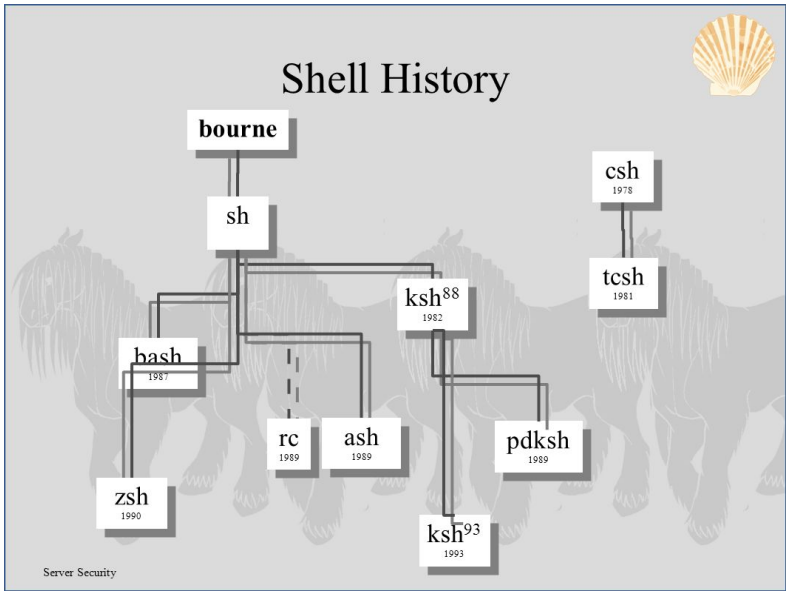
Shell et ligne de commandes

Christophe Brun <christophe@cbrun.fr>

2018

introduction

- Interpréteur de commandes en mode console
- mode interactif et mode script
- pas 1 shell, mais des shells



Définition d'une commande

`[assignation] [commande] [arguments] [redirections]`

- Assignation : de valeur ex: 'DISPLAY'
- commande à lancer
- arguments, attributs, options
- redirections entrantes, sortantes, pipe

```
>  TOTO="bonjour le monde"; echo $TOTO >test.txt
>  cat test.txt
bonjour le monde
```

Variables

- 2 types principaux
  - ordinaires à portée locale
  - d'environnement à portée globale
- une variable locale devient variable d'environnement en utilisant export

```
> mavariable="bonjour la devTeam"
> echo $mavariable
bonjour la devTeam
> env | grep mavariable
> export mavariable
> env | grep mavariable
mavariable=bonjour la devTeam
>
```

Opérations sur les variables

- `‘${var:-text}‘` : Si var existe alors contenu de var sinon text
- `‘${var:+text}‘` : Si var existe alors text sinon null
- `‘${var:?}‘` : Si var existe alors contenu de var sinon exception
- `‘${#var}‘` : Longueur du contenu de var
- `‘${var%motif}‘` : contenu le plus court à gauche répondant à motif
- `‘${var%%motif}‘` : contenu le plus long à gauche répondant à motif
- `‘${var#motif}‘` : contenu le plus court à droite répondant à motif
- `‘${var##motif}‘` : contenu le plus long à droite répondant à motif

Opérations sur les variables (exemples)

```
>  echo $mavariable
bonjour la devTeam
>  echo ${mavariable}
bonjour la devTeam
>  echo ${mavariable:-toto}
bonjour la devTeam
>  echo ${variableNoExist:-toto}
toto
>  echo ${variableNoExist:+toto}

>  echo ${mavariable:+toto}
toto
>  echo ${mavariable:?}
bonjour la devTeam
>  echo ${variableNoExist:?}
zsh: variableNoExist: parameter not set
```

Opérations sur les variables (exemples)

```
>  echo $mavariable
bonjour la devTeam
>  echo ${#mavariable}
18
>  echo ${mavariable%%la*}
bonjour
>  echo ${mavariable%a*}
bonjour la devTe
>  echo ${mavariable##*la}
devTeam
>  echo ${mavariable##*d}
evTeam
```

