Menu

My projects

Holy Graph

List projects

Available Cursus

Your projects

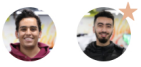
CPP Module 08

Philosophers

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 benefits of the entire community.

SCALE FOR PROJECT MINISHELL

You should evaluate 2 students in this team



Git repository

`git@vogsphere-v2-bg.13`

Introduction

Please respect 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 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

Mandatory Part

Simple Command & global

- Execute a simple command with an absolute path like /bin/ls or any other command without options
- How many global variable ? why ? concrete example of why it feels mandatory or logical.

Yes | No

Arguments

- Execute a simple command with an absolute path like /bin/ls or any other command with arguments but without quotes and double quotes
- Repeat multiple times with different commands and arguments

Yes | No

echo

- Execute the echo command with or without arguments or options
- Repeat multiple times with different arguments

Yes | No

exit

- Execute exit command with or without arguments
- Repeat multiple times with different arguments
- Don't forget to relaunch the minishell

Yes | No

Return value of a process

- Execute a simple command with absolute path like /bin/ls or any other command with arguments but without quotes and double quotes then execute echo $?
- Check the printed value. You can repeat the same in bash and compare it.
- Repeat multiple times with different commands and arguments, use some failing commands like '/bin/ls filethatdoesntexist'

Yes | No

Semicolons

- Execute multiple simple commands with absolute path with arguments but separate them with semicolons
- Repeat multiple times with different commands and don't forget to try with or without whitespaces around the semicolons

Yes | No

Signals

- Try ctrl-C in an empty prompt
- Try ctrl-\ in an empty prompt
- Try ctrl-D in an empty prompt
- Try ctrl-C in a prompt after you wrote some stuff

- Try ctrl-D in a prompt after you wrote some stuff
- Try ctrl-\ in a prompt after you wrote some stuff
- Try ctrl-C after running a blocking command like cat or grep without arguments
- Try ctrl-\ after running a blocking command like cat or grep without arguments
- Try ctrl-D after running a blocking command like cat or grep without arguments
- Repeat multiple times with different commands

Yes | No

Double Quotes

- Execute a simple command with absolute path with arguments but this time double quotes (you should include whitespaces and semicolons in the quotes)
- Think about empty arguments or a weird use of '\'
- Do not try multiline strings

Yes | No

env

- Check if env shows you the current environment variables

Yes | No

export

- Export environment variables, create new ones and replace old ones
- Check them with env

Yes | No

unset

- Export environment variables, create new ones and replace old ones
- Use unset to remove some of them
- Check the result with env

Yes | No

Environment Variables

- Execute echo with some $ variables as arguments
- Check if double quotes around $ variables is working correctly (like in bash)

Yes | No

cd

- Use the command cd to move the working directory and check if you are in the right directory with /bin/ls
- Repeat multiple times with working and not working cd
- try '.' '..' as arguments too

Yes | No

pwd

- Use the command pwd
- Repeat multiple times in multiple directories

Yes | No

Relative Path

- Execute commands but this time use a relative path
- Repeat multiple times in multiple directories with complex relative path (lots of ..)

Yes | No

Environment Path

- Execute commands but this time without any path. (ls, wc, awk etc...)
- Unset the $PATH and check if it is not working anymore
- Set the $PATH to a multiple directory value (directory1:directory2) and check that directories are checked in order from left to right

Yes | No

Simple Quotes

- Execute commands with simple quotes as argument
- Try empty arguments
- Try environment variables, whitespaces and semicolons in the simple quotes

Yes | No

Redirection

- Execute commands with redirections < and/or >
- Repeat multiple times with different commands and arguments and sometimes change > with >>
- Check if multiple of the same redirections fail

Yes | No

Pipes

- Execute commands with pipes like 'cat file | grep bla | more'
- Repeat multiple times with different commands and arguments
- Try some failing commands like 'ls filethatdoesntexist | grep bla | more'
- Try to mix pipes and redirections.

Yes | No

Go Crazy and history

- Can we navigate through history with up and down and retry some command
- Execute commands that should not work like 'dsbksdgbksdghsd' and check if the shell doesn't crash and prints an error
- Try to execute a really really really long command with a ton of arguments
- Have fun with that beautiful minishell and enjoy it

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.

double left redirection

- Check if << is working fine

Yes | No

Line editing

- Can we move the cursor left and right and edit the line by inserting or deleting characters at cursor location
- Can we copy paste all/part of a line using a key sequence
- Can we move word by word with ctrl+left or ctrl+right
- Go directly to the beginning or the end of the line with home or end
- Write and edit commands with multilines

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

And, Or

- Use &&, || and parenthesis with commands and check if it works like bash
- For each working flag give 1 point
- If all flags are working give 1 bonus point

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

WildCard

- Use wildcards in arguments
- Try things like */*
- Go crazy with wildcards

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