(https://profile.intra.42.fr)

Remember that the quality of the defenses, hence the quality of the of the school on the labor market depends on you. The remote defences during the Covid crisis allows more flexibility so you can progress into your curriculum, but also brings more risks of cheat, injustice, laziness, that will harm everyone's skills development. We do count on your maturity and wisdom during these remote defenses for the bene fits of the entire community.

SCALE FOR PROJECT GET_NEXT_LINE (/PROJECTS/42CURSUS-GET_NEXT_LINE)

You should evaluate 1 student in this team

Git repository

Introduction

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

- Identify with the person (or the group) evaluated the eventual dysfunctions of the work. Take the time to discuss and debate the problems you have identified.

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

Guidelines

- Only grade the work that is in the student or group's GiT repository.

- Double-check that the GiT repository belongs to the student or the group. Ensure that the work is for the relevant project and 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 other than the content of the official repository.

- To avoid any surprises, carefully check that both the evaluating and the evaluated students have reviewed the possible scripts used to facilitate the grading.

- If the evaluating student has not completed that particular project yet, it is mandatory for this student to read the entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository, non-functioning program, a norm error, cheating etc. In these cases, the grading is over and the final grade is 0 (or -42 in case of cheating). However, with the exception of cheating, you are encouraged to continue to discuss your work (even if you have not finished it) in order to identify any issues that may have caused this failure and avoid repeating the same mistake 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.

Attachments

 subject.pdf (https://cdn.intra.42.fr/pdf/pdf/15640/en.subject.pdf)

Mandatory Part

Error Management

Carry out AT LEAST the following tests to try to stress the error management

- Pass an arbitrary file descriptor to the get_next_line function on which it is not possible to read, for example 42. The function must return -1.
- Check the error returns for read and malloc.

If there is an error, you must stop the evaluation.

Yes No

Testing

As the evaluator, you are expected to provide a main which will always check:
- The return value of the get_next_line.
- The line read is the line sent to the input, without the \n. Test all the possible combinations of the following rules:

- Large BUFFER_SIZE (>1024)
- Small BUFFER_SIZE (< 8, and 1)
- BUFFER_SIZE exactly the length of the line to read
- 1 byte variant (+/-) between the line and the BUFFER_SIZE

- Read on stdin
- Read from a file

- (Multiple/Single) Long line (2k+ characters)
- (Multiple/Single) Short line (< 4 characters, even 1)
- (Multiple/Single) Empty line

These tests should allow you to verify the strength of the student's get_next_line. If any fails, grade 0 on the project.

Yes No

Bonus

We will look at your bonuses if and only if your mandatory part is excellent. This means that you must complete the mandatory part, beginning to end, and your error management must be flawless, even in cases of twisted or bad usage. So if you didn't score all the points on the mandatory part during this defence bonuses will be totally ignored.

Multiple fd reading

Perform the same tests as you did before, this time launch multiple instances of get_next_line with a different file descriptor on each. Make sure that each get_next_line is returning the correct line, combine with a non-existing fd to check for errors.

Yes No

Single static variable

Check the code and verify if there is indeed a single static variable. Give the points if that's the case.

Yes No

Ratings

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

Ok Outstanding project

Empty work No author file Invalid compilation Norme Cheat Crash

Leaks Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation