Le filtres puissants find grep sed

1 Introduction

Vous êtes maintenant bien plus à l'aise avec votre ligne de commande. Vous gagnez en rapidité et en efficacité. Vous avez un certain nombre de commandes dans votre boite à outils avec lesquels vous arrivez à exploiter votre système d'exploitation. Il est donc temps pour vous de passer à la vitesse supérieure et de maîtriser toute la puissance de votre système. Pour rappel, la philosophie des commandes UNIX dont les systèmes Linux ont hérité est : **Chaque commande fait une seule chose et le fait bien**. Cette section va introduire des filtres puissants par opposition aux filtres simples vus dans la précédente. Les commandes que vous allez voir sont des commandes (basiques) qui font toute la puissance de la ligne de commande.

2 La commande find

1 Généralités

Trouver un fichier dans son arborescence est l'une des tâches les plus fréquentes que l'on effectue devant un ordinateur. Jusqu'ici, pour trouver un fichier, vous vous servez de la commande **ls**. Cependant,cette commande ne vous permet pas d'explorer simplement une arborescence à la recherche d'un fichier ou d'un groupe de fichier précis. L'utilisation de l'option**-r** de la commande ls donne une liste récursive complète (par opposition à filtrée) de tous les fichiers et répertoires de l'arborescence sur laquelle est lancée la commande. Une autre commande est bien plus adaptée pour cette tâche de recherche : la commande **find**. Cette commande permet de retrouver dans un répertoire, ou une liste de répertoires, un fichier ou un ensemble de fichiers possédant certaines caractéristiques comme : le nom, les droits, les dates, la taille, etc. ou satisfaisant à une expression donnée en argument.

La commande **find** possède beaucoup d'options dont les principales sont :

> **-name fichier** où fichier est le nom du fichier à trouver. Ce nom peut comporter des caractères spéciaux. L'option **-iname fichier** indique de ne pas être sensible à la casse.
>
> **-perm nombre** où nombre équivaut aux droits d'accès.
>
> -**user nom** où nom correspond à un nom d'utilisateur et le résultat de la recherche listera tous les fichiers appartenant à cet utilisateur.
>
> **-size n** où n correspond à la taille du fichier

`-mtime n` où n correspond à une date ou un temps.

2 Exemples simples

Donnons quelques exemples courants d'utilisation de la commande **find**

```
psimier@b107PSR:~$  find /etc -name "av*"
```

Liste tous entités (fichiers et répertoires) dans /etc dont le nom commence par av. Les guillemets sont fondamentaux afin que le shell n'interprète pas le motif de recherche à la place de find.

```
psimier@b107PSR:~$ find . -iname  "sn*" -type f
```

Liste tous les fichiers (option -type f) dans le répertoire courant (option .) dont le nom commence par **sn** sans tenir compte de la casse (option iname).

```
psimier@b107PSR:~$ find $HOME -mtime -1
```

Liste toutes les entités du répertoire personnel qui ont été modifiées il y a moins d'un jour. L'option -mtime +1 avec un nombre signé positivement indique de retrouver les fichiers modifiés il y a plus de 1 jour.

```
psimier@b107PSR:~/2020$ find . -size +1M
```

Liste toutes les entités du répertoire personnel 2020 dont la taille dépasse le méga octet. Une option **-size -1G** listera les entités de moins de 1 giga octets.

```
psimier@b107PSR:~/2020$ find . ! -user $LOGNAME
```

Liste toutes les entités du répertoire ~/2020 dont vous n'êtes pas le propriétaire, l'option ! Exprime la négation.

3 Intersection (ET) et union (OU) avec find

Il est possible de combiner les critères avec la commande find. Il suffit de saisir toutes les options. Attention, néanmoins, dans ce cas, ces critères sont pris comme des intersections (ET). Par exemple

```
psimier@b107PSR:~$ find . -size +15M -size -1G
```

Liste toutes les entités du répertoire courant dont la taille dépasse quinze méga octets mais dont la taille est aussi inférieure à un giga octet.

L'option **-o** de la commande **find** permet de chercher sur des unions de critères (OU).

```
psimier@b107PSR:~/2020$ find . -name "*.c" -o -name "*.cpp"
```

Liste toutes les entités du répertoire courant 2020 dont le nom se termine par **.c** ou par **.cpp**.

4 L’option -exec

On peut faire exécuter à la commande **find** n’importe quelle commande en utilisant l’option **-exec**.

Exemple comptage des lignes des fichiers trouvés.

```
psimier@b107PSR:~/2020/snir1$ find . -name "*.txt" -exec  wc -l {} ';'
14 ./Operating System/AdminLinux/tp/masquerade.txt
3 ./Operating System/TP2/liste de noel.txt
3 ./Operating System/TP2/listeDeNoel.txt
```

Tous les {} sont remplacés par le chemin trouvé en protégeant automatiquement les espaces. Le ; indique la fin de la commande que l’on veut exécuter. Il faut le protéger en l’écrivant entre quote ” car ce symbole est aussi le séparateur de commandes.

