

Les bases du shell

Sébastien GAGNÉ, Université d'Orléans

L1 Pratique du système UNIX — S2

Expansion sous UNIX

L'**expansion** sous UNIX est un mécanisme puissant qui permet d'utiliser des motifs (patterns) ou de créer des commandes complexes.

L'expansion peut être appliquée

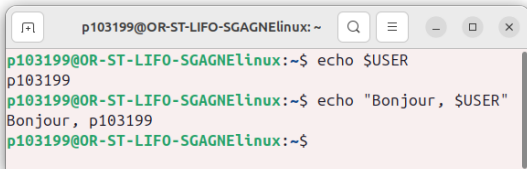
- aux variables ;
- aux accolades ;
- aux chemins de fichiers ;
- au caractère ~ ;
- aux commandes ;
- aux expressions arithmétiques ;

Expansion des variables

Dans le cours précédent on a évoqué les **variables** (d'environnement, comme PATH, ou locales).

On a vu que pour afficher une variable VAR, on exécutait la commande **echo \$VAR**;

L'**expansion des variables** sous UNIX est ce mécanisme qui remplace une variable par sa valeur, quand elle est évoquée précédée d'un symbole **\$**.

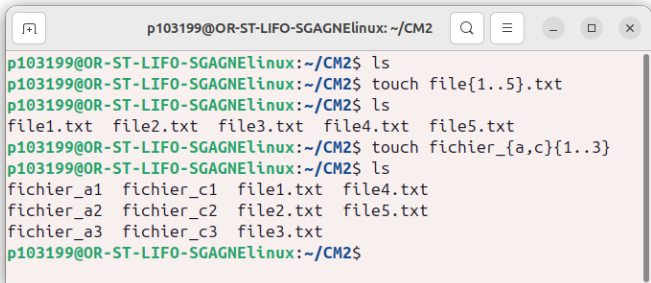
A terminal window titled 'p103199@OR-ST-LIFO-SGAGNElinux: ~' with standard window controls. It shows two commands being executed. The first command is 'echo \$USER', which outputs 'p103199'. The second command is 'echo "Bonjour, \$USER"', which outputs 'Bonjour, p103199'. The prompt 'p103199@OR-ST-LIFO-SGAGNElinux:~\$' is visible at the end of the second line.

```
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo $USER
p103199
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "Bonjour, $USER"
Bonjour, p103199
p103199@OR-ST-LIFO-SGAGNElinux:~$
```

Expansion des accolades

L'**expansion des accolades** permet de remplacer

- {1,2,5,a} par 1 2 5 a ;
- {abc,de,fg} par abc de fg ;
- {1..3} par 1 2 3 ;
- {f..k} par f g h i j k ;



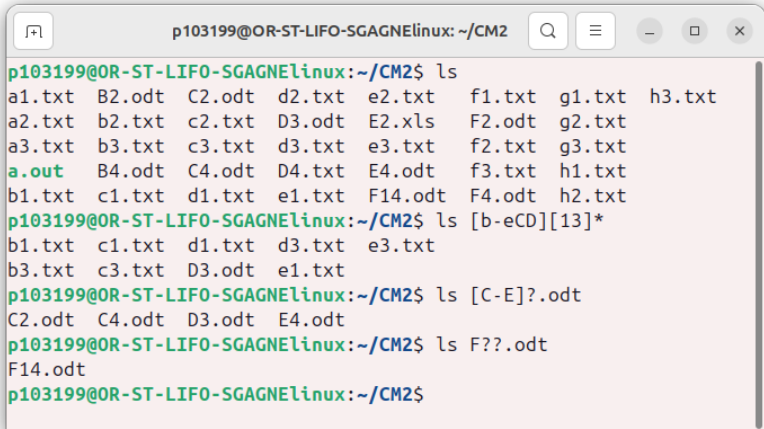
```
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ touch file{1..5}.txt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls
file1.txt file2.txt file3.txt file4.txt file5.txt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ touch fichier_{a,c}{1..3}
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls
fichier_a1 fichier_c1 file1.txt file4.txt
fichier_a2 fichier_c2 file2.txt file5.txt
fichier_a3 fichier_c3 file3.txt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$
```

Expansion des chemins de fichiers

L'**expansion des chemins de fichiers** permet de décrire de façon générique des chemins de fichiers en utilisant les caractères magiques (cf cours précédent).

- ***** désigne un nombre éventuellement nul de caractères quelconques ;
- **?** désigne un unique caractère quelconque ;
- **[abc]** désigne l'un des caractères a, b ou c ;
- **[a-eB-D]** désigne l'un des caractères a, b, c, d, e, B, C, D ;

Expansion des chemins de fichiers

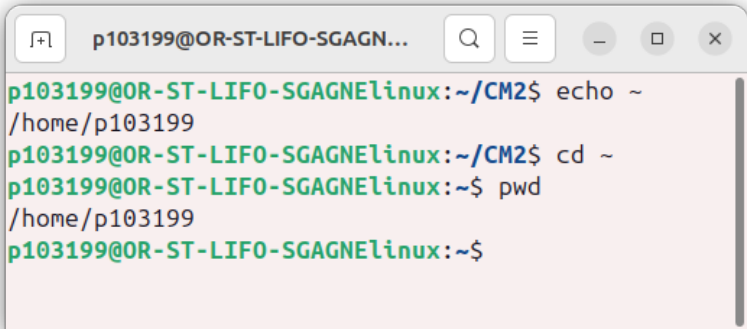


A terminal window titled "p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2" with standard window controls. The terminal shows a series of commands and their outputs:

```
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls
a1.txt  B2.odt  C2.odt  d2.txt  e2.txt  f1.txt  g1.txt  h3.txt
a2.txt  b2.txt  c2.txt  D3.odt  E2.xls  F2.odt  g2.txt
a3.txt  b3.txt  c3.txt  d3.txt  e3.txt  f2.txt  g3.txt
a.out   B4.odt  C4.odt  D4.txt  E4.odt  f3.txt  h1.txt
b1.txt  c1.txt  d1.txt  e1.txt  F14.odt F4.odt  h2.txt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls [b-eCD][13]*
b1.txt  c1.txt  d1.txt  d3.txt  e3.txt
b3.txt  c3.txt  D3.odt  e1.txt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls [C-E]? .odt
C2.odt  C4.odt  D3.odt  E4.odt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls F?? .odt
F14.odt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$
```

Expansion du caractère `~` (tilde)

L'expansion du caractère `~` remplace ce caractère par le répertoire personnel de l'utilisateur :



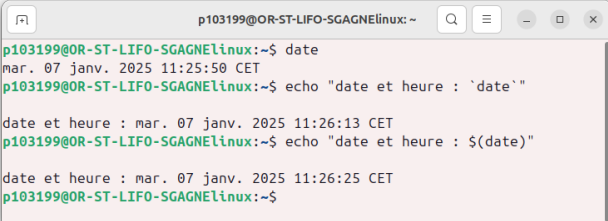
```
p103199@OR-ST-LIFO-SGAGN...  
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ echo ~  
/home/p103199  
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ cd ~  
p103199@OR-ST-LIFO-SGAGNElinux:~$ pwd  
/home/p103199  
p103199@OR-ST-LIFO-SGAGNElinux:~$
```

Expansion des commandes

Le mécanisme d'expansion s'applique aussi aux **commandes**. Il remplace une commande par son résultat.

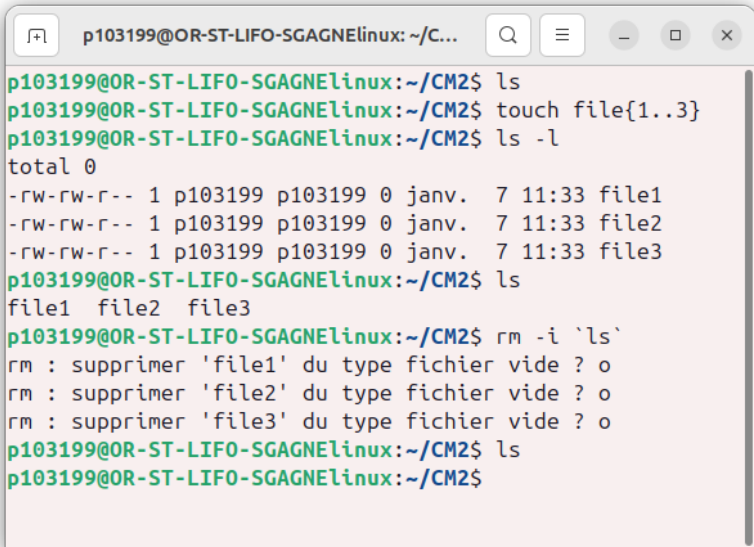
Cette expansion de commande se fait :

