B2- Unix System Programmation

B-PSU-156

42sh

man tcsh

v1.2

42sh

man tcsh

binary name:	42sh
repository name:	PSU_$YEAR_42sh
repository rights:	ramassage-tek
language:	C
group size:	4-5
compilation:	via Makefile, including re, clean and fclean rules

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- 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).

You must write a Unix SHELL.
The project consists of two sections:

1. a mandatory section, which MUST be completed:
   display a prompt, parse and execute some commands (*see below*),
2. an optional section, which will only be evaluated during the keynote (if the mandatory section is fully functionnal).

Stability and usability of the entire project will be greatly considered. It would be in your best interest to conform to the usages and customs.

Authorized functions: all functions included in the libC or the ncurses library.

Mandatory Section

This section must be **COMPLETELY FUNCTIONAL**. Otherwise your grade will be 0.
Concerning command parsing and execution, you must handle:

- spaces and tabs,
- $PATH and environment,
- errors and return value,
- redirections ('<', '>', '«' and '»'),
- pipes ('|'),
- builtins: `cd`, `echo`, `exit`, `setenv`, `unsetenv`,
- separators: ';', '&&', '||'.

For instance, you should be able to execute the following command:

```
Terminal
~/B-PSU-156> ./42sh
42sh> cd ; </etc/hosts od -c | grep xx | wc » /tmp/z -l ; cd - && echo "OK"
```

Optional Section

You will get most of your points in this section.
You have free-rein and can do what you want. However, the entire project's coherence will be taken into consideration. Once more, **stability** will be much more important that quantity. Don't include an option that will cause a problem for the rest of the program (especially for the mandatory section!).

For the different commands and compatibility (syntax), the reference shell used will be **tcsh**.

Here is a list of desired extras:

- inhibitors ('\'),
- globbings ('*', '?', '[', ']'),
- job control ('&', fg),
- backticks ('`'),
- parentheses ('(' and ')'),
- variables (local and env),
- special variables (*term*, *precmd*, *cwdcmd*, *cwd*, *ignoreof* for instance),
- history ('!'),
- aliases,
- line edition (multiline, dynamic rebinding, auto-completion dynamic,
- scripting (a bit harsh though).

These extras are not bonuses! Bonuses will be evaluated only after you've correctly implemented every item on this list!

Advice

Form a solid group!

- make sure that you can really work together (times, amount of time, personality),
- really work together as a group (together and through discussion),
- spend a lot of time analyzing things on all levels,
- compare your ideas before starting to create,
- make sure that you, and the other group members, completely understood the same thing,
- speak with other groups,
- get together frequently (at least once or twice a week) and code side-by-side.

Progress step by step!

- don't code anything if it's not clear,
- don't code anything until all of your minishells are completely functional (for all group members). We would advise you to even completely redo them as a group, in order to see how you code together,
- make completely functioning scenarios for your shell. Perform tests for everything you are going to code. Look for all the specific cases (we'll test them during your oral presentation),
- compare your case lists with other groups,
- make a clear list of the extras you would like to do and separate them into steps,
- make a general outline on paper before writing the first line,
- test everything over the course of your functionalities' development. Don't wait until you are "finished",
- don't hesitate to delete sections that seem weird, shakey, or poorly written (even if functional). This will help you out for the rest,
- don't do anything that you haven't completely understood.

When your project seems functional, have it tested by the other groups.