Filtres (suite)

1 - Sélection de contenu

Il est souvent utile de pouvoir extraire d'un fichier, les lignes qui contiennent une chaîne de caractères donnée.

Il existe une commande très puissante, et donc très utile, qui permet de faire ce type de recherche et d'extraction. Ses possibilités sont telles qu'elle mérite un chapitre à elle seule !

grep et egrep

On a introduit les filtres comme étant des utilitaires basiques, rudimentaires.

La fonction de grep est basique. Même si ses capacités d'extraction sont très évoluées, sa fonction reste simple : extraire les lignes contenant une chaîne donnée.

egrep = extended grep. Nous utiliserons seulement egrep

Des capacités d'extraction évoluées, par la manière dont on peut décrire la chaîne recherchée :

- Texte simple
- Texte défini, décrit, par un motif, une "formule" appelée une expression rationnelle ou expression régulière (regex ou regexp).

grep signifie : globally search regular expression and print. Recherche globalement une expression régulière et affiche (le résultat). Les regex sont ici un élément clé.

Texte simple

Syntaxe : egrep [option] <texte>

La recherche se fait par ligne.

Toute ligne contenant <texte>, n'importe où, provoque l'affichage de la ligne complète.

Exemple - Fichier etu_2018 :

```
ALAIN:VALERIE:H2
PETIT:LUDOVIC:G1
MOREAU:FLAVIE:G2
DUBOIS:ALAIN:F2
```

La commande `egrep ALAIN < etu_2018` affiche :

```
ALAIN:VALERIE:H2
DUBOIS:ALAIN:F2
```

Comment n'extraire que la 2ème ligne ?

~~ALAIN:VALERIE:H2~~
DUBOIS:ALAIN:F2

Comment n'extraire que la 2ème ligne ?

~~ALAIN:VALERIE:H2~~
DUBOIS:ALAIN:F2

grep :ALAIN < etu_2018

Comment n'extraire que la 1ère ligne ?

ALAIN:VALERIE:H2
~~DUBOIS:ALAIN:F2~~

Comment n'extraire que la 1ère ligne ?

ALAIN:VALERIE:H2
~~DUBOIS:ALAIN:F2~~

Aucun moyen avec une recherche d'un texte simple. Il faut un moyen plus puissant de décrire ce qu'on cherche.

Expression régulière

Syntaxe : egrep [option] <regex>

La recherche se fait toujours par ligne.

Toute ligne concordant avec <regex> provoque l'affichage de la ligne complète.

Que signifie “concordant” (matching en anglais) ?

Ce terme signifie qu’on ne cherche pas une égalité au sens strict du terme.

On cherche la présence d’une chaîne décrite, pas uniquement par son contenu, mais aussi par des règles de constitution de la chaîne, ou de positionnement.

Par exemple, on peut chercher :

- un texte de 3 caractères quelconques suivis du symbole “:”
- un texte précis, mais uniquement situé en début de ligne.
- un texte composé uniquement de lettres minuscules.
- un texte composé de 5 groupes de 2 chiffres, séparés par des “:” ou des “-” ou des espaces. (C’est à dire : un numéro de téléphone)

Options

Syntaxe : egrep [option] <texte> :

- -v : négation. On cherche l'absence de la chaîne.
- -i : insensible à la casse (min et MAJ sont considérées identiques).
- -n : affiche les numéros des lignes où apparaissent le texte.
- -l : affiche seulement les noms des fichiers qui contiennent le texte sur une des lignes.

2 - Expressions régulières

Motif

Le motif (pattern en anglais) décrit les règles à respecter pour qu'il y ait concordance dans la recherche, pour qu'une ligne "match" avec ce qu'on cherche.

On utilise aussi des jokers, comme avec le Shell.

Attention, même s'ils se ressemblent, ils ne fonctionnent pas de la même manière, ils ne faut pas les confondre !