- par les guillemets **`commande`**
- par la syntaxe **\$(commande)**



```
p103199@OR-ST-LIFO-SGAGNElinux: ~  
p103199@OR-ST-LIFO-SGAGNElinux:~$ date  
mar. 07 janv. 2025 11:25:50 CET  
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "date et heure : `date`"  
  
date et heure : mar. 07 janv. 2025 11:26:13 CET  
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "date et heure : $(date)"  
  
date et heure : mar. 07 janv. 2025 11:26:25 CET  
p103199@OR-ST-LIFO-SGAGNElinux:~$
```


Expansion des commandes

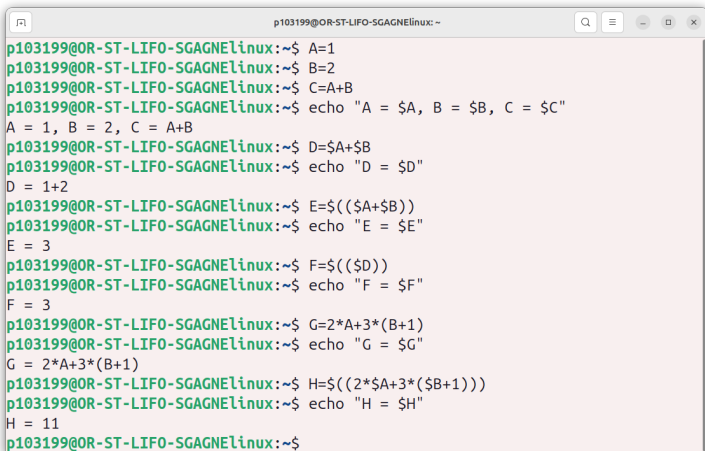


A terminal window with a title bar showing the user 'p103199@OR-ST-LIFO-SGAGNElinux' and the current directory '~/C...'. The terminal displays a sequence of commands and their outputs. The 'ls' command is used to list files, and 'touch' is used to create three files named 'file1', 'file2', and 'file3'. The 'ls -l' command shows the details of these files, including permissions, ownership, and timestamps. Finally, the 'rm -i' command is used to delete the files, and the terminal shows the confirmation prompts for each file.

```
p103199@OR-ST-LIFO-SGAGNElinux: ~/C...
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2$ ls
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2$ touch file{1..3}
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2$ ls -l
total 0
-rw-rw-r-- 1 p103199 p103199 0 janv.  7 11:33 file1
-rw-rw-r-- 1 p103199 p103199 0 janv.  7 11:33 file2
-rw-rw-r-- 1 p103199 p103199 0 janv.  7 11:33 file3
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2$ ls
file1 file2 file3
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2$ rm -i `ls`
rm : supprimer 'file1' du type fichier vide ? o
rm : supprimer 'file2' du type fichier vide ? o
rm : supprimer 'file3' du type fichier vide ? o
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2$ ls
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2$
```

Expansions arithmétiques

L'expansion évalue les **expressions arithmétiques** grâce à la syntaxe **$\$((...))$**

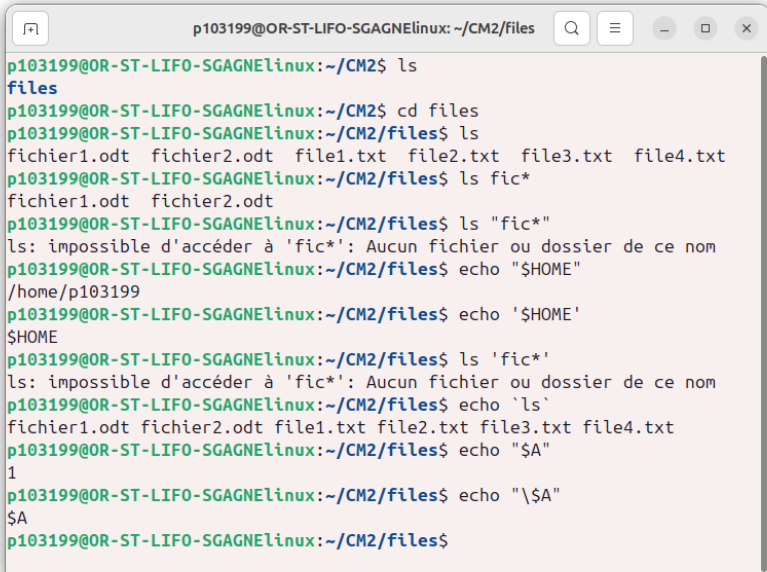


```
p103199@OR-ST-LIFO-SGAGNElinux: ~  
p103199@OR-ST-LIFO-SGAGNElinux:~$ A=1  
p103199@OR-ST-LIFO-SGAGNElinux:~$ B=2  
p103199@OR-ST-LIFO-SGAGNElinux:~$ C=A+B  
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "A = $A, B = $B, C = $C"  
A = 1, B = 2, C = A+B  
p103199@OR-ST-LIFO-SGAGNElinux:~$ D=$A+$B  
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "D = $D"  
D = 1+2  
p103199@OR-ST-LIFO-SGAGNElinux:~$ E=$(( $A+$B ))  
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "E = $E"  
E = 3  
p103199@OR-ST-LIFO-SGAGNElinux:~$ F=$(( $D ))  
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "F = $F"  
F = 3  
p103199@OR-ST-LIFO-SGAGNElinux:~$ G=2*$A+3*(B+1)  
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "G = $G"  
G = 2*$A+3*(B+1)  
p103199@OR-ST-LIFO-SGAGNElinux:~$ H=$(( 2*$A+3*( $B+1) ))  
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "H = $H"  
H = 11  
p103199@OR-ST-LIFO-SGAGNElinux:~$
```

