

# ZSH: Pimp my shell !

Philippe Wambeke - LoliGrUB (19 octobre 2019)

## ZSH, c'est quoi ?

ZSH est un "shell" qui offre des fonctionnalités simples et puissantes avec un grand niveau de personnalisation.

## Et c'est quoi un shell ?

Un "shell" est un logiciel qui permet à un humain de communiquer avec un ordinateur. On parle d'"interface homme-machine" en bon français.

Il en existe de deux types:

- Les shells graphiques
- Les shells textuels

## Les shells graphiques

Tout le monde connaît: c'est l'interface graphique qui permet l'interaction avec l'ordinateur.

- KDE, Gnome, XFCE, Mate (fork de gnome 2), Cinnamon (fork de gnome 3), ...
- Windows

## Les shells textes

C'est la fenêtre noire qui fait peur où il faut taper des trucs bizarres.

- sh (Bourne Shell) - Stephen Bourne: 1977
- csh (C Shell) - Bill Joy: 1978
- tcsh - version moderne de csh. Systèmes BSD
- ksh (Korn Shell) - David Korn: 1983. Compatible avec sh. Inclut des fonctionnalités de csh
- Bash (Bourne Again Shell) - Brian Fox: 1988. Implémentation libre de sh pour le projet GNU
- Zsh (Z Shell) - Paul Falstad: 1990. Le shell ultime.

## A quoi ça ressemble ?

### Exemple de shell traditionnel

### Exemple de zsh

## Installation

Deux grandes étapes à suivre:

- Installer le paquet zsh: inclus dans toutes les distributions GNU/Linux
- Installer Oh my zsh !

### Installation du paquet

Debian et dérivés:

```
sudo apt-get install zsh
```

Fedora et dérivés:

```
sudo yum install zsh
```

## Oh my zsh

Oh my zsh est un framework de personnalisation qui permet de gérer facilement la configuration de zsh. Inclut:

- Plus de 200 plugins optionnels
- Plus de 140 thèmes
- Mise à jour automatique

### Installation de Oh my zsh

Facile: un coup de copier-coller dans n'importe quel shell ;)

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

Après on se retrouve dans zsh, mais c'est encore assez éloigné de la capture d'écran.

## PowerLevel9K

PowerLevel9K est un des nombreux thèmes pour zsh. Si le paquet existe pour votre distribution, l'installation se fait avec un simple:

```
sudo apt-get install powerlevel9k  
sudo yum install powerlevel9k  
sudo pacman -S powerlevel9k  
...
```

Sinon, il est possible de l'installer par:

```
git clone https://github.com/bhilburn/powerlevel9k.git ~/.oh-my-zsh/custom/themes/powerlevel9k
```

## Les polices PowerLine

Ces polices de caractères sont nécessaires pour l'affichage correct des petits symboles. Il faut donc les installer via:

```
sudo apt-get install fonts-powerline
sudo yum install powerline
sudo pacman -S powerline-fonts
...
```

## Les polices comprenant les glyphes

La police PowerLine ne contient pas suffisamment de glyphes. Il faut donc installer une police de caractère spéciale faisant partie de la collection <https://github.com/ryanoasis/nerd-fonts> ["nerd font"].

Parmi toutes les <https://github.com/ryanoasis/nerd-fonts/releases/tag/v2.0.0#polices>, j'ai choisi la police <https://github.com/ryanoasis/nerd-fonts/releases/download/v2.0.0/Hack.zip#Hack>:

```
wget
"https://github.com/ryanoasis/nerd-fonts/releases/download/v2.0.0/Hack.zip"
```

## Installer la police Hack

Pour installer cette nouvelle police de caractères, suivre les étapes suivantes:

```
mkdir hack
unzip Hack.zip -d hack
sudo mv hack /usr/share/fonts
fc-cache
```

## La configuration

Maintenant le gros morceau: la configuration. Tout se fait dans le fichier ~/.zshrc

```
export TERM="xterm-256color"
ZSH_THEME="powerlevel9k/powerlevel9k"
```

Si le thème PowerLevel9k est packagé par la distribution, la seconde ligne est inutile.

## Premier plugin: autosuggestions

Ce plugin permet d'afficher des suggestions basées sur l'historique des commandes. Installation:

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

Et activation (toujours dans le fichier ~/.zshrc):

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

Après, une suggestion basée sur l'historique apparaîtra au fur et à mesure de la frappe. L'appui sur la flèche droite (ou end) validera la suggestion.

## Un historique qui marche

Et tant qu'on y est, on peaufine la gestion de l'historique:

```
HISTSIZE=3000          # Nombre de lignes en mémoire
HISTFILE=~/.zsh_history # Fichier de sauvegarde
SAVEHIST=3000          # Nombre d'entrées à enregistrer
HISTDUP=erase          # Suppression des doublons
setopt HIST_IGNORE_SPACE # Ne pas enregistrer les commandes commençant par
un blanc
setopt appendhistory    # Ajout des entrées en mode append
setopt sharehistory     # Partage de l'historique entre terminaux
setopt incappendhistory # Ajout immédiat à l'historique (pas à la fermeture
du terminal)
setopt correct          # Propose la commande la plus proche en cas
d'erreur de frappe
```

## Deuxième plugin: coloration syntaxique

Ce plugin permet d'avoir une coloration syntaxique à mesure de la frappe au clavier. Installation:

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

Et activation (toujours dans le fichier ~/.zshrc):

```
plugins=(
  zsh-autosuggestions
  zsh-syntax-highlighting
)
```

## Troisième plugin: les pages man en couleur

Ce plugin permet la consultation des pages man en couleur.

