

4. La commande **grep** (ne demandez pas pourquoi grep!)

4.1. Ou alors si. Grep de la commande d'édition **g/RE/p 'globally search for RE and print it'** ou RE est un raccourci pour *RegularExpression*.

La commande grep *chaîne fichier* permet d'extraire de *fichier* toutes les lignes contenant *chaîne*. (Ca vous paraît familier? Qu'est-ce que ça vous rappelle?)

Si *chaîne* contient des espaces, elle doit être encadrée par deux guillemets simples. Ainsi,

- grep to Sha ou
- grep 'o b' Sha

afficheront la première ligne to be de Sha, et

- grep t Sha, ou
- grep ' ' Sha

afficheront les deux lignes to be et or not (qui contiennent toutes deux aussi bien un t qu'un espace). Sachant que grep *chaîne fichier* peut être redirigée vers (éventuellement, le même) *fichier* à l'aide de >>, en déduire la commande permettant de transformer en une seule étape le contenu de Sha en

- to be
or not
to be

La commande grep est sensible à la casse.

```
grep Science science.txt et  
grep science science.txt
```

ne donnent pas les mêmes résultats. Afin que la commande grep ignore la différence entre Majuscule et minuscule, utilisez l'option -i.

```
grep -i science science.txt
```

Afin de chercher une phrase, ou un mot en entier, vous devez l'entourer de guillemets simples ou doubles :

```
grep -i 'eruption volcanique' total.txt  
grep -i "volcan" total.txt
```

Afin d'afficher toutes les lignes qui ne contiennent pas un motif, utiliser l'option -v. On pourrait ainsi combiner plusieurs commandes grep à la suite à l'aide de | :

```
grep -i volcan VOLCFR.txt | grep -v Rittman | more
```

L'option -n permet d'afficher le numéro de ligne dans le fichier auquel on a trouvé le motif recherché.

```
grep -i -n VOLCFR.txt
```

L'option -c permet d'afficher uniquement le nombre total de lignes correspondant au motif.

```
grep -c volcan science.txt
```

est équivalente à
`grep volcan.txt | wc -l`

On peut combiner plusieurs options à la fois ainsi :
`grep -ivc science science.txt` est la même chose que
`grep -i -v -c science science.txt`

L'option `-l` permet d'afficher juste les noms des fichiers dans lesquels un motif apparaît, sans afficher les occurrences trouvées.

L'option `-m NUM` permet de s'arrêter après NUM solutions trouvées.
`grep -m 20 volcan VOLCEN.txt`
nous permet de limiter notre recherche à 20 résultats.

L'option `-H` permet d'avoir le nom du fichier où l'on cherche un motif en tête de ligne :
`grep -H volcan corpus_EN/*.txt`

4. 2. Extension : l'utilisation de la commande `grep` avec les expressions régulières.

La syntaxe de la commande `grep` que nous utiliserons sera de la forme

`grep -E "expression" fichier.`

- L'argument *expression* est ce qu'on appelle une *expression régulière*, une suite de symboles décrivant un ensemble (fini ou infini) de mots. Noter que cet argument est encadré par des guillemets doubles.
- L'argument `-E` indique que cette expression est une expression *étendue*, par opposition aux expressions dites *de base*, dont la syntaxe est différente de celle décrite ci-dessous (mais certainement pas plus basique).
- En sortie, `grep` renvoie toutes les lignes de *fichier* contenant au moins un mot décrit par *expression*.

4.3. Rappel sur la syntaxe des expressions régulières

Exemple	Explication
A	retrouve la lettre "a"
[a-z]	retrouve n'importe quelle lettre en minuscule
[A-Z]	retrouve n'importe quelle lettre en majuscule
[0-9]	retrouve n'importe quel chiffre

[0123456789]	retrouve n'importe quel chiffre
[aeiouAEIUO]	retrouve n'importe quelle voyelle
[^aeiouAEIOU]	retrouve n'importe quel caractère sauf une voyelle
.	retrouve n'importe quel caractère
^	le début de ligne
\$	la fin de ligne
x^*	une suite quelconque d'occurrences de x (0 ou plus)
x^+	au moins une occurrence de x (1 ou plus)
$x\{n, m\}$	Entre n et m occurrences de x (au moins n, au plus m)
$x^?$	une occurrence optionnelle de x (0 ou 1)
$x y$	X ou y

Court rappel sur les expressions régulières.

