



# UNIX

## Cours 3: Commande grep et expressions régulières

Filipe Vasconcelos<sup>1</sup>

<sup>1</sup>ESME, Lille, [filipe.vasconcelo@esme.fr](mailto:filipe.vasconcelo@esme.fr)

- ① grep
- ② Expressions régulières



### AAV3

Utiliser et enchaîner, au moyen de tubes, les principales commandes de filtre (grep, cut, head, tail ...) pour manipuler des flux de données qu'ils proviennent ou non dans des fichiers



1 grep

2 Expressions régulières



## grep : « :g/re/p »

- ❑ Usage : Cette commande cherche et affiche toutes les lignes qui satisfont un motif (chaine de caractère ou expression régulière).
- ❑ Syntaxe : `grep [options] <motif> <fichier>`
- ❑ Résultat : Lignes du fichier contenant le motif
- ❑ Remarque : `grep -E` est équivalent à `egrep`

À voir : la petite vidéo youtube de Computerphile sur votre Moodle



## Options de la commande grep

- c : donne seulement le nombre de lignes trouvées obéissant au critère
- l : donne seulement le nom des fichiers où le critère a été trouvé
- v : donne les lignes où le critère n'a pas été trouvé
- i : ne pas tenir compte de la casse (ne pas différencier majuscules minuscules)
- n : pour n'afficher que les numéros des lignes trouvées
- w : pour imposer que le motif corresponde à un mot entier d'une ligne

Placer vous dans le répertoire de la séance03 ~/unix-etudiant/seance03 :

```
cat lettre.txt
```

Juchée à l'extrême nord-est de la table qui me sert de bureau,  
une lampe imbibe de sa lumière tamisée les quelques papiers  
privilégiés qui n'ont pas encore leur place dans le fatras des  
classeurs habitant l'étagère à laquelle je tourne le dos avec une  
feinte obstination.

Rechercher le mot **une** dans ce texte

```
grep une lettre.txt
```

**une** lampe imbibe de sa lumière tamisée les quelques papiers  
classeurs habitant l'étagère à laquelle je tourne le dos avec **une**



# Exemples d'utilisation de grep

Recherche d'un mot exact

```
grep -w le lettre.txt
```

privilégiés qui n'ont pas encore leur place dans **le** fatras des  
classeurs habitant l'étagère à laquelle je tourne **le** dos avec une

```
grep -1 leur lettre.txt
```

une lampe imbibe de sa lumière tamisée les quelques papiers  
privilégiés qui n'ont pas encore **leur** place dans le fatras des  
classeurs habitant l'étagère à laquelle je tourne le dos avec une

```
grep -c que lettre.txt
```

```
2
```

Placer vous dans le répertoire de la séance03 ~/unix-etudiant/seance03 :

```
cat notes.txt
```

```
crepetna:Crepet, Nathalie:CREN1807750:92:87:88:54:70
yosnheat:Yos Nhean, Trakal:YOST19087603:84:73:70:50:73
benelaur:Benel, Aurelien:BENA80207700:84:73:89:45:100
soucypas:Soucy, Pascal:SOUP14067502:95:90:89:87:99
```

On peut extraire les lignes qui contiennent une note comprise entre 90 et 99

```
grep :9 notes.txt
```

```
crepetna:Crepet, Nathalie:CREN1807750:92:87:88:54:70
soucypas:Soucy, Pascal:SOUP14067502:95 :9 0:89:87:99
```



# Limitations du motif

Comment faire pour extraire les lignes où la dernière note est comprise entre 90 et 99 ?

```
crepetna:Crepet, Nathalie:CREN1807750:92:87:88:54:70
yosnheat:Yos Nhean, Trakal:YOST19087603:84:73:70:50:73
benelaur:Benel, Aurelien:BENA80207700:84:73:89:45:100
soucypas:Soucy, Pascal:SOUP14067502:95:90:89:87:99
```

Solution : Expressions régulières !!!



1 grep

2 Expressions régulières



### Définition

- Formule qui représente une chaîne de caractères
- Composée de caractères et d'opérateurs

### Utilisation

- On recherche alors non pas un mot ou une simple chaîne de caractères mais une suite de caractères qui correspondent au critères énoncés par la formule

### Culture Générale

- Expression régulière, "regular expression" en anglais, se traduit en bon français par "expression rationnelle" mais l'usage est de dire "régulière"
- Fondamentalement lié aux automates finis



## Commandes et langages utilisant les RE

- grep
- sed
- awk
- python
- java
- vi, emacs

C'est un très bon investissement de savoir maîtriser les expressions régulières !!

