Unix / Linux III
& Introduction

Ziyati

Université Hassan II
Faculté des Sciences Ain Chock, Casa

17 juillet 2018

Plan

Recherche d'un fichier **find**

La commande `find` permet de retrouver des fichiers à partir de certains critères.

Syntaxe :

```
find <répertoire de recherche> <critères de recherche>
```

critères de recherche :

`-name`	recherche sur le nom du fichier
`-perm`	recherche sur les droits d'accès
`-link`	recherche sur le nombre de liens
`-user`	recherche sur le propriétaire
`-group`	recherche sur le groupe auquel appartient le fichier
`-type`	recherche sur le type
`-size`	recherche sur la taille
`-atime`	recherche sur la date de dernier accès en lecture
`-mtime`	recherche sur la date de dernière modification du fichier
`-ctime`	recherche sur la date de création du fichier

Recherche d'un fichier **find**

On peut combiner les critères avec des opérateurs logiques :

- `critère1 critère2` ou `critère1 -a critère2` ≡ au ET logique,
- `!critère` ≡ NON logique,
- `\(critère1 -o critère2\)` ≡ OU logique,

L'option `-print` est indispensable pour obtenir une sortie.

Remarque

La commande `find` est récursive, *i.e.* scruter dans les répertoires, et les sous répertoires qu'il contient.

Recherche d'un fichier **find**

Recherche par nom de fichier

Pour chercher un fichier dont le nom contient la chaîne de caractères `toto` à partir du répertoire `/usr` :

```
find /usr -name toto -print
```

- Si le(s) fichier(s) existe(nt) ⇒ sortie : `toto`
- En cas d'échec, vous n'avez rien.

Pour rechercher tous les fichiers se terminant par `.c` dans le répertoire `/usr` :

```
find /usr -name " *.c " -print
```

⇒ toute la liste des fichiers se terminant par `.c` sous les répertoires contenus dans `/usr` (et dans `/usr` lui même).

Recherche d'un fichier **find**

Recherche suivant la date de dernière modification

Ex :Les derniers fichiers modifiés dans les 3 derniers jours dans toute l'arborescence (/) : `find / -mtime 3 -print`

Recherche suivant la taille

Ex :Connaître dans toute l'arborescence, les fichiers dont la taille dépasse 1Mo (2000 blocs de 512Ko) :

```
find / -size 2000 -print
```

Recherche combinée

Ex :Chercher dans toute l'arborescence, les fichiers ordinaires appartenant à olivier, dont la permission est fixée à 755 :

```
find / -type f -user olivier -perm 755 -print
```

Ex : Recherche des fichiers qui ont pour nom `a.out` et des fichiers se terminant par `.c` :

```
find . \ ( -name a.out -o -name " *.c " \ ) -print
```

Recherche d'un fichier **find**

Commandes en option :
En dehors de `-print` on dispose de l'option `-exec`. Le `find` couplé avec `exec` permet d'exécuter une commande sur les fichiers trouvés d'après les critères de recherche fixés. Cette option attend comme argument une commande, suivie de {}\ .
Ex : recherche des fichiers ayant pour nom `core`qu'on efface
`find . -name core -exec rm {}\`
Ex : les fichiers ayant pour nom `core` seront détruits, pour avoir une demande de confirmation avant l'exécution de `rm` :
`find . -name core -ok rm {}\`
Autres subtilités : Une fonction intéressante de `find` est de pouvoir être utilisé avec d'autres commandes
Ex : `find . -type f -print | xargs grep toto`
Rechercher dans le répertoire courant tous les fichiers normaux (sans fichiers spéciaux), et rechercher dans ces fichiers tous ceux contenant la chaîne `toto`.

Recherche d'un fichier **find**

Appliquons ! ! !

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)

Recherche d'un fichier **find**

Appliquons ! ! !

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)
   `find /var/log/ -name "syslog"`

Recherche d'un fichier **find**

Appliquons ! ! !

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)
2. $\cdots$ Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?

Recherche d'un fichier **find**

Appliquons ! ! !

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)
2. $\cdots$ Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?
   `find / -name ‘‘syslog’’`

Recherche d'un fichier **find**

Appliquons ! ! !

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)
2. $\cdots$ Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?
3. rechercher tous les fichiers qui font plus de 10 Mo dans mon `home`

Recherche d'un fichier **find**

Appliquons ! ! !

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)
2. $\cdots$ Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?
3. rechercher tous les fichiers qui font plus de 10 Mo dans mon `home`

   `find ~ -size +10M`

Recherche d'un fichier **find**

Appliquons ! ! !