Les expressions régulières sont des chaînes de caractères qui peuvent être utilisées pour retrouver un **ensemble** de chaînes de caractères. Par exemple, pour retrouver toutes les occurrences du verbe *écrire* dans un texte on aurait besoin d'un nombre considérable de requêtes pour chacune des formes : *écris*, *écrirais*, *écrit*, *écrivant*, etc. En utilisant une expression régulière comme **écri[a-z]^+** on peut retrouver toutes ces formes en même temps. De la même façon en utilisant l'expression **[Ww]ork(s|ing|ed)?** on peut retrouver les formes *Work*, *work*, *Works*, *works*, *Working*, *working*, *Worked* ou *worked*.

Dans cet exemple, la suite de caractères [a-z] désigne l'intervalle a-z (n'importe quel caractère - un seul caractère - dans cette intervalle) alors que le caractère + est un quantificateur indiquant 'une ou plusieurs' occurrences de la sous-expression précédente. L'expression **écri[a-z]^+** dans son ensemble correspond à l'ensemble de chaînes de caractères comportant le motif *écri* suivi d'au moins une lettre minuscule.

Autres éléments de syntaxe des expressions régulières :

- Le caractère . remplace n'importe quel caractère.

- `b.lle` - > correspond aux chaînes de caractères *balle*, *belle*, *bulle*, *bille* mais aussi *b.lle*, *bplle*, *b3lle* etc.
- `[ab2X]` décrit l'ensemble de caractères `a,b,2,X` et l'expression `x[ab2X]y` décrit les chaînes *xay*, *xby*, *x2y*, *xXy*
- `[a-c],[0-38]` et `[a-d5-8X-Z]` désignent respectivement l'ensemble de caractères (n'importe quel caractère de l'ensemble) *a,b,c*, un des caractères numériques *0,1,2,3,8* (les caractères dans l'intervalle 0-3 plus le caractère 8) et un des caractères suivants : *a,b,c,d,5,6,7,8,X,Y,Z*
- Il est possible d'exclure un ensemble de caractères d'une requête à l'aide du caractère `^` : `[^0-9]` décrit n'importe quel caractère qui n'est pas un chiffre, `[^]` décrit n'importe quel caractère qui n'est pas un espace, etc.
- Les caractères `^` et `$` permettent de retrouver un motif en début respectivement en fin de ligne.

NOTE : le caractère `^` n'a la signification de *non* qu'entre crochets, comme ci-dessus.

QUESTION : Quelle est la différence entre `[^abc]` et `^[abc]` ?

- `X*` désigne une suite quelconque d'occurrences de `X` (0 ou plus)
- `X+` désigne au moins une occurrence de `X` (1 ou plus)
- `X?` désigne une occurrence optionnelle de `X` (0 ou 1)
- `X{n, m}` entre `n` et `m` occurrences de `x` (au moins `n`, au plus `m`)
- Enfin le caractère `|` (*ou* logique) permet de retrouver des expressions régulières alternatives : *chats/chiens/souris*

NOTE : étant donné que les **guillemets** (`"` ou `'`) font partie de la syntaxe de la commande `grep`, afin de les inclure dans une commande, et les caractères spéciaux `*`, `+`, `.`, `?`, `[`, `]`, `{`, `}` ont un rôle dans la syntaxe des expressions régulières, on doit les protéger avec un `\`

La plupart des concordanciers (wall, IMS Corpus Workbench, Unitex) intègrent la possibilité d'effectuer des requêtes en utilisant des expressions régulières. Mais ce qui nous intéresse dans le cadre de ce cours est leur utilisation pour des requête et substitutions par un nombre d'utilitaires Unix : **grep** et **sed**.

4.4. Exemples d'utilisation avec la commande Grep

Exemple	Explication
grep gh	retrouver les lignes contenant “gh”
grep -E '^con'	retrouver les lignes commençant avec “con”
grep -E 'ing\$'	retrouver les lignes finissant en “ing”
grep -v gh	ignorer les lignes contenant “gh”
grep -v -E '^con'	ignorer les lignes commençant en “con”
grep -v -E 'ing\$'	ignorer les lignes finissant en “ing”
grep -E '[A-Z]'	les lignes comportant une majuscule
grep -E '^[A-Z]'	les lignes commençant avec une majuscule
grep -E '[A-Z]\$'	les lignes finissant avec une majuscule
grep -E '^[A-Z]*\$'	les lignes comportant uniquement des Majuscules
grep -E '[aeiouAEIOU]'	les lignes comportant une voyelle
grep -E '^[aeiouAEIOU]'	les lignes commençant avec une voyelle
grep -E '[aeiouAEIOU]\$'	les lignes finissant avec une voyelle
grep -i -E '[aeiou]'	
grep -i -E '^[aeiou]'	
grep -i -E '[aeiou]\$'	
grep -i -E '^[^aeiou]'	les lignes commençant avec une non-voyelle
Grep -i -E '[^aeiou]\$'	les lignes finissant avec une non-voyelle

