{EPITECH}

MINISHELL1

TO THE 42SH AND BEYOND...

MINISHELL1

Preliminaries

binary name: mysh
language: C
groupe size: 1
compilation: via Makefile, including re, clean and fclean rules
authorized functions: malloc, free, exit, opendir, readdir, closedir, getcwd, chdir, fork, stat, lstat, fstat, open, close, getline, strtok, strtok_r, read, write, execve, access, isatty, wait, wait-pid, wait3, wait4, signal, kill, getpid, strerror, perror, strsignal

- ✓ The totality of your source files, except all useless files (binary, temp files, objfiles,...), must be included in your delivery.
- ✓ All the bonus files (including a potential specific Makefile) should be in a directory named bonus.
- ✓ Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Project

You have to program a **UNIX command interpreter** based on TCSH.
This is one of the first step to be able to do 42sh project.

The interpreter is expected to display a prompt ($> for example) and then wait for you to write a command line, which must be validated by a newline.
The prompt must be displayed again only after the command execution.

Only basic command lines are expected to be executed; no pipes, redirections or any other advanced features.
The commands should be those found in the path, as indicated in the PATH variable, or with the direct path to the command (Like /bin/ls).

If the command cannot be found, you must display an error message and display the prompt again.
Errors must be dealt with and must display the appropriate message on the error output.

What does TCSH write to indicate the errors ?

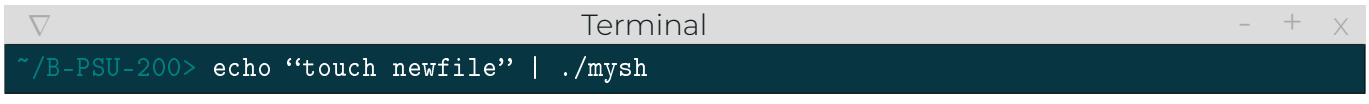
You must correctly handle the PATH and the environment (by copying and restoring the initial env).

You must implement the following builtins: **cd, setenv, unsetenv, env, exit**.

Your **env** builtin must not take any argument and your **unsetenv** builtin must not support the “*” wildcard.

Examples

You can use any prompt as we are testing your minishell as followed :
echo “command” | ./mysh



But your minishell have to work like any shells.

```
Terminal
~/B-PSU-200> ./mysh
$> pwd
/tmp
$> cd test
$> pwd
/tmp/test
```

```
Terminal
~/B-PSU-200> ./mysh
$> ./my_ls .
dir my_ls '#test#' test~
```

The exit status of your minishell must be the same as the exit status of the command you have executed.

In this example the ls command return 0 and the minishell return 0 also.

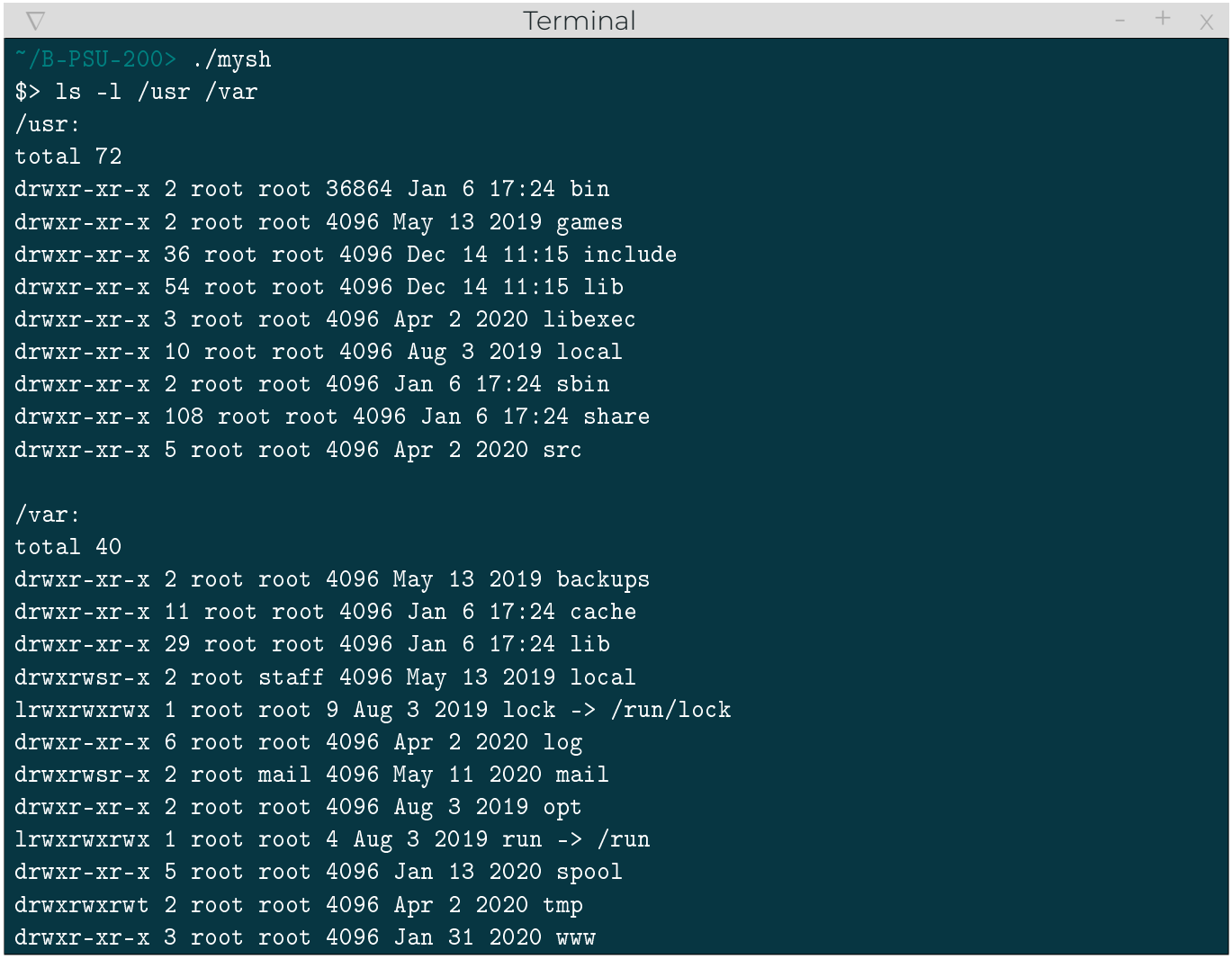
```
Terminal
~/B-PSU-200> echo "ls" | ./mysh
tata
```

```
Terminal
~/B-PSU-200> echo $?
0
```

In this one, the ls command return 2 and the minishell return 2.

```
Terminal
~/B-PSU-200> echo "ls nofile" | ./mysh
ls: cannot access 'nofile': No such file or directory
```

```
Terminal
~/B-PSU-200> echo $?
2
```

Do not hesitate to share your tests with the other students

v1