Linux Worflow

> Différentes techniques pour gagner du temps sur un terminal linux.

Fichier rc

Le fichier **rc** (pour run commands) est un fichier qui est situé dans le home directory (correspondand à la variable `$HOME` du shell) et est exécuté systématiquement à l'ouverture du shell de l'utilisateur. Il permet de customiser différentes fonctionnalités du terminal (couleurs, invite de commandes, titre de la fenêtre) ainsi que de sourcer (charger) d'autres fichiers de configuration.

```
# on source un fichier avec source
source /home/user/.aliases.sh
```

Dans le fichier rc, on peut aussi modifier la variable `$PATH` ce qui permet de rendre des exécutables accessibles par leur nom à l'utilisateur (sans avoir à donner le chemin absolu du fichier).
Le fichier rc est différent selon chaque shell : ainsi pour bash, il s'agit de *~/.bashrc* et pour zsh de *~/.zshrc*

Alias et fonctions utilisateurs

Le fichier rc est souvent utilisé pour sourcer un ou plusieurs fichiers d'alias et de fonctions utilisateurs.
Sous linux il est possible de définir des alias de commandes avec cette syntaxe :

```
alias commande="<une commande très longue et compliquée à mémoriser>"
```

Le shell étant avant tout un langage de programmation, il est aussi possible de définir des fonctions pour opérer des tâches plus complexes

par exemple une fonction qui push des modifications sur un repository git:

```
function dotupdate() {
    if [ -z "$1" ]
    then
        folder="dotfiles"
    else
        folder=$1
    fi
```

```
    homeshick cd $folder
    git add .
    git commit -m "added new dotfiles things"
    git push
    cd -
}
```

A noter les arguments de la fonction sont accessibles avec les variables `$1` , `$2` etc .. . La variable spéciale `$@` permet de forwarder tous les arguments à une autre fonction

tmux

Un autre moyen d'accélerer les choses sous linux est de sauvegarder plusieurs fenêtres de terminal et d'y revenir plus tard. C'est à cela que sert un multiplexeur de terminal comme tmux .
Tmux est un serveur auquels plusieurs clients (utilisateurs) peuvent se rattacher. Tmux permet d'ouvrir des **sessions** qui sont en fait un ensemble de **windows** un peu comme un navigateur est un ensemble d'onglets.
En plus de cela tmux permet de définir des **panes** qui permettent de diviser une fenêtre en plusieurs sections de différentes tailles ce qui donne la possibilité de taper des commands tout en voyant simultanément un `tail -F` d'un fichier de log.
Le véritable intérêt de tmux est qu'il garde en mémoire les sessions ouvertes et permet de s'y rattacher à n'importe quel moment. On peut ainsi retrouver une session sur un serveur plusieurs mois après l'avoir quittée.
Tmux peut être scripté pour créer une session au démarrage de l'ordinateur, comme ceci :

```
#!/bin/bash

SESSIONNAME="projet"
tmux has-session -t $SESSIONNAME &> /dev/null

if [ $? != 0 ]
 then
    # general window
    tmux new-session -s $SESSIONNAME -n "general" -c "$HOME/dev/projet" -d

    # mysql window
    tmux new-window -t $SESSIONNAME -n "mysql"
    tmux send-keys -t $SESSIONNAME "mysql -u homestead -psecret projet" C-m

    # logs window
    tmux new-window -t $SESSIONNAME -n "logs"
    tmux split-window -h -p 50
    tmux send-keys -t $SESSIONNAME "tail -F /var/log/nginx/projet.dev-ssl-error.log error" C-m
    tmux split-window -t %2 -v -p 50
    tmux send-keys -t $SESSIONNAME "sudo vim /etc/nginx/sites-available/projet.dev" C-m

fi

tmux attach -t $SESSIONNAME
```

Pour plus d'informations Voici un guide sympathique

Accélérer le déploiement d’un environement shell

Frameworks shell

De nombreux projets collaboratifs existent centrés sur les manières de cutomiser un shell. Un exemple est oh-my-zsh qui une fois installé permet de charger divers plugins qui ajoutent de nombreuses commandes utiles au shell ou des fonctionnalités comme l’autocomplétion ou la colorisation.

Shell package management

Afin de pouvoir exporter sa configuration facilement, on peut utiliser un système de synchronisation de fichiers. Une possibilité est d’utiliser git manuellement en créant un repository dans son $HOME. Une autre est d'utiliser un système comme [homeshick](https://github.com/andsens/homeshick) qui utilise aussi git ainsi que des liens symboliques pour installer rapidement une configuration shell dans son $ HOME.

En une commande

Pour aller encore plus vite, il peut être utile de créer un fichier de code hosté (comme un gist), de le télécharger et de l’exécuter en arrivant sur un serveur.
Cette commande va installer zsh, tmux, vim, git puis oh-my-zsh. Elle va ensuite télécharger des fichiers de configuration (.zshrc, fichiers d’aliases) et les linker dans le $HOME :

```
sh -c "$(curl -fsSL https://gist.githubusercontent.com/baxson/b145bbdbe44afa297515737521e88c7c/raw/44b0aa6ee93ddcb1e45900e1d7da09b370de6603/install%2520shell)"
```