Grep -i -E '[aeiou].*[aeiou]'	les lignes avec deux ou plusieurs voyelles
Grep '^[^aeiou]*[aeiou][^aeiou]*\$' -iE	les lignes comportant exactement une voyelle

4.5/ Exercices : Grep

a/ De retour dans le répertoire Linux/Exos, choisir un terme à chercher dans les fichiers corpus. (par exemple *volcan*). Trouver à l'aide de la commande `grep` une suite d'au maximum 35 caractères, suivis du motif *volcan*, suivis d'une autre suite d'au maximum 35 caractères. Utilisez l'option `-l` pour n'afficher que le motif recherché. Le résultat devrait ressembler à une concordance KWIC (Key Word in Context)

Combien y a-t-il de lignes vides dans le corpus ?

Y a-t-il des symboles `*` dans ce corpus ?

Un des mots *true*, *True*, *TRUE* est-il présent aussi ?

b/ sur le site <http://www.eila.jussieu.fr/~avolansk>, télécharger dans linux/exos le fichier html correspondant au lien <http://bowland-files.lancs.ac.uk/monkey/ihe/linguistics/contents.htm>

- obtenez la liste des balises html de ce document.
- Y a-t-il des caractères en *italique* dans ce document ?
- Quelle commande taper pour obtenir la liste des urls se trouvant dans cette page ?

c/ Dans une expression régulière étendue, le symbole `^` correspond à un début de ligne, le symbole `$` correspond à une fin de ligne. Ainsi, `grep -E "[ab]$"` affichera les lignes de son entrée réduites à la seule lettre a ou b, et seulement celles-ci.

Trouver les trois commandes permettant d'afficher la liste des fichiers du répertoire courant faisant :

- exactement quatre lettres,
- au plus quatre lettres.
- dont le nom commence par un a, et se finit par un e.

Tester ces commandes dans le répertoire `/usr/bin`. Si une liste est trop longue, vous pouvez rediriger une nouvelle fois la sortie de la commande correspondante vers `more`.

d/ Exercices : grep, sort, uniq, cut, head, tail, wc

De retour dans le répertoire Linux/Exos, choisir un terme à chercher dans les fichiers corpus. (par exemple *volcan*). Trouver à l'aide de la commande `grep` une suite d'au maximum 35 caractères, suivis du motif *volcan*, suivis d'une autre suite d'au maximum 35 caractères.

Utilisez l'option `-l` pour n'afficher que le motif recherché. Le résultat devrait ressembler à une concordance KWIC (Key Word in Context)

Combien y a-t-il de lignes vides dans le corpus ?

Trouver les mots dans le corpus anglais finissant en `-ing`.

Y a-t-il des symboles `*` dans ce corpus ?

Un des mots *true*, *True*, *TRUE* est-il présent aussi ?

Combien de lignes contiennent deux ou plus occurrences du mot *volcan*? (le mot *volcan*, mais pas ses dérivés). NOTE : afin de trouver les occurrences du mot *volcan*, le rechercher précédé et suivi de blancs (espace, retour chariot, ou tab - vous pouvez utiliser le raccourci `\b`) ou ponctuation forte : `.!?`

Combien de lignes contiennent exactement deux occurrences du mot *volcan* (pas plus)?

Combien de lignes comportent le motif *volcan*?

Combien de lignes contiennent le motif *volcan* mais pas les dérivés de *erupt*?

Combien de lignes contiennent le motif *volcan* séparé de *erupt* par pas plus de 10 caractères. (le motif *erupt* précède ou suit le mot *volcan*).

