



Commandes avancées pour utilisateurs Linux

24/06/2017

1. La commande

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 illustrer 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 prenom.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 prenom.txt | uniq -c
  3 antoine
  1 patrick
  2 xavier
```

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

2. La commande

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 concrèt 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
```