Guillemets et protection

- Les guillemets doubles (") groupent un ensemble de mots. Ils permettent l'expansion des variables mais pas des patterns ;
- Les guillemets simples (') groupent un ensemble de mots, et empêchent toute expansion ;
- Les anti-guillemets (`) permettent l'expansion d'une commande ;
- Le caractère \ permet de protéger un caractère spécial (donc de le voir comme un caractère).

Guillemets et protection



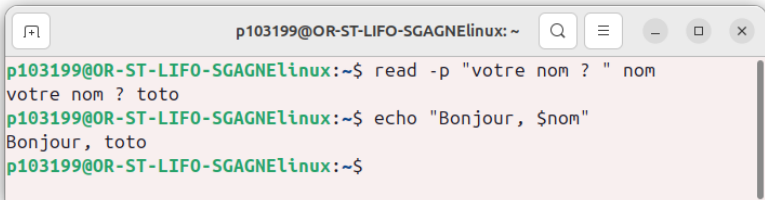
```
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2/files
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls
files
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ cd files
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$ ls
fichier1.odt fichier2.odt file1.txt file2.txt file3.txt file4.txt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$ ls fic*
fichier1.odt fichier2.odt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$ ls "fic*"
ls: impossible d'accéder à 'fic*': Aucun fichier ou dossier de ce nom
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$ echo "$HOME"
/home/p103199
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$ echo '$HOME'
$HOME
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$ ls 'fic*'
ls: impossible d'accéder à 'fic*': Aucun fichier ou dossier de ce nom
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$ echo `ls`
fichier1.odt fichier2.odt file1.txt file2.txt file3.txt file4.txt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$ echo "$A"
1
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$ echo "\$A"
$A
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2/files$
```

Les flux standards

Il existe sous UNIX trois **flux standards**. Ils permettent la gestion :

- des entrées de données (sur l'entrée standard) ;
 - des sorties de résultats (sur la sortie standard) ;
 - des erreurs (sur la sortie erreurs standard).
-
- L'entrée standard est par défaut le clavier ;
 - La sortie standard est par défaut le terminal ;
 - La sortie erreur est par défaut le terminal.

Les flux standards



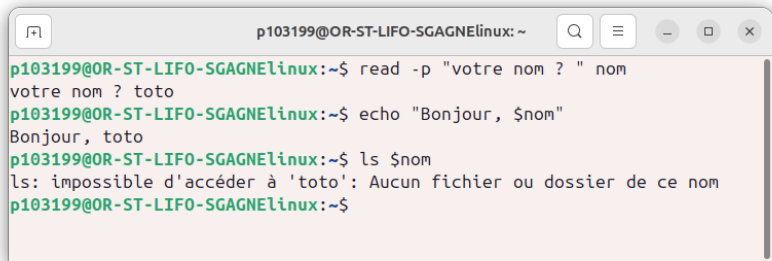
```
p103199@OR-ST-LIFO-SGAGNElinux: ~  
p103199@OR-ST-LIFO-SGAGNElinux:~$ read -p "votre nom ? " nom  
votre nom ? toto  
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "Bonjour, $nom"  
Bonjour, toto  
p103199@OR-ST-LIFO-SGAGNElinux:~$
```

L'instruction `read -p "votre nom ? " nom`

1. affiche sur la sortie standard (terminal) le message :
`votre nom ?`
2. lit sur l'entrée standard (clavier) la valeur de la variable
`nom` (ici la valeur "toto")

L'instruction `echo "Bonjour, $nom"` affiche sur la sortie standard un message avec expansion de `nom` en "toto"

