

# CLI is fun !

Philippe Wambeke - LoLiGrUB (20 novembre 2021)

## Préambule

La ligne de commande, c'est magique. Il existe des tas de sortilèges. Si vous êtes perdu, vous pouvez toujours demander votre chemin à ``man`` (le manuel).

## Exemple 1

Démarrage en douceur:

Affichage des fichiers triés par extension:

---

```
ls -lX
```

---

Beuh, c'est moche ! Installons exa !

La commande devient:

---

```
exa -l -icons -sort=extension
```

---

## Un petit mot concernant les arguments

Lorsqu'on appelle une commande avec des arguments, quelle est la différence entre ``-h`` et ``-human-readable`` ?

[%step] \* ``-h``: c'est une option courte (1 lettre): permet d'ajouter d'autres options (`-hla ...`) \*  
``-human-readable``: c'est une option longue: plus facile à lire / retenir

## Un petit alias peut-être ?

exa c'est bien, mais c'est long à taper: créons un alias. Dans votre fichier `~/.bashrc`` ou `~/.zshrc``, ajoutez:

---

```
alias ll="exa -l -icons -group-directories-first -sort=extension"
```

---

## Quelques trucs de base

Notations particulières couramment utilisées:

[%step] \* dossiers ``.`` et `..`` : dossier courant et parent \* `~`` : raccourci pour signifier “mon dossier personnel” (home) \* `/`` : caractère de séparation de dossiers dans un chemin \* fichiers commençant par ``.`` : fichiers cachés

## Exemple 2

Détecter un “trou” dans une séquence de fichiers:

---

```
ls PW1_{4700..4750}.jpg > /dev/null
```

---

## Les redirections

Chaque programme peut rediriger la sortie écran (stdout) vers un fichier via l'opérateur `>``.

[%step] \* `/dev/null`` : pseudo-fichier qui ignore tout ce qu'on lui envoie \* les erreurs restent visibles à l'écran: seuls sont affichés les fichiers manquants

## Exemple 3

On passe la deuxième:

Dans un log de serveur web (nginx), affichage des 100 urls les plus consultées:

---

```
awk '{print $7}' access.log | sort | uniq -c | sort -rn | head -100
```

---

## awk

Du nom de ses concepteurs: Al Aho, Peter Weinberger et Brian Kernighan. Outil de traitement et d'extraction de texte possédant son propre langage.

[https://en.wikipedia.org/wiki/The\\_AWK\\_Programming\\_Language](https://en.wikipedia.org/wiki/The_AWK_Programming_Language)[The AWK Programming Language]

En gros, les mots sont séparés par un caractère blanc et sont numérotés de `$1`` à `$x``.

La ligne `{print $7}`` signifie:

affiche le septième champ (l'url).

## Le pipe

Concept clé du shell UNIX, le pipe permet de rediriger la sortie d'un programme vers l'entrée d'un autre.

Il est représenté par le caractère `|`.

## Exemple 4: encore awk

Dans un log de serveur web (nginx), affichage des 30 urls générant le plus de code http 404:

---

```
awk '$9 == "404" {print $7}' access.log | sort | uniq -c \ | sort -rn | head -n 30
```

---

La ligne `'\$9 == "404" {print \$7}` signifie:

Si le neuvième champ de chaque ligne est `404`, alors affiche le septième champ.

## Exemple 5

Générateur de phrase de passe composées de 2 mots:

---

```
look . | grep -E "[a-z]{4,8}$" | shuf | head -40 | xargs -n2
```

---

### look

Outil (apparu dans l'édition 7 de UNIX) permettant de rechercher un mot dans un fichier.

Si aucun fichier n'est spécifié, recherche dans un dictionnaire.

`.` signifie "n'importe quel terme"

### grep

Get Regular Expression and Print: recherche toute chaîne répondant à l'expression régulière et l'affiche.

[quote, Wikipedia] Chaîne de caractères, qui décrit selon une syntaxe précise, un ensemble de chaînes de caractères possibles.

