

Plan

1

- Expression régulière
- grep

2

Expressions régulières

Expressions régulières

3

- Chaîne de caractères, générique, désignant un ensemble de chaînes de caractères respectant une syntaxe particulière.
- Décrivent un motif générique, avec certaines commandes, permet d'identifier les chaînes de caractères répondant à ce motif.
- Très utile dans la manipulation de texte, pour rechercher, remplacer ou supprimer.
- Plusieurs commandes avancées utilisent les expressions régulières.

Exemples: `ed`, `vi`, `sed`, `awk`, `expr` et `grep` ...

Il existe des expressions ordinaires (`grep`) et des expressions régulières étendues (`egrep`)

Expressions régulières

4

Métacaractères	Fonction
?	0 ou 1 fois l'expression précédente
*	0, 1 ou plusieurs fois l'expression précédente
+	1 ou plusieurs fois l'expression précédente
{n}	n fois l'expression précédente
{m,n}	m fois minimum et n fois maximum l'expression précédente
[^ ...] ou ^	Négation ou début de ligne
\$	Fin de ligne
[]	Un ou plusieurs des objets entre crochets doivent être présents
()	
	Opérateur « ou ».

Expressions régulières

5

- Méta-caractères désignant un seul caractère
 - . Un caractère quelconque unique, sauf le retour chariot (`\n`)
 - [. . .] Un des caractères entre crochets
 - [^ . . .] Un caractère différent de ce qui sont entre crochets
 - `\char` Pour désigner un des caractères spéciaux
 - – Désignant un intervalle
- Les méta-caractères (`/`, `*`, `$`, `.`, `[`, `]`, `{`, `}`, `!`,...etc) peuvent être l'objet d'une recherche dans une expression régulière. Pour cela il faut le faire précéder par `\`(backslash). `\B` (étendu)

Exemples avec [] et [^]

6

- [A-Z] retrouve n'importe quelle lettre en majuscule
- [A-D] intervalle de A à D (A, B, C, D) par exemple **Porte[A-D]** donne **PorteA, PorteB, PorteC, PorteD**
- [**a-dA-D**] intervalle de a à d et A à D (a, b, c, d, A, B, C, D)
par exemple **X[a-dA-D]y** donne **Xay, Xby, Xcy, Xdy, XAy, XBy, XCy, XDy**
- [aeiouAEIOU] retrouve n'importe quelle voyelle
- [^aeiouAEIOU] retrouve n'importe quel caractère sauf voyelles
- [^a-zA-Z] n'importe quel caractère sauf une lettre minuscule ou majuscule.

Exemples avec [] et [^]

7

- `[0-9]` ou `[0123456789]` retrouve n'importe quel chiffre
- `[^0-9]toto` identifie les lignes contenant une chaîne toto, le caractère juste avant ne doit pas être un 1toto, 5toto).
- **`[2-56]`** intervalle de 2 à 5 et 6 (et non pas 56) (2, 3, 4, 5, 6)
par exemple `12[2-56]2` donne `1222, 1232, 1242, 1252, 1262`
- **`[1-3-]`** intervalle de 1 à 3 et - (1, 2, 3, -)
par exemple `[1-3-]3` donne `13, 23, 33, -3`

Expressions régulières

8

- Méta-caractères désignant une position
 - `^` Le début de ligne
 - `$` La fin de ligne
 - `\<` Début d'un mot
 - `\>` Fin d'un mot
- Méta-caractères en mode étendu avec `egrep` ou avec `-E`
 - `\b` : Indique une limite de mot pour s'assurer que le mot est entier.
 - `\w+` : Représente un mot (une séquence de caractères alphanumériques).
 - `\d+` : représente un nombre (une séquence de chiffre décimaux)
 - `\1`, `\2`, ... représente une référence arrière à un groupe de capture

Expressions régulières

9

- Méta-caractères désignant Les occurrences
 - ? 0 ou 1 fois l'expression précédente (optionnel)
 - * Désigne une suite quelconque d'occurrences de (0 ou plus)
 - + au moins une occurrence de x (1 ou plus)
 - {N} Exactement n occurrences
 - {N, } au moins n occurrences
 - {min, max} Entre n et m occurrences

- | des expressions alternatives **expression1 | expression2** désigne une chaîne correspondant à l'expression 1 ou 2.