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)
2. $\cdots$ Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?
3. rechercher tous les fichiers qui font plus de 10 Mo dans mon `home`
4. uniquement les rep qui s'appellent `syslog` (et pas les fichiers)

Recherche d'un fichier **find**

Appliquons ! ! !

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)
2. $\cdots$ Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?
3. rechercher tous les fichiers qui font plus de 10 Mo dans mon `home`
4. uniquement les rep qui s'appellent `syslog` (et pas les fichiers)

   `find /var/log -name ‘‘syslog‘‘ -type d`

Recherche d'un fichier **find**

Appliquons !!!

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)
2. $\cdots$ Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?
3. rechercher tous les fichiers qui font plus de 10 Mo dans mon `home`
4. uniquement les rep qui s'appellent `syslog` (et pas les fichiers)
5. Imaginons que je souhaite mettre un `chmod` à 600 pour chacun de mes fichiers `jpg`, pour que je sois le seul à pouvoir les lire

Recherche d'un fichier **find**

Appliquons ! ! !

1. Retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-rep)
2. $\cdots$ Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?
3. rechercher tous les fichiers qui font plus de 10 Mo dans mon `home`
4. uniquement les rep qui s'appellent `syslog` (et pas les fichiers)
5. Imaginons que je souhaite mettre un `chmod` à 600 pour chacun de mes fichiers `jpg`, pour que je sois le seul à pouvoir les lire

```
find ~ -name "*.jpg" -exec chmod 600 {} +
```

`sed` est éditeur ligne non interactif, il lit les lignes d'un fichier une à une (ou provenant de l'entrée standard) leur applique un certain nombre de commandes d'édition et renvoie les lignes résultantes sur la sortie standard. Il ne modifie pas le fichier traité, il écrit tout sur la sortie standard.

Syntax `sed -e 'programme sed' fichier-a-traiter`
ou `sed -f fichier-programme fichier-a-traiter`

On dispose de l'option `-n` qui supprime la sortie standard par défaut, `sed` va écrire uniquement les lignes concernées par le traitement (sinon il écrit tout même les lignes non traitées).
L'option `-e` n'est pas nécessaire quand on a une seule fonction d'édition.
`sed` est une commande très riche (pour plus de détails `man sed`)

La commande **sed**

La fonction de substitution : `s`

`s` permet de changer la 1^{re} ou toutes les occurences d'une chaîne par une autre.

Syntaxe :

- `sed "s/toto/TOTO/" fichier` va changer la 1^{re} occurence de la chaîne `toto` par `TOTO`
- `sed "s/toto/TOTO/3" fichier` va changer la 3^{me} occurence de la chaîne `toto` par `TOTO`
- `sed "s/toto/TOTO/g" fichier` va changer toutes les occurences de la chaîne `toto` par `TOTO`
- `sed "s/toto/TOTO/p" fichier` en cas de remplacement imprime les lignes concernées
- `sed "s/toto/TOTO/w resultat" fichier` en cas de substitution la ligne en entrée est inscrite dans un fichier résultat

La commande **sed**

La fonction de substitution peut être utilisée avec une expression régulière.
`sed -e "s/[Ff]raise/FRAISE/g" fichier` substitue toutes les chaînes `Fraise` ou `fraise` par `FRAISE`

La commande **sed**

La fonction de suppression : `d`

La fonction de suppression `d` supprime les lignes comprises dans un intervalle donné.

Syntaxe : `sed "20,30d" fichier`

Cette commande va supprimer les lignes 20 à 30 du fichier `fichier`. On peut utiliser les expressions régulières :

- `sed "/toto/d" fichier` : supprime les lignes contenant la chaîne `toto`
- `sed "/toto/!d" fichier` : supprime toutes les lignes ne contenant pas la chaîne `toto`

En fait les lignes du fichier d'entrée ne sont pas supprimées, elles le sont au niveau de la sortie standard.

La commande **sed**

Les fonctions : `p`, `l` et`=`

- `p` (print) affiche la ligne sélectionnée sur la sortie standard. Elle invalide l'option `-n`.
- `l` (list) affiche la ligne sélectionnée sur la sortie standard avec en plus les caractères de contrôles en clair avec leur code ASCII (deux chiffres en octal).
- `=` donne le num de la ligne sélectionnée sur la sortie standard.

Ces trois commandes sont utiles pour le débogage, (mise au point des programmes `sed`)
`sed "/toto/=" fichier` : afficher le numéro de la ligne contenant la chaîne `toto`.