À voir : une vidéo youtube sur les expressions régulières que vous retrouverez sur Moodle  
<https://www.youtube.com/watch?v=M3x5Z3iIoSU>



Une expression régulière est une chaîne de caractères composée de caractères normaux et de caractères spéciaux, appelés "métacaractères des expressions régulières"  
**à ne pas confondre avec les métacaractères du shell !!**

## Comment représenter ...

- un caractère quelconque : .
- une ou une infinité d'occurrences : +
- zéro ou une infinité d'occurrences : \*
- zéro ou une occurrences : ?
- un choix parmi un ensemble : [<liste>] ex. : [ab] ou [a-z]
- tout sauf un certain caractère : [^<caractère>]
- de n à m répétitions d'une occurrence du caractère précédent : {n,m}



## Comment représenter ...

- un début de la ligne : ^
- une fin de ligne : \$
- l'alternative (ou) : | ex. : L[y|i]s identifie Lys ou Lis
- le complément (i.e l'opposé) : [^] ex. : [^a-z] identifie tous les caractères sauf une lettre minuscule.

On utilisera les parenthèses (...) pour limiter la portée d'un masque ou de l'alternative.

## Combinaison

- .\* : Zéro ou une infinité de caractères quelconques
- a+b\* : Au moins un 'a' suivi de 0 ou une infinité de 'b'
- [ab]+ : Au moins un 'a' ou 'b' ou une infinité
- ^text\$ : identifie les lignes qui contiennent strictement la chaîne text.
- ^\$ : identifie une ligne vide.



Voici les motifs prédéfinis et reconnus par les expressions régulières :

- ❑ `[:alpha:]` n'importe quelle lettre
- ❑ `[:digit:]` n'importe quel chiffre
- ❑ `[:alnum:]` n'importe quelle lettre ou chiffre
- ❑ `[:space:]` n'importe quel espace blanc
- ❑ `[:punct:]` n'importe quel signe de ponctuation
- ❑ `[:lower:]` n'importe quelle lettre en minuscule
- ❑ `[:upper:]` n'importe quelle lettre capitale
- ❑ `[:blank:]` espace ou tabulation
- ❑ `[:graph:]` caractères affichables et imprimables
- ❑ `[:cntrl:]` caractères d'échappement
- ❑ `[:print:]` caractères imprimables exceptés ceux de contrôle

Comment faire pour extraire les lignes où la dernière note est comprise entre 90 et 99 ?

```
crepetna:Crepet, Nathalie:CREN1807750:92:87:88:54:70
yosnheat:Yos Nhean, Trakal:YOST19087603:84:73:70:50:73
benelaur:Benel, Aurelien:BENA80207700:84:73:89:45:100
soucypas:Soucy, Pascal:SOUP14067502:95:90:89:87:99
```

```
grep -E ".*:.*:.*:.*:9" notes.txt
```

soucypas:Soucy, Pascal:SOUP14067502:95:90:89:87:99

Alternative plus condensée :

```
grep -E "( *:){5}9" notes.txt
```

# Exemple

```
crepetna:Crepet, Nathalie:CREN1807750:92:87:88:54:70
yosnheat:Yos Nhean, Trakal:YOST19087603:84:73:70:50:73
benelaur:Benel, Aurelien:BENA80207700:84:73:89:45:100
soucypas:Soucy, Pascal:SOUP14067502:95:90:89:87:99
```

Remarquons que le code de Nathalie ne contient que 7 chiffres au lieu de 8 (pour les autres). Comment peut-on identifier cela en utilisant une expression régulière ?  
Un code contient 4 majuscules suivies de 8 chiffres : [A-Z]{4}[0-9]{8}

```
grep -E "[A-Z]{4}[0-9]{8}" notes.txt
```

```
yosnheat:Yos Nhean, Trakal:YOST19087603:84:73:70:50:73
benelaur:Benel, Aurelien:BENA80207700:84:73:89:45:100
soucypas:Soucy, Pascal:SOUP14067502:95:90:89:87:99
```



## Chiffres romains

Déterminer l'expression régulière équivalente à l'ensemble des nombres romains compris entre : I II III IV V VI VII VIII IX On testera avec le fichier

~/unix-etudiant/romain.txt Références :

<https://www.youtube.com/watch?v=M3x5Z3iIoSU>

