42 Evals

Back to all evaluation sheets

Points earned
0

pipex

You should evaluate 1 student in this team

Introduction

Please follow the rules below:

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 being 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 if the peer-evaluation is done seriously.

Guidelines

Please follow the guidelines below:

Only grade the work that was turned in to 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 u
evaluating something that is not the content of the of

To avoid any surprises and if applicable, review t
facilitate the grading (scripts for testing or automatio

Points earned
0

If you have not completed the assignment you are going to evaluate, you must 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 the case of cheating. However, except for cheating, students 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 defense, no segfaults or other unexpected, premature, or uncontrolled terminations of the program will be tolerated, 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 explain 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.

Attachments

Please download the attachments below:

subject.pdf

Points earned
0

Mandatory Part

Prerequisites

Please verify the following points:

The conditions for proceeding with the defense are met: the evaluated person(s) are present, and a non-empty submission that rightfully belongs to them is available, etc.

If no work has been submitted (or if incorrect files, incorrect directory, or incorrect file names were used), the grade is 0, and the evaluation process ends.

No empty repository (i.e., the Git repository is not empty).

No Norm errors.

Cheating results in a score of -42.

No compilation errors or a Makefile that re-links.

If all these points are validated, check "Yes" and continue with the evaluation.

If not, use the appropriate flag at the end of the evaluation!

Yes No

General Instructions

If a crash or error occurs unexpectedly (segmentation fault, bus error,

abnormal display, etc.), use the "Crash" flag!

The Makefile correctly compiles the executable with the required options.

The executable is named "pipex."

No forbidden functions are used.

Points earned
0

Yes No

Error and Argument Handling

The command `./pipex file1 cmd1 cmd2 file2` should behave exactly like the following command: `< file1 cmd1 | cmd2 > file2`.

The program accepts exactly 4 arguments, no more, no less (except for the bonus part), and they must be in the correct order.

Error handling is correct: existing or non-existing files, file permissions, existing or non-existing command binaries, etc.

If these points are met, check "Yes" and continue the evaluation.

If not, the evaluation stops here. Use the 'Incomplete work' flag or any other appropriate flag.

Yes No

The Program

The Program

The program performs as required by the subject without displaying any additional information or steps compared to the shell command.

Conduct your own tests and compare the program
the original command. Feel free to review the subj
examples.

If no errors are detected, check "Yes" and continu
evaluation stops here.

Points earned
0

Yes No

Bonus Part

Multiple Pipes

The bonus will only be considered if the mandatory part is excellent. This means the mandatory section must be completed from start to finish, with perfect error handling, even in unexpected use cases. If all mandatory points have not been awarded during this evaluation, no bonus points will be counted.

The program handles the use of multiple pipes consecutively.

As with the mandatory part, test the program using shell commands to compare the output.

If it works correctly for only 2 pipes in the same command but not for 5, this bonus should not be awarded.

Yes No

`<<` and `>>` with the `here_doc` Parameter

`<<` and `>>` with the `here_doc` Parameter

The program replicates the use of `<<` and `>>`.

Test this multiple times with a command like:

```
CMD << STOP_VALUE | CMD1 >> file1
```

Ensure that the program behaves as expected when using these redirection operators.

Yes No

Points earned
0

Ratings

OK Outstanding Empty Work

Incomplete Work Invalid Compilation

Norme Cheat Crash

Concerning Situations Leaks

Forbidden Functions

Points earned

0