Go to homepage

SCALE FOR PROJECT MINITALK

You should evaluate 1 student in this team

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 were 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 before 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, to identify any mistakes that shouldn't be repeated in the future.

Attachments

subject.pdf

Preliminary tests

If cheating is suspected, the evaluation stops here. Use the "Cheat" flag to report it. Take this decision calmly, wisely, and please, use this button with caution.

Prerequisites

- Defense can only happen if the evaluated student or group is present. This way everybody learns by sharing knowledge.
- If no work has been submitted (or wrong files, wrong directory, or wrong filenames), the grade is 0, and the evaluation process ends.
- No empty repository (= nothing in Git repository).
- No Norm error.
- Cheating (= -42).
- No compilation error. Also, the Makefile must not re-link.

If all of these requirements are passed, check Yes and go on.
Otherwise, use the appropriate flag at the end of the scale!

Yes | No

General instructions

General instructions

- The Makefile compiles both executables -> 1 point
- The server name is 'server' and it prints his PID at launch -> 2 points
- The client name is 'client' and is run as follows: './client PID_SERVER STRING_TO_PASS' -> 2 points

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

Mandatory part

This project is an introduction to signals. Check the code and ensure the signals are used only for the communication between the server and the client.

Message transmission

It's possible to pass on a message of any size.
Received messages must be displayed by the server, and must be obviously corrects!
The server should never get stuck or print wrong characters.

Yes | No

Simple setup

- The server can receive multiple strings without needing to be restarted. -> 1 point
- Only one global variable per program is allowed, or no global. Ask for explanations. -> 1 point
- The communication is done only using the signals SIGUSR1 and SIGUSR2. -> 3 points

Received messages must be displayed by the server, and must be obviously corrects!

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

Bonus part

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 ignored.

Unicode characters support

Unicode characters are supported both by the client and the server.

Yes | No

Acknowledgement

The server acknowledges every message received by sending back a signal to the client.

Yes | No

Ratings

Don't forget to check the flag corresponding to the defense

Ok | Outstanding project

Empty work | Incomplete work | Invalid compilation | Norme | Cheat

Crash | Concerning situation | Leaks | Forbidden function

Can't support / explain code

Conclusion

Give this repository a star.