Cours des Systèmes d'Exploitation LINUX

ENSAM – Casablanca

2022-2023

Chapitre 7

Expressions régulières

Introduction

Les **expressions régulières** permettent de décrire des motifs formés de caractères. Ce mécanisme est très similaire au mécanisme de **génération de noms de fichiers** par le shell. Il en diffère toutefois par les aspects suivants : la génération de noms est un mécanisme rudimentaire qui décrit de façon générique des **noms de fichiers**, alors que les expressions régulières représentent une description générique très sophistiquée de chaînes de **caractères contenues dans des fichiers (ou un flot de caractères).**

Exemples :

```
grep '[a-zA-Z]' fich1.f
grep '[a-zA-Z]' *.f
grep '[a-zA-Z]' [A-Z]*.f
```

Expression régulière

Génération des noms de fichiers

Introduction

Une expression régulière est une description d'une chaîne de caractères. Elle se présente elle-même sous forme de chaîne, mais son contenu est symbolique et doit être interprété comme tel. La syntaxe des expressions régulières est en outre différente de la syntaxe de génération de noms. Ces expressions sont utilisées par un grand nombre d'utilitaires : non seulement grep, sed et awk, mais également par le langage perl (the Practical Extraction and Report Language) et tout éditeur de texte, dont vi et emacs.

Les expressions régulières servent à décrire de façon concise et claire des motifs de caractères, comme par exemple :

« toutes les chaînes de 1 à 8 caractères alphanumériques, le premier étant alphabétique ».

Cette phrase se traduit en expression régulière devient :

```
[A-Za-z][A-Za-z0-9]{0,7}
```

Les conventions

- Premièrement, Il existe deux sortes d'expressions régulières : **expressions régulières de base** et les **expressions régulières étendues (ere)**.
- Aussi dans ce chapitre, on dira qu'une chaîne de caractères vérifie une expression régulière, et le terme caractère exclut le caractère <nouvel ligne>. Nous pouvons distingué les deux types de caractères :
  - **sp** désigne un caractère spécial. Les caractères spéciaux pour les expressions régulières sont les suivants : `| . * + ? ^ $ ( ) [ ] {} \ -`
  - **ch** désigne un caractère quelconque sauf un caractère spécial.
- Les expressions régulières sont construites progressivement, à partir de briques de base appelées **expressions régulières atomiques**.

Expressions régulières atomiques (ERA)

Une (ERA) Un motif constitué **d'un seul caractère** appartenant à un ensemble précis.

ERA	Définit l'ensemble de caractères suivant:
`ch`	l'ensemble constitué du seul caractère **ch**
`\sp`	l'ensemble constitué du seul caractère spécial **sp**
`.`	l'ensemble de tout les caractères
`[erp]`	l'ensemble constitué des caractères placés entre **[]**
`[^erp]`	l'ensemble constitué des caractère autres que **e,r,p**. (Le caractère **^** placé après **[** indique la négation).
`[d-m]`	l'ensemble constitué des caractères compris entre **d** et **m** (ordre alphabétique)
`[^d-m]`	l'ensemble constitué des caractères non compris entre **d** et **m**

Expressions régulières atomiques (ERA)

Exemples d'ERA:

`a`	le caractère a
`[i-n]`	un caractère appartenant à l'ensemble ijklmn
`[-i-n]`	un caractère appartenant à l'ensemble -ijklmn
`.`	un caractère quelconque
`\.`	le caractère .
`[^0-9]`	tout caractère autre qu'un chiffre
`[^0-9^]`	tout caractère autre qu'un chiffre ou ^ car le dernier ^ n'est pas après [.
`[a-zA-Z]`	un caractère appartenant à l'ensemble des majuscules et des minuscules .

