42 Evals

Back to all evaluation sheets

ft_irc

You should evaluate 2 student in this team

Introduction

Please follow the rules below:

- Remain polite, courteous, respectful, and constructive throughout the evaluation process. The community's well-being depends on it.
- Work with the student or group being evaluated to identify potential issues in their project. Take time to discuss and debate the problems identified.
- Understand that there may be differences in how peers interpret the project instructions and scope. Always keep an open mind and grade as honestly as possible. Pedagogy is effective only when peer evaluations are taken seriously.

Guidelines

Please follow the guidelines below:

- Only grade the work submitted to the Git repository of the evaluated student or group.

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

- Carefully verify that no malicious aliases are used to deceive the evaluator into grading non-official content.

- If applicable, review any scripts used for testing or automation together with the student.

- If you haven’t completed the assignment you’re evaluating, read the entire subject before starting the evaluation.

- Use the available flags to report an empty repository, a non-functioning program, a Norm error, or cheating. The evaluation process ends with a final grade of 0 (or -42 for cheating). However, except in cases of cheating, students are encouraged to review the work together to identify mistakes to avoid in the future.

- Remember that no segfaults or other unexpected program terminations will be tolerated during the evaluation. If this occurs, the final grade is 0. Use the appropriate flag.

- You should not need to edit any files except the configuration file, if it exists. If editing a file is necessary, explain the reasons to the evaluated student and ensure mutual agreement.

- Verify the absence of memory leaks. All memory allocated on the heap must be properly freed before the program ends.

- You may use tools like leaks, valgrind, or e_fence to check for memory leaks. If memory leaks are found, tick the appropriate flag.

Attachments

Please download the attachments below:

subject.pdf

bircd.tar.gz

Mandatory Part

Basic checks

Basic checks

- [ ] - There is a Makefile, the project compiles correctly with the required options, is written in C++, and the executable is called as expected.
- [ ] - 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 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 basic

Client Commands basic

- With both nc and the reference IRC client, check that you can authenticate, set a nickname, a username, join a channel. This should be fine (you should have already done this previously).

- Verify that private messages (PRIVMSG) are fully functional with different parameters.

Yes No

Client Commands channel operator

Client Commands channel operator

- With both nc and the reference IRC client, check that a regular user does not have privileges to do channel 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)

0 1 2 3 4 5

Bonus Part

File transfer

Evaluate the bonus part if, and only if, the mandatory part has been entirely and perfectly done, and the error management handles unexpected or bad usage. In case all the mandatory points were not passed during the defense, bonus points must be totally ignored.

File transfer works with the reference IRC client.

Yes No

A small bot

A small bot

There's an IRC bot.

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

0 1 2 3 4 5

Ratings

OK

Outstanding

Incomplete Work

Invalid Compilation

Cheat

Crash

Concerning Situations

Leaks

Forbidden Functions

Cannot Support/Explain code

© 2024 42evals. All rights reserved.