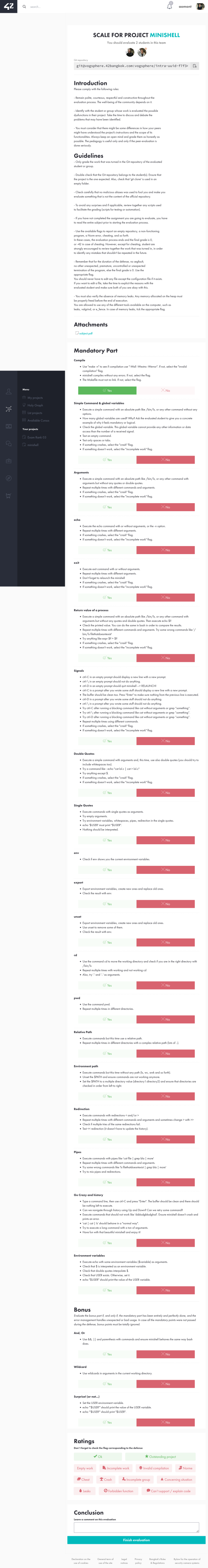

SCALE FOR PROJECT MINISHELL

You should evaluate 2 students in this team

Git repository

`git@vogsphere.42bangkok.com:vogsphere/intra-uuid-f7f3`

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 was 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 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 case of cheating. However, except for cheating, student 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 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

Compile

- Use "make -n" to see if compilation use "-Wall -Wextra -Werror". If not, select the "invalid compilation" flag.
- minishell compiles without any errors. If not, select the flag.
- The Makefile must not re-link. If not, select the flag.

✓ Yes ✗ No

Simple Command & global variables

- Execute a simple command with an absolute path like /bin/ls, or any other command without any options.
- How many global variables are used? Why? Ask the evaluated student to give you a concrete example of why it feels mandatory or logical.
- Check the global variable. This global variable cannot provide any other information or data access than the number of a received signal.
- Test an empty command.
- Test only spaces or tabs.
- If something crashes, select the "crash" flag.
- If something doesn't work, select the "incomplete work" flag.

✓ Yes ✗ No

Arguments

- Execute a simple command with an absolute path like /bin/ls, or any other command with arguments but without any quotes or double quotes.
- Repeat multiple times with different commands and arguments.
- If something crashes, select the "crash" flag.
- If something doesn't work, select the "incomplete work" flag.

✓ Yes ✗ No

echo

- Execute the echo command with or without arguments, or the -n option.
- Repeat multiple times with different arguments.
- If something crashes, select the "crash" flag.
- If something doesn't work, select the "incomplete work" flag.

✓ Yes ✗ No

exit

- Execute exit command with or without arguments.
- Repeat multiple times with different arguments.
- Don't forget to relaunch the minishell
- If something crashes, select the "crash" flag.
- If something doesn't work, select the "incomplete work" flag.

✓ Yes ✗ No

Return value of a process

- Execute a simple command with an absolute path like /bin/ls, or any other command with arguments but without any quotes and double quotes. Then execute echo $?
- Check the printed value. You can do the same in bash in order to compare the results.
- Repeat multiple times with different commands and arguments. Try some wrong commands like '/bin/ls filethatdoesntexist'
- Try anything like expr $? + $?
- If something crashes, select the "crash" flag.
- If something doesn't work, select the "incomplete work" flag.

✓ Yes ✗ No

Signals

- ctrl-C in an empty prompt should display a new line with a new prompt.
- ctrl-\ in an empty prompt should not do anything.
- ctrl-D in an empty prompt should quit minishell --> RELAUNCH!
- ctrl-C in a prompt after you wrote some stuff should display a new line with a new prompt.
- The buffer should be clean too. Press "Enter" to make sure nothing from the previous line is executed.
- ctrl-D in a prompt after you wrote some stuff should not do anything.
- ctrl-\ in a prompt after you wrote some stuff should not do anything.
- Try ctrl-C after running a blocking command like cat without arguments or grep “something“.
- Try ctrl-\ after running a blocking command like cat without arguments or grep “something“.
- Try ctrl-D after running a blocking command like cat without arguments or grep “something“.
- Repeat multiple times using different commands.
- If something crashes, select the "crash" flag.
- If something doesn't work, select the "incomplete work" flag.

✓ Yes ✗ No

Double Quotes

- Execute a simple command with arguments and, this time, use also double quotes (you should try to include whitespaces too).
- Try a command like : echo "cat lol.c | cat > lol.c"
- Try anything except $.
- If something crashes, select the "crash" flag.
- If something doesn't work, select the "incomplete work" flag.

✓ Yes ✗ No

Single Quotes

- Execute commands with single quotes as arguments.
- Try empty arguments.
- Try environment variables, whitespaces, pipes, redirection in the single quotes.
- echo '$USER' must print "$USER".
- Nothing should be interpreted.

✓ 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 the result 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

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
- Also, try '.' and '..' as arguments.

✓ Yes ✗ No

pwd

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

✓ Yes ✗ No

Relative Path

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

✓ Yes ✗ No

Environment path

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

✓ Yes ✗ No

Redirection

- Execute commands with redirections < and/or >
- Repeat multiple times with different commands and arguments and sometimes change > with >>
- Check if multiple tries of the same redirections fail.
- Test << redirection (it doesn't have to update the history).

✓ Yes ✗ No

Pipes

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

✓ Yes ✗ No

Go Crazy and history

- Type a command line, then use ctrl-C and press "Enter". The buffer should be clean and there should be nothing left to execute.
- Can we navigate through history using Up and Down? Can we retry some command?
- Execute commands that should not work like 'dsbksdgbksdghsd'. Ensure minishell doesn't crash and prints an error.
- 'cat | cat | ls' should behave in a "normal way".
- Try to execute a long command with a ton of arguments.
- Have fun with that beautiful minishell and enjoy it!

✓ Yes ✗ No

Environment variables

- Execute echo with some environment variables ($variable) as arguments.
- Check that $ is interpreted as an environment variable.
- Check that double quotes interpolate $.
- Check that USER exists. Otherwise, set it.
- echo "$USER" should print the value of the USER variable.

✓ Yes ✗ No

Bonus

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.

And, Or

- Use &&, || and parenthesis with commands and ensure minishell behaves the same way bash does.

✓ Yes ✗ No

Wildcard

- Use wildcards in arguments in the current working directory.

✓ Yes ✗ No

Surprise! (or not...)

- Set the USER environment variable.
- echo "'$USER'" should print the value of the USER variable.
- echo '"$USER"' should print "$USER".

✓ Yes ✗ No

Ratings

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

Conclusion

Leave a comment on this evaluation