L’exemple ci dessus trouve toutes les entités dont le nom se termine par **.txt**, puis exécute la commande **wc -l** sur les entités trouvées (représentée par{})

3 la commande grep

1 Utilisation classique

La commande **find** vue dans la section précédente permet de trouver des fichiers avec certaines caractéristiques, mais ne permet pas de rechercher dans le contenu du fichier. La commande **grep** affiche toutes les lignes des fichiers données en argument, ou sur l’entrée standard, contenant une chaîne de caractères elle aussi donnée en argument. Prenons le fichier slogan.txt suivant comme exemple :

```
$ cat slogan.txt
There is nothing UNIX can’t buy
Give that man a Bash
Proudly powered by linux
Crunch All you want. We’ll make Bash
UNIX or nothing
Kids are stronger with linux
If you really want to know, look into linux
We’re always low UNIX
Pleasing linux the world over
Linux inside
Live free or die UNIX
```

Voici un exemple d’utilisation de la commande grep sur ce fichier

```
$ grep UNIX slogan.txt
There is nothing UNIX can’t buy
UNIX or nothing
We’re always low UNIX
Live free or die UNIX
```

Dans cet exemple, on cherche dans le fichier *slogan.txt* toutes les lignes contenant le mot ’UNIX’. Ici nous ne donnons qu’un seul fichier en argument mais il est possible d’en donner plusieurs.

```
$ grep UNIX fichier1.txt fichier2.txt
```

Les options de la commande les plus utilisées sont -v pour inverser la sélection, -n pour afficher le numéro de ligne et -c pour donner le nombre de lignes trouvées.

```
$ grep -v UNIX slogan.txt
Give that man a Bash
Proudly powered by linux
Crunch All you want. We’ll make Bash
Kids are stronger with linux
If you really want to know, look into linux
Pleasing linux the world over
Linux inside

$ grep -nv UNIX slogan.txt
2:Give that man a Bash
3:Proudly powered by linux
4:Crunch All you want. We’ll make Bash
6:Kids are stronger with linux
7:If you really want to know, look into linux
9:Pleasing linux the world over
10:Linux inside

$ grep -vc UNIX slogan.txt
7
```

Les motifs de recherche utilisés par la commande grep sont les expressions régulières. Il existe des livres entiers traitant des expressions régulières. Ce sont des chaînes syntaxiques qui donnent tout leur sens aux commandes comme *grep*. Les expressions régulières utilisées avec la commande *grep* permettent par exemple de rechercher des expressions complexes comme : trouver les lignes contenant les mots commençant par "th" ; les lignes contenant les mots de 4 lettres se terminant par "w" ; etc. Nous n’allons pas examiner ici toutes les possibilités des expressions régulières, mais seulement donner quelques exemples basiques.

L’accent circonflexe permet de chercher une expression en début de ligne :

```
$ grep ^Li slogan.txt
Linux inside
Live free or die UNIX
```

A l’inverse, le caractère $ permet de rechercher une expression en fin de ligne.

```
$ grep linux$ slogan.txt
Proudly powered by linux
Kids are stronger with linux
If you really want to know, look into linux
```

La commande suivante permet de lister toutes les lignes commençant par "K" ou commençant par "U". Notez dans ce cas que l’expression régulière est placée entre guillemets simples.

```
$ grep ’^[KU]’ slogan.txt
UNIX or nothing
Kids are stronger with linux
```

Les crochets permettent aussi de définir une plage de caractères. Dans l’exemple suivant on liste toutesles lignes commençant par un caractère entre "K" et "U".

```
$ grep ’^[K-U]’ slogan.txt
There is nothing UNIX can’t buy
Proudly powered by linux
UNIX or nothing
Kids are stronger with linux
Pleasing linux the world over
Linux inside
Live free or die UNIX
```

2 Filtres puissants

La commande grep s’utilise souvent comme filtre. L’exemple suivant permet de rechercher "motif" dans les fichiers f1,f2,f3, et f4.

```
$ cat f1 f2 f3 f4 | grep "motif"
```

4 La commande sed

La commande **sed** est un éditeur de texte non-interactif, c'est-à-dire qu'il ne permet pas la saisie de texte mais plutôt d'éditer le contenu du fichier de façon automatisée. C'est en fait un filtre de la même nature que grep. La commande sed (stream editor) permet de modifier un flux de données de taille illimitée en utilisant très peu de mémoire. La commande sed est de ce fait un outil très rapide pour l'édition complexe de fichier.

