `/send_file _nickname_ _file_` : sends a file to a user.
- `/accept_file _nickname_` : accepts the reception of a file from a user from the channel.

Constraints

Your code will not only be non-blocking, but will also **use** circular buffer to secure and optimize the sending as the receiving of the various commands and responses.

It is your responsibility to produce a "clean" code, checking absolutely every error and every case that could cause problems. Otherwise, we will have no difficulty making your server inoperable (and therefore non-functional).

Indices `man nc + Ctrl-D`

Your client may be graphic. You have the possibility to use a graphics library (GTK, SDL, ...) as it is a C library or C++.

However, the network portion must imperatively be achieved through the C library functions... (no QtNetwork for example).

Forbidden functions

- fork

Allowed functions

- the C library