

Terminal

The terminal I use on a daily basis is zsh. The Z Shell is a Unix shell which is in the family of the sh shell (shell command line interpreter) that has many improvements on top of bash, ksh, and tcsh. I also use the default terminal that is provided by my macbook air. I find it important to keep a minimalist setup to make sure everything is light and efficient. Zsh is installed on all of my computers (Window and macOS).

Brew

I strongly believe that brew is mandatory for programming. Whether you have macOS, Linux or Windows, you should familiarize yourself with a package manager. I use brew, because I use a macbook, but it can be chocolatey for Windows.

If Homebrew (or brew) is not installed on your computer: Linux or Windows (WSL), macOS

Oh My Zsh

The Oh my zsh is an open source project that helps us manage the Zsh configuration with plugins, themes, alias and more. To install it, simply run -

```
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

Ps- The Wiki holds additional information like cheatsheets and more.

Zsh comes with a configuration file called `.zshrc`. the file is located in the user's home directory. To edit the configuration, simple run -

```
code ~/.zshrc
```

Theme

The theme I am using is called `minimal` on the themes available. Simply make the environment variable `ZSH_THEME` to equal the theme you want-

```
ZSH_THEME="minimal"
```

Zsh syntax highlighting

My terminal has syntax highlighting which helps visualize the available commands and makes everything much more elegant. To set it up, simply add `zsh-syntax-highlighting` to your plugins array like-

```
plugins=(
  zsh-syntax-highlighting
  # ...
)
```

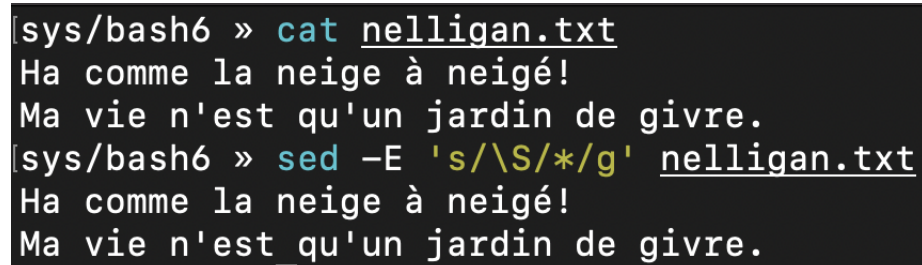
The default colors where not to my liking, so I changed the default colors with-

```
ZSH_HIGHLIGHT_STYLES[suffix-alias]=fg=cyan,underline  
ZSH_HIGHLIGHT_STYLES[precommand]=fg=cyan,underline  
ZSH_HIGHLIGHT_STYLES[arg0]=fg=cyan
```

which are added in the configuration after the aliases.

BSD and GNU

The problem with macOS is that it comes with the command line utility from the BSD world, which has different (say minimal) interpretation of commands. For example, `sed` does not accept group-

A terminal window with a dark background. The prompt is [sys/bash6 ». The first command is cat nelligan.txt, followed by its output: Ha comme la neige à neigé! and Ma vie n'est qu'un jardin de givre. The second command is sed -E 's/\S/*/g' nelligan.txt, which produces the same output as the first command, indicating it failed to replace the characters.

```
[sys/bash6 » cat nelligan.txt  
Ha comme la neige à neigé!  
Ma vie n'est qu'un jardin de givre.  
[sys/bash6 » sed -E 's/\S/*/g' nelligan.txt  
Ha comme la neige à neigé!  
Ma vie n'est qu'un jardin de givre.
```

Figure 1: Trying to replace every characters with *

As you can see, the characters aren't replaced with *. To fix this issue, I have found the GNU alternative package on the brew repository and installed them.

for `sed`, simply run-

```
brew install gnu-sed
```

A terminal window with a dark background. The prompt is sys/bash6 ». The command brew install gnu-sed is entered.

```
sys/bash6 » brew install gnu-sed
```

Figure 2: Installing gnu-sed

and add an alias to the Zsh configuration to make the `gnu sed` the default `sed`-

```
alias sed="gsed"
```

And now, the `sed` command works as expected

Change each characters to a *.

I did the same for `grep` and `egrep` with-

```
brew install grep
```

and aliases are-

```
[sys/bash6 » cat nelligan.txt
Ha comme la neige à neigé!
Ma vie n'est qu'un jardin de givre.
[sys/bash6 » sed -E 's/\S*/g' nelligan.txt
** ***** ** ***** * *****
** *** ***** ***** ***** ** *****
```

Figure 3: GNU sed works

```
alias egrep="grep --color"
alias grep="grep --color"
```

The `--color` is because it was not displaying the reddish color when using `grep`.

```
[sys/bash6 » grep 'neige' nelligan.txt
Ha comme la neige à neigé!
```

Figure 4: Example of GNU grep

Git

For my `zsh` plugins, I also use the `git` plugin. It helps with the shortcuts when typing certain commands. Furthermore, it makes `git` command available on the terminal. The shortcuts can be found here. Also, I like to add my own alias to `git push --set-upstream origin $(current_branch)`. Instead of `gpsup`, it is `gps`. The `git status` command is `gs` with a alias.

```
plugins=(
  git
  # ...
)
alias gps="gpsup"
alias gs="git status"
```

Zsh autosuggestions

I also get the `zsh-autosuggestions` to make sure I can keep track of my last commands. It saves me a lot of time when writing complex (repetitive) commands.

```
plugins=(
  zsh-autosuggestions
```

```
# ...  
)
```

A terminal window with a dark background. The prompt is '~ »'. The command 'git commit -m "docs: add readme"' is entered. The word 'git' is highlighted in green, and 'commit' is highlighted in grey, indicating a shell completion suggestion.

```
~ » git commit -m "docs: add readme"
```

Figure 5: Image showing the suggestion

Gitignore

A new plugin in my configuration is the **gitignore** plugin. It helps me create ignore files. For example, I need a ignore file for node, I simply run-

```
gi node >> .gitignore
```

Here is the plugins array-

```
plugins=(  
  gitignore  
  # ...  
)
```

In summary, we saw how to configure your terminal to be minimal, whilst still having access to powerful tools such as oh my zsh.