



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, waitpid, 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

```
Terminal
~/B-PSU-200> echo "touch newfile" | ./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
```

```
Terminal
~/B-PSU-200> ./mysh
$> ls -l /usr /var
/usr:
total 72
drwxr-xr-x 2 root root 36864 Jan 6 17:24 bin
drwxr-xr-x 2 root root 4096 May 13 2019 games
drwxr-xr-x 36 root root 4096 Dec 14 11:15 include
drwxr-xr-x 54 root root 4096 Dec 14 11:15 lib
drwxr-xr-x 3 root root 4096 Apr 2 2020 libexec
drwxr-xr-x 10 root root 4096 Aug 3 2019 local
drwxr-xr-x 2 root root 4096 Jan 6 17:24 sbin
drwxr-xr-x 108 root root 4096 Jan 6 17:24 share
drwxr-xr-x 5 root root 4096 Apr 2 2020 src

/var:
total 40
drwxr-xr-x 2 root root 4096 May 13 2019 backups
drwxr-xr-x 11 root root 4096 Jan 6 17:24 cache
drwxr-xr-x 29 root root 4096 Jan 6 17:24 lib
drwxrwsr-x 2 root staff 4096 May 13 2019 local
lrwxrwxrwx 1 root root 9 Aug 3 2019 lock -> /run/lock
drwxr-xr-x 6 root root 4096 Apr 2 2020 log
drwxrwsr-x 2 root mail 4096 May 11 2020 mail
drwxr-xr-x 2 root root 4096 Aug 3 2019 opt
lrwxrwxrwx 1 root root 4 Aug 3 2019 run -> /run
drwxr-xr-x 5 root root 4096 Jan 13 2020 spool
drwxrwxrwt 2 root root 4096 Apr 2 2020 tmp
drwxr-xr-x 3 root root 4096 Jan 31 2020 www
```



Do not hesitate to share your tests with the other students

{EPITECH}