Exercises — Minishell

version **#1.0.0**

The way is lit. The path is clear.
We require only the strength to follow it.

Assistants C/Unix 2023 <assistants@tickets.assistants.epita.fr>

Copyright

Contents

*https://intra.forge.epita.fr

File Tree

```
./
├─ Makefile   (to submit)
│
├─ src/
│  └─ **/
│     └─ *   (to submit)
│
└─ tests/
   └─ **/
      └─ *   (to submit)
```

Compilation : Your code must compile with the following flags

- -Wall -Wextra -Werror -pedantic -std=c99 -Wvla

Forbidden functions : You can use all the functions of the standard C library except

- glob
- popen
- regexec
- syscall
- system
- wordexp

Makefile

- minishell: Produce the minishell binary

Obligations

*Obligations are **fundamental** rules shared by all subjects. They are non-negotiable and to not apply them means to face sanctions. Therefore, do not hesitate to ask for explanations if you do not understand one of these rules.*

Obligation #0: **Cheating**, as well as sharing source code, tests, test tools or coding-style correction tools is **strictly forbidden** and penalized by not being graded, being flagged as a cheater and reported to the academic staff.

Obligation #1: The coding-style needs to be respected at all times.

Obligation #2: If you do not submit your work before the deadline, it will not be graded.

Obligation #3: Anything that is not **explicitly** allowed is **disallowed**.

Obligation #4: When examples demonstrate the use of an output format, you must follow it scrupulously.

Obligation #5: Your submission repository must be **clean**. Except for special cases, which (if any) are **explicitely** mentioned in this document, an *unclean* repository may contain:

- binary files;[1]
- files with inappropriate privileges;
- forbidden files: `*~`, `*.swp`, `*.o`, `*.a`, `*.so`, `*.class`, `*.log`, `*.core`, etc.;
- a file tree that does not follow our specifications.

Obligation #6: All your files must be encoded in ASCII or UTF-8 without BOM.

Advice

- ▷ Do **not** wait for the last minute to start your project!
- ▷ Read the *whole* subject.

[1]If an executable file is required, please provide its sources **only**. We will compile it ourselves.

1 Introduction

For this miniproject, you have to develop a minishell. You probably already know what a shell is with the 42sh project. As we are talking about a ***mini*** shell, you will not have to implement all the features of a shell. In this project, some features are not independent: if you do not validate any of them, the following parts will be affected as we are using some of the previous features to evaluate another. Please note the following:

- The produced executable must be named **minishell** and be generated at the **root** of your project with the command `make minishell`, your Makefile should define at least the target `minishell`.

```
42sh$ ls
Makefile  src  tests
42sh$ make minishell
42sh$ ls
Makefile  minishell  src  tests
42sh$ ./minishell
minishell$
```

- You are free with the project architecture as long as it respects the minimum submission tree and the directive above.
- The return code of your minishell must be the same as the reference, which is *bash –posix*.

```
42sh$ echo "Test." | minishell
minishell: line 1: Test.: command not found
42sh$ echo $?
127
```

- Remember that each feature from bash that is not mentionned in the subject must not be implemented. You don't even have to parse them. For example:

```
minishell$ echo &
&
minishell$ exit
42sh$ echo &
[1] 10212

42sh$
```

For this project, you are allowed to use all the functions of the standard C library except:

- glob
- regexec
- wordexp
- system
- popen
- syscall

2 Inputs

Your shell must be able to run in interactive mode or to get inputs directly from a pipe. It should also be able to read a script if passed as argument. Note that this part has to work if you want all the following parts to work. As an example, both of the following commands should work:

```
42sh$ echo "echo Test." | ./minishell
Test.
42sh$ cat test.sh
echo "Test."
42sh$ ./minishell test.sh
Test.
42sh$ ./minishell
minishell$ echo test
test
```

3 Execution

Your minishell must be able to execute **basic commands** and **commands with arguments**. Here are some examples of commands you must be able to handle:

```
minishell$ /bin/echo -e "test\ntest"
test
test
minishell$ /bin/ls
Makefile  src  tests
minishell$ ls
Makefile  src  tests
minishell$
```

In this part, you can just use `execvp(3)`. In the event of an execution failure, refer to *bash* for the return code.

4 Builtins

For this part, we want you to implement some simple builtins, obviously, you will not be allowed to use the binaries or bash builtins:

4.1 echo

Refer to the man of `echo(1)`. Only the `-n` option is mandatory.

```
minishell$ /bin/echo -e "test\ntest"
test
test
minishell$ echo -e "test\ntest"
-e test\ntest
minishell$ echo -n test
testminishell$
```

Be careful!

The builtin *echo* is a very basic builtin, be careful to not fail it as it could severely impact your other features.

