Commande `grep`

Les expressions régulières sont utilisées fréquemment par les développeurs et les ingénieurs systèmes pour explorer des fichiers de logs (à la main ou par la biais de scripts).

Sous Unix, la commande `grep` permet d'identifier les lignes d'un fichier qui nous intéressent sur la base d'un motif qu'elles contiennent. Ce motif est décrit par une expression régulière.

Vous pouvez trouver les principales syntaxes des expressions régulières manipulées par Grep à l'adresse suivante : `http://www.linux-france.org/article/web/egraffin/regexp.php`

1 Trouver ses mots

Le fichier `mots.txt` contient un ensemble de mots (un par ligne) que nous allons essayer de retrouver par des expressions régulières données à la commande `grep`. Nous utiliserons l'option `-E` qui permet d'utiliser les parenthèses et les accolades sans avoir à les déspécialiser, par exemple `grep -E "(abc)+" mots.txt` demande au système d'indiquer les mots du fichier qui sont composés d'une ou plusieurs occurences du facteur `abc`

1. Quelle expression régulière permet d'obtenir les mots contenant le facteur `ab` ? Essayer dans le terminal.

   grep -E "ab" mots.txt

2. Combien de mots du fichier correspondent à cette description ? Comment faire pour que la commande `grep` les compte pour vous ?

   10

3. Comment trouver les mots ayant `ab` pour préfixe ? et ceux l'ayant pour suffixe ?

   grep -E "^ab" mots.txt grep -E "ab$" mots.txt

4. Comment trouver les mots `abba` ?

   grep -E "^abba$" mots.txt

5. comment trouver les mots correspondant à l'expression régulière $(a|b)c^+(a|b)$?

   grep -E "(alb)c+(alb)" mots.txt

2 Entre parenthèses

Vous trouverez dans le manuel unix (commande `man`), sinon dans vos cours Systèmes/Unix, ou bien encore dans les tutoriels sur internet une description de la syntaxe des expressions régulières de la commande `grep`.

1. Ecrivez une commande permettant de lister les accolades ouvrantes dans un fichier de code `.c` et testez-le sur les trois programmes C.

   grep -E "{" prog2.c

2. Modifiez la commande pour qu'elle compte maintenant le nb d'accolades ouvrantes.

   grep -E "{" prog2.c -c

3. Même chose pour le nombre d'accolades fermantes. Pouvez-vous déduire si la compilation de l'un des trois programmes C va échouer ? lequel et pourquoi ?

   grep -E "}" prog2.c

   prog1.c va échouer car pas le même nombre d'ouvrantes que fermantes.

4. Est-ce que vous pouvez trouver une situation ou cette stratégie ne suffira pas pour détecter des *nombres* différents d'accolades ouvrantes et fermantes ?

   Fonctionne pas si plusieurs accolades sur la même ligne.

5. Comment pouvez-vous faire afficher dans la même commande toutes les accolades (ouvrantes et fermantes) d'un programme C ?

   grep -E "{|}" prog3.c

6. En regardant le résultat de cette commande sur chacun des trois programmes séparément, pouvez-vous détecter un soucis à venir lors de la compilation de l'un d'entre eux ?

   On voit qu'il manque une fermante dans prog1.c

7. Si l'on s'attaque maintenant aux parenthèses, indiquez pourquoi la commande suivante ne marche pas : `grep ( prog1.c`

   Car il y a un espace et que le regex n'est pas entre quotes. Aussi, il faut échaper la parenthèse car c'est un élément connu par les regex.

8. Corrigez cette commande pour compter le nb de parenthèses ouvrantes dans les trois programmes C. Vérifiez si votre solution passe dans la commande `grep -E`

```
grep -E "\(" prog3.c -c
```

3 Variables

La déclaration d'une variable est en général un type suivi d'un identificateur (une lettre suivi de caractères alphanumériques) suivi d'un point virgule. Bien-sûr des espaces (`[[:space:]]`) peuvent venir se glisser entre ces différentes parties.

1. Trouver l'expression régulière qui permet d'obtenir le noms des variables déclarées de type `int` dans le programme `prog2.c` (attention, il y en a 3, soyez sûrs que votre expression permet de les obtenir toutes).

```
grep -E "int\ +([[:alnum:]]|_)*\ *;" prog2.c
```

2. Comment maintenant avoir les variables qui sont des pointeurs sur des entiers (par ex : `int *truc`) ?

```
grep -E "int\ +\*\ *([[:alnum:]]|_)*\ *;" prog2.c
```

3. Comment avoir par une même commande les variables de type `int` ou `int *` à la fois ?

```
grep -E "int\ +\*?\ *([[:alnum:]]|_)*\ *;" prog2.c
```

4. comment compléter la commande pour avoir aussi les variables de type `float` ?

```
grep -E "(int|float)\ +\*?\ *([[:alnum:]]|_)*\ *;" prog2.c
```