Travaux pratiques: les filtres programmables

1. Exemples simples d'utilisation de `sed`

En utilisant le fichier `/etc/passwd` par exemple:

1. supprimer un # en début de ligne,
2. ajouter un # en début de ligne,
3. ajouter un ! en fin de ligne,
4. remplacer un # en début de ligne par `<--`,
5. ajouter `-->` en fin de ligne.

3. Utilisation de `awk`

3.1 Exploitation de fichiers de log

Écrire un script shell exploitant un fichier de log d'un serveur web. Utilisez le fichier `access_log.1`.

- Affichez les différentes adresses IP des machines s'étant connectées au serveur Web.
- Comptez le nombre d'adresses IP différentes de machines s'étant connectées au serveur Web.
- Quelle est la page la plus fréquemment lue?

- Comptez le nombre d'accès de pages moyen par utilisateur.
- Comptez le nombre moyen d'utilisateurs accédant à une page.

3.2 Génération de page web

Ecrire un script shell générant une page comportant un lien vers chaque document dont l'URL figure dans le fichier `liste_images.txt`.

3.3 Statistiques sur le fichier `auto`

Le fichier `auto` contient une liste de voitures avec pour chacune le constructeur, le modèle, l'année, le kilométrage, le prix en francs et la couleur. Le séparateur de champs est le caractère «:». Ecrire un fichier de commande `awk` pour afficher:

1. les caractéristiques des voitures de moins de 30 000 francs,
2. les caractéristiques des voitures de marque Opel ayant moins de 45000 km,
3. le prix moyen des voitures,

4. l'année, le modèle et le prix des voitures ayant entre 30 000 et 50 000 km et le nombre de voitures sélectionnées,
5. le prix moyen des voitures d'après 1985,
6. le prix moyen des voitures par année,
7. la voiture de couleur bleue la moins chère,
8. le nombre de voitures de chaque marque,
9. la liste des modèles de marque Opel triée par ordre alphabétique.

```
awk -F: 'BEGIN {print "Voitures de moins de 30 000F :"} ; $5 < 30000 {
print }' auto
awk -F: 'BEGIN {print "Voitures Opel de moins de 45 000 km :"} ; $1 ==
"Opel" && $4 < 45000 { print }' auto
  awk -F: 'BEGIN {print "Prix moyen :"} ; {S+=$5} ; END {print S/NR}' auto
awk -F: 'BEGIN {print "Voitures entre 30 000 et 50 000 km :"} ; $4>30000 &&
$4 < 50000 { print $2,$3,$5; nb++} ; END {print "Nombre de voitures
selectionnees = ",nb}' auto
 awk -F: 'BEGIN {print "Prix moyen des voitures post 1985 :"} ;  $3>85
{prix+=$5;nb++} ; END {print prix/nb}' auto
 awk -F: 'BEGIN {print "Prix moyen par annee :"} ;
{prix[$3]+=$5;nb[$3]++;an[$3]=$3} ; END {print "Annee\t Prix"; for (j in
prix) {print an[j],"\t",prix[j]/nb[j]|"sort -n"}}' auto
awk -F: 'BEGIN {print "Voiture de couleur bleue la moins chere :";
prix=1000000} ; $6=="bleu" {if (prix >= $5) {prix=$5;voit=$0}} ; END {print
"Prix : ",prix; print "Caracteristiques : ", voit}' auto
awk -F: '{nb[$1]++} ; END {print "Nombre de voitures de chaque marque
:";for (v in nb) print v, ":", nb[v]}' auto
awk -F: '$1=="Opel" {liste[$2]=$2} ; END {print "Modeles Opel :"; for (i in
liste) {print liste[i]|"sort"}}' auto
```

3.4 Inversion des champs

Ecrire un script shell qui inverse l'ordre des champs de chaque ligne d'un fichier.

```
f_inverse.awk

# FS = separateur de champs en entree, ORS = separateur de champs en sortie
BEGIN   {FS=":";ORS=":"}
        {for (i=NF;i>0;--i) print $i;printf "\n"}    # NF = nombre de
champs
```

3.5 Références croisées

Ecrire un script shell qui pour chaque mot d'un fichier affiche les numéros des lignes contenant les occurrences de ce mot.

```
# NF = nombre de champs, NR = numero de ligne
#
# En awk, on n'utilise pas de $ devant les noms des variables comme en
shell
#
# On construit un tableau associatif mot en utilisant chaque valeur de
champ
# rencontree comme clef, et en lui associant dans le tableau la valeur
obtenue
# en concatenant (operateur "espace") au fur et a mesure les numeros de
ligne
```

```
# separes par des virgules ",".

        {for (i=1;i<=NF;i++) mot[$i]=mot[$i] "," NR}

        # boucle sur toutes les clefs du tableau : affichage clef, valeur
END     {for (x in mot) print x,mot[x]}
```

