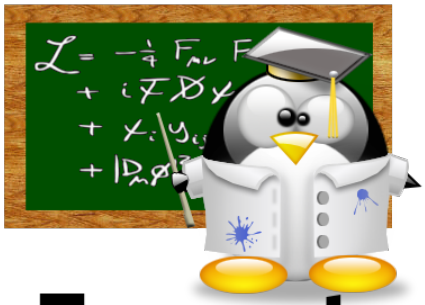

Formatux

Commandes avancées pour utilisateurs Linux

Table des matières

Chapitre 1. La commande uniq

La commande **uniq** est une commande, utilisée avec la commande **sort** très puissante, notamment pour l'analyse de fichiers de logs. Elle permet de trier et d'afficher des entrées en supprimant les doublons.

Pour illustre le fonctionnement de la commande **uniq**, utilisons un fichier prenoms.txt contenant une liste de prénoms :

```
antoine
xavier
patrick
xavier
antoine
antoine
```

uniq réclame que le fichier d'entrée soit trié car il ne compare que les lignes consécutives.

Sans argument, la commande uniq ne va pas afficher les lignes identiques qui se suivent du fichier prenoms.txt :

```
$ sort prenoms.txt | uniq
antoine
patrick
xavier
```

Pour n'afficher que les lignes n'apparaissant qu'une seule fois, il faut utiliser l'option **-u** :

```
$ sort prenoms.txt | uniq -u
patrick
```

A l'inverse, pour n'afficher que les lignes apparaissant au moins deux fois dans le fichier, il faut utiliser l'option **-d** :

```
$ sort prenoms.txt | uniq -d
antoine
xavier
```

Pour simplement supprimer les lignes qui n'apparaissent qu'une seule fois, il faut utiliser l'option **-D** :

```
$ sort prenoms.txt | uniq -D
antoine
antoine
antoine
xavier
xavier
```

Enfin, pour compter le nombre d'occurrences de chaque ligne, il faut utiliser l'option **-c** :

```
$ sort prenoms.txt | uniq -c
      3 antoine
      1 patrick
      2 xavier
```

```
$ sort prenoms.txt | uniq -cd
      3 antoine
      2 xavier
```

Chapitre 2. La commande xargs

La commande **xargs** permet la construction et l'exécution de lignes de commandes à partir de l'entrée standard.

La commande xargs lit des arguments délimités par des blancs ou par des sauts de ligne depuis l'entrée standard, et exécute une ou plusieurs fois la commande (**/bin/echo** par défaut) en utilisant les arguments initiaux suivis des arguments lus depuis l'entrée standard.

Un premier exemple le plus simple possible serait le suivant :

```
$ xargs
utilisation
de
xargs
<CTRL+D>
utilisation de xargs
```

La commande xargs attend une saisie depuis l'entrée standard **stdin**. Trois lignes sont saisies. La fin de la saisie utilisateur est spécifiée à xargs par la séquence de touches <CTRL+D>. Xargs exécute alors la commande par défaut **echo** suivi des trois arguments correspondants à la saisie utilisateur, soit :

```
$ echo "utilisation" "de" "xargs"
utilisation de xargs
```

Il est possible de spécifier une commande à lancer par xargs.

Dans l'exemple qui suit, xargs va exécuter la commande *ls -ld* sur l'ensemble des dossiers qui seront spécifiés depuis l'entrée standard :

```
$ xargs ls -ld
/home
/tmp
/root
<CTRL+D>
drwxr-xr-x. 9 root root 4096  5 avril 11:10 /home
dr-xr-x---. 2 root root 4096  5 avril 15:52 /root
drwxrwxrwt. 3 root root 4096  6 avril 10:25 /tmp
```

En pratique, la commande xargs a exécuté la commande *ls -ld "/home" "/tmp" "/root"*.

Que se passe-t-il si la commande à exécuter n'accepte pas plusieurs arguments comme c'est le cas pour la commande find ?

```
$ xargs find /var/log -name
*.old
*.log
find: les chemins doivent précéder l'expression : *.log
```

La commande xargs a tenté d'exécuter la commande *find* avec plusieurs arguments derrière l'option *-name*, ce qui fait généré par find une erreur :

```
$ find /var/log -name "*.old" "*.log"
find: les chemins doivent précéder l'expression : *.log
```

Dans ce cas, il faut forcer la commande xargs à exécuter plusieurs fois (une fois par ligne saisie en entrée standard) la commande find. L'option **-L** suivie d'un **nombre entier** permet de spécifier le nombre maximal d'entrées à traiter avec la commande en une seule fois :

```
$ xargs -L 1 find /var/log -name
*.old
/var/log/dmesg.old
*.log
/var/log/boot.log
/var/log/anaconda.yum.log
/var/log/anaconda.storage.log
/var/log/anaconda.log
/var/log/yum.log
/var/log/audit/audit.log
/var/log/anaconda.ifcfg.log
/var/log/dracut.log
/var/log/anaconda.program.log
```

Si nous avions voulu pouvoir spécifier sur la même ligne les deux arguments, il aurait fallut utiliser l'option -n 1 :

```
$ xargs -n 1 find /var/log -name
*.old *.log
/var/log/dmesg.old
/var/log/boot.log
/var/log/anaconda.yum.log
/var/log/anaconda.storage.log
/var/log/anaconda.log
/var/log/yum.log
/var/log/audit/audit.log
/var/log/anaconda.ifcfg.log
/var/log/dracut.log
/var/log/anaconda.program.log
```

Cas concret d'une sauvegarde avec un tar en fonction d'une recherche :

```
$ find /var/log/ -name "*.log" -mtime -1 | xargs tar cvfP /root/log.tar
$ tar tvfP /root/log.tar
-rw-r--r-- root/root      1720 2017-04-05 15:43 /var/log/boot.log
-rw------- root/root    499270 2017-04-06 11:01 /var/log/audit/audit.log
```

Chapitre 3. Le paquet yum-utils

Le paquet **yum-utils** est une collection d'utilitaires de différents auteurs pour yum, qui le rendent plus simple et plus puissant à utiliser.

