WORKSHOP

Travailler dans un terminal

Alexandre CALVET – Arnaud GUEROUT

2021-2022

Contexte:

Afin de développer, vous avez pris l’habitude d’utiliser emacs, code, bash... Mais êtes-vous allé plus loin? Avoir des informations supplémentaires dans votre terminal, savoir en un seul coup d’oeil le retour de votre programme et bien plus encore.

Toutes les tâches ne nécessitent pas de lancer code, un simple emacs, une utilisation de nano et le problème est réglé.

Sommaire:

Un environnement adapté

Oh my zsh

Par défaut, vous utilisez bash. Une alternative à ce shell : zsh. Son avantage est la surcouche que vous pouvez ajouter : Oh my zsh (https://github.com/ohmyzsh).

Cette surcouche open source vous offre un ensemble de fonctionnalités supplémentaires et des thémes prêt à l'emploi (https://github.com/ohmyzsh/ohmyzsh/wiki/Themes).

Comment l'installer ? Rien de plus simple qu'une simple commande.

Method	Command
curl	`sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"`
wget	`sh -c "$(wget -O- https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"`
fetch	`sh -c "$(fetch -o - https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"`

Pas besoin de faire chacune de ces methodes, prenez celle que vous voulez.

A partir de maintenant vous utiliserez .zshrc et non plus .bashrc pour configurer votre shell. Vous pouvez déjà choisir votre thème : pour ca changez le nom utilisez dans la variable ZSH_THEME (ligne 11 du .zshrc).

Prenez un peu de temps pour lire votre .zshrc et activer les options qui vous interressent.

Nous allons ajouter deux plugins pour faciliter le dev: une auto complétion afin de retrouver rapidement les commandes déjà utilisée et une coloration de la syntaxe afin de savoir si une commande existe avant même de l'exécuter.

ZSH-AUTOSUGGESTIONS:

git clone https://github.com/zsh-users/zsh-autosuggestions.git $ZSH_CUSTOM/plugins/zsh-autosuggestions

ZSH-ZYNTAX-HIGHLIGHTING:

git clone https://github.com/zsh-users/zsh-syntax-highlighting.git $ZSH_CUSTOM/plugins/zsh-syntax-highlighting

Ouvrez maintenant votre .zshrc et trouvez la ligne avec "plugins=(git)" (ligne 71 par défaut). Ajoutez "zsh-autosuggestions zsh-syntax-highlighting" dans les parenthèses. Il ne vous reste plus qu'à reset votre shell (utilisez la commande "source").

Si vous voulez revenir en arrière utiliser la commande chsh pour reprendre bash.

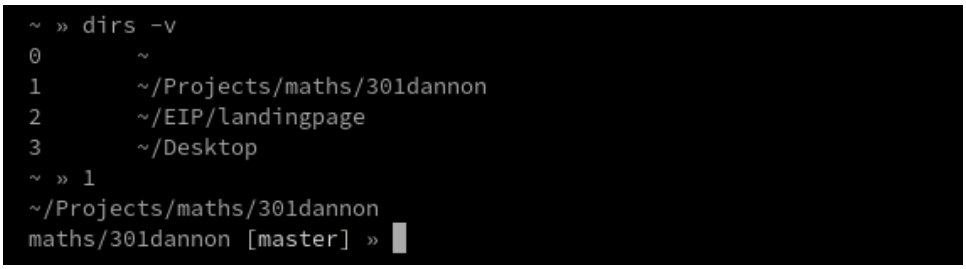
Utilisez efficacement zsh

Voyons maintenant quelques tips utils pour gagner du temps.

La navigation avec zsh

Tout d'abord oh-my-zsh vous permet de choisir directement une complétion de la commande en appuyant sur TAB. Par exemple pour faire un cd:

En plus de la navigation simplifiée par l'autocomplétion de oh-my-szh, vous pouvez utilisez la commande "dirs", qui permet de lister vos récents dossiers utilisés et d'y retourner rapidement.