La commande **sed** possède plusieurs options offrant une grande variété de fonctionnalités d'édition et de traitements. En effet, la commande **sed** peut utiliser les expressions régulières qui en font un filtre très puissant et dont les possibilités sont infinies. La commande **sed** lit les fichiers dont les noms sont indiqués en argument. Si aucun nom n'est indiqué, elle lit l'entrée standard. On lui indique les traitements à effectuer en utilisant l'option **-e** (la présence de cette option est facultative s'il n'y a qu'une seule directive de traitement). Par défaut les fichiers indiqués ne sont pas modifiés et le résultat de la transformation est écrit sur la sortie standard. Mais avec l'option **-i** on peut obtenir que le résultat soit écrit dans chacun des fichiers.

L'utilisation principale de la commande **sed** est la substitution d'une expression régulière par une chaîne de caractères, la forme syntaxique pour exprimer ce traitement est la suivante :

s/expression-reguliere/chaîne/g

L'option **/g** indique que toutes lignes validant l'expression régulière seront traitées, sinon le traitement s'arrêtera après la première occurrence trouvée. La commande sed permet d'utiliser les expressions régulières étendues avec l'option **-r**, nous n'entrerons pas dans ce type d'expressions régulières ici. Pour utiliser les caractères spéciaux tel que /ou **&** dans les expressions ou la chaîne, il faut utiliser le caractère d'échappement \. Il est possible d'effectuer plusieurs substitutions à la suite, elles seront alors traitées les unes après les autres et de gauche à droite.
Voici les exemples d'utilisation les plus courants :

```
$sed 's/alice/bob/g' fichier1 fichier2
```

Lit le contenu des fichiers fichier1 et fichier2 et l'écrit sur la sortie standard en remplaçant tous les **alice** par **bob**. Pour supprimer un mot il suffit de substituer le mot par une chaîne vide. Par exemple, pour supprimer le mot alice la commande est :

```
$sed 's/alice//g' fichier1 fichier2
```

Il est possible d'utiliser la commande sed comme filtre :

```
$cat fichier | sed -e 's/a/A/g' -e 's/TA/ta/g'
```

La commande ***cat*** affiche le contenu du fichier *fichier* sur la sortie standard, qui est reprise par la commande ***sed*** (c'est le cas classique d'utilisation d'une commande filtre).
Notez ici que nous avons utilisé l'option -e car la commande contient plusieurs directives de traitement. La première substitution change tous les ***a*** en ***A*** et la seconde tous les ***TA*** en ***ta***. Ce qui signifie que si *fichier* contient une ligne ***a Table***, cette ligne sera transformée en ***A table***. En effet, la première substitution transformera :

a Table --> A Table

et la deuxième substitution agira sur le résultat de la première substitution :

A TAble --> A table

1 0ptions classiques

Dans les exemples précédents, les substitutions sont effectuées sur l'ensemble des lignes. La syntaxe de substitution de la commande ***sed*** permet de spécifier les numéros des lignes sur lesquelles la substitution doit être effectuée. Par exemple, la commande suivante permet de remplacer 'UNIX' par '*nix' seulement sur les lignes **1 à 5 incluses** :

```
$ sed '1,5s/UNIX/*nix/g' slogan.txt
```

la commande inverse pour rétablir le fichier d'origine

```
$ sed '1,5s/*nix/UNIX/g' slogan.txt
```

S'il est possible d'effectuer les substitutions seulement sur des lignes données en argument, il est aussi possible de spécifier les lignes en fonction d'un motif de recherche.

```
$ sed -e '/^C/s/Bash/BASH/g' slogan.txt
```

La commande précédente permet de remplacer '***Bash***' par '***BASH***' sur les lignes commençant par '***C***'.
La commande suivante permet de faire ce changement sur les lignes contenant le mot '***that***'.

```
$ sed -e '/that/s/Bash/BASH/g' slogan.txt
```

Jusqu'ici nous avons effectué des substitutions en utilisant un motif de recherche simple. Mais il est possible, comme pour la commande grep, d'utiliser des expressions régulières complexes.
La commande suivante remplacera toutes les chaînes de caractères contenant 'L' et 'x' avec n'importe quel nombre de caractère entre ces deux lettres par "XXX"

```
$ sed -e 's/L.*x/XXX/g' slogan.txt
```

sed permet aussi de supprimer des lignes trouvées, au lieu de faire une substitution comme on l'a fait jusqu'ici. La commande suivante permet de supprimer, grâce à l'action ***/d***
La commande suivante effectue la suppression de toutes les ligne contenant le mot Linux.

```
$ sed -e '/Linux/d' slogan.txt
```

5 Conclusion

Les commandes que nous avons vu ici (***find***, ***grep***, ***sed***) sont, selon moi, les commandes les plus puissantes de linux car elles permettent vraiment de passer d'un stade de débutant à un stade d'utilisateur avancé sous linux. Maîtriser ces commandes, et les filtres puissants en général, font partie des fondamentaux de linux et vous venez d'acquérir ces fondamentaux. Les activités suivantes vous permettront de mieux exploiter votre ligne de commande.