Principes de base :

- Tous les caractères du texte source sont significatifs, y compris les espaces, les tabulations, et même les fins de lignes.
- La concordance est recherchée sur chaque ligne de la source.
- La concordance est recherchée n'importe où sur une ligne, sauf si le motif décrit qu'il s'agit d'une ligne complète.

Liste - [ ]

Un caractère parmi une liste de caractères possibles.
C'est similaire au joker [ ] du Bash.
Exemple :

grep [aeiouy] < fic

Cherche les lignes de fic où apparaît au moins une voyelle, n'importe où sur la ligne.

Autre exemple :

grep [0123456789]TTC < tarifs

Cherche les lignes de tarifs où apparaît au moins un chiffre avec le mot TTC collé derrière, et n'importe où sur la ligne.

Intervalle - [ ]

Un caractère parmi un intervalle de caractères possibles.
C'est similaire au joker intervalle [ ] du Bash.
Exemple :

grep [A-Z] < fic

Cherche les lignes de fic où apparaît au moins une lettre MAJUSCULE, n'importe où sur la ligne.

Les intervalles peuvent aussi être combinés. Exemple :

grep [A-Za-z0-9] < liste

Cherche les lignes de liste où apparaît au moins une lettre MAJUSCULE ou minuscule ou un chiffre, n'importe où sur la ligne.

Quantificateurs

Un quantificateur indique que l'élément précédent peut/doit être trouvé ou répété un certain nombre de fois, en fonction de la nature du quantificateur :

- ? : 0 ou 1 fois.
- + : 1 fois au moins.
- * : 0 à N fois.
- {N} : exactement N fois.
- {N, M} : entre N et M fois.

Exemples :

- jours? : match “jour” ou “jours”.
- [A-Z]+ : match 1 ou plusieurs MAJUSCULES.
  Important : il n’est pas nécessaire que ce soit plusieurs fois la même lettre ! C’est une MAJUSCULE suivie éventuellement d’une autre MAJUSCULE, etc.
- [0-9]* : Rien, un ou plusieurs chiffres successifs.

Position “début”

Le symbole ^ permet de préciser que le motif doit être trouvé au début de la ligne.

Exemple :

^[0-9] signifie : début de ligne, suivi d’un chiffre. Ensuite, il peut y avoir des choses derrière.

Position “fin”

Le symbole $ permet de préciser que le motif doit être trouvé en fin de la ligne. Exemple :

[0-9]$ signifie : un chiffre, suivi de la fin de ligne. Donc, tout simplement “un chiffre en bout de ligne”. Il peut y avoir des choses devant le chiffre.

Evidemment il est possible d'avoir ces 2 symboles en même temps. Exemple :

^[0-9]{3}$ signifie : début de ligne, suivi de 3 chiffres, suivi de la fin de ligne. Donc, des lignes contenant uniquement 3 chiffres consécutifs et rien d'autre, ni devant, ni derrière, même pas des espaces.

Symboles spéciaux

Il y a quelques symboles qui ont une signification spéciale :

- . : n'importe quel caractère.
- \ : pour retirer au caractère suivant sa signification particulière dans le motif. Exemple :
  * pour représenter l'étoile, \? pour le "?", \\ pour le "\", \$ pour le dollar, etc.

Cas du ^

Le “^” signifie “début de ligne”.
Cependant, il a une autre signification dans une liste ou un intervalle [ ].
Il permet de dire “tout sauf”. Exemple :

- [a-z] : une lettre minuscule.
- [^a-z] : tout sauf une lettre minuscule.

Groupage

Il est possible de définir une partie d'un motif comme un groupe, un ensemble à considérer comme unitaire.

Un groupe est encadré par des ( ). Il est alors possible de quantifier aussi un groupe. Exemple :

([0-9][a-z]){2} : Un chiffre suivi d'une minuscule, répété 2 fois (pas forcément les mêmes chiffres et lettres).

