Bases de Données Avancées (ADB'17)

TD 1 : Expressions Régulières et `grep`

Exercice 1 : Les expressions régulières de base. Télécharger de la page du cours le fichier `abc.txt` et le sauvegarder dans un répertoire choisi (par exemple `~/adb18/td1/`). Ouvrir le fichier dans un éditeur de texte et dans un terminal changer le répertoire courant à celui où se trouve le fichier (`cd adb18/td1`). Exécuter la commande

```
grep -E -n "^(E)$" abc.txt
```

en remplaçant *E* par chacune des expressions suivantes :

1. `a`
2. `b`
3. `ab`
4. `(a|b)`
5. `a*`
6. `a+`
7. `a?`
8. `(a|b)*`
9. `a*b*`
10. `(ab)*`
11. `()`
12. `(a|b| )*`

Pour chaque expression noter les numéros des lignes du fichier qui lui correspondent et observer attentivement leur sémantique. Remarquer les différences entre la syntaxe des expressions `grep` et la syntaxe des expressions `DTD` présentées pendant le cours.

Exercice 2 : Les expressions "bracket". Utiliser les expressions `[ab]` et `[ab]*` avec `grep` et le fichier de l'exercice précédent. Ces expressions sont-elles équivalentes à certaines expressions de la liste de l'exercice 1, et si oui lesquelles ? (Deux expressions sont équivalentes si elles donnent les mêmes résultats). Proposer une expression équivalente à `(a|b| )*` qui n'utilise pas de parenthèses `(` et `)` mais uniquement des crochets `[` et `]`. Tester ces expressions en utilisant `grep` avec le fichier `abc.txt`.

Exercice 3 : Complément. L'expression `[abc]` correspond à un caractère parmi `a`, `b` et `c` et l'expression `[^abc]` correspond à tout caractère différent de `a`, `b` et `c`. On dit alors que l'expression `[^abc]` est le *complément* de l'expression `[abc]` parce que leur union (`[abc]|[^abc]`) est un caractère quelconque. Utiliser le complément pour construire des expressions régulières qui correspondent à 1) des mots qui ne contiennent pas de caractère `b` 2) des mots qui ne contiennent pas les caractères `b` et `c`. Tester ces expressions en utilisant `grep` avec le fichier `abc.txt`.

Exercice 4 : Les intervalles de caractères. Le langage des expressions régulières utilisé par `grep` permet de spécifier un intervalle de caractères consécutifs d'une manière très simple et pratique : l'expression `[a-d]` est ainsi équivalente à `[abcd]` et donc les deux expressions correspondent à des mots qui contiennent uniquement une seule occurrence d'un des caractères `a`, `b`, `c` ou `d`. Quelques exemples d'intervalle souvent utilisés :

- `[0-9]` correspond à un chiffre entre 0 et 9 ;
- `[1-9]` correspond à un chiffre entre 1 et 9 ;
- `[a-z]` correspond à une lettre minuscule (sans accent) ;
- `[A-Z]` correspond à une lettre majuscule (sans accent) ;
- `[a-zA-Z]` et `[A-Za-z]` qui sont équivalents et correspondent à une lettre (non accentuée) quelconque.

Les intervalles de caractères ci-dessus sont souvent utilisés pour construire des expressions plus complexes. Par exemple, `[1-9][0-9]*` est une expression régulière qui correspond aux nombres entiers strictement positifs ($1, 2, 3, \ldots, 10, 11, \ldots$) parce que tout nombre entier positif commence par un chiffre entre 1 et 9 suivi d'une suite de chiffres quelconques (donc entre 0 et 9). Remarquer que 01 et 0010 ne sont pas bien formés. Proposer des expressions régulières pour les classes de nombres suivantes :

(i) Les nombres entiers naturels (0, 1, 2, 3, 4, $\ldots$)

(ii) Les nombres entiers ($\ldots$, -2, -1, 0, 1, 2, $\ldots$)

(iii) Les nombres décimaux ayant obligatoirement une partie entière et une partie décimale (par exemple $12{,}234$, $0{,}234$, $-12{,}254$, $-0{,}234$, $0{,}0$; mais $,1$ et $24,$ ou même 12 n'appartiennent pas à cette classe)

Vérifier les expressions sur le fichier `chiffres.txt` disponible sur le site du cours. Les expressions devraient sélectionner les lignes suivantes : (i) 1, 2, 5, 6, 7, 8 ; (ii) toutes les lignes de (i) et 9, 11, 12, 14 ; (iii) 15, 16, 18, 21, 22.

Exercice 4 : Recherche des noms. Utiliser les intervalles `[a-z]` et `[A-Z]` pour capturer les chaînes de caractères avec des noms complets bien formés : un ou deux prénoms suivi par le nom de famille. Chaque nom doit se commencer par une lettre majuscule. On peut supposer que les noms n'utilisent ni espace ni tiret (-). Vérifier les expressions en utilisant `grep` avec un fichier contenant quelques noms bien formés et quelques nom mal-formés.

Exercice 5 : Recherche d'information. Télécharger le fichier `musica.csv` et se renseigner sur le format CSV (comma-separated values) utilisé par ce fichier pour stocker un catalogue de disques. Ouvrir le fichier dans un éditeur de texte et repérer la structure générale des lignes et remarquer que la première et la dernière lignes ne la suivent pas. Proposer une expression régulière qui corresponde uniquement aux lignes bien formées et utiliser la commande `grep` avec l'option `-c` qui retourne le nombre des lignes qui correspondent à l'expression donnée (il devrait y en avoir 857). Ensuite, écrire des expressions régulières qui permettent de trouver des lignes avec les informations suivantes :

1. les disques de Bob Dylan ; même question avec AC DC, The Rolling Stones, David Bowie, Jacques Higelin.
2. les disques avec la chanson « Brown Sugar » ; même question avec « Little Red Lobster », « Elisa » et « Start me up ».
3. les disques sortis en 2005.

Exercice 6* : Pour les geeks confirmés. Dans l'exercice précédent utiliser les programmes `cut`, `sort` et `uniq` pour afficher uniquement les titres des disques et ses artistes.