Commandes de filtres : grep, cut, tr, sed

Généralités

- Un filtre est une commande qui lit les données sur l'entrée standard, effectue des traitements sur les lignes reçues et écrit le résultat sur la sortie standard.
- Bien sûr les entrées/sorties peuvent être redirigées, et enchainées avec des tubes.
  A noter que le caractère d'indirection < en entrée n'est pas obligatoire pour les filtres
  Ainsi, dans `# cat /etc/*.conf > tous.conf` cat va bien lire les fichiers qui correspondent au modèle `/etc/*.conf` et les contaténer dans le fichier `tous.conf`
- Dans ce chapitre, on va revoir ou découvrir les principaux filtres utilisés dans le monde UNIX
  - cat, more et less
  - grep
  - cut
  - wc
  - tr
  - sed
  - awk

Les commandes cat, more et less

```
cat f1 f2 .. > f concaténe f1, f2 .. dans le nouveau fichier f
less f1 f2 .. > f  concaténe les fichiers f1 f2 .. en un seul fichier f (comme cat)
less f3 >> f    ajoute le contenu du fichier f3 à la suite du contenu du fichier f
```

La commande grep : sélection de lignes

Cet utilitaire (*General Regular Expression Parser*, analyseur général d'expression régulière) sélectionne toutes les lignes qui satisfont une expression régulière (ou rationnelle).

Syntaxe

`grep [options] expreg [fichiers]`
Cette commande recherche dans les fichiers ou sur son entrée standard des lignes de texte qui satisfont l'expression régulière `expreg` indiquée.
Sa sortie peut être redirigée dans un fichier.

options
-c donne seulement le nombre de lignes trouvées obéissant au critère
-l donne seulement le nom des fichiers où le critère a été trouvé
-v donne les lignes où le critère **n'a pas** été trouvé
-i ne pas tenir compte de la casse (ne pas différencier majuscules minuscules)
-n pour n'afficher que les numéros des lignes trouvées
-w pour imposer que le motif corresponde à un mot entier d'une ligne

constructions
grep est souvent inclus dans un tube qui lui fournit en entrée le fichier à étudier.

```
Exemple : quelle est la question posée ?
cat /etc/passwd | cut -d: -f1 | grep -w "jean" > sortie
```

Expressions reconnues

Grep ne reconnait pas toutes les expressions rationnelles étendues.
Voici la liste des symboles utilisables par `grep` : . * [ ] [^] ^ $

- `.` signifie un caractère quelconque
- `*` répétition du caractère situé devant
- `^` début de ligne
- `$` fin d'une ligne (donc "e$" mots se terminant par e)
- `[...]` contient une liste ou un intervalle de caractères cherchés
- `[^..]` caractères interdits.

Attention
Pour éviter une confusion entre les interprétations de ces symboles spéciaux par grep ou par le shell, il est indispensable de "verrouiller" `expreg` en plaçant l'expression entre guillemets " " (et non entre quotes !).

Exemples

Etudier et commenter les commandes suivantes :

1. *cherche dans fichier, les lignes dont la 1ère lettre est qcq et la 2ème doit être o*
   grep "^.o" fichier
2. *cherche dans le fichier passwd les lignes commençant par t*
   grep "^t" /etc/passwd
3. *cherche les lignes ne commençant pas commençant par t*
   grep -w "^t" /etc/passwd
4. *cherche les lignes contenant les mots suivant le modèle T.t.*
   grep "T.t." /etc/passwd
5. *cherche dans le fichier des groupes, ceux qui commencent par a ou b .. ou j*
   less /etc/group | grep "^[a-j]"
6. *pour lister les s-répertoires du rép. /etc*
   ll /etc | grep "^d"
7. *compter les lignes saisies au clavier qui se termine par a*
   grep -c "a$"
8. *afficher les lignes des fichiers essai?.txt qui contiennent a, b ou c*
   grep [abc] "essai?.txt"
9. *détourne le flot de sortie du moniteur pour l'envoyer sur l'entrée de wc pour ?*
   grep [abc] "essai?.txt" | wc -l

Exercices

1. donner une version sans `cut` de la commande précédente
2. *Comment ne sélectionner que "root" avec*

```
cat /etc/passwd | cut -d : -f 1 | grep  "r"
```

3. on sait que `ps aux` donne la liste des processus. La commande `/etc/X11/X` est celle qui lance le serveur X. Il est nécessaire de connaitre son **`PID`**, en cas de plantage du serveur X
   Ecrire la commande qui retourne la ligne correspondante.
4. Se placer dans `/etc`. En une seule commande, faire calculer et afficher le nombre de sous-répertoires de /etc, sous la forme :
   "Il y a 33 répertoires dans /etc"
5. Créer un fichier `essai1` contenant quelques lignes dont des lignes vides Avec **`grep`** générer le

fichier essai2 à partir de essai1 sans ligne vide

```
cat essai1 .....  >essai2
```

Réponses

---

cut : sélection de colonnes

La commande **cut** présente 2 formes suivant que l'on sélectionne des colonnes de caractères ou qu'on distingue des champs séparés par un caractère précis.

sélection_colonne
cut -c(sélection_colonnes) [fichiers]
Exemples

- *affiche le 5ième caractère*
  cut -c5 fichier
- *affiche du 5ième **au** 10ème caractères*
  cut -c5-10 fichier
- *affiche le 5ième **et** le 10ème caractères*
  cut -c5-10 fichier
- *affiche à partir du 5ième (jusqu'à la fin)*
  cut -c5- fichier

sélection champs
cut -d(séparateur) -f(sélection_champs) [fichiers]

