find

- Objet : find
- Niveau requis :
  débutant, avisé
- Commentaires : *Recherche de fichiers.*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !
- Suivi :
  - Création par martin_mtl le 09/12/2012
  - Testé par smolski 26-08-2013
- Commentaires sur le forum : Lien vers le forum concernant ce tuto[1)]

Introduction

Les exemples de ce tuto sont tous à faire en *user* sauf spécification.

Cette commande permet de faire des recherches de fichier ou de dossier dans une hiérarchie de répertoires.

Par exemple, je voudrais chercher le log messages, mais je sais pas où se trouve ce fichier, faites :

```
find / -name 'messages'
```

résultat de la commande précédente

```
/var/log/messages
```

Voilà la réponse :
Il