Les flux standards

A terminal window with a title bar showing the username 'p103199@OR-ST-LIFO-SGAGNElinux' and standard window controls. The terminal text shows a script that prompts for a name, echoes a greeting, and attempts to list the contents of a directory named with the input. The final command results in an error message because the directory does not exist.

```
p103199@OR-ST-LIFO-SGAGNElinux:~$ read -p "votre nom ? " nom
votre nom ? toto
p103199@OR-ST-LIFO-SGAGNElinux:~$ echo "Bonjour, $nom"
Bonjour, toto
p103199@OR-ST-LIFO-SGAGNElinux:~$ ls $nom
ls: impossible d'accéder à 'toto': Aucun fichier ou dossier de ce nom
p103199@OR-ST-LIFO-SGAGNElinux:~$
```

L'instruction `ls $nom` affiche sur la sortie erreur standard (terminal) un message d'erreur disant que le répertoire dont on veut lister le contenu n'existe pas.

Les redirections

- Les **redirections** permettent de modifier les flux standards (entrée, sortie ou erreur).
- La redirection classique **remplace** l'un des flux standards par un **fichier** (écriture dans un fichier, ajout à un fichier ou lecture d'un fichier).
- Les flux standards ont chacun un **descripteur** de fichier : **0** (entrée), **1** (sortie) et **2** (erreur). Ces descripteurs de flux peuvent servir lors des redirections.

Redirections principales

- Écrire le résultat dans un fichier (modifie la sortie standard)

echo "Bonjour" 1> fichier.txt

- Écrire l'erreur dans un fichier (modifie la sortie erreur)

ls toto 2> erreurs.txt

- Ajouter le résultat à un fichier (modifie la sortie standard)

echo "Hello" 1>> fichier.txt

- Ajouter les erreurs à un fichier (modifie la sortie erreur)

ls titi 2>> erreurs.txt

- Lire les données à partir d'un fichier (modifie l'entrée standard)

read nom prenom 0< personne.txt

Redirections principales

Remarque : Les descripteurs sont optionnels.
S'il n'y a pas d'ambiguïté, on peut les omettre.

echo "Bonjour" > fichier.txt

ls toto 2> erreurs.txt

echo "Hello" >> fichier.txt

ls titi 2>> erreurs.txt

read nom prenom < personne.txt

Philosophie UNIX

Elle peut être résumée en 3 règles :

1. Les programmes effectuent une seule chose, et de façon correcte ;
2. Les programmes collaborent, communiquent ;
3. Les programmes gèrent des flux de texte, qui sont une interface universelle.

Ces règles de base sont illustrées par le mécanisme de l'enchaînement de commandes !

Enchaînement de commandes

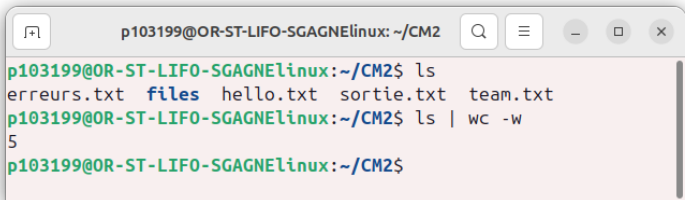
- Une commande simple utilise des données en entrée pour produire des résultats en sortie ;
- La **communication entre les programmes** : les résultats (sorties) d'une commande servent de données (entrées) à la commande suivante.

La communication entre commandes se fait par le biais du **pipe** représenté par le symbole |.

Syntaxe : **commande1 | commande2**

Un exemple de pipe

- La commande **ls** fournit la liste des fichiers du répertoire courant ;
- La commande **wc** (pour **w**ord **c**ount) compte, avec l'option **-w**, le nombre de mots de l'entrée ;
- L'enchaînement **ls | wc -w** affiche donc le nombre de fichiers du répertoire courant.

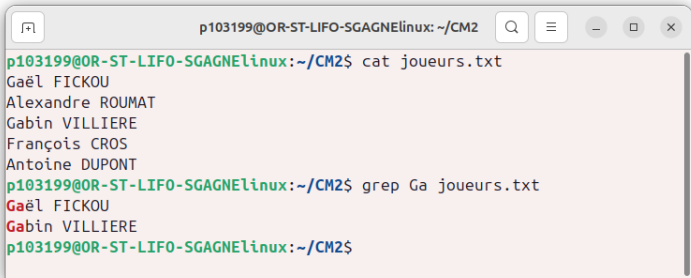


```
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls
erreurs.txt  files  hello.txt  sortie.txt  team.txt
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ ls | wc -w
5
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$
```

Un autre exemple

La commande **grep** recherche, sur l'entrée fournie, les lignes qui contiennent le motif qui est donné en paramètre.