Constituer la liste triée des mots composés dans votre corpus. Chacun des composants de ces mots doit faire au moins 2 caractères. Constituez maintenant la liste de premiers composants de ces mots (liste des mots qui précèdent le `-`).

e/ sur le site <http://www.eila.jussieu.fr/~avolansk>, télécharger dans Linux/Exos le fichier html correspondant au lien

<http://bowland-files.lancs.ac.uk/monkey/ihe/linguistics/contents.htm>

- obtenez la liste des balises html de ce document. Comment obtenir une liste de ces balises ou chacune apparaît une seule fois?
- Y a-t-il des caractères en *italique* dans ce document ?
- Quelle commande taper pour obtenir la liste des URLs se trouvant dans cette page ?

f/ Réutiliser les motifs : **les variables \1, \2, etc**

Les variables `\1`, `\2`, `\3`, etc. permettent de faire référence à la 1ère, 2ème, 3ème, etc. sous-expression d'une expression régulière sans la répéter. Par exemple, afin de retrouver les phrases comportant plusieurs occurrences de *volcan*, on peut utiliser l'expression suivante :

```
grep -E "\b(volcan)\b.*\b\1\b" fichier
```

Ici, le symbole `\1` fait référence au premier motif enfermé entre parenthèses.

EXERCICE g : réécrire la commande permettant de retrouver les phrases comportant 2 occurrences de *volcan*, mais pas 3.

EXERCICE h : retrouver dans votre corpus le mots composés construits sur le patron : `mot1-mot2-mot1`, par exemple : *arm-in-arm*, *peer-to-peer*, *porte-à-porte* et uniquement ces mots. (revoir les options de la commande `sort`).

Ensuite, trier ces mots par ordre alphabétique, compter le nombre d'occurrences de chacun de ces composants, et trouver les 20 les plus fréquents. Quel est le patron le plus fréquent pour

ces composants? Compléter la commande `grep` avec une autre commande `grep` qui permet de éliminer de la requête ce patron (en utilisant l'opérateur `|`).

EXERCICE i : deux options utiles de la commande `sort`. L'option `-f` permet à la commande `sort` d'ignorer la différence entre majuscules et minuscules. L'option `+1` par exemple permet de faire le tri sur le premier champ d'un fichier organisé en colonnes. Exemple : trouver l'expression régulière permettant de retrouver les occurrences de *volcan* suivi d'un autre mot et uniquement ces deux mots (revoir les options de la commande `grep`). Trouver les 30 mots qui apparaissent le plus souvent à droite du motif *volcan* en ignorant la différence entre minuscules et majuscules.

EXERCICE j : En utilisant les commandes `head/tail`, imprimez les lignes 25 à 75 de votre corpus. Mettre le résultat dans un fichier appelé `petit.txt`. Combien de lignes/mots/caractères fait `petit.txt`?

EXERCICE k : extraction de termes très simplifiée. Quelques options intéressantes de la commande `grep`. On a vu lors de la dernière séquence comment extraire une liste de fréquence de mots ou de deux mots à partir de vos corpus. Étant donné que vos corpus ne sont pas étiquetés, on n'a pas moyen de préciser qu'on ne cherche que les suites Nom Nom ou Nom de Nom, ou alors qu'on ne s'intéresse pas aux mots très fréquents comme *le*, *the*, *il*, etc. Afin de trouver des mots-clés de vos corpus, ou des termes, on peut construire des listes d'exclusion (*stop lists*) comportant ces mots et ne pas les compter.

Construisez la liste d'exclusion de mots fonctionnels les plus fréquents à partir de la liste des 100 mots les plus communs de votre corpus (comme vu en cours la dernière fois ; utiliser la commande `cut` pour ne garder que les mots, sans leur fréquence). Mettez-la dans un fichier *exclusion.txt* à l'aide de l'opérateur `>`. Si nécessaire (si la liste contient des mots clés que vous ne voudriez pas exclure), vous pouvez l'éditer avec l'éditeur `gedit`. Vous pourrez compléter et mettre à jour cette liste plus tard.

Maintenant on peut essayer de construire la liste de mots les plus fréquents de votre corpus en excluant les mots de cette liste. Les options de `grep` qui vont nous permettre d'obtenir cette nouvelle liste sont :

`-v` pour imprimer les lignes correspondant **pas** à un patron.

`-f` dit à la commande `grep` de lire le patron à chercher dans un fichier (en l'occurrence *exclusion.txt*) et non pas sur la ligne de commande.

`-x` l'option dit à `grep` que le motif recherché doit correspondre à toute la ligne, non seulement à une partie. Cette option est nécessaire afin d'éviter d'éliminer des mots comportant la sous-chaîne *to*, *the*, etc. comme par exemple **touch** or **therapy**.

Utiliser ainsi la commande `grep -Evix -f exclusion.txt` après avoir segmenté le texte en un token par ligne, mais avant de compter le nombre d'occurrences.