# B2 - Shell Programming

B-PSU-210

# 42sh

man tcsh

2.0

{EPITECH.}

# 42sh

| | |
|---|---|
| **binary name:** | 42sh |
| **repository name:** | PSU_42sh_$ACADEMICYEAR |
| **repository rights:** | ramassage-tek |
| **language:** | C |
| **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:

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

**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-210> ./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, really work together as a group (together and through discussion)

**Split the work-load intelligently:** what if a member of your team get sick? Can you finish his part of the job, or is it a "black box" for you?

**Write unit tests:** if you write them as you add features, you will be more confident about the stability of your project as the code base grows.

**Never hesitate to delete or rewrite some old code:** unit tests are there to ensure your new code has the same behavior as the old one when you're refactoring

**Git is your friend:** commit often! Do you know how to navigate in previous revisions? Do you know how to use branches?