🔍

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

**(https://profile.intra.42.fr/searches)**     **(https://signin.intra.42.fr/users**

Scale for project minishell (/projects/minishell)

**You should correct 1 student in this team**

★

**Git repository**

vogsphere@vgs.42.us.org:intra/2017/activities/minishell/bbauer  📋

# Introduction

Please respect the following rules:

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

- Identify with the person (or the group) graded 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 if
the 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 correcting
and the corrected students have reviewed the possible scripts used
to facilitate the grading.

- If the correcting 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.

# Attachments

🗎 Subject (https://cdn.intra.42.fr/pdf/pdf/949/minishell.en.pdf)

Sections

Mandatory part

Reminder : Remember that for the duration of the defence, no segfault, nor other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag. This rule is active thoughout the whole defence.

## Author file

Reminder : Remember that for the duration of the defence, no segfault, nor other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag. This rule is active thoughout the whole defence.

⊘ Yes                                                    ✕ No

## Memory leaks

Throughout the defence, pay attention to the amount of memory used by minishell (using the command top for example) in order to detect any anomalies and ensure that allocated memory is properly freed. If there is one memory leak (or more), the final grade is 0.

⊘ Yes                                                    ✕ No

## Fork and execve

"fork" et "execve" form the basis of a minimalist shell like the minishell. Therefore, those functions should be used in the code. Otherwise, sit means that the instructions were misunderstood.
Defence is finished and final grade is 0. The only way to succeed this project to use the list of authorized functions. There is no other solution.

Execute the following 4 tests:

- Run minishell, then run the following command "$> foo". It must fail with a proper error message and then give back the prompt.

- Run the following command "$> /bin/ls". ls must be properly executed and then give back the prompt.

- Run the following command "$> /bin/ls -laF". ls must be properly executed with the -l, -a, -F flags and then give back the prompt.

- Run the following command "$> /bin/ls -l -a -F". ls must be properly executed with the -l, -a, -F flags and then give back the prompt.

If at least one fails, no points will be awarded for this section. Move to the next one.

⊘ Yes                                                    ✕ No

## Builtins

In this section we'll evaluate the implementation of built-ins such as "exit", "echo" et "cd". A shell needs to include these basic features, even if it sounds prehistoric to you,

Execute the following 8 tests:

- Run minishell, then run the following command "$> exit". The program must terminate proprely and give back the parent's shell. Run the minishell again.

- Run a command such as "$> echo "It works"". The message must be properly displayed.

- Run a command such as "$> echo It works" (without the double quotes). The message must be properly displayed.

- Run a command such as "$> cd /absolute/path/of/your/choice", then run the following command "$> /bin/pwd". /bin/pwd must confirm that the current folder was updated.

- Run a command such as "$> cd relative/path/of/your/choice", then run the following command "$> /bin/pwd". /bin/pwd must confirm that the current folder was updated.

- Run the following command "$> cd", then run "$> /bin/pwd".

/bin/pwd must confirm that the current folder is the
user's home folder.

- Run the following command "$> cd -", then run "$> /bin/pwd".
/bin/pwd must confirm that the current folder is the
folder relative/path/of/your/choice used before.

- Run the following command "$> cd ~/path/of/your/choice",
then run "$> /bin/pwd". "$> /bin/pwd". /bin/pwd must
confirm that the current folder was updated.

If at least one fails, no points will be awarded for this
section. Move to the next one.

⊘ Yes                                                                                                    ✕ No

## Environment management

In this section we'll evaluate the implementation of specific
built-ins such as "env", "setenv" et "unsetenv".

Execute the following 7 tests:

- Run the following command "$> env". Environment variables
must be displayed as key=value.

- Run a command such as "$> setenv FOO bar" or
"$> setenv FOO=bar" depending on the implemented syntax.
Then run the following command "$> env". The environment
must display a FOO variable with the value bar.

- Run the following command "$> echo $FOO". The value bar
must be displayed.

- Run the following command "$> /usr/bin/env". Minishell
must send the appropriate environment to ran binaries.
/usr/bin/env must display environment including FOO and
its value bar.

- Run the following command "$> unsetenv FOO". Then run
"$> env". The environment variable FOO must not be
displayed anymore.

- Run the following command again "$> unsetenv FOO". Then
run "$> env". Environment must not change.

- Run the following command again "$> /usr/bin/env".
/usr/bin/env must not display variable FOO anymore.()

If at least one fails, no points will be awarded for
this section. Move to the next one.

⊘ Yes                                                                                                    ✕ No

## PATH management

In this section we'll evaluate the implementation of PATH in
your shell.