Variables spéciales

- `?` : code retour de la dernière commande
- `!` : pid du dernier process d'arrière plan
- `$` : pid du shell courant

Variables spéciales utilisables uniquement dans un script

- `\*` : ensemble des paramètres (dans une chaine de caractères)
- `@` : ensemble des paramètres (multi string)
- `#` : nombre de paramètres
- `\-` : option courante
- `0` : nom de la commande
- `1,2,3, ..` : paramètre un à un

Variables spéciales (exemples)

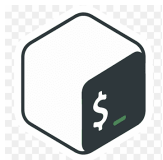
```
>  echo test
test
>  echo $?
0
>  echo $!
0
>  echo "test" &
test
[1] 13485
[1]  + 13485 done       echo "test"
>  files echo $!
13485
>  echo $$
31931
```

Substitution de commandes

- Permet de récupérer le résultat d'une commande
- 2 mode d'écriture :
  - `$(commande)`
  - `‘commande‘` : back quote

```
>  ll
total 12K
-rw-rw-r-- 1 cbrun cbrun 179 20 dec.  18:12 compagnies.csv
-rw-rw-r-- 1 cbrun cbrun 178 20 dec.  16:59 origine_compagnies.csv
-rw-rw-r-- 1 cbrun cbrun  17 21 dec.  06:53 test.txt
>  echo Il y a $(ls  | wc -l) fichiers
Il y a 3 fichiers
>  echo Il y a ‘ls -l  | wc -l‘ fichiers
Il y a 4 fichiers
```

What ?



Redirections (1/2)

- Par défaut le shell utilise 3 entrée / sorte (fichiers)
  - stdin : entrée standard
  - stdout : sortie standard
  - stderr : sortie des erreurs
- Le shell autorise les redirections de ses entrées/sorties vers fichiers, pipe, ...

Redirections (2/2)

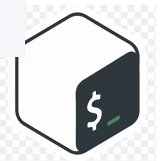
- Les modes de redirections
  - `[n]> fichier` : Redirection de la sortie n (par défaut standard vers un fichier (écrase le fichier)
  - `[n]>> fichier` : idem ci-dessus mais ajoute la sortie au fichier
  - `[n]< fichier` : Le contenu du fichier est injecté dans l'entrée standard
  - `[n1]>&n2` : Redirige la sortie n1 dans n2 : utilisé pour injecter les erreurs dans la sortie standard (puis dans un fichier)
  - `[n1]<&n2` : Redirige l'entrée de n2 dans n1
  - `[n]>\-` : Fermer la sortie standard (ou n). on utilise parfois '[n] > /dev/null'
  - `[n]<> fichier` : Redirection de la sorte et entrée n avec le fichier
  - `[n]<<chaine` : Injecte le contenu dans l'entrée n (standard par défaut) jusqu'à ce que chaine apparaisse

Exemples de redirections

```
>  ll
total 8,0K
-rw-rw-r-- 1 cbrun cbrun 179 20 dec.  18:12 compagnies.csv
-rw-rw-r-- 1 cbrun cbrun 178 20 dec.  16:59 origine_compagnies.csv
>  echo "Bonjour la dev Team" > test.txt
>  ll
total 12K
-rw-rw-r-- 1 cbrun cbrun 179 20 dec.  18:12 compagnies.csv
-rw-rw-r-- 1 cbrun cbrun 178 20 dec.  16:59 origine_compagnies.csv
-rw-rw-r-- 1 cbrun cbrun  20 21 dec.  07:07 test.txt
>  cat test.txt
Bonjour la dev Team
```

Exemples de redirections

```
>  cat << __fin__
heredoc> Bonjour
heredoc> la
heredoc> Dev
heredoc> Team
heredoc> __fin__
Bonjour
la
Dev
Team
>  echo "Et les autres" >> test.txt
>  cat test.txt
Bonjour la dev Team
Et les autres

>  rm ll.log
>  ll
total 12K
-rw-rw-r-- 1 cbrun cbrun 179 20 dec.  18:12 compagnies.csv
-rw-rw-r-- 1 cbrun cbrun 178 20 dec.  16:59 origine_compagnies.csv
-rw-rw-r-- 1 cbrun cbrun  35 21 dec.  07:09 test.txt
>  ll toto > ll.log 2>&1
>  cat ll.log
ls: toto: No such file or directory
```