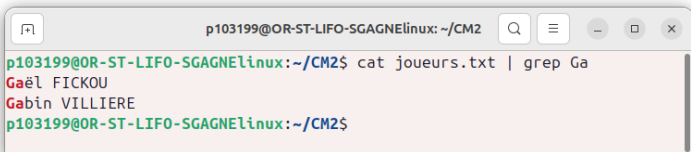
L'entrée peut être un fichier (ici **joueurs.txt**)



```
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ cat joueurs.txt
Gaël FICKOU
Alexandre ROUMAT
Gabin VILLIERE
François CROS
Antoine DUPONT
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ grep Ga joueurs.txt
Gaël FICKOU
Gabin VILLIERE
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$
```

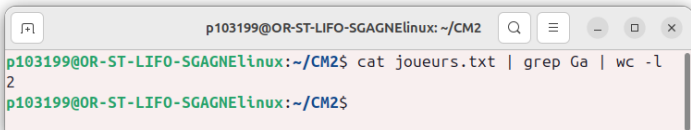
Un autre exemple

Cette commande **grep** peut être lancée via un **pipe** :

A terminal window titled 'p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2'. The prompt is 'p103199@OR-ST-LIFO-SGAGNElinux:~/CM2\$'. The command entered is 'cat joueurs.txt | grep Ga'. The output shows two lines: 'Gaël FICKOU' and 'Gabin VILLIERE'. The prompt returns to 'p103199@OR-ST-LIFO-SGAGNElinux:~/CM2\$'.

```
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ cat joueurs.txt | grep Ga
Gaël FICKOU
Gabin VILLIERE
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$
```

On peut même enchaîner avec la commande **wc -l** qui compte les lignes de l'entrée :

A terminal window titled 'p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2'. The prompt is 'p103199@OR-ST-LIFO-SGAGNElinux:~/CM2\$'. The command entered is 'cat joueurs.txt | grep Ga | wc -l'. The output is '2'. The prompt returns to 'p103199@OR-ST-LIFO-SGAGNElinux:~/CM2\$'.

```
p103199@OR-ST-LIFO-SGAGNElinux: ~/CM2
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$ cat joueurs.txt | grep Ga | wc -l
2
p103199@OR-ST-LIFO-SGAGNElinux:~/CM2$
```

Un exercice

- La commande **head** affiche les premières lignes de l'entrée. L'option **-n 7** affiche les 7 premières ;
- La commande **tail** affiche les dernières lignes de l'entrée. L'option **-n 4** affiche les 4 dernières ;



```
p103199@OR-ST-LIFO-SGAGNELinux: ~/CM2
p103199@OR-ST-LIFO-SGAGNELinux:~/CM2$ ls -l files
total 0
-rw-rw-r-- 1 p103199 p103199 0 janv. 7 16:02 fichier1.odt
-rw-rw-r-- 1 p103199 p103199 0 janv. 7 16:02 fichier2.odt
-rw-rw-r-- 1 p103199 p103199 0 janv. 8 13:44 sujetA1.pdf
-rw-rw-r-- 1 p103199 p103199 0 janv. 8 13:44 sujetA2.pdf
-rw-rw-r-- 1 p103199 p103199 0 janv. 8 13:44 sujetA3.pdf
-rw-rw-r-- 1 p103199 p103199 0 janv. 8 13:44 sujetB1.pdf
-rw-rw-r-- 1 p103199 p103199 0 janv. 8 13:44 sujetB2.pdf
-rw-rw-r-- 1 p103199 p103199 0 janv. 8 13:44 sujetB3.pdf
p103199@OR-ST-LIFO-SGAGNELinux:~/CM2$ ls -l files | grep pdf | head -n 5 | tail -n 3 > resultat.txt
p103199@OR-ST-LIFO-SGAGNELinux:~/CM2$
```

Que fait la dernière ligne de commandes ?