Execute the following 6 tests:

- Run the following command "$> unsetenv PATH", then run
"$> setenv PATH "/bin:/usr/bin" or "$> setenv
"PATH=/bin:/usr/bin" depending on the implemented syntax.
Then run the following command "$> ls". /bin/ls must be
properly executed.

- Run the following command "$> emacs". /usr/bin/emacs
must be properly executed.

- Run the following command "$> unsetenv PATH", then run
"$> ls". It must fail.

- Run now the following command "$> emacs". It must also
fail.

- Run the following command "$> /bin/ls". /bin/ls must
be properly executed.

- Run the following command "$> /usr/bin/emacs".
/usr/bin/emacs must be properly executed.

If at least one fails, no points will be awarded for this
section. Move to the next one.

☑ Yes                                                          ✕ No

## Command line management

Nous allons évaluer dans cette section la gestion de la ligne de
commande. Effectuez les 4 tests suivants. Si l'un au moins de ces tests échoue,
alors toute la section est échouée, passez à la suivante :

- Lancez une commande vide "$> ". Le shell ne doit rien faire et
réafficher le prompt.

- Lancez une commande composée uniquement d'un seul espace "$> ".
Le shell ne doit rien faire et réafficher le prompt.

- Lancez une commande composée uniquement d'espaces et de
tabulations. Le shell ne doit rien faire et réafficher le prompt.

- Lancez une commande avec plusieurs espaces et tabulations
avant le nom du binaire, entre chaque argument passé au
binaire, et après le dernier argument. Ces espaces et
tabulations inutiles ne doivent pas perturber l'éxécution de
la commande.

☑ Yes                                                          ✕ No

# Bonus

Reminder : Remember that for the duration of the defence, no segfault, nor other unexpected, premature, uncontrolled or unexpected termination of the program, else the final
grade is 0. Use the appropriate flag. This rule is active thoughout the whole defence. We will look at your bonuses if and only if your mandatory part is EXCELLENT. This means that
your 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 the mandatory part didn't score
all the point during this defence bonuses will be totally IGNORED.

## Signal

In this section we'll evaluate signal management and more
specifically Ctrl-C.

Execute the following 3 tests:

- Instead of typing a command press Ctrl-C. The shell
must just give back the prompt

- Type a random command but instead of running it press
Ctrl-C. The minishell must give back and empty prompt.

- Run the following command "$> cat", then when cat waits
for inputs on the standard input, press Ctrl-C. The
minishell must kill cat's process and give back the
prompt.

If at least one fails, no points will be awarded for this
section. Move to the next one.

☑ Yes                                                          ✕ No

## PATH\'s rights

In this section we'll evaluate PATH's rights management.

Execute the following test:

- Create a new folder /tmp/bin/ and add this folder to the
PATH environment variable. Create a program named
'test_exec_rights' inside that folder that will just display
'KO'. Give this program the following rights 644 (meaning no
execution rights). From another folder, run the following
command "$> test_exec_rights". Check that the minishell
refuses to run the program because of the missing execution
rights.

If it fails, no points will be awarded for this section. Move
to the next one.

☑ Yes                                                          ✕ No

## Auto-completion

In this section we'll evaluate auto-completion.

Execute the following 2 tests:

- Type the following beginning of command "$> ec", then
press tabulation. The minishell must complete the

command into "$> echo".

- Type the following beginning of command "$> ema", then press tabulation. The minishell must complete the command into "$> emacs".

If at least one fails, no points will be awarded for this section. Move to the next one.

☑ Yes                                                                    ✕ No

## Semi colon

In this section we'll evaluate semi colon management. Allowing the user to run 2 commands, separated by a semi colon, in a row.

Execute the following 2 tests:

- Run the following command "$> echo TOP ; ls ; echo MIDDLE ; ls ; echo BOTTOM". The 5 commands must be executed without any errors in the order they were written.

- Run the following command "$> ;". The minishell must either do nothing and give the prompt back or display a syntax error and give the prompt back.

If at least one fails, no points will be awarded for this section. Move to the next one.

☑ Yes                                                                    ✕ No

## Other features

If the project has other operational bonuses, you can evaluate them and grade them in this section.

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

# Ratings

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

✔ Ok          ▮ Empty work          ▰ Incomplete work          🗩 No author file          ☣ Invalid compilation          ♫ Norme          ▤ Cheat          ♛ Crash

# Conclusion

**Leave a comment on this correction**
**\* (required) Comment**

|  |
|  |

| Finish correction |