La commande **sed**

Les fonctions : `q, r` etw

- `q` (quit) interrompre l'exécution de `sed`, la ligne en cours de traitement est affichée sur la sortie standard (uniquement si `-n` n'a pas été utilisée).
- `r` (read) lit le contenu d'un fichier et écrit le contenu sur la sortie standard.
- `w` (write) écrit la ligne sélectionnée dans un fichier.

`sed "/^ toto/w resultat" fichier` : Ecrire dans le fichier `resultat` toutes les lignes du fichier `fichier` commençant par la chaîne `toto`.

La commande **sort**

La commande `sort` se révèle bien utile lorsqu'on a besoin de trier le contenu d'un fichier

sort : `sort sort.txt`

Le contenu du fichier est trié alphabétiquement et le résultat est affiché dans la console. Vous noterez que `sort` ne fait pas attention à la casse (majuscules / minuscules).

Le fichier en lui-même n'a pas été modifié lorsque nous avons lancé la commande. Seul le résultat était affiché dans la console.

Vous pouvez faire en sorte que le fichier soit modifié en précisant un nom de fichier avec l'option `-o` :

```
sort -o nomstries.txt sort.txt
```

La commande **wc**

La commande `wc` signifie *Word Count*. C'est donc a priori un compteur de mots, mais en fait on lui trouve plusieurs autres utilités : compter le nombre de lignes (très fréquent) et compter le nombre de caractères.
Comme les précédentes, la commande `wc` travaille sur un fichier. Sans paramètres, les résultats renvoyés par `wc` sont un peu obscurs :

`wc sort.txt` $\Longrightarrow$ `27 27 220 sort.txt`

Ces 3 nombres signifient, dans l'ordre :

1. Le nombre de lignes
2. Le nombre de mots
3. Le nombre d'octets

La commande **wc**

Les options :

1. `-l` : compter le nombre de lignes
2. `-w` : compter le nombre de mots
3. `-c` : compter le nombre d'octets
4. `-m` : compter le nombre caractères

La commande **uniq**

Parfois, certains fichiers contiennent des lignes en double et on aimerait pouvoir les détecter ou les supprimer. La commande `uniq` est toute indiquée pour cela. Nous devons travailler sur un fichier trié. En effet, la commande `uniq` ne repère que les lignes successives qui sont identiques.

```
uniq doubl.txt
```

Vous pouvez demander à ce que le résultat sans doublons soit écrit dans un autre fichier plutôt qu'affiché dans la console :

```
uniq doubl.txt sansdoubl.txt
```

La commande **uniq**

Les options :

1. `-d` : afficher uniquement les lignes présentes en double
2. `-c` : compter le nombre d'occurences

La commande **cut**

Vous avez déjà coupé du texte dans un éditeur de texte, non ?
La commande **cut** vous propose de faire cela au sein d'un fichier, afin de conserver uniquement une partie de chaque ligne.

Couper selon le nombre de caractères

Par exemple, si vous souhaitez conserver uniquement les caractères 2 à 5 de chaque ligne du fichier :

```
cut -c 2-5 sort.txt
```

Pour conserver du 1er au 3me caractère

```
cut -c -3 sort.txt
```

pour conserver du 3me au dernier caractère

```
cut -c 3- sort.txt
```

La commande **cut**

Couper selon un délimiteur

Faisons maintenant quelque chose de bien plus intéressant.
Plutôt que de s'amuser à compter le nombre de caractères, on va travailler avec ce qu'on appelle un délimiteur. Prenons un cas pratique : les fichiers notes.
Imaginons que nous souhaitons extraire de ce fichier la liste des prénoms. Comment nous y prendrions-nous ?

La commande **cut**

Couper selon un délimiteur

Faisons maintenant quelque chose de bien plus intéressant.
Plutôt que de s'amuser à compter le nombre de caractères, on va travailler avec ce qu'on appelle un délimiteur. Prenons un cas pratique : les fichiers notes.
Imaginons que nous souhaitons extraire de ce fichier la liste des prénoms. Comment nous y prendrions-nous ?
Nous allons donc nous servir du fait que nous savons que la virgule sépare les différents champs dans ce fichier. Vous allez avoir besoin d'utiliser 2 paramètres :

- `-d` :indique quel est le délimiteur dans le fichier
- `-f` :indique le numéro du ou des champs à couper

La commande **cut**

Dans notre cas, le délimiteur qui sépare les champs est la virgule. Le numéro du champ à couper est 1 (c'est le premier).

```
cut -d , -f 1 notes.txt
```

Après le `-d`, nous avons indiqué quel était le délimiteur (à savoir la virgule).
Après le `-f`, nous avons indiqué le numéro du champ à conserver (le premier).