Les expressions rationnelles

Thierry Vaira

v1.1 - 30 octobre 2017

Les expressions rationnelles

- Les expressions rationnelles (*Regular Expressions*) sont beaucoup utilisées sous UNIX, et notamment avec les outils d'éditions de texte et les filtres (`sed`, `grep`, `awk`, ...).
- Il s'agit d'un mécanisme qui permet de décrire des ensembles de caractères dans le cadre d'une recherche ou d'un remplacement de texte.
- Une expression rationnelle est une suite de caractères qu'on appelle plus simplement `motif` (*pattern*) pour trouver une correspondance (*match*).
- Les mécanismes de base pour former un motif sont basés sur des caractères spéciaux de substitution, de groupement et de quantification.

Les quantificateurs I

- ? qui définit un groupe qui existe zéro ou une fois : `toto?` correspondant (*match*) alors à « tot » ou « toto » mais pas « totoo » ;
- * qui définit un groupe qui existe zéro ou plusieurs fois : `toto*` correspondant à « tot », « toto », « totoo », « totooo », etc. ;
- + qui définit un groupe qui existe une ou plusieurs fois : `toto+` correspondant à « toto », « totoo », « totooo », etc. mais pas « tot ».
- `{n}` qui définit exactement *n* occurrences de l'expression précédant les accolades : `a{3}` correspondant à « aaa » mais pas à « aa », ni « aaaa » etc . ;

Les quantificateurs II

- `{n,m}` qui définit entre *n* et *m* occurrences de l'expression précédant les accolades : `a{2,4}` correspondant à « aa », « aaa », « aaaa » mais pas à « a », ni « aaaaa » ;
- `{n,}` qui définit au moins *n* occurrences de l'expression précédant les accolades : `a{3,}` correspondant à « aaa », « aaaa », « aaaaa » mais pas à « aa » ;

Remarque : Pour neutraliser un caractère spécial, il faut l'« échapper », c'est-à-dire le faire précéder du caractère \ (anti-slash).

Les opérateurs de base I

- l'opérateur de concaténation de deux expressions (implicite) : ab correspondant à « ab » mais pas à « a », ni « b », ni une chaîne vide ;
- . qui définit un caractère et un seul : . correspondant à « a », « b », ... mais pas une chaîne vide, ni « ab » ;
- | qui est l'opérateur de choix entre plusieurs alternatives. Il peut être combiné autant de fois que nécessaire pour chacune des alternatives possibles. Il fait correspondre l'une des expressions placées avant ou après l'opérateur : a|b correspondant à « a », « b » mais pas à une chaîne vide, ni « ab », ni « c » ;

Les opérateurs de base II

- `[]` qui définit un des caractères entre crochets (« classe de caractères ») : `[aeiou]` correspondant à « a », « e », « i », ... mais pas à une chaîne vide, ni « b », ni « ae » ; Entre ces crochets, un intervalle de caractères peut être indiqué en donnant le premier et le dernier caractère, séparés par un tiret : `[a-d]` est équivalent à `[abcd]`
- `[^]` qui définit un caractère n'étant pas entre crochets (négation) : `[^aeiou]` correspondant à « b », ... mais pas à une chaîne vide, ni « a », ni « e », ... ni « bc » ;
- `()` qui définit un groupement de l'expression entre parenthèses : `(détecté)` correspondant à « détecté » mais pas à « détect », « détecta », « détectés » ;

Les opérateurs de base III

- ^ qui ne correspond à aucun caractère mais fixe une condition en indiquant que ce doit être au début d'une ligne : ^a trouve « a » en début de ligne mais pas dans « ba » ;
- $ qui ne correspond à aucun caractère mais fixe une condition en indiquant que ce doit être à la fin d'une ligne : a$ trouve « a » en fin de ligne mais pas dans « ab » ;
- Entre les crochets [], les métacaractères sont interprétés de manière littérale : [.?*] désigne l'ensemble constitué des caractères « . », « ? » et « * ».
- Les sous-ensembles placés entres (et) peuvent êtres rappelés par un numéro correspondant à leur ordre de déclaration, précédé du caractère \.

Les standards I

Le standard POSIX propose plusieurs normes :

- BRE (*Basic Regular Expressions*) pour les expressions rationnelles basiques. C'est par exemple le standard par défaut pour `sed` et `grep`.
- ERE (*Extended Regular Expressions*) pour les expressions rationnelles étendues. C'est l'option `-E` pour `grep` et `-r` pour `sed`.

Les expressions rationnelles de **Perl** sont également un standard de fait, en raison de leur richesse et de leur puissance (elles ont donné la bibliothèque **PCRE**). C'est l'option `-P` pour `grep`.

Les notations peuvent varier légèrement d'un moteur d'expressions rationnelles à l'autre. Un moteur d'expressions rationnelles est un outil permettant de manipuler des expressions rationnelles.

Les standards II

- Dans BRE, les accolades, les parenthèses, le symbole « ? » et le symbole « + » ne sont pas des métacaractères : ils ne représentent qu'eux même. Pour prendre leur notion de métacaractères, ils ont besoin d'être échappés par le symbole « \».
- Contrairement aux expressions rationnelles basiques, la norme ERE reconnaît les caractères vus précédemment comme des métacaractères. Ils doivent ainsi être échappés pour être interprétés littéralement.
- La plupart des exemples donnés ici étaient des expressions régulières étendues POSIX.

Les classes de caractères I

Les classes de caractères les plus utilisées sont généralement fournies avec le moteur d'expression régulière.

Quelques classes de caractères POSIX prédéfinies :

- `[:digit:]` = chiffres décimaux
- `[:alpha:]` = caractères alphabétiques
- `[:alnum:]` = caractères alphanumériques
- `[:blank:]` = espace et tabulation
- `[:space:]` = caractères d'espacement
- `[:punct:]` = caractères de ponctuation
- etc ...

➠ `[[:alnum:]]` est équivalent à `[0-9A-Za-z]` pour des caractères ASCII.

Les commandes

- Grep (*Global Regular Expression Printer*) permet de faire des recherches de lignes contenant une chaîne correspondant à une expression rationnelle.
- Sed (*Stream Editor*) est un éditeur qui possède les mêmes fonctionnalités que l'éditeur ed mais qui ne travaille pas en mode interactif. Il permet donc, contrairement à grep, de modifier le flux de lignes qui lui est passé.
- etc ...

Exemple

Utilisation d'expressions rationnelles avec grep et sed sur un fichier texte contenant des codes postaux :

```
$ cat liste.txt
Sarrians 84260
Avignon 84000
Carpentras 84200
Jonquières 84150
Marseille 13000
Istres 13800
Vitrolles 13127
Paris 75000
```

```
// sed en mode ERE (extended)
$ cat liste.txt | sed -rn '/(84|13)[[:digit:]]{3}/p'
// grep en mode BRE (basic)
$ cat liste.txt | grep '\(84\|13\)[[:digit:]]\{3\}'
// grep en mode ERE (extended)
$ cat liste.txt | grep -E '(84|13)[[:digit:]]{3}'
Sarrians 84260
Avignon 84000
Carpentras 84200
Jonquières 84150
Marseille 13000
Istres 13800
Vitrolles 13127
```