Ce n'est pas de la magie, c'est une utilisation de cd intégrée dans votre shell. Pour vous en rendre compte tapez "alias | grep cd"

Les alias:

Voyons maintenant une autre fonctionnalité, les alias. Les alias vous permettent de créer des raccourcis pour vos commandes. Par exemple pour avoir les droits et les fichiers cachés lorsque l'on fait un ls:

```
alias ls='ls -la'
```

Il existe également alias basés sur les extensions de fichiers, qui permettent de définir comment ouvrir un type de fichier:

```
alias -s {c,h}=emacs
```

Enfin nous pouvons faire des alias globaux. La différence avec un alias classique et que ceux la vont être utilisé n'importe où dans la ligne de commande et pas juste au debut:

```
alias -g G='| grep -i'
```

Vous pouvez maintenant recharger votre shell et tester vos alias. Votre ls fait maintenant un ls –la, en tapant le nom d'un fichier .c vous l'ouvrirez avec emacs, et si vous utilisez G vous pourrez faire une recherche rapide avec grep:

```
~ » ls -l G ba
-rwxr-xr-x 1 tic tic  3990472  7 oct.  11:25 babel_client
-rwxr-xr-x 1 tic tic 89631317  3 sept. 15:58 balenaEtcher-1.5.107-x64.AppImage
```

Voici une petite liste d'alias pouvant vous aider:

```
alias zshconfig="nano ~/.zshrc; source ~/.zshrc"
alias cls='clear; ls -l'
alias ne='emacs -nw'
alias please='sudo'
alias reload='source ~/.zshrc'
```

L'historique des commandes:

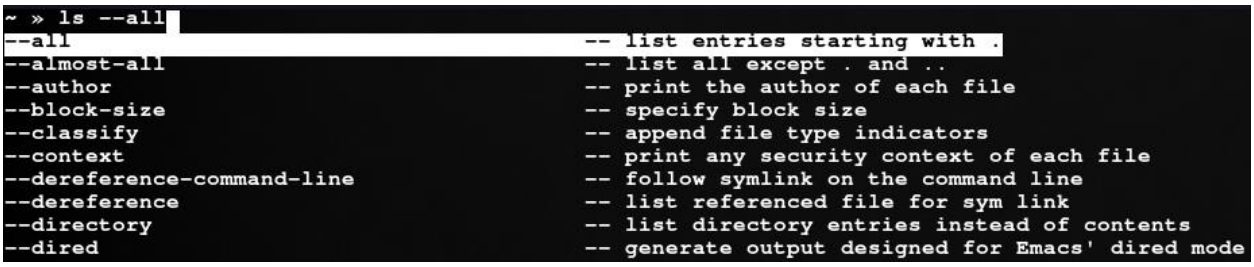
Quand vous entrez une commande, celle-ci va être enregistrée dans votre historique. C'est notamment grâce à cela que vous pouvez retrouver une commande en appuyant sur la flèche du haut.

Cette fonctionnalité est utilisable même avec une commande en particulier ! Par exemple prenons ls. Si vous tapez ls et que vous remontez l'historique avec la flèche du haut, vous retrouverez uniquement les commandes ls déjà entrées.

Pour effacer votre historique, vous allez utiliser la commande "history -c" et recharger votre terminal.

TAB

Nous avons vu qu'utilisez TAB permet de trouvez un dossier avec cd. Mais ce n'est pas son utilisation. Utilisez TAB vous permet de compléter n'importe quelle commande et de parcourir les options de cette dernière:



Emacs –nw

Vous pouvez utiliser emacs directement dans votre terminal avec son option –no-window-system, aussi utilisable via –nw.

Cette option vous permet de lancer emacs sans quitter le terminal. Cela vous permet de mettre emacs en suspend pour tester avec un Ctrl-Z et reprendre votre travail avec la commande “fg”.

Utiliser emacs c’est bien, mais par défaut ce logiciel n’est pas aussi attirant qu’un IDE comme code, pourtant nous pouvons le rendre tout aussi puissant.

Customiser Emacs

La base de la customisation Emacs :

Afin de rendre votre première année plus agréable, il est indispensable de customiser votre Emacs. Cela peut passer par plusieurs moyens : l’ajout de package et la modification de votre thème.

Tout dabord l’ajout de package. Le plus simple est de passer par un gestionnaire de package, le plus connus étant Melpa.

La premiere chose a faire est d’allé sur l’intranet et de telecharger l’archive epitech-emacs. Decompressez la et lancez le script qui se trouve à l’interieur. Celui-ci vous permet d’avoir une installation “de base” plus ou moins conforme aux standards Epitech.

Une fois cela fait, allez dans l’onglet “Getting Started” de Melpa et copiez ceci:

```
(require 'package)
(add-to-list 'package-archives '("melpa" . "https://melpa.org/packages/") t)
;; Comment/uncomment this line to enable MELPA Stable if desired.  See `package-archive-p
riorities`
;; and `package-pinned-packages`. Most users will not need or want to do this.
;;(add-to-list 'package-archives '("melpa-stable" . "https://stable.melpa.org/packages/")
t)
(package-initialize)
```

Il faut ensuite le coller en base de votre fichier .emacs qui se trouve dans votre home “emacs ~/.emacs” (si vous ne l’avez pas, créez le).

