TP n° 5 : Expressions régulières

Préparation : trois fichiers issus du [projet Gutenberg http ://www.gutenberg.org/] sont utilisés pour illustrer ce TP : `cyrano.txt, fable1.txt` et `fable2.txt`. Ils sont disponibles dans le répertoire `data/txt/`

Nettoyage des fichiers
Les fichiers récupérés devront être nettoyés (c'est l'objectif de l'exercice 1) avant d'être utilisés. À l'aide de la commande `file`, on s'informe sur le codage :

```
$ file cyrano.txt
cyrano.txt: UTF-8 Unicode (with BOM) English text, with CRLF line terminators
```

Les fins de ligne sont codées CR + LF, comme on peut le voir à l'aide de la commande `hexdump` (noter les`0d 0a`, respectivement `\r : CR` et `\n : LF`, en fin de ligne) :

```
$ hexdump -C cyrano.txt | head
00000000  ef bb bf 54 68 65 20 50  72 6f 6a 65 63 74 20 47  |...The Project G|
00000010  75 74 65 6e 62 65 72 67  20 45 42 6f 6f 6b 20 6f  |utenberg EBook o|
00000020  66 20 43 79 72 61 6e 6f  20 64 65 20 42 65 72 67  |f Cyrano de Berg|
00000030  65 72 61 63 2c 20 62 79  20 45 64 6d 6f 6e 64 20  |erac, by Edmond |
00000040  52 6f 73 74 61 6e 64 0d  0a 0d 0a 54 68 69 73 20  |Rostand....This |
00000050  65 42 6f 6f 6b 20 69 73  20 66 6f 72 20 74 68 65  |eBook is for the|
00000060  20 75 73 65 20 6f 66 20  61 6e 79 6f 6e 65 20 61  | use of anyone a|
00000070  6e 79 77 68 65 72 65 20  61 74 20 6e 6f 20 63 6f  |nywhere at no co|
00000080  73 74 20 61 6e 64 20 77  69 74 68 0d 0a 61 6c 6d  |st and with..alm|
00000090  6f 73 74 20 6e 6f 20 72  65 73 74 72 69 63 74 69  |ost no restricti|
```

Nous sommes sous Linux, où le `LF` est suffisant, donc il convient de supprimer le CR... De plus, le texte est entouré de méta-données en début et fin de document, séparées du texte par des lignes contenant respectivement `**START OF THIS PROJECT**` et `**END OF THIS PROJECT**`.

Exercice 1. Écrire un script `cleaner` qui corrige les fins de ligne, à l'aide de la commande `tr -d` et qui supprime en-tête et pied de document, à l'aide des commandes `head` et `tail`.

Pas-à-pas :

- commencer par trouver les numéros des lignes `**START...**` et `**END...**` à l'aide de `grep` et `cut`, et les stocker dans des variables ;
- retirer tous les `CR` du texte ;
- enfin, couper les blocs de début et de fin de projet.

Utiliser le script pour créer deux fichiers contenant pour l'un, les textes de Cyrano de Bergerac et pour l'autre, les Fables de La Fontaine :

```
$ ./cleaner cyrano.txt > cyrano.clean
$ (./cleaner fable1.txt ; ./cleaner fable2.txt) > fables.clean
```

À l'aide de `grep` :
Les exercices qui suivent ont pour but de se familiariser avec `grep`, commande dont il convient tout d'abord de consulter le manuel `(man grep)`.

Exercice 2. Compter les répliques de chaque personnage de Cyrano de Bergerac et afficher les 5 premiers résultats triés par ordre décroissant d'occurrence.

```
$ ./repliques cyrano.clean
    612 CYRANO
    333 ROXANE
    193 CHRISTIAN
    145 DE GUICHE
    109 LE BRET
```

Pas-à-pas :

- Repérer dans le texte à quoi ressemble une ligne de début de réplique ;
- récupérer uniquement ces lignes grâce à `egrep` (et à une expression régulière...) ;
- récupérer le nom du personnage dans la ligne ;
- Reprendre les mécanismes de tri et décompte vus dans une séance précédente...

Exercice 3. Afficher dans l'ordre du texte les noms des actes et des scènes ainsi que les descentes de rideau de Cyrano de Bergerac (à l'aide de la commande `egrep`).
Pas-à-pas :

- repérer dans le texte à quoi ressemble une ligne de début d'acte ou scène ou rideau ;
- récupérer uniquement ces lignes grâce à `egrep` (et à une expression régulière...)

Exercice 4. Écrire un script qui prend en paramètre le nom d'une scène, compte les répliques de chaque personnage dans cette scène et affiche les résultats triés par ordre croissant d'occurrence.
Pas-à-pas :
Commencer par écrire un script qui extrait le texte d'une scène à partir du numéro de celle-ci

- regarder à quoi ressemble une ligne de début de scène ;
- écrire un code qui trouve les numéros de ligne de début et fin de scène (on pourra s'aider de l'exercice 3, en récupérant en plus les numéros de ligne).

Il ne reste qu'à compter les répliques, comme vu précédemment.

À l'aide de `sed` :

Exercice 5. Afficher les lignes contenant le mot `fourmi` dans les Fables de la Fontaine, à l'aide de `grep` puis à l'aide de `sed`. Enfin, afficher les mêmes lignes où le mot `fourmi` est remplacé par le mot `carpe`. Notons que par défaut, `sed` affiche toutes les lignes du texte (si utilisé en mode substitution). Si on ne veut afficher que les lignes pertinentes, il faut passer en mode silencieux (option `-n`) et demander l'affichage des lignes correspondant au motif (commande `"p"` dans la dernière rubrique de la substitution). Un exemple vous sera donné...

Exercice 6. Écrire un script shell `html2txt` à l'aide de `sed` qui, appliqué au fichier d'exemple `sample.html`, produit un fichier identique au fichier `sample.txt`. On pourra s'aider de la commande `diff` pour comparer le fichier produit au fichier souhaité.

Il s'agira tout simplement d'empiler des commandes de substitution. Pour ce faire, on pourra passer par un script `sed` avec une substitution par ligne, plutôt que par une ligne interminable...

Exercice 7. Si vous avez fini et que vous vous ennuyez...

À l'aide de la commande `find`, écrire un script qui recherche récursivement dans les fichiers des répertoires passés en argument, les adresses email qui y apparaissent, puis affiche ces adresses par ordre décroissant d'occurrence.