Scale for project minishell

You should evaluate 2 students in this team

Git repository

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 before starting the defence.

- Use the flags available on this scale to signal an empty repository,
non-functioning program, norm error, cheating etc. In these cases,

the grading is over and the final grade is 0 (or -42 in case of
cheating). However, except for cheating, you are
encouraged to continue to discuss your work (even if you have not
finished it) 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 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 use invalid compilation flags
- minishell Compile without errors if not use flags
- makefile must not re-link

Simple Command & global

- Execute a simple command with an absolute path like /bin/ls or any other command without options
- How many global variables? why? Give a concrete example of why it feels mandatory or logical.
- Test an empty command.
- Test only spaces or tabs.
- if something crashes use the crash flag.
- if something is not working use the incomplete work flag.

Arguments & history

- 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
- if something crashes use the crash flag.
- if something is not working use the incomplete work flag.

echo

- Execute the echo command with or without arguments or -n
- Repeat multiple times with different arguments
- if something crashes use the crash flag.
- if something is not working use the incomplete work flag.

exit

- Execute exit command with or without arguments
- Repeat multiple times with different arguments
- Don't forget to relaunch the minishell
- if something crashes use the crash flag.
- if something is not working use the incomplete work flag.

Return value of a process

- Execute a simple command with an 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'
- anything like expr $? + $?
- if something crashes use the crash flag.
- if something is not working use the incomplete work flag.

Signals

- Try ctrl-C in an empty prompt should show a new line with a new prompt
- Try ctrl-\ in an empty prompt should not do anything
- Try ctrl-D in an empty prompt should quit minishell --> RELAUNCH!
- Try ctrl-C in a prompt after you wrote some stuff should show a new line with a new prompt
- The buffer should be clean too, press "enter" to make sure nothing from the old line is executed.
- Try ctrl-D in a prompt after you wrote some stuff should not do anything
- Try ctrl-\ in a prompt after you wrote some stuff should not do anything!
- 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
- if something crashes use the crash flag.
- if something is not working use the incomplete work flag.

Double Quotes

- Execute a simple command with arguments but this time double quotes (you should include whitespaces)
- a command like : echo "cat lol.c | cat > lol.c"
- anything except $.
- if something crashes use the crash flag.
- if something is not working use the incomplete work flag.

Single Quotes

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

env

- Check if env shows you the current environment variables

export

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

unset

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

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

pwd

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

Relative Path

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

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

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
- Test << redirection (it doesn't need to update history).

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.

Go Crazy and history

- type a command line then use ctrl-C then press enter the buffer should be clean and nothing try to execute.
- 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 long command with a ton of arguments
- Have fun with that beautiful minishell and enjoy it

Environment Variables

- Execute echo with some $ variables as argumentss
- Check that $ is interprated as an environment variable
- Check that double quotes interpolate $
- Check that $USER exist or set it.
- echo "$USER" should print the value of $USER

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 defense bonuses will be totally ignored.

And, Or

- Use &&, || and parenthesis with commands and check if it works as bash

WildCard

- Use wildcards in arguments for the local directory.

Surprise (or not...)

- set USER environment variable.
- Test echo "'$USER'" this should print 'USER_VALUE'
- Test echo "'$USER'" this should print "$USER"

Ratings

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