 **Exercices**

Etudier les commandes suivantes

1. `cut -d":" -f1,6 /etc/group`
2. *Que réalise la ligne suivante ? Vérifiez*
   `grep "^st" /etc/passwd | cut -d":" -f1,3-4,6`

La commande wc

Exemples

```
cat /etc/paswwd | grep /bin/bash/ | wc -l
    pour compter les titulaires d'un compte pouvant se connecter avec le login shell
less /etc/paswwd | grep -vc /bin/bash/
    négation de la question précédente (revoir les rôles ds options -c et -v)
```

La commande tr

tr=Translate, est un filtre ne reconnaissant pas les expr. régulières.
Cette commande est le plus souvent associée à des redirections
Les caractères entrés sont traités et le résultat est envoyé sur la sortie standard
On peut utiliser les intervalles du type `a-z` et les codes ASCII des caractères en notation octale `\0xx`

Syntaxe

1. **`tr [options] ch1 ch2 <fich1 >fich2`**
   Remplace toutes les occurrences de TOUS les caractères de ch1 par le caractère de ch2, de même rang, dans le flot d'entrée.

2. Exemple

   ```
   # pour convertir et afficher la ligne saisie au clavier en minuscules
   read ligne; echo $ligne | tr 'A-Z' 'a-z'
   ```

3. `tr -c chaine car` remplace tout caractère NON INCLUS dans la chaine `chaine` par le caractère `car`

   ```
   # remplace supprime tous les caractères différents de a,b, ..z par un espace
   echo $ligne | tr -c a-z ' '
   ```

4. `tr -d chaine` supprime tout caractère entré, appartenant à la chaine `chaine`

   ```
   # supprime toutes les minuscules non accentuées
   echo $ligne | tr -d a-z
   ```

5. `tr -s chaine` supprime toute répétition des caractères contenus dans `chaine`

   ```
   # supprime les espaces multiples entre les mots
   echo $ligne | tr -s ' '
   ```

 **Exercices**

Expliquer les effets de ces commandes :

1. tr 'a,/' 'A;_' <fich1 >fich2
2. tr 'a-z' 'A-Z' <fich1 >fich2
3. tr -d '\011\015\032' <fich1 >fich2
4. tr -s ' \011\012' <fich1 >fich2
5. tr -cs 'a-zA-Z0-9' '\n' <fich1 | sort | uniq >fich2

Réponses

L'utilitaire sed

Il s'agit d'un utilitaire (*sed = "Stream EDitor"*) qui sélectionne les lignes d'un fichier texte (ou d'un flot provenant d'un pipe) vérifiant une expression régulière et qui leur applique un traitement ou un remplacement.

Syntaxe

```
sed [-n] [-e script] [-f fichier-commandes] fichier-source
```

- L'option -n empêche la sortie à l'écran du résultat (souvent associé à l'option p)
- Le fichier source est traité ligne par ligne conformément à la liste des commandes (-e) ou au fichier de commandes (-f)

Commande de substitution

- La commande s permet d'effectuer des substitutions suivant la syntaxe :
  `sed [adresse]s/expr-régulière/remplacement/options`

- Attention ! contrairement à ce que l'on pourrait attendre, cette commande laisse passer toutes les lignes et ne sélectionne pas celles qui ont satisfait l'expression régulière et donc subi la substitution. Pour sélectionner, voir la commande de destruction.

- Options
  - Sans précision, la commande ne s'applique qu'à la 1ère occurence de chaque ligne.
  - 0...9 : indique que la substitution ne s'applique qu'à la nième occurence
  - g : effectue les modifications sur toutes les occurences trouvées.

- Exemple : `sed s/moi/toi/g fich.moi > fich.toi`
  le fichier `fich.moi` est parcouru, à chaque occurrence de "moi", ce mot est remplcé par "toi" et le nouveau fichier est sauvegardé sous le nom `fich.toi`

- Délimitateur
  Le slash `/` étant très utilisé au niveau du shell comme séparateur de niveau de répertoire, il est possible d'utiliser à la place tout autre caractère comme `#`
  `sed s#/home#/rep_perso#g /etc/passwd > /tmp/passwd.new`

Destruction ou sélection

- Cette option permet de filtrer les lignes qui satisfont une expression régulière. Ces lignes ne sont pas détruites dans le fichier d'origine, mais ne sont pas transmise en sortie.
- Comment modifier alors le fichier à traiter ?

```
cp fichier copie
sed /.../d copie
```

- Par exemple, pour détruire toutes les lignes vide d'un fichier :

```
sed /^$/d
```

Ajout, insertion et modification

Pour utiliser ces commandes, il est nécessaire de les saisir sur plusieurs lignes

```
sed [adresse] commande\
expression
```

La commande peut être :

```
a pour ajout ;
     i pour insertion ;
     c pour modification.
```

L'utilitaire awk

Son nom vient de ses 3 auteurs Aho, Weinberger et Kernighan C'est l'implémentation GNU du langage awk, petit langage interprété par le programme awk.

Voir cette page

Réponses

grep

1. less /etc/group | grep "^[a-j]"
2. ps aux | grep "/etc/X11/X"
3. cat essai1 | grep -v "^$" >essai2

tr

1. remplace resp. a , et / par A ; et _
2. met en majuscules
3. suppression des caractères ASCII tab (9), retour-chariot (13) et Ctrl-Z (26)
4. suppression des répétitions des espaces, des tab et des passages à la ligne (10)
5. découpage en mots de la chaine entrée, trie des lignes et suppression des doublons