Une fois cela fait il vous suffit de commenter la seconde ligne (ajoutez “;;” devant) et de décommenter l’avant dernière ligne (retirez les “;;”)

Fermez votre emacs et re-ouvrez le en tappant simplement “emacs” dans un terminal.
Si vous avez bien fait les chose vous ne devriez avoir aucun warning à ce moment la.

Appuyez donc sur “Alt”+”X” et afin d’ouvrir un invité de commande et tappez “package-refresh-contents” (appuyez sur tab il y a de l’auto-completion)

Comment installer un package ?

Une fois Melpa installé, rendez vous sur l’onglet “package” de melpa et trouvez un package que vous souhaitez avoir.
Pour l’exemple nous allons installer le package “NeoTree” (Lien Melpa) (Lien GitHub)
Appuyez a nouveau sur “alt” et “X”, ensuite tappez “package-install” et appuyez sur “Enter”. Tappez maintenant le nom de votre package, ici “neotree”

Une fois que le package sera installé, il y aura marqué “Done” (ou plus rien si le message est deja partis). Il ne vous reste plus qu’une chose à faire, aller sur le lien “source de votre package (pour NeoTree c’est le lien github donne plus haut) et faîtes en sorte que le package se lance

dès l'ouverture de emacs. Il y a en general une section "copiez ceci dans votre .emacs ou votre init.el"

Make sure you have something like the following in your Emacs startup file (`~/.emacs.d/init.el`, or `~/.emacs`):

```
(add-to-list 'package-archives
             '("melpa" . "http://melpa.org/packages/"))
```

Ok c'est cool les packages mais pour le theme ?

Si vous ne voulez pas vous embeter a creer votre theme, je vous invite a suivre ce lien : https://pawelbx.github.io/emacs-theme-gallery/

Sinon vous pouvez créer un thème sur ce site: https://mswift42.github.io/themecreator/

Une fois votre theme créé, créez un dossier "themes" dans votre dossier .emacs.d situé dans votre home.
Une fois le dossier creer, copier collez votre theme dans ce dossier et ajoutez ces lignes a votre .emacs :

```
;;
;; Theme
;;
(add-to-list 'custom-theme-load-path "~/.emacs.d/themes/")
(load-theme 'my t)
```

Faîtes attention à remplacer "my" par le nom que vous aurez donné à votre theme dans ThemeCreator

Pour finir, voila quelques lignes que j'ai personnellement ajouté à mon Emacs pour la norme :

```
;;Norme Epitech
;;
(require 'whitespace)
 (setq whitespace-style
       (quote (face trailing tab-mark lines tail)))
       (add-hook 'find-file-hook 'whitespace-mode)

       (global-set-key (kbd "<f6>") 'whitespace-mode)
       (add-to-list 'load-path "~/.emacs.d/lisp")
       (global-set-key (kbd "<f7>") 'global-linum-mode)
       (global-linum-mode t)
       (setq column-number-mode t)
       (setq linum-format "%4d \u2502 ")

(electric-pair-mode 1)
(show-paren-mode 1)

;;
;;Spaces instead of Tabs
;;
(setq-default indent-tabs-mode nil)
(setq-default c-basic-offset 4)
(setq-default tab-width 4)
(setq indent-line-function 'insert-tab)
```

Enfin, vous pouvez retrouver ici quelques packages que j'ai utilisé durant ma première année:

- NeoTree (Package que l'on a utilise pour ce workshop)
- Smooth Scrolling (pas smooth scrool attention)
- Auto Complete
- Highlight ToDo

Si vous vous formez à doxygen, sachez que je me suis créé un package doxygen que je peux vous partager si vous le souhaitez.

PS: vous trouverez la liste de tous les raccourcis sur ce site https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf

Nano

Nano est un autre éditeur souvent installé par défaut. Il n’a pas de couleur pour le moment. Commençons par récupérer le fichier de configuration:

cp /etc/nanorc ~/.nanorc

Ajoutons maintenant de la couleur:

find /usr/share/nano/ -iname ".nanorc" -exec echo include {} \; >> ~/.nanorc*

Avec cette commande nous avons ajouté une coloration en fonction des types de fichiers.

Nous pouvons maintenant ouvrir notre .nanorc et utiliser les options que nous voulons. Pour cela prenez le temps de lire les options et de tester.