La construction d'une expression régulière

Si l'on recherche une chaîne de caractères (ou un "mot") qui vérifie certaines conditions (on dit plutôt : respecte un certain **motif**), il faut pouvoir définir :

- l'ensemble des caractères constituant ce « mot »
- le nombre de caractères dans ce mot
- à quel endroit le chercher

En effet, il nous faut les mécanismes qui peuvent rependre a ces questions (quoi ? combien ? où ?).

Par exemple : un mot de 3 à 6 lettres minuscules au début de la ligne.

Les briques élémentaires (**ERA**) définissent des ensembles de caractères (quoi ?) et seront combinées entre elles par trois opérations élémentaires : la concaténation, la quantification (combien ?) et l'ancrage (où ?).

La concaténation

Une expression régulière (**ER**) peut être obtenue par concaténation (juxtaposition de l'une après l'autre) d'expressions régulières atomiques (**ERA**). La concaténation est décrite **sans opérateur ou caractère spécial.**

Exemples:

`smi3`	signifie : s suivi de m suivi de i suivi de 3.
`[Oo]ui`	la chaîne Oui ou la chaîne oui
`[A-Z][0-9]..`	une majuscule suivie d'un chiffre suivi de deux caractères quelconques : S3S5 ou T911 mais pas g1Lu
`[A-Z][0-9].\.`	une majuscule suivie d'un chiffre suivi d'un caractère quelconque suivi d'un point : S32. ou L4e. mais pas h7fu ou HL4.

La quantification

Les quantifieurs permettent de définir combien de fois l'**ERA** qui précède est répétée. les quantifieurs utilisables sont les suivants :

Quantifieur	Signification
`ERA*`	tout mot de 0 à N caractères vérifiant **ERA**
`ERA+`	tout mot de 1 à N caractères vérifiant **ERA**
`ERA?`	tout mot de 0 à 1 caractère vérifiant **ERA**
`ERA{n}`	tout mot de n caractères vérifiant **ERA**
`ERA{n,m}`	tout mot de n à m caractères vérifiant **ERA**
`ERA{n,}`	tout mot d'au moins n caractères vérifiant **ERA**
`ERA{,m}`	tout mot de 0 à m caractères vérifiant **ERA**

La quantification

Dans ce tableau, la phrase « tout mot de n caractères vérifiant ERA » signifie exactement : « tout mot de n caractères appartenant à l'ensemble défini par l'expression régulière atomique ERA ». Voilà quelques exemples :

`S*`	rien ou S ou SS ou SSS ou ...
`S.*`	S suivi de n'importe quelle chaîne (même vide) (S, Sds, S13ER,…). L'ERA (.) se répète 0 ou plusieurs fois.
`S+`	S ou SS ou SSS ou ...
`S?`	rien ou S
`S{4}`	la chaine SSSS
`[b-d]{2}`	bb ou bc ou bd ou cb ou cc ou cd ou db ou dc ou dd

L'encrage

Deux caractères spéciaux servent à préciser la position de la chaîne recherchée dans la ligne :

^ désigne le début de la ligne (s'il est placé au début de l'expression)

$ désigne la fin de la ligne (s'il est placé à la fin de l'expression)

Exemples :

`^Smi`	Smi au début de la ligne
`Smi$`	les lignes se terminant par Smi
`^Smi$`	les lignes ne contenant que Smi
`^$`	les lignes vides

ER Vs Génération de noms de fichier

Les deux principes de génération utilisent les mêmes caractères spéciales mais avec des significations différentes. Comme il est résumé dans le tableau suivant :

	Génération de noms de fichier	Expressions régulières
?	un caractère quelconque, sauf <new-line>	remplace zéro ou une fois le caractère qui précède
.	le caractère point	un caractère quelconque sauf <new-line>
*	zéro ou un nombre quelconque de caractè-res	remplace zéro fois ou n fois le caractère qui précède
[a-i]	un caractère entre a et i	un caractère entre a et i
[!a-i]	un caractère qui n'est pas entre a et i	un caractère entre a et i, ou !
[^a-i]	un caractère entre a et i, ou ^	un caractère qui n'est pas entre a et i
\	banalise le caractère qui suit	banalise le caractère qui suit
^	le caractère ^	ce qui suit est en début de ligne
$	référence une variable	ce qui précède est en fin de ligne

Combinaison d'expressions régulières

D'autres opérations sont essentielles pour résoudre certaines recherches plus complexes par la combinaison de plusieurs **ER**. Les deux opérations qui permettent de combiner entre elles des expressions régulières sont:

l'alternative et le **groupage**.