Installation: rien à faire car il est déjà fourni par Oh my zsh !

Ne reste plus qu'à l'activer: (toujours dans le fichier ~/.zshrc):

```
plugins=(  
  zsh-autosuggestions  
  zsh-syntax-highlighting  
  colored-man-pages  
)
```

## Personnalisation du thème

### Ajout de l'icone du système

Ajouter cette ligne dans ~/.zshrc:

```
POWERLEVEL9K_LEFT_PROMPT_ELEMENTS=(os_icon dir)
```

Si un texte s'affiche au lieu de l'icône (Arc, Deb,...) il est possible de forcer l'icône à utiliser:

```
os_icon='\uf306'
```

Les icônes et leur numéros peuvent être recherchés sur le site <http://nerdfonts.com>

### Un prompt mutli-lignes

Ajouter ceci dans ~/.zshrc:

```
POWERLEVEL9K_PROMPT_ON_NEWLINE=true  
POWERLEVEL9K_PROMPT_ADD_NEWLINE=true  
POWERLEVEL9K_MULTILINE_FIRST_PROMPT_PREFIX="%F{014}\u256D\u2500%f"  
POWERLEVEL9K_MULTILINE_LAST_PROMPT_PREFIX="%F{014}\u2570%f{cyan}\uF460%f{073}  
}\uF460%f{109}\uF460%f "
```

### Partie droite du prompt

Ajouter ceci dans ~/.zshrc:

```
POWERLEVEL9K_TIME_FORMAT="%D{\ue383 %H:%M \uf073 %d.%m.%y}"  
POWERLEVEL9K_RIGHT_PROMPT_ELEMENTS=(status background_jobs time battery)
```

## Toujours plus fort !

Modifier la couleur en fonction de l'état de charge de la batterie:

```
POWERLEVEL9K_BATTERY_STAGES=('$\uf244' '$\uf243' '$\uf242' '$\uf241'  
'$\uf240')  
POWERLEVEL9K_BATTERY_LEVEL_BACKGROUND=(196 208 226 118 46)  
POWERLEVEL9K_BATTERY_LOW_FOREGROUND=232  
POWERLEVEL9K_BATTERY_CHARGING_FOREGROUND=232  
POWERLEVEL9K_BATTERY_CHARGED_FOREGROUND=232  
POWERLEVEL9K_BATTERY_DISCONNECTED_FOREGROUND=232
```

Consultez la <https://github.com/bhilburn/powerlevel9k/wiki/Stylizing-Your-Prompt>[référence] et autres exemples pour styliser le prompt.

## Et maintenant, qu'est-ce qu'on en fait ?

Quelques exemples d'usage de zsh:

### Navigation dans les dossiers

- navigation rapide dans les dossiers grâce aux combinaisons uniques (zsh n'est pas case-sensitive !)
- un deuxième appui sur la touche TAB présente une liste de possibilités dans laquelle on peut naviguer
- pas besoin de faire “cd dossier” pour aller dans “dossier”
- les horribles commandes du genre “cd ../../..” pour remonter de 3 niveaux sont remplacées par “...”
- pour retourner dans le 'n' dernier dossier, il suffit d'un simple “cd -n”

### Aide à la saisie

- Remplacement des variables: oubliez “echo \${ma\_variable}”: tapez directement “\${ma\_variable}” suivi de la touche TAB
- Proposition de commande si faute de frappe
- Coloration syntaxique en cours de frappe (commandes ET variables)
- besoin de voir les fichiers du dossier pendant la frappe ? appuyer deux fois sur la touche TAB
- besoin de connaître les arguments d'une commande sans quitter le prompt ? taper la commande suivie d'un tiret et appuyer 2 fois sur TAB

### Un historique qui tue

- L'historique fonctionne toujours même avec plusieurs instances de zsh simultanées
- Le plugin de suggestion est juste génial
- Rappel de la dernière commande qui commence par ce qu'on a tapé: flèche vers le haut
- CTRL+R de bash fonctionne aussi dans ZSH

## La mort de find

zsh propose un système de filtres (très) puissant.

Recherche de tous les fichiers sh de manière récursive

```
ls **/*.sh
```

Recherche de tous les fichiers modifiés il y a moins d'une heure

```
ls **/*(mh-1)
```

Recherche des fichiers de plus de 1 Mio dans le dossier courant:

```
ls *(Lm+1)
```

## Pour aller plus loin

- [https://ohmyz.sh/\[ohmyzsh\]](https://ohmyz.sh/[ohmyzsh])
- <https://github.com/robbyrussell/oh-my-zsh> [ohmyzsh depuis github]
- <https://github.com/bhilburn/powerlevel9k> [Personnalisation]
- <http://reasoniamhere.com/2014/01/11/outrageously-useful-tips-to-master-your-z-shell/> [Utilisation]
- <https://github.com/romkatv/powerlevel10k> [powerlevel10k]

## Merci

Questions ?

From:  
<https://www.loligrub.be/wiki/> - LoLiGrUB

Permanent link:  
<https://www.loligrub.be/wiki/atelier20191019-zsh-pimp-my-shell-run?rev=1570779084>

Last update: 2019/10/11 07:31