Un deuxième exercice

- La commande **cut** permet de découper les lignes données en entrée, selon un délimiteur (option -d) et affiche les champs sélectionnés (option -f) ;
- La commande **sort** permet de trier les données, par défaut par ordre alphabétique. On peut trier selon une colonne (option -k) grâce à un délimiteur (option -t). On peut trier en tenant compte des valeurs numériques (option -n) et par ordre décroissant (option -r) ;

Un deuxième exercice

Voici le fichier `departements.csv`

```
Cher;18;297000;7235;Bourges  
Eure-et-Loir;28;432000;5880;Chartres  
Indre;36;218000;6791;Châteauroux  
Indre-et-Loire;37;620000;6127;Tours  
Loir-et-Cher;41;329000;6343;Blois  
Loiret;45;677000;6775;Orléans
```

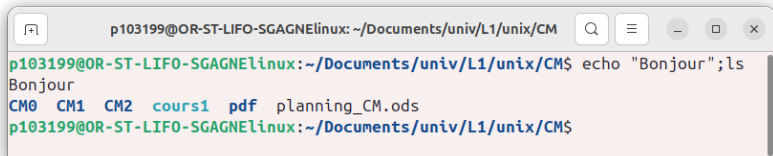
On y voit le nom, le numéro, la population, la superficie et la préfecture des départements de la région Centre Val de Loire.

Comment créer une commande complexe qui affiche le nom des départements, le numéro et la préfecture, selon un tri par population décroissante ?

Opérateurs logiques

L'enchaînement de commandes est un mécanisme essentiel, mais il existe d'autres opérateurs entre commandes :

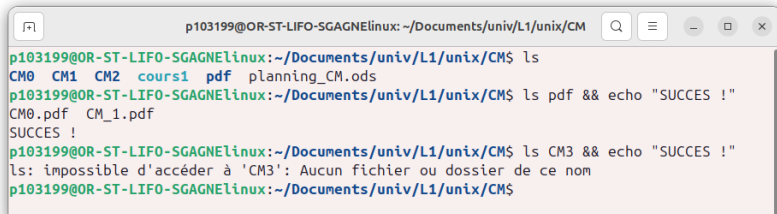
- l'opérateur `;` permet d'exécuter l'une après l'autre plusieurs commandes, indépendamment du succès ou de l'échec de chacun ;

A terminal window with a title bar showing the user 'p103199' and the path '~/Documents/univ/L1/unix/CM'. The terminal text shows a command 'echo "Bonjour";ls' being executed. The output is 'Bonjour' followed by a directory listing: 'CM0 CM1 CM2 cours1 pdf planning_CM.ods'.

```
p103199@OR-ST-LIFO-SGAGNElinux: ~/Documents/univ/L1/unix/CM
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ echo "Bonjour";ls
Bonjour
CM0 CM1 CM2 cours1 pdf planning_CM.ods
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$
```

Opérateurs logiques

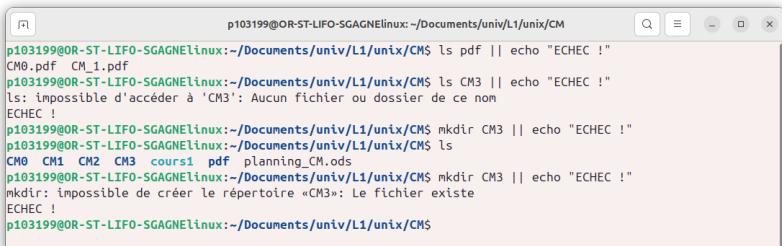
- l'opérateur **&&** exécute la première commande, et la deuxième **SI** la première a réussi (avec un code de retour 0) ;

A terminal window titled 'p103199@OR-ST-LIFO-SGAGNElinux: ~/Documents/univ/L1/unix/CM'. The window shows a series of commands and their outputs. The first command is 'ls', which lists 'CM0', 'CM1', 'CM2', 'cours1', 'pdf', and 'planning_CM.ods'. The second command is 'ls pdf && echo "SUCCES !"', which lists 'CM0.pdf' and 'CM_1.pdf' and then prints 'SUCCES !'. The third command is 'ls CM3 && echo "SUCCES !"', which results in an error message 'ls: impossible d'accéder à 'CM3': Aucun fichier ou dossier de ce nom' and does not print the success message. The prompt is 'p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM\$' for each command.

```
p103199@OR-ST-LIFO-SGAGNElinux: ~/Documents/univ/L1/unix/CM
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls
CM0 CM1 CM2 cours1 pdf planning_CM.ods
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls pdf && echo "SUCCES !"
CM0.pdf CM_1.pdf
SUCCES !
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls CM3 && echo "SUCCES !"
ls: impossible d'accéder à 'CM3': Aucun fichier ou dossier de ce nom
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$
```