L’alternative

Les expressions régulières permettent de désigner "ceci ou cela" en séparant deux ER par le caractère | comme dans : "ceci|cela". Cette combinaison est une nouvelle expression régulière.

Exemples :

`Linux|Unix`	le mot Linux ou le mot Unix
`^Linux|^Unix`	commence par Linux ou Unix
`[A-Z]{8}|[0-9]{4}`	8 lettres majuscules ou 4 chiffres

Le groupage par ( ) et la notation \N

En principe, rien ne change si on met ou pas entre parenthèses une expression régulière ER. Puisque l'expression (ER) représente toute chaîne qui vérifie ER. Les parenthèses semblent inutiles.

Cependant Cette notation offre deux possibilités :

- les quantifieurs vus plus haut peuvent s'appliquer à un ensemble de ERA.

- la notation **\N**, où **N** est un nombre de 1 à 9, désigne la chaîne de caractères qui a vérifié l'expression placée dans la N^{iéme} paire de parenthèses.

Le groupage par ( ) et la notation \N

Exemples :

`FST`	le mot FST est aucun autre mots.
`FST{2}`	le mot FSTT et aucune autre chaîne : Le quantifieur {2} n'est appliqué qu'au dernier T.
`(FST){2}`	le mot FSTFST et aucune autre chaine : le quantificateur {2} agit sur tout ce qui est entre parenthèses
`(FST|Smi){2}`	FSTSmi ou FSTFST ou SmiFST ou SmiSmi
`(FST|SMI)\1`	FSTFST ou SmiSmi MAIS PAS FSTSmi ni SmiFST

Dans le groupage par (), la notation \N peut être complétée par * + ou ? Avec les significations habituelles :

- \1* signifie 0 à N fois la chaîne qui a vérifié le premier sous-motif
- \3+ signifie 1 à N fois la chaîne qui a vérifié le troisième sous-motif
- \2? signifie 0 ou 1 fois la chaîne qui a vérifié le deuxième sous-motif

D’autres expressions régulières atomiques

D’autres possibilités permettent de rendre les expressions régulières atomiques plus complètes et utilisables avec des jeux de caractères autre que **l’ASCII US** (en particulier avec des caractères accentués).

L’extension des possibilités des **ERA** permettent de résoudre les problèmes suivants:

- comment désigner un caractère spécial ?
- comment définir une classe de caractères indépendamment de la langue locale ?

Nous citons alors quelques caractères spéciaux non affichables à travers la séquence d’échappement **\X**

Les caractères spéciaux et la notation \X

Il est possible de désigner des caractères spéciaux de la façon suivante :

Caractères spéciaux	Significations
`\0`	le caractère <**null**>
`\a`	le caractère <**alert**> (bell)
`\b`	le caractère <**backspace**>
`\f`	le caractère <**form-feed**>
`\n`	le caractère <**new-line**>
`\r`	le caractère <**carriage-return**>
`\t`	le caractère <**tab**>
`\v`	le caractère <**vertical-tab**>
`\103`	le caractère dont le code ASCII est 1**03** en octal

Les classes de caractères

Les classes de caractères sont définies par la notation **[:code:]** et ne peuvent être utilisées que dans la définition d'une liste de caractères (ERA). La correspondance entre ces classes et les éléments qui les composent sont donnés ici pour la langue anglaise :