Voici quelques exemples d'utilisation :

- La commande package-cleanup :

Elle permet (entre autre) la suppression des anciens noyau.

Syntaxe de la commande package-cleanup

```
package-cleanup --oldkernels --count=1
```

Table 1. Options de la commande package-cleanup

Options	Commentaires
--oldkernels	Demande la suppression des anciens noyaux.
--count=X	Nombre de noyaux à conserver (par défaut = 2)

- La commande repoquery :

La commande **repoquery** interroge les dépots.

Exemples d'utilisation :

- Connaître les dépendances d'un paquet non-installé :

```
repoquery --requires <package>
```

- Connaître les fichiers fournis par un paquet non-installé :

```
$ repoquery -l yum-utils
/etc/bash_completion.d
/etc/bash_completion.d/yum-utils.bash
/usr/bin/debuginfo-install
/usr/bin/find-repos-of-install
/usr/bin/needs-restarting
/usr/bin/package-cleanup
/usr/bin/repo-graph
/usr/bin/repo-rss
/usr/bin/repoclosure
/usr/bin/repodiff
/usr/bin/repomanage
/usr/bin/repoquery
/usr/bin/reposync
/usr/bin/repotrack
/usr/bin/show-changed-rco
/usr/bin/show-installed
/usr/bin/verifytree
/usr/bin/yum-builddep
/usr/bin/yum-config-manager
/usr/bin/yum-debug-dump
/usr/bin/yum-debug-restore
/usr/bin/yum-groups-manager
/usr/bin/yumdownloader
...
```

- La commande yumdownloader :

La commande **yumdownloader** télécharge les paquets RPM depuis les dépôts.

Cette commande est très pratique pour construire un dépôt local de quelques rpm !

Exemple : yumdownloader va télécharger le paquet rpm de repoquery ainsi que toutes ses dépendances.

```
$ yumdownloader --destdir /var/tmp -- resolve repoquery
```

Table 2. Options de la commande yumdownloader

Options	Commentaires
--destdir	Les paquets téléchargés seront conservés dans le dossier spécifié.
--resolve	Télécharge également les dépendances du paquet.

Chapitre 4. Le paquet psmisc

Le paquet **psmisc** contient des utilitaires pour gérer les processus du système :

- **pstree** : la commande pstree affiche les processus en cours sur le système sous forme de structure en forme d'arbre.
- **killall** : la commande killall envoie un signal d'extinction à tous les procesuss identifiés par un nom.
- **fuser** : la commande fuser identifie les PIDs des processus qui utilisent les fichiers ou les systèmes de fichiers spécifiés.

Exemples :

```
$ pstree
systemd─┬─NetworkManager───2*[{NetworkManager}]
        ├─agetty
        ├─auditd───{auditd}
        ├─crond
        ├─dbus-daemon───{dbus-daemon}
        ├─firewalld───{firewalld}
        ├─lvmetad
        ├─master─┬─pickup
        │        └─qmgr
        ├─polkitd───5*[{polkitd}]
        ├─rsyslogd───2*[{rsyslogd}]
        ├─sshd───sshd───bash───pstree
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-udevd
        └─tuned───4*[{tuned}]
```

```
# killall httpd
```

Tue les processus (option -k) qui accèdent au fichier */etc/httpd/conf/httpd.conf* :

```
# fuser -k /etc/httpd/conf/httpd.conf
```

Chapitre 5. La commande watch

La commande **watch** exécute régulièrement une commande et affiche le résultat dans le terminal en plein écran.

L'option **-n** permet de spécifier le nombre de secondes entre chaque exécution de la commande.

Pour quitter la commande watch, il faut saisir les touches : <CTRL>+<C> pour tuer le processus.

Exemples :

- Afficher la fin du fichier /etc/passwd toutes les 5 secondes :

```
$ watch -n 5 tail -n 5 /etc/passwd
```

Résultat :

```
Toutes les 5,0s: tail -n 5 /etc/p...

lightdm:x:620:620:Light Display Manager:/var/lib/lightdm:/usr/bin/nologin
clamav:x:64:64:Clam AntiVirus:/dev/null:/bin/false
systemd-coredump:x:994:994:systemd Core Dumper:/:/sbin/nologin
ceph:x:993:993::/run/ceph:/sbin/nologin
dnsmasq:x:992:992:dnsmasq daemon:/:/sbin/nologin
```

- Surveillance du nombre de fichier dans un dossier :

```
$ watch -n 1 'ls -l | wc -l'
```

- Afficher une horloge :

```
$ watch -t -n 1 date
```