Exemples

10

- **'^a'** va identifier les lignes commençant par le caractère a.
- **'a\$'** va identifier les lignes se terminant par le caractère a.
- **^chaine\$** identifie les lignes qui contiennent strictement la chaîne chaine.
- **\$** identifie une ligne vide.
- **x*** une suite quelconque d'occurrences de x (0 ou plus)
- **x+** au moins une occurrence de x (1 ou plus)
- **x{n, m}** Entre n et m occurrences de x (au moins n, au plus m)
- **x?** Une occurrence optionnelle de x (0 ou 1)
- **x|y** X ou y

Expressions régulières : *

11

- **a*** correspond aux lignes comportant 0 ou plusieurs caractère **a**. Son utilisation est à proscrire, car toutes les lignes, même celles ne contenant pas le caractère **a**, répondent aux critères de recherche.
- **x*** est une source de problèmes, il vaut mieux éviter de l'employer.
- **aa*** correspond aux lignes comportant 1 ou plusieurs caractères **a**.
- **.*** correspond à n'importe quelle chaîne de caractères.
- **[a-z][a-z]*** va chercher les chaînes de caractères contenant 1 ou plusieurs lettres minuscules (de a à z).
- **[][]*** est équivalent à tout sauf un blanc.

Expressions régulières : priorité

12

- Dans l'évaluation d'une expression régulière l'ordre de priorité est
 - ▣ [et],
 - ▣ puis *,
 - ▣ +
 - ▣ ?
 - ▣ la concaténation
 - ▣ enfin |
- (...) les parenthèses permettent de modifier l'ordre de priorité des opérateurs. **(a|b)c : ac ou bc**

Les métacaractères \ (\)

13

□ \ (... \) et \n

▣ Pour représenter une sous chaîne, on utilise la syntaxe

**\ (expression régulière) **

▣ **\ (expression \)** qui désigne la même chose que l'expression elle-même

▣ **\n** qui désigne la nième expression entourée précédemment par \ (et \).

□ Exemples

▣ **" \ (^ . * \) \1 \$ "** : désigne une ligne formée d'une chaîne répétée deux fois.

▣ **\ ([a-z] [a-z] *) ** est une sous chaîne identifiant les lignes contenant une ou plusieurs lettres minuscules

Autres : Concaténation

14

- On peut construire des expressions régulières à partir d'autres expressions
- La concaténation d'expressions régulières désigne la concaténation des chaînes désignées par chacune des expressions.
- Exemple :
 $\wedge[0-9]\{3\}A.*T\$$: désigne une chaîne qui commence par trois chiffres suivis de la lettre "A" et qui se termine par la lettre "T" en fin de ligne.

15

La commande grep

Famille grep (**Get Regular Expression Pattern**)

16

- **grep** permet de retrouver dans un ou plusieurs fichiers l'ensemble des lignes qui répondent à certains critères de recherche

```
$grep [ option ] < pattern > <filename >
```

```
$ cmd | grep [ option ] <pattern >
```

On peut utiliser les expressions régulières pour la recherche de pattern

- **egrep** Accepte la définition complète des expressions régulières y compris

« + », « ? », « | » et « () ». la syntaxe est la même que celle de **egrep**

- **fgrep** ne permet de rechercher que des chaînes explicitement indiquées. N'admet aucune expression régulière.

La commande grep

17

- Le code de retour de la commande grep est:
 - ▣ 0 : si une ou plusieurs lignes ont été trouvées
 - ▣ 1 : si aucune ligne n'a été trouvée
 - ▣ 2 : si une erreur de syntaxe ou un fichier mentionné en argument est inaccessible

Chercher avec grep

18

- Afin de chercher une phrase, ou un mot en entier, vous devez l'entourer de guillemets simples ou doubles :
 - ▣ `grep 'texte a chercher' file.txt`
 - ▣ `grep " mot " file.txt`

- Chercher selon un pattern exprimé par une expression régulière
 - ▣ `grep 'expression_reg ' file.txt`
 - ▣ `grep '^A' file.txt`

L'option `-i`

19

- La commande `grep` est sensible à la casse.

`grep Aaa file.txt` et `grep aaa file.txt`

ne donnent pas les mêmes résultats

- Afin que la commande `grep` ignore la différence entre Majuscule et minuscule, on utilise l'option `-i`.