4.2 cd

Refer to the man of `bash(1)`. No option are asked, not even `cd` without any argument (equivalent to `cd $HOME`).

4.3 exit

Refer to the man of `bash(1)`.

4.4 kill

Refer to the man of `kill(1)`. Only `kill -signal pid [...]` with numeric signals and positive PIDs is required. If no signal is provided to the `-signal` option, you must use the same default one as bash.

5 Command sequences

This part requires the *execution* part to be tested.

Your program must handle the `;`, `&&` and `||` tokens:

```
minishell$ echo -n Y; echo -n A ; echo -n K ;echo A
YAKA
minishell$ true && echo OK;
OK
minishell$ false || echo OK
OK
minishell$ echo -n Y; echo -n A || echo -n A && echo -n K || echo U; echo A
YAKA
```

6 Redirections

This part requires the *execution* part to be tested.

Your minishell must handle:

- simple redirections (standard and error outputs);
- double redirections (standard and error outputs);
- simple left redirections (standard input).

After this part, it must be able to execute commands such as:

```
minishell$ echo xavier.login > AUTHORS
minishell$ cat AUTHORS
xavier.login
minishell$ cat /etc/passwd >> users_list
minishell$ grep ^root users_list
root:x:0:0:root:/root:/bin/bash
rootme:x:812:800:rootme lse:/home/rootme/:/bin/tcsh
minishell$ < users_list wc -l
    9364
minishell$ ls -l
total 8
--w------- 1 xavier.login assistants 30 May 15 17:15 no_right
-rw-r--r-- 1 xavier.login assistants 35 May 15 17:15 rd_right
minishell$ cat no_right rd_right > Test.
cat: no_right: Permission denied
minishell$ cat Test.
Votai.
minishell$ cat no_right rd_right 2>> Test.
Votai.
minishell$ cat Test.
Votai.
cat: no_right: Permission denied
minishell$
```

6.1 Remarks:

- Redirections can be found anywhere: before, after or in the middle of commands.
- In the case of multiple redirections of a stream in a command, the last one prevails:

```
minishell$ ls
minishell$ echo foo > bar > baz
minishell$ ls
bar  baz
minishell$ cat bar
minishell$ cat baz
foo
minishell$
```

  Here, `bar` was created but is empty.

- The simultaneous use of a redirection, an indirection for the same file and the same command has an undefined behaviour.

7 Pipe

This part requires the *execution* part to be tested.

Your minishell must also correctly implement the *pipe* `|`:

```
minishell$ cat /etc/passwd | head -1
root:x:0:0:root:/root:/bin/zsh
minishell$ echo turanic | cat -e | cat -e
turanic$$
minishell$ cat Makefile | cat | head -2 | cat -e
# Makefile for minishell$
CC= gcc$
```

8 Advanced grammar

Here are examples of commands your minishell must be able to execute without problem:

```
minishell$ ls -l -a -F /usr | cat -e | grep ^drw | tail -4
drwxr-xr-x   2 root    wheel  5120 Aug 24  2007 sbin/$
drwxr-xr-x   6 daemon  wheel   512 Sep  3 14:58 school/$
drwxr-xr-x   6 daemon  wheel   512 Sep  3 14:58 school.old/$
drwxr-xr-x  27 root    wheel   512 Aug 24  2007 share/$
minishell$ ypcat passwd | grep /a1 > a1_list
minishell$ < /etc/passwd cat -n | grep -v root | grep smmsp > /tmp/out 2> /dev/null
minishell$ cat /tmp/out
    14  smmsp:*:25:25:Sendmail Submission User:/var/spool/clientmqueue:/usr/sbin/nologin
minishell$ </etc/passwd cat -n|grep -v root|grep smmsp>/tmp/out 2>/dev/null
minishell$ cat /tmp/out
    14  smmsp:*:25:25:Sendmail Submission User:/var/spool/clientmqueue:/usr/sbin/nologin
minishell$
```

9 Environment variables

You must handle environment variables expansion. For example:

```
minishell$ /bin/echo $HOME
/u/prof/yaka
minishell$
```

The following code should also work:

```
42sh$ ./minishell
minishell$ /bin/echo $TEST

minishell$ TEST=Votai /bin/sh
sh-4.4$ /bin/echo $TEST
Votai
sh-4.4$ exit
minishell$ /bin/echo $TEST

minishell$
```

You should take a look at `setenv(3)`, `getenv(3)` and `putenv(3)`.

10 Return codes

Your minishell must support the special variable `$?` that stores the return code of the last executed command:

```
minishell$ true
minishell$ /bin/echo $?
0
minishell$ false
minishell$ /bin/echo $?
1
minishell$
```

The way is lit. The path is clear. We require only the strength to follow it.