Classes	Significations
`[[:alnum:]]`	un alphanumérique (**[0-9a-zA-Z]**)
`[[:alpha:]]`	un alphabétique (**[a-zA-Z]**)
`[[:cntrl:]]`	un caractère de contrôle (**[\a\b\r\f\t\n\v]**)
`[[:digit:]]`	un digit (**[0-9]**)
`[[:graph:]]`	un caractère autre qu'alphanumérique ou ponctuation
`[[:lower:]]`	une lettre minuscule (**[a-z]**)
`[[:punct:]]`	un caractère de ponctuation
`[[:print:]]`	un caractère imprimable
`[[:space:]]`	un caractère d'espacement (**[\t\r\n\f]**)
`[[:upper:]]`	une lettre majuscule (**[A-Z]**)
`[[:xdigit:]]`	un digit hexa (**[0-9A-Fa-f]**)

Les classes de caractères

Par exemple en français, **[[:alpha:]]** désigne toutes les lettres alphabétiques **y compris** les caractères accentués, alors que **[a-zA-Z]** ne désigne que les 26 lettres de l'alphabet en minuscule ou majuscule.

Attention

Ne pas oublier **[:** et **:]** dans les classes :

`[[:print:]]`	correct : un caractère imprimable.
`[^[:print:]]`	correct : un caractère non imprimable.
`[^:print:]`	incorrect.
`[:^print:]`	incorrect.
`[[:lower:][:digit:]]`	correct : une minuscule (y compris accentuée)ou un chiffre.en anglais : **[a-z0-9]**

Exemple détaillé

L'exemple suivant illustre l'utilisation des classes de caractères dans le cas des caractères accentués français : rechercher dans un texte les chaînes représentant un prénom suivi d'un nom (dans la même ligne).
Un nom est une suite d'au moins deux lettres majuscules précédées d'un espace et d'un prénom. Un prénom est une suite d'au moins deux lettres, la première étant majuscule et les suivantes minuscules. Le prénom est précédé d'un espacement ou du début de la ligne. Le nom est suivi d'un espace ou d'une ponctuation ou de la fin de la ligne. Les lignes suivantes en sont des exemples :

Ce monsieur s'appelle Rayan MIRI, il est endormie au centre de l'amphi.
Meryem CHAKIR préfère le BASH : c'est bien compréhensible.
C'est un exemple qui a était bien compris par Hicham AMMARI.
Mais cette ligne ne contient PAS de nom prénom.

Exemple détaillé

L'expression régulière est donc construite de la façon suivante :
un nom : `[[:upper:]]{2,}`
un prénom : `[[:upper:]][[:lower:]]+`
un prénom suivi d'un nom (séparés par un espace) :
`[[:upper:]][[:lower:]]+ [[:upper:]]{2,}`
Il faut maintenant exprimer ce qui peut se trouver avant :
`(^|[[:space:]])`
... et ce qui peut se trouver après :
`([[:space:][:punct:]]|$)`
Et voila : on peut tout mettre ensemble en une seule ligne.
(replié en deux lignes ci-dessous pour des raisons typographiques) :
Pour tester

```
grep -E '(^|[[:space:]])[[:upper:]][[:lower:]]+
[[:upper:]]{2,}([[:space:][:punct:]]|$)' texte.txt
```

Exemple détaillé

```
smi@ubuntu:~$ cat texte.txt
Ce monsieur s'appelle Rayan MIRI, il est endormie au centre de l'amphi.
Meryem CHAKIR préfère le BASH : c'est bien compréhensible.
C'est un exemple qui a était bien compris par Hicham AMMARI.
Mais cette ligne ne contient PAS de nom prénom.

smi@ubuntu:~$ grep -E '(^|[[:space:]])[[:upper:]][[:lower:]]+ [[:upper:]]
{2,}([[:space:][:punct:]]|$)' texte.txt
Ce monsieur s'appelle Rayan MIRI, il est endormie au centre de l'amphi.
Meryem CHAKIR préfère le BASH : c'est bien compréhensible.
C'est un exemple qui a était bien compris par Hicham AMMARI.
```