`grep -i aaa file.txt`

`grep -i Aaa file.txt`

Les deux expressions sont équivalentes

grep -v -l -L -H

20

- L'option `-v` est utilisée pour inverser la correspondance, affiche toutes les lignes qui ne correspondent pas au modèle spécifié. `-v` affiche les lignes ne contenant pas la chaîne

```
grep -v [0-9] file.txt
```

- L'option `-l` permet d'afficher juste les noms des fichiers dans lesquels un motif apparaît, sans afficher les occurrences trouvées.
- L'option `-L`, affiche les noms de fichiers où ne figurent pas la correspondance cherchée .
- L'option `-H` permet d'avoir le nom du fichier ou l'on cherche un motif en tête de ligne :
`grep -H 'expression' file/*.txt`

grep -c -n -m

21

- L'option **-c** compte le nombre de lignes contenant la correspondance cherhcée

```
grep -c 'expression' file.txt
```

```
grep 'expression' file.txt | wc -l
```

- L'option **-n** chaque ligne contenant la chaîne est numérotée

```
grep -n 'expression' file.txt
```

- L'option **-m NUM** permet de s'arrêter après NUM solutions trouvées.

```
grep -m 20 'expression' file.txt
```

permet de limiter notre recherche à 20 résultats.

Options grep pour les lignes de contexte

qui précèdent et/ou suivent la ligne où figure le mot)

22

□ Plusieurs possibilités

- ▣ `-num` : le numéro indique le nombre de lignes de contexte que l'on veut voir figurer avant et après la ligne où figure le mot recherché. `$grep -3 ...`
- ▣ `-A num (after)` : `grep -A 4 ...`
- ▣ `-B num (before)` : `grep -B 10 ...`
- ▣ `-C num (context)` : équivalent à `-num`.

□ En fait, les trois lignes suivantes sont strictement équivalentes :

- ▣ `grep -2 ...`
- ▣ `grep -C 2 ...`
- ▣ `grep -A 2 -B 2 ...`

Combiner les options

23

- On peut combiner plusieurs options
 - ▣ `grep -ivc 'expression' file.txt` est la même chose que
 - ▣ `grep -i -v -c 'expression' file.tx`
- Exemples
 - ▣ `grep -nC violon verlaine.tex`

La commande grep : Exemple

24

```
$cat file.txt
```

```
ligne 1: Ceci est un exemple de fichier.
```

```
ligne 2: Il contient plusieurs lignes de texte.
```

```
ligne 3: Nous allons utiliser grep pour rechercher des mots spécifiques.
```

```
ligne 4: Par exemple, nous pouvons chercher le mot "exemple".
```

```
ligne 5: Ou nous pouvons chercher le mot "grep" pour voir comment cela  
fonctionne.
```

```
Ligne contexte
```

```
ligne 6: Un autre exemple de ligne avec le mot "exemple".
```

```
ligne 7: Voici une ligne qui ne contient pas le mot que nous recherchons.
```

```
ligne 8: La commande grep est très utile pour filtrer des informations.
```

```
ligne 9: EXEMPLE en majuscule
```

```
(123) Exemple de texte avec des parenthèses.
```

```
(456) Une autre ligne avec des informations.
```

```
(789) Une troisième ligne.
```

La commande grep

25

```
$grep "exemple" file.txt
```

```
ligne 1: Ceci est un exemple de fichier.
```

```
ligne 4: Par exemple, nous pouvons chercher le mot "exemple".
```

```
ligne 6: Un autre exemple de ligne avec le mot "exemple".
```

```
$grep -c "exemple" file.txt
```

```
3
```

```
$grep -i "exemple" file.txt
```

```
ligne 1: Ceci est un exemple de fichier.
```

```
ligne 4: Par exemple, nous pouvons chercher le mot "exemple".
```

```
ligne 6: Un autre exemple de ligne avec le mot "exemple".
```

```
ligne 9: EXEMPLE en majuscule
```

La commande grep : Exemple

26

Ouioui

NonNonNon

MiaouMiaouMiaouMiaou

Aligne 1: Ceci est un exemple de fichier.

aligne 2: Il contient plusieurs lignes de texte.

aaligne 3: Nous allons utiliser grep pour rechercher des mots spécifiques.

ligne 4: Par exemple, nous pouvons chercher le mot "exemple".

ligne contexte

ligne 5: Ou nous pouvons chercher le mot "grep" pour voir comment cela fonctionne.