Avec le groupage on peut aussi réutiliser le contenu d'un groupe à un autre endroit dans le motif. Exemple :

`([0-9]{2})_\1` :

- Un chiffre, 2 fois de suite (pas forcément le même chiffre).
- Suivi d'un "_".
- Suivi des mêmes chiffres que ce qui a été trouvé dans le (groupe).

Donc ça match "12_12" mais pas "12_34".

Alternative

Une alternative, représentée par le symbole “|” (pipe), permet d’indiquer 2 solutions possibles dans le motif, soit la partie à gauche du “|”, soit la partie à droite.
Exemples :

0|1 : Un 0 ou un 1.
([a-z]{5})|([A-Z]{2}) : 5 minuscules ou 2 MAJUSCULES.

Retour au problème :
Comment n'extraire que la 1ère ligne ?

ALAIN:VALERIE:H2
~~DUBOIS:ALAIN:F2~~

Retour au problème :
Comment n'extraire que la 1ère ligne ?

ALAIN:VALERIE:H2
~~DUBOIS:ALAIN:F2~~

grep ^ALAIN: < etu_2018

3 - Trouver des objets

Les sources, les objets, les fichiers à traiter ne sont pas toujours placés dans le répertoire courant, ni forcément dans un même répertoire.

Parfois les sources potentielles sont très nombreuses et il est difficile, voire impossible de les lister manuellement.

Il faut pouvoir construire cette liste dynamiquement, en fonction de certains critères.

Find

La commande find permet de localiser des objets (fichiers ou dossiers) dans l'arborescence, en fonction des critères définis en paramètres.

Syntaxe : find <départ> <critères> <action>

Exemple #1 : find . -type f -print

- Départ : ".", le répertoire courant.
- Critère : type f, on ne cherche que des fichiers (files).
- Action : print, on affiche le chemin+nom des fichiers trouvés.

Exemple #2 : find / -type d -print 2>/dev/null

- Départ : “/”, la racine.
- Critère : type d, on ne cherche que des dossiers (directories).
- Action : print, on affiche le chemin+nom des fichiers trouvés.
- 2>/dev/null : on envoie les erreurs (canal 2, STDERR) vers /dev/null (le trou noir Unix).

Exemple #3 : find src -type f -mtime -24 -exec ls -l {} \;

- Départ : “src”, le dossier ./src.
- Critère 1 : type f, on ne cherche que des fichiers.
- Critère 2 : mtime, fichiers modifiés il y a - de 24H.
- Action : exec, on execute la commande “ls -l” sur chaque fichier trouvé. Le {} sera remplacé par le nom de chaque fichier. Le \; est nécessaire pour que la commande ls -l puisse s’exécute. C’est bizarre, mais c’est la syntaxe de find.

Exemple #4 : find ~ -type f -size +10M -print

- Départ : “~”, le home.
- Critère 1 : type f, on ne cherche que des fichiers.
- Critère 2 : size, fichiers dont la taille (size) est supérieure à 10 Mo.
- Action : print, on affiche.

Pratique pour localiser ce qui prend de la place. Ca pourrait bien vous être utile ! :-)

4 - Jouons un peu

Que peut matcher cette expression ?

[0-9]{2}/[0-9]{2}/[0-9]{4}

Que peut matcher cette expression ?

[0-9]{2}/[0-9]{2}/[0-9]{4}

12/02/1967 : Une date.

Que peut matcher cette expression ?

```
([A-Za-z0-9_\.-]+)@([A-Za-z0-9\.-]+)\.([A-Za-z\.]{2,6})
```

Que peut matcher cette expression ?

([A-Za-z0-9_\.-]+)@([A-Za-z0-9\.-]+)\.([A-Za-z\.]{2,6})

jean-charles.dupont@univ-rennes1.fr : Une adresse mail.

https://commons.wikimedia.org/wiki/File:Thats_all_folks.svg