Opérateurs logiques

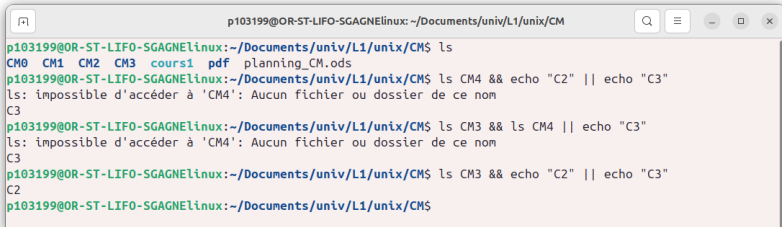
- l'opérateur `||` exécute la première commande. La deuxième commande n'est exécutée que **SI** la première a échoué (avec un code de retour $\neq 0$);



```
p103199@OR-ST-LIFO-SGAGNElinux: ~/Documents/univ/L1/unix/CM
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls pdf || echo "ECHEC !"
CM0.pdf  CM_1.pdf
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls CM3 || echo "ECHEC !"
ls: impossible d'accéder à 'CM3': Aucun fichier ou dossier de ce nom
ECHEC !
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ mkdir CM3 || echo "ECHEC !"
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls
CM0  CM1  CM2  CM3  cours1  pdf  planning_CM.ods
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ mkdir CM3 || echo "ECHEC !"
mkdir: impossible de créer le répertoire «CM3»: Le fichier existe
ECHEC !
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$
```

Opérateurs logiques

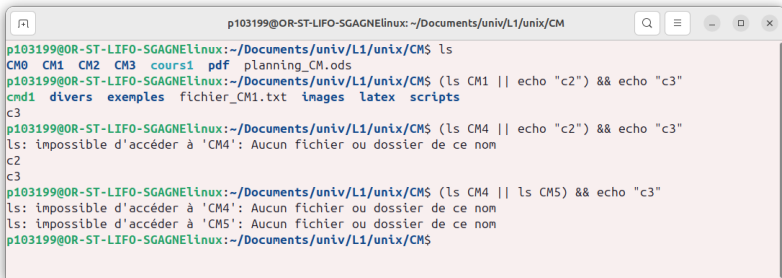
- combinaison **c1 && c2 || c3** :
 - ▶ Si c1 échoue, c2 n'est pas exécutée, et c3 est exécutée ;
 - ▶ Si c1 réussit et c2 échoue, c3 est exécutée ;
 - ▶ Si c1 réussit et si c2 réussit, c3 n'est pas exécutée.



```
p103199@OR-ST-LIFO-SGAGNElinux: ~/Documents/univ/L1/unix/CM
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls
CM0 CM1 CM2 CM3 cours1 pdf planning_CM.ods
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls CM4 && echo "C2" || echo "C3"
ls: impossible d'accéder à 'CM4': Aucun fichier ou dossier de ce nom
C3
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls CM3 && ls CM4 || echo "C3"
ls: impossible d'accéder à 'CM4': Aucun fichier ou dossier de ce nom
C3
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$ ls CM3 && echo "C2" || echo "C3"
C2
p103199@OR-ST-LIFO-SGAGNElinux:~/Documents/univ/L1/unix/CM$
```

Opérateurs logiques

- parenthèses de regroupement de commandes :
 - ▶ Par défaut, **&&** a la priorité sur **||** ;
 - ▶ On peut regrouper des commandes entre **parenthèses** pour modifier les priorités ;
 - ▶ Dans la commande **c1 && (c2 || c3)**, on évalue d'abord **c2 || c3**, puis on combine le résultat avec **c1** par **&&**.



```
p103199@OR-ST-LIFO-SGAGNELinux: ~/Documents/univ/L1/unix/CM
p103199@OR-ST-LIFO-SGAGNELinux:~/Documents/univ/L1/unix/CM$ ls
CM0 CM1 CM2 CM3 cours1 pdf planning_CM.ods
p103199@OR-ST-LIFO-SGAGNELinux:~/Documents/univ/L1/unix/CM$ (ls CM1 || echo "c2") && echo "c3"
cmd1 divers exemples fichier_CM1.txt images latex scripts
c3
p103199@OR-ST-LIFO-SGAGNELinux:~/Documents/univ/L1/unix/CM$ (ls CM4 || echo "c2") && echo "c3"
ls: impossible d'accéder à 'CM4': Aucun fichier ou dossier de ce nom
c2
c3
p103199@OR-ST-LIFO-SGAGNELinux:~/Documents/univ/L1/unix/CM$ (ls CM4 || ls CM5) && echo "c3"
ls: impossible d'accéder à 'CM4': Aucun fichier ou dossier de ce nom
ls: impossible d'accéder à 'CM5': Aucun fichier ou dossier de ce nom
p103199@OR-ST-LIFO-SGAGNELinux:~/Documents/univ/L1/unix/CM$
```