## N'importe quel mot de 4 à 8 lettres

[%step] \* `^` : rien avant \* `[a-z]` : n'importe quelle lettre de a jusqu'à z \* `{4,8}` : répétée de 4 à 8 fois \* `\$` : rien après

## xargs

Parfois, il n'est pas possible que la sortie d'une commande corresponde à l'entrée d'une autre. xargs permet de se sortir de situations parfois difficiles où il n'est pas possible d'enchaîner les commandes avec des `|`.

Par défaut, xargs affiche ce qu'il reçoit sur 1 ligne. L'argument -n2 lui indique de grouper 2 éléments par ligne.

---

```
cd /usr/bin ; ls -1 | shuf | xargs man
```

---

## Autre exemple d'expression régulière

Afficher toutes les lignes qui ne sont pas des commentaires dans un fichier de configuration:

---

```
grep '^[^#]' /etc/pacman.conf
```

---

[%step] \* `^` : rien avant \* `[^x]` : qui n'est pas le caractère `x`.

## Du fun, du fun, du fun

Ça ne sert à rien, mais c'est tellement bien !

Les outils indispensables:

[%step] \* Mettez de la couleur dans vos terminaux: `lolcat` \* Inspectez votre machine: `neofetch` \* Réalisez des bannières avec style: `figlet` \* Invitez une vache dans le terminal: `cowsay` \* Faites parler chuck norris: `fortune-mod-chucknorris`

## La météo

Rapide, facile et sans pub:

---

```
curl fr.wttr.in/Boussu
```

---

curl: outil d'interrogation de serveur web en ligne de commande.

## Base combo

---

neofetch catimg loligrub-asbl.png chuck | cowsay | lolcat -F 1 figlet -tc -f shadow "Merci de votre attention \!" | lolcat

---

La sortie de figlet peut être redirigée vers `/etc/motd` (message of the day).

## Ultra combo !

---

yes "\$(seq 231 -1 16)" | while read i; do printf "\x1b[48;5;\${i}m\n";\

```
sleep .03; done
```

---

---

grep -ao "[/\]" /dev/urandom | sed -e 's,\\,\\,' -e 's,/,/, ' | \

```
tr -d \\n | lolcat -F 0.001
```

---

---

## Le meilleur pour la fin

---

for p in {36..1..4}; do espeak-ng -v en -p \$p\

```
"We are the Borg. Lower your shields and surrender your ships...\nYour biological and technological distinctiveness will be added to our own.\nResistance is futile."\n& sleep 0.007; done
```

---

---

yes \$COLUMNS \$LINES|awk 'BEGIN{x=y=e=f=1}{if(x==\$1||!x)\

```
{e*=-1};if(y==$2||!y){f*=-1};x+=e;y+=f;\ printf "\033[%s;%sH",y,x;system("sleep .02")}'
```

## Quelques références

\* [https://en.wikipedia.org/wiki/The\\_AWK\\_Programming\\_Language](https://en.wikipedia.org/wiki/The_AWK_Programming_Language)[The AWK Programming Language] \*  
\* <https://www.rexegg.com/regex-quickstart.html>[Regex Cheat Sheet] \*  
\* <https://mywiki.woledge.org/BashGuide>[Bash Guide] \*  
\* <https://blog.zenika.com/2019/02/14/shell-mon-amour/>[Shell mon amour] \*  
\* <https://adamdrake.com/command-line-tools-can-be-235x-faster-than-your-hadoop-cluster.html>[Command-line Tools can be 235x Faster than your Hadoop Cluster]

## Merci

Questions ?

From:

<https://loligrub.be/wiki/> - **LoLiGrUB**

Permanent link:

[https://loligrub.be/wiki/atelier20211120-cli\\_is\\_fun-run?rev=1638791175](https://loligrub.be/wiki/atelier20211120-cli_is_fun-run?rev=1638791175)

Last update: **2021/12/06 11:46**

