commandes unix utilitaires

Sommaire

Étendre une partition avec LVM lvextend

```
lvextend -L +3G -r /dev/vg0/home
```

Cette commande permet d'ajouter 3 Go à la partition `/dev/vg0/home` mais aussi d'étendre la partition `ext4` sous-jacente pour qu'elle occupe l'espace nouvellement alloué à cette partition — c'est l'option `-r (--resisefs)`.

Avec awk

- Afficher la somme de la place occupée par les fichiers trouvés par *find*

Somme des tailles des fichiers trouvés par find

```
find /rep -type f -print0 | xargs -0 ls -al | awk '{sum += $5} END {print sum}'
```

Commentaire : le cinquième champ ($5) correspond à la taille en octets du fichier.

- N'afficher que la taille d'un fichier avec son nom complet (même s'il y a des espaces dans le nom)

Afficher taille et nom de fichier

```
find . -type f -print0 | xargs -0 ls -al | awk '{chaine = "" ; for (i=5; i<=NF;i++) if (i == 5 || i >= 9)
chaine = chaine $i OFS ; print chaine}'| less
```

Commentaire : le nom du fichier commence au neuvième champ et les suivants.

dig

Utiliser dig pour trouver tous les noms d'une machine par son ip

```
dig +nostats +nocomment +nocmd any -x 194.214.217.94
```

Commentaire : pas d'affichage superflux.

N'avoir que les noms dns de la machine, sans le point final et les types

```
dig +nostats +nocomment +nocmd  -x 194.214.217.94 | grep PTR | awk '{print $5}' | sed -e "s/\.$//g"
```

Trucs et astuces avec vim

tri d'une liste de lignes – quelques exemples

```
# tri alphabétique de la zone sélectionnée
:'<,'> sort
# tri des lignes 7 à 204
:7,204 sort
# tri de toutes les lignes du fichier
:% sort
```

Sauver un fichier en lecture seule en utilisant sudo *dans* vim

Parfois, on oublie de mettre un *sudo* devant une édition d'un fichier qui ne nous appartient pas. Alors, voici comment remédier à la situation quand on a déjà fait des modifs dans le fichier en question :

```
# pour sauver et en supposant avoir les droits sudo qui vont bien
:% ! sudo -u utilisateur
# ou, plus simplement, en root
:% ! sudo
# ou mieux (repris de commandlinefu)
:w ! sudo tee %
```

Efficace et ça marche correctement ! Plus de problèmes d'oubli d'un sudo avant d'entrer dans vim.

- http://www.commandlinefu.com/commands/view/5147/save-a-file-you-edited-in-vim-without-the-needed-permissions-no-echo
- http://www.commandlinefu.com/commands/view/1204/save-a-file-you-edited-in-vim-without-the-needed-permissions

sed multiligne

Exemple

Le but est de réécrire un fichier shell (c'est un exemple) en changeant un peu la convention de codage.

Au départ, nous avons des blocs **if …** puis passage à la ligne puis un **then** tout seul sur la sienne.

À la fin, nous voulons un **if … ; then** sur une seule ligne. Le but est de traiter le fichier en une fois en utilisant **sed**.

```
sed -i.bak  '/^[[:space:]]*if[[:space:]].*$/{N;s/^\([[:space:]]*\)if\([[:space:]].*\)\n[[:space:]]*then[[:
space:]]*$/\1if\2 ; then/}' fichier_a_traiter.sh
```

Explications

La première partie permet de délimiter le début de traitement (ligne avec un **if** précédé d'espaces et suivi par des choses.

Ensuite, on ouvre **{** pour commencer une séquence de commandes sed. La première, **N** permet d'accumuler la ligne suivant celle qui vient de correspondre au premier motif. On a donc maintenant la ligne **if** plus celle **then** dans le même buffer où seront faites la substitution suivante. La seule chose, c'est qu'il faut penser que cette substitution se fait sur une chaîne avec 2 lignes, donc séparée par un passage à la ligne. Au final, on referme **}** pour terminer les séquences.

Autre essai

```
# transformation d'une structure de déclaration d'utilisateurs du type :
# user:
#   sshkeys:
#     sshkey_nom_numero:
#       key: '<clé ssh>'
#
# en
# user:
#   sshkeys:
#     - 'ssh-rsa <clé ssh> ssh_nom_numero'

sed -i.bak '/^[[:space:]]*sshkeys:[[:space:]]*$/{N;N;s/^[[:space:]]*sshkeys:\n[[:space:]]*sshkey_\([^_][^_]*_[0-
9]*\):\n[[:space:]]*key: .\([^[:space:]]*\).$/    sshkeys:\n      - '"'"'ssh-rsa \2 sshkey_\1'"'"'/}' users.yaml
```

Références

- http://commandlinefu.com/