SCALE FOR PROJECT FT_IRC (/PROJECTS/FT_IRC)

Introduction

Please comply with the following rules:

- Remain polite, courteous, respectful and constructive throughout the evaluation process. The well-being of the community depends on it.

- Identify with the student or group whose work is evaluated the possible dysfunctions in their project. Take the time to discuss and debate the problems that may have been identified.

- You must consider that there might be some differences in how your peers might have understood the project's instructions and the scope of its functionalities. Always keep an open mind and grade them as honestly as possible. The pedagogy is useful only and only if the peer-evaluation is done seriously.

Guidelines

- Only grade the work that was turned in the Git repository of the evaluated student or group.

- Double-check that the Git repository belongs to the student(s). Ensure that the project is the one expected. Also, check that 'git clone' is used in an empty folder.

- Check carefully that no malicious aliases was used to fool you and make you evaluate something that is not the content of the official repository.

- To avoid any surprises and if applicable, review together any scripts used to facilitate the grading (scripts for testing or automation).

- If you have not completed the assignment you are going to evaluate, you have to read the entire subject prior to starting the evaluation process.

- Use the available flags to report an empty repository, a non-functioning program, a Norm error, cheating, and so forth.
In these cases, the evaluation process ends and the final grade is 0, or -42 in case of cheating. However, except for cheating, student are strongly encouraged to review together the work that was turned in, in order to identify any mistakes that shouldn't be repeated in the future.

- Remember that for the duration of the defence, no segfault, no other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag.
You should never have to edit any file except the configuration file if it exists. If you want to edit a file, take the time to explicit the reasons with the evaluated student and make sure both of you are okay with this.

- You must also verify the absence of memory leaks. Any memory allocated on the heap must be properly freed before the end of execution.
You are allowed to use any of the different tools available on the computer, such as leaks, valgrind, or e_fence. In case of memory leaks, tick the appropriate flag.

Mandatory Part

Basic checks

- There is a Makefile, the project compiles correctly, is written in C++, and the executable is called "ircserv".
- Ask and check how many poll() (or equivalent) are present in the code. There must be only one.
- Verify that the poll() (or equivalent) is called every time before each accept, read/recv, write/send. After these calls, errno should not be used to trigger specific action (e.g. like reading again after errno == EAGAIN).
- Verify that each call to fcntl() is done as follows:
fcntl(fd, F_SETFL, O_NONBLOCK);

Any other use of fcntl() is forbidden.
- If any of these points is wrong, the evaluation ends now and the final mark is 0.

✓ Yes ✗ No

Networking

Check the following requirements:
- The server starts, and listens on all network interfaces on the port given from the command line.
- Using the 'nc' tool, you can connect to the server, send commands, and the server answers you back.
- Ask the team what is their reference IRC client.
- Using this IRC client, you can connect to the server.
- The server can handle multiple connections at the same time. The server should not block. It should be able to answer all demands. Do some test with the IRC client and nc at the same time.
- Join a channel thanks to the appropriate command. Ensure that all messages from one client on that channel are sent to all other clients that joined the channel.

✓ Yes ✗ No

Networking specials

Network communications can be disturbed by many strange situations.
- Just like in the subject, using nc, try to send partial commands. Check that the server answers correctly. With a partial command sent, ensure that other connections still run fine.
- Unexpectedly kill a client. Then check that the server is still operational for the other connections and for any new incoming client.
- Unexpectedly kill a nc with just half of a command sent. Check again that the server is not in an odd state or blocked.
- Stop a client (^-Z) connected on a channel. Then flood the channel using another client. The server should not hang. When the client is live again, all stored commands should be processed normally. Also, check for memory leaks during this operation.

✓ Yes ✗ No

Client Commands

- With both nc and a regular IRC client, check that you can authenticate, set a nickname, a username, join a channel. This should be ok (you should have already done this previously).
- Verify that private messages (PRIVMSG) and notices (NOTICE) are fully functional with different parameters.
- Check that a regular user does not have privileges to do operator actions. Then test with an operator. All the channel operation commands should be tested (remove one point for each feature that is not working).

Rate it from 0 (failed) through 5 (excellent)

File transfer

File transfer works with the reference IRC client.

✓ Yes ✗ No

A small bot

There's an IRC bot.

✓ Yes ✗ No