ligne 6: Un autre exemple de ligne avec le mot "exemple".

ligne 7: Voici une ligne qui ne contient pas le mot que nous recherchons.

ligne 8: La commande grep est très utile pour filtrer des informations

ligne 9: EXEMPLE en Majuscule

ligneTestligneTest

Exemples avec les options de grep

27

```
$grep -n "^a" testgrep
```

```
5:aligne 2: Il contient plusieurs lignes de texte.
```

```
6:aaligne 3: Nous allons utiliser grep pour rechercher des mots spécifiques.
```

```
$grep -n -i "^a" testgrep
```

```
4:Aligne 1: Ceci est un exemple de fichier.
```

```
5:aligne 2: Il contient plusieurs lignes de texte.
```

```
6:aaligne 3: Nous allons utiliser grep pour rechercher des mots spécifiques.
```

```
$grep -m 4 'ligne' testgrep
```

```
Aligne 1: Ceci est un exemple de fichier.
```

```
aligne 2: Il contient plusieurs lignes de texte.
```

```
aaligne 3: Nous allons utiliser grep pour rechercher des mots spécifiques.
```

```
ligne 4: Par exemple, nous pouvons chercher le mot "exemple".
```

Exemples avec les options de grep

28

```
grep -2 "contexte" testgrep
```

ligne 3: Nous allons utiliser grep pour rechercher des mots spécifiques.

ligne 4: Par exemple, nous pouvons chercher le mot "exemple".

ligne **contexte**

ligne 5: Ou nous pouvons chercher le mot "grep" pour voir comment cela fonctionne.

ligne 6: Un autre exemple de ligne avec le mot "exemple".

Recherche de fichiers

29

- Pour chercher des fichiers dont le nom satisfait à une expression régulière, il suffit de combiner les commandes `grep` et `ls` à l'aide d'un pipeline (« | »).
- Par exemple, pour énumérer les fichiers du répertoire courant dont le nom est `abc`, `abcbc`, `abcbcbc`, etc., il suffit d'utiliser la ligne suivante :

```
ls -R | grep -E "a(bc)+"
```

Chercher un mot -w

30

pour que grep ne recherche que les lignes où figure le mot tel quel, et non pas ses variantes ?

- C'est l'option -w (comme word) qui sert à cela : un mot complet est délimité comme suit :
 - Début : la chaîne de caractères est placée au début d'une ligne, ou précédée d'un blanc, d'une tabulation ou d'une ponctuation.
 - Fin : la chaîne de caractère est placée en fin de ligne, ou suivie d'un blanc, d'une tabulation ou d'une ponctuation.
- Si donc on veut chercher «travail» et aucune forme dérivée de ce mot, on écrit :
grep -w travail mon-fichier

Autres

31

- Les listes par exemple `[:alpha:]` ou `[:space:]`

```
grep `^[[:alpha:]]*$` file
```

- `grep `\<1` file`

Examples

32

- $abc?$: ab ou abc.
- abc^* : ab, abc, abcc, ...
- abc^+ : abc, abcc, abccc, ...
- $abc\{2\}$: abcc.
- $abc\{2, \}$: abcc, abccc, ...
- $abc\{2, 4\}$: abcc, abccc, abccccc.

Exemples

33

- **^début** : chaîne commençant par « début ».
- **fin\$** : chaîne se terminant par « fin ».
- **^abc+** : abc, abcc, abccc, ...
- **^chaîne\$** : chaîne qui commence et se termine par « chaîne ».

Exemples

34

- $[abc]$: a, b ou c; équivalent à $(a \mid b \mid c)$.
- $[a-z]$: une des lettres de l'alphabet.
- $[a-zA-Z]$: union de $[a-z]$ et $[A-Z]$.
- $a(bc)^*$: a, abc, abcbcb, ...
- $un \mid le$: « un » ou « le ».
- $(un \mid le) \text{ chien}$: « un chien » ou « le chien ».
- $(a \mid b)^*$: chaîne contenant une suite de a ou de b.
- $^{\wedge}\{3\}\$$: chaîne de 3 caractères exactement.

Expressions régulières: exemples

35

- ❑ Lignes commençant par un nombre
- ❑ FreeBSD, OpenBSD ou NetBSD
- ❑ Chaîne commence "Bonjour" puis se termine par "au revoir."
- ❑ Une chaîne de caractères de lettres (même vide)