

# Cours UNIX

## Chapitre 5

### Filtres

## 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 que nous allons voir dans ce chapitre, ainsi que dans nombre de langages de programmation (en particulier dans Javascript, Perl, PHP et Ruby).

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. Sans les expressions rationnelles, la manipulation de fichiers texte en ligne de commande ou par programmation est un véritable cauchemar.

Les expressions rationnelles sont extrêmement utiles sous UNIX, étant donné que tous les fichiers de configuration et de journalisation (logs) sont des fichiers texte.

### → Caractères spéciaux

Pour neutraliser un caractère spécial, le précéder du caractère `\` (*anti-slash*).

- [ ] définition d'un ensemble de caractères (voir ci-dessous)
- ^ deux significations:
  - Début de ligne lorsqu'il est au début de l'expression rationnelle
  - Complément (négation) de l'ensemble de caractères si il est juste après le caractère [
- \$ fin de ligne lorsqu'il est en fin d'expression rationnelle
- .
- \*
- +
- ?

### → Ensembles

Les ensembles sont délimités par les caractères [ et ]. Une série de caractères peut être exprimée avec le caractère -.

Exemples:

- [012345679] n'importe quel chiffre.
- [0-9] n'importe quel chiffre.
- [a-z\_] n'importe quel caractère de a à z (minuscules) ou le caractère \_.
- [^a-z] n'importe quel caractère SAUF les caractères de a à z en minuscule.

### → Exemples

- abc\$ lignes se terminant par abc
- ^abc lignes commençant par abc
- [abc] lignes contenant les lettres a ou b ou c
- ^[abc] lignes commençant par a ou b ou c
- .\* toutes les lignes
- ^...\$ toutes les lignes de 3 caractères.

### → Sous ensembles

Permettent de définir des sous-ensemble dans un ensemble de caractère. Les sous-ensembles sont placés entre `(` et `)` et peuvent être rappelés par un numéro correspondant à leur ordre de déclaration, précédé du caractère `\`.

Exemples:

- \(25[0-9]\*\).\*\1.\*\1 toutes les lignes contenant 3 fois le premier chiffre trouvé commençant par 25.

## Grep

Grep (Global Regular Expression Printer) permet de faire des recherches de lignes contenant une chaîne correspondant à une expression rationnelle.

Syntaxe:

```
grep [options] 'expressionrationnelle' [fichiers...]
```

Si aucun fichier n'est précisé (en dernier argument) alors c'est l'entrée standard qui est filtrée.

Options couramment utilisées:

- v inverse de la recherche (toutes les lignes ne contenant pas l'expression).
- c affiche le nombre de lignes trouvées.

Exemples:

```
# grep '^[aw-z]' /etc/services
zip          6/ddp      #Zone Information Protocol
auditd      48/tcp     #Digital Audit Daemon
auditd      48/udp     #Digital Audit Daemon
xns-time    52/tcp     #XNS Time Protocol
...

# cat /etc/services | grep '^t[^t]o'
troff       2014/tcp

# ls -l /etc | grep 'rc\..*$'
-rw-r--r--  1 root  wheel    4903 Nov 20  2000 rc.atm
-rw-r--r--  1 root  wheel    217 May 21  09:31 rc.conf
-rw-r--r--  1 root  wheel    1667 Nov 20  2000 rc.devfs
...
```

## Sed

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.

Syntaxe:

```
sed [options] 'commande' [fichiers..]
```

Les commandes sont de la forme:

```
[adresse1[,adresse2]] fonction [arguments...]
```

Les adresses sont:

- des nombres décimaux correspondant à des numéros de lignes.
- Le caractère `$` désignant la dernière ligne

Exemple: `10,$` désigne de la 10ème à la dernière ligne.

Si les adresses ne sont pas spécifiées, la commande est exécutée sur toutes les lignes.

Commandes couramment utilisées:

```
y/texte1/texte2
```

substitution (remplacement) d'un texte par un autre.

```
s/expressionrationnelle/texte/flags
```

substitution (remplacement) d'un texte par un autre à l'aide d'une expression rationnelle.

Flags peut être, entre autre:

- `g` remplacer toutes les occurrences dans une même ligne.

Options couramment utilisées:

- n ne pas afficher la ligne qui vient d'être traitée.

Exemples:

```
# cat /etc/rc.conf
...
ifconfig_xl0="inet 10.0.0.1 netmask 255.255.255.0"
...
# cat /etc/rc.conf | sed 's/255\./xxx\./g'
...
ifconfig_xl0="inet 10.0.0.1 netmask xxx.xxx.xxx.0"
...
```