4. Utilisation de sed

4.1 Création de liste à partir d'adresses e-mail

Écrire un script shell utilisant `sed` qui, à partir d'une liste obtenue à partir de la réponse de sympa, à une requête du type «rev liste» produit un fichier comprenant pour chaque entrée le nom suivi du prénom (séparés par un espace). On utilisera ici le fichier liste sympa.txt.

```
#!/bin/bash
#
# Script permettant de convertir un fichier contenant une liste d'e-mails
# obtenue a partir de la reponse de sympa a une requete du type
# en un fichier comprend sur chaque ligne un nom et un prenom separes par
des
# espaces.
#
# Auteur: Daniel Millot

if [ $# -ne 1 ]
  then
  echo "ERREUR: Usage : $(basename $0) fichier"
  exit
fi

if [ ! -r "$1" ]
  then
  echo "ERREUR: Fichier $1 inaccessible "
  exit
fi

sed 's/@.*//' "$1" | sed "s/^\([^.]*\)\.\(.*\)/\2 \1/"

# sed 's/@.*//'  supprime tout ce qui suit @ : n'importe quel caractere (.)
en
# nombre quelconque (*). On recupere donc Prenom.Nom sur chaque ligne

# la suite isole les chaines avant et apres le . en les designant par \1 et
\2.
# Il n'y a plus qu'a remplacer Prenom.Nom par 'Nom Prenom', i.e. \2 \1  !!!

#             sed "s/^\([^.]*\)\.\(.*\)/\2 \1/"
# s = substituer

# ^ = debut de ligne
# \([^.]*\) = 1ere sous-expression correspondant a Prenom, avec
# \(  \) = pour delimiter la sous-expression (designee par \1 dans la
suite)
# [^.]* = caractere different de . en nombre quelconque (comme dans Prenom)

# \. = le point, avec \ pour ne pas l'interpreter comme un caractere
quelconque
```

```
# \(.*\) = 2eme sous-expression correspondant a Nom, avec
# \(  \) = pour delimiter la sous-expression (designee par \2 dans la
suite)
# .* = caractere quelconque en nombre quelconque
```

4.2 Liste des utilisateurs d'un groupe

Écrire un script shell qui produit, à partir des fichiers d'administration `passwd` et `group`, la liste des utilisateurs d'un groupe dont le nom est passé comme argument. Utilisez les fichiers passwd et group fournis. Dans ces fichiers, le cas de l'utilisateur `maing` présente des incohérences entre ces deux fichiers; un tel script permettrait de les corriger.

```
# Comande List_ut affichant la liste des utilisateurs
# appartenant a un groupe passe en argument a la commande
# format de ligne de group = nom_groupe:passwd:id_groupe:liste_utils
# format de ligne de passwd = nom:passwd:uid:gid:commentaire:login_shell
#
# Auteur: Daniel Millot

if [ $# -ne 1 ]; then
echo usage : $(basename $0) nom_du_groupe
exit 1
fi

# recuperation du groupe dans /etc/group
DIR=.   # a remplacer par /etc pour un fonctionnement normal

# on verifie que le nom du groupe $1 figure bien en debut d'une ligne (^)
et on
# recupere la ligne correspondante dans la variable x :
x=$(grep "^$1:" $DIR/group)

if [ "$x" ]        # la variable contient la ligne contenant le groupe
then              # recuperation du GID du groupe dans la variable gid
gid=$(expr "$x" : '.*:.*:\(.*\):.*')

# ici l'operateur : de expr permet d'extraire une sous-chaine de la chaine
$x
# cette sous-chaine est representee par \(.*\) ou  \(  \) la delimite, .*
# represente une chaine quelconque  et les ':' permette de selectionner le
3e
# champ de la ligne (cf format de ligne de group)

#*************************************************************
echo "Liste des membres du groupe $1 (1ere methode avec grep)"
#*************************************************************

# grep liste les lignes avec le numero de GID et cut ne garde que le 1er
champ
grep "^.*:.*:.*:$gid:" $DIR/passwd |cut -d":" -f1

# l'expression reguliere utilisee par grep repose sur les ingredients :
# ^ = debut de ligne
# .* = chaine quelconque ne contenant pas ":" (qui est utilise
explicitement)
# format de ligne de passwd => 4e champ pour le gid
```

```
#*************************************************************
echo "Liste des membres du groupe $1 (2eme methode avec sed)"
#*************************************************************

# avec sed, on remplace les lignes selectionnees par leur seul premier
champ

sed -n "/^.*:.*:.*:$gid:/s/^\([^:]*\):.*/\1/p" $DIR/passwd

# la meme expression reguliere que pour grep permet de selectionner les
lignes
# contenant le gid du groupe
# derriere, le s = substituer (s/ch1/ch2/ remplace ch1 par ch2)
# ch1 = ^\([^:]*\):.* = chaine en debut de ligne (^) ne contenant pas ":"
# (representee par [^:]* ) et allant jusqu'a ":" (c'est le nom, designable
# par \1), et suivie de tout le reste (.*) de la ligne
# ch2 = \1  i.e. le nom comme  indique ci-dessus.

else    # correspond a if [ "$x" ]

# la variable x est vide, ce qui veut dire que $1 ne figure a aucune ligne
echo "Le groupe $1 n'existe pas"
fi
```

Auteurs: Dominique Bouillet, Daniel Millot
Modifications: 2006 Frédérique Silber-Chaussumier
Last modified: Thu Dec 18 16:54:28 CET 2008