Composition de commandes (1/3)

La composition de commande permet

- de lancer une liste de commande
- d'exécuter un groupe de commande dans un sous-shell (ou shell courrant)
- d'exécuter une fonction de contrôle
- d'exécuter une fonction

Composition de commandes (2/3)

Modes de composition

- `[commande1] && [commande2]` : lance [commande1] puis [commande2] seulement si le code retour de [commande1] = 0 (réussi)
- `[commande1] || [commande2]` : lance [commande1] puis [commande2] seulement si [commande1] la code retour de [commande1] <> 0 (échec)
- `[commande1] ; [commande2]` : lance [commande1] puis [commande2]
- `[commande1] &` : lance [commande1] en arrière-plan
- `[commande1] | [commande2]` : lance [commande1] et [commande2] en parallèle et en redirigeant la sortie de [commande1] vers l'entrée de [commande2] (pipeline)

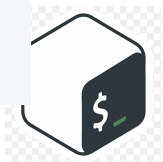
Composition de commandes (3/3)

Modes de lancement

- `{ liste; }` pour une exécution dans le shell courant
- `(liste)` pour une exécution dans un sous-shell

Composition de commandes exemples

```
>  ll
total 16K
-rw-rw-r-- 1 cbrun cbrun 179 20 dec.  18:12 compagnies.csv
-rw-rw-r-- 1 cbrun cbrun  63 21 dec.  07:15 ll.log
-rw-rw-r-- 1 cbrun cbrun 178 20 dec.  16:59 origine_compagnies.csv
-rw-rw-r-- 1 cbrun cbrun  35 21 dec.  07:09 test.txt
>  cat ll.log && echo "Fin du log"
ls: tot': No such file or directory
Fin du log
>  cat aa.log && echo "Fin du log"
cat: aa.log: No such file or directory
>  cat ll.log || echo "Erreur de lecture"
ls: toto: No such file or directory
>  cat aa.log || echo "Erreur de lecture"
cat: aa.log: No such file or directory
Erreur de lecture
>  ll ; cat ll.log
total 16K
-rw-rw-r-- 1 cbrun cbrun 179 20 dec.  18:12 compagnies.csv
-rw-rw-r-- 1 cbrun cbrun  63 21 dec.  07:15 ll.log
-rw-rw-r-- 1 cbrun cbrun 178 20 dec.  16:59 origine_compagnies.csv
-rw-rw-r-- 1 cbrun cbrun  35 21 dec.  07:09 test.txt
ls: toto: No such file or directory
>  cat ll.log | sed 's/No such file or directory/Le fichier inexistant/'
ls: toto: Le fichier inexistant
```



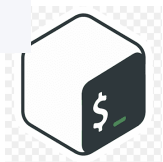
Fonctions de contrôles : If

Structure

```
    if [condition];
    then [liste de commande]
    elif [conditions]
    then [liste de commande]
    else [liste de commande]
    fi
```

Exemple

```
>  if test "$DEBUG"
then cat ll.log
else echo "Pas mode debug"
fi
Pas mode debug
>  DEBUG=1
>  if test "$DEBUG"
then cat ll.log
  else echo "Pas mode debug"
fi
ls: toto: No such file or directory
```



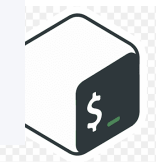
fonctions de contrôle : Case

Structure

```
    case mot in
    motif) liste ;;
    ...
    esac
```

Exemple

```
>  echo $OS
Linux Redhat
>  case $OS in
Linux*) echo "Un linux cool";;
Mac*) echo "Ouais bof";;
Windows) echo "non mais what ?";;
*) echo "Connait pas";;
esac
Un linux cool
>  OS="test"
>  case $OS in
Linux*) echo "Un linux cool";;
Mac*) echo "Ouais bof";;
Windows) echo "non mais what ?";;
*) echo "Connait pas";;
esac
Connait pas
```



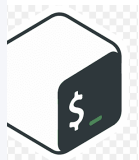
Boucles : For

Structure

```
for [variable] in [lite de mot]; do
   liste
done
```

Exemple

```
>  for compteur in 1 2 3 4 5; do
for> echo $compteur
for> done
1
2
3
4
5
>  for fichier in `ls`; do
echo $fichier ; cat $fichier | wc -l
for> done
compagnies.csv
6
ll.log
1
origine_compagnies.csv
6
test.txt
3
```



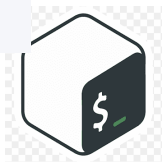
Boucles : While

Structure

```
while [conditions]; do
    [commandes]
done
```

Exemple

```
>  i=0
>  while test $i -le 2
while> do
while> echo i=$i; i=$((i+1))
while> done
i=0
i=1
i=2
>  nbcarac=‘cat ll.log | wc -w‘
>  while test $nbcarac -le 13
while> do
while>  nbcarac=‘cat ll.log | wc -w‘
while> done; echo "ecriture dans le fichier"
ecriture dans le fichier
```



Scripts

- C'est une simple liste de commandes dans un fichier
- Le fichier doit être exécutable (ou lancer dans un sous-shell)
- si lancé avec un point, il affecte le shell courant

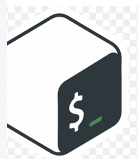
```
source .bashrc
```

On peut définir un shell particulier avec le Shenbang (#)

```
#!/bin/bash
echo "Un script"
```

Exemple script

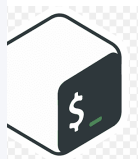
```
#!/bin/bash
. lib/functions.sh
. ./env.cfg

function update_env()
{
    env_to_update="$1"
    sql_files=\$(ls ./sql/prod/\$env_to_update/*.sql 2>/dev/null)
    OUT=$?
    if [ $OUT -ne 0 ]; then
        die "Pas de scripts de mise a jour"
    fi
    for i in $sql_files;
    do
        query_name=$(awk '/query_name: (.*)/ {print $3}' \$i)
        query_info=$(awk '/query_info:/ {split($0,a,":");print a[2]}' $i)
        echo "$start -- Start $query_name : $query_info $normal"
        query_steps=$(awk 'match($0, /.*query_step: (.*)\*\//,a) {printf "
      %s\n\r",a[1]}' $i)
        echo "$subitem $query_steps"
        tput dim
        $psql_cmd $i
        OUT=$?
        tput sgr0
        if [ $OUT -eq 0 ]; then
            echo "$ok -- End $query_info"
        else
            echo "$nok -- End $query_info"
            die "Erreur dans le script "
        fi
        echo $normal
```



cp rm mv touch

- `cp` : copie de fichiers
- `mv` : déplacement/rename fichier
- `rm` : suppression fichiers
- `touch` : Change le timestamp d'un fichier (cool pour init fichier, ou trigger un event)

```
>  ll
total 0
>  touch toto.bak
>  ll
total 0
-rw-rw-r-- 1 cbrun cbrun 0 20 dec.  17:31 toto.bak
>  mv toto.bak tata.bak
>  ll
total 0
-rw-rw-r-- 1 cbrun cbrun 0 20 dec.  17:31 tata.bak
>  cp tata.bak titi.bak
>  ll
total 0
-rw-rw-r-- 1 cbrun cbrun 0 20 dec.  17:31 tata.bak
-rw-rw-r-- 1 cbrun cbrun 0 20 dec.  17:31 titi.bak
>  rm *.bak
>  ll
total 0
```



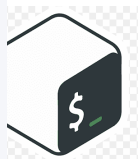
chmod, chown

- `chown` : changement de propriétaire
- droits au format numérique
- ou au format : +w -r

```
>  ll
total 0
-rw-rw-r-- 1 cbrun cbrun 0 20 dec.  17:54 toto.bak
>  chown root toto.bak
>  ll
total 0
-rw-rw-r-- 1 root cbrun 0 20 dec.  17:54 toto.bak
```

* chmod : changement des droits

```
>  chmod o-r toto.bak
>  ll
total 0
-r-xr-x--- 1 cbrun cbrun 0 20 dec.  17:54 toto.bak
>  chmod u+w toto.bak
>  ll
total 0
-rwxr-x--- 1 cbrun cbrun 0 20 dec.  17:54 toto.bak
>  ll
total 0
-rw-rw-r-- 1 cbrun cbrun 0 20 dec.  17:54 toto.bak
>  chmod a-w toto.bak
>  ll
total 0
-r--r--r-- 1 cbrun cbrun 0 20 dec.  17:54 toto.bak
```



pwd

- Affiche le répertoire courant
- C'est le contenu de la variable spéciale '$PWD'

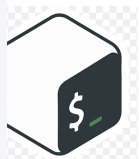
```
>  pwd
/home/cbrun/Projets/slides/Shell/assets/files

>  echo \$PWD
/home/cbrun/Projets/slides/Shell/assets/files
```

mkdir, rmdir, cd

Commande de manipulation de l'arborescence

- mkdir : création d'un répertoire
- rmdir : suppression d'un répertoire (vide)
- cd : changement de répertoire

```
>  mkdir tempo
>  ll
total 8,0K
-rw-rw-r-- 1 cbrun cbrun  152 18 dec.  19:48 compagnies.csv
drwxrwxr-x 2 cbrun cbrun 4,0K 18 dec.  19:55 tempo
>  cd tempo
>  tempo ll
total 0
>  tempo cd ..
>  ll
total 8,0K
-rw-rw-r-- 1 cbrun cbrun  152 18 dec.  19:48 compagnies.csv
drwxrwxr-x 2 cbrun cbrun 4,0K 18 dec.  19:55 tempo
>  rmdir tempo
>  ll
total 4,0K
-rw-rw-r-- 1 cbrun cbrun 152 18 dec.  19:48 compagnies.csv
>  mkdir -p tempo/tempo
>  ll
total 8,0K
-rw-rw-r-- 1 cbrun cbrun  152 18 dec.  19:48 compagnies.csv
drwxrwxr-x 3 cbrun cbrun 4,0K 18 dec.  19:56 tempo
>  cd tempo
```



fg, bg, kill, killall, ...

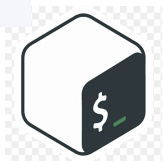
- manipulation de processus
  - ‘ps‘ : Affiche la liste des processus
  - ‘fg‘ : repasse le processus (dernier) en avant plan
  - ‘bg‘ : passe le processus en arrière plan
  - ‘kill‘ : détruit un processus
  - ‘killall‘ : idem kill mais avec le nom du processus et non le pid
- il y a d’autres commandes
  - renice : changement de priorité du processus

cat / less / column

[containsverbatim]

- Affiche le contenu d'un fichier
- less idem que cat, mais permet la navigation dans le fichier
- column peu connu, mais très bien pour du csv

```
>  cat compagnies.csv
id,name,nb_users,nb_project,created_at
1,Tilkee,24,200,2017-01-01
2,King Jouet,12,1345,2017-06-23
3,Conforama,1,13,2017-11-01
4,EDF,432,1543,2016-12-25
>  column -s, -t compagnies.csv
id  name         nb_users  nb_project  created_at
1   Tilkee       24        200         2017-01-01
2   King Jouet   12        1345        2017-06-23
3   Conforama    1         13          2017-11-01
4   EDF          432       1543        2016-12-25
5   Jouetclub    18                    2017-06-05
```



sort (1/2)

Tri de données

```
>  column -s, -t compagnies.csv
id  name         nb_users  nb_project  created_at
1   Tilkee       24        200         2017-01-01
2   King Jouet   12        1345        2017-06-23
3   Conforama    1         13          2017-11-01
4   EDF          432       1543        2016-12-25
5   Jouetclub    18                    2017-06-05
>  sort compagnies.csv|  column -s, -t
1   Tilkee       24        200         2017-01-01
2   King Jouet   12        1345        2017-06-23
3   Conforama    1         13          2017-11-01
4   EDF          432       1543        2016-12-25
5   Jouetclub    18                    2017-06-05
id  name         nb_users  nb_project  created_at
```

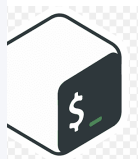
sort (2/2)

```
>  sort -t , -k 3 compagnies.csv|  column -s, -t
3   Conforama   1          13           2017-11-01
2   King Jouet  12         1345         2017-06-23
5   Jouetclub   18                      2017-06-05
1   Tilkee      24         200          2017-01-01
4   EDF         432        1543         2016-12-25
id  name        nb_users   nb_project   created_at
>  sort -t , -k 3 -r compagnies.csv|  column -s, -t
id  name        nb_users   nb_project   created_at
4   EDF         432        1543         2016-12-25
1   Tilkee      24         200          2017-01-01
5   Jouetclub   18                      2017-06-05
2   King Jouet  12         1345         2017-06-23
3   Conforama   1          13           2017-11-01
>  cat compagnies.csv| sed 1,1d |  sort -t , -k 3 |  column -s, -t
3  Conforama   1    13    2017-11-01
2  King Jouet  12   1345  2017-06-23
5  Jouetclub   18         2017-06-05
1  Tilkee      24   200   2017-01-01
4  EDF         432  1543  2016-12-25
```

cut

- Permet d'extraire une partie de ligne d'un fichier
- Paramètres importants
  - -d : délimiteur
  - -f : champs

```
>  cut -d , -f 2 compagnies.csv
name
Tilkee
King Jouet
Conforama
EDF
Jouetclub
>  cut -d , -f 2,5 compagnies.csv | column -s , -t
name         created_at
Tilkee       2017-01-01
King Jouet   2017-06-23
Conforama    2017-11-01
EDF          2016-12-25
Jouetclub    2017-06-05
>  cat compagnies.csv | cut -d , -f 2,5  | column -s , -t
name         created_at
Tilkee       2017-01-01
King Jouet   2017-06-23
Conforama    2017-11-01
EDF          2016-12-25
Jouetclub    2017-06-05
>  cat compagnies.csv | cut -d , -f -3  | column -s , -t
id  name         nb_users
1   Tilkee       24
```

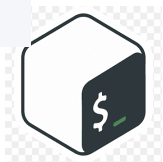


tr

Transcoding de caractère

- rencoding
- suppression des accents
- ...

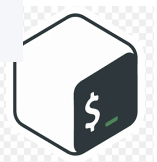
```
>  cat compagnies.csv | tr e a | column -s , -t
id  nama         nb_usars  nb_project  craatad_at
1   Tilkaa       24        200         2017-01-01
2   King Jouat   12        1345        2017-06-23
3   Conforama    1         13          2017-11-01
4   EDF          432       1543        2016-12-25
5   Jouatclub    18                    2017-06-05
>  cat compagnies.csv | tr ea ae | column -s , -t
id  nema         nb_usars  nb_project  craetad_et
1   Tilkaa       24        200         2017-01-01
2   King Jouat   12        1345        2017-06-23
3   Conforeme    1         13          2017-11-01
4   EDF          432       1543        2016-12-25
5   Jouatclub    18                    2017-06-05
```



grep

- extraction de données
- filtre de données

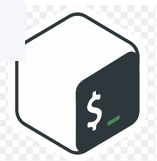
```
> cat compagnies.csv |  column -s , -t
id  name         nb_users  nb_project  created_at
1   Tilkee       24        200         2017-01-01
2   King Jouet   12        1345        2017-06-23
3   Conforama    1         13          2017-11-01
4   EDF          432       1543        2016-12-25
5   Jouetclub    18                    2017-06-05
>  grep Jouet compagnies.csv| column -s , -t
2  King Jouet   12  1345  2017-06-23
5  Jouetclub    18        2017-06-05
>  grep -v Jouet compagnies.csv| column -s , -t
id  name         nb_users  nb_project  created_at
1   Tilkee       24        200         2017-01-01
3   Conforama    1         13          2017-11-01
4   EDF          432       1543        2016-12-25
>  grep -E 'e[te]'  compagnies.csv| column -s , -t
1  Tilkee       24  200   2017-01-01
2  King Jouet   12  1345  2017-06-23
5  Jouetclub    18        2017-06-05
```



sed

Editor de text en mode flux (Stream EDitor)

```
>  column -s , -t compagnies.csv
id  name         nb_users  nb_project  created_at
1   Tilkee       24        200         2017-01-01
2   King Jouet   12        1345        2017-06-23
3   Conforama    1         13          2017-11-01
4   EDF          432       1543        2016-12-25
5   Jouetclub    18                    2017-06-05
>  sed 's/Jouet/Toto/g' compagnies.csv | column -s , -t
id  name         nb_users  nb_project  created_at
1   Tilkee       24        200         2017-01-01
2   King Toto    12        1345        2017-06-23
3   Conforama    1         13          2017-11-01
4   EDF          432       1543        2016-12-25
5   Totoclub     18                    2017-06-05
>  sed 's/\\([0-9]\\{1,4\\}\\)-\\([0-9]\\{1,2\\}\\)-\\([0-9]\\{1,2\\}\\)
     /\\3\/\\2\/\\1/' compagnies.csv | column -s , -t
id  name         nb_users  nb_project  created_at
1   Tilkee       24        200         01/01/2017
2   King Jouet   12        1345        23/06/2017
3   Conforama    1         13          01/11/2017
4   EDF          432       1543        25/12/2016
5   Jouetclub    18                    05/06/2017
```

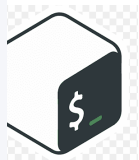


wc

Compte le nombre de

- -L : caractères de la plus grande ligne
- -l : lignes
- -w : mots
- -c : bytes
- -m : caractères

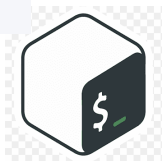
```
>  cat compagnies.csv
id,name,nb_users,nb_project,created_at
1,Tilkee,24,200,2017-01-01
2,King Jouet,12,1345,2017-06-23
3,Conforama,1,13,2017-11-01
4,EDF,432,1543,2016-12-25
5,Jouetclub,18,2017-06-05
>  wc -L compagnies.csv
38 compagnies.csv
>  wc -l compagnies.csv
6 compagnies.csv
>  wc -w compagnies.csv
7 compagnies.csv
>  wc -c compagnies.csv
178 compagnies.csv
>  wc -m compagnies.csv
178 compagnies.csv
```



find

Recherche de fichiers

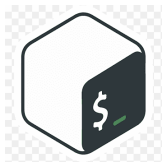
```
>  ll -i
total 8,0K
4265 -rw-rw-r-- 1 cbrun cbrun 178 20 dec.  16:53 compagnies.csv
4266 -rw-rw-r-- 1 cbrun cbrun 178 20 dec.  16:59 origine_compagnies.csv
>  find ./ -name 'compa*'
./compagnies.csv
>  find ./ -name 'compa*' -exec cat {} \;
id,name,nb_users,nb_project,created_at
1,Tilkee,24,200,2017-01-01
2,King Jouet,12,1345,2017-06-23
3,Conforama,1,13,2017-11-01
4,EDF,432,1543,2016-12-25
5,Jouetclub,18,2017-06-05
>  find ./ -inum 4265
./compagnies.csv
>  find ./ -name '*.bak' -exec rm {} \;
>  find . -type f -exec file '{}' \;
./compagnies.csv: ASCII text
./origine_compagnies.csv: ASCII text
./test.txt: ASCII text
./ll.log: UTF-8 Unicode text
```



watch

Lance un commande de manière régulière et affiche le résultat à l'écran.

```
watch 'ls -l *.bak'
```



man

Interface de consultation des manuels de référence

alias

alias de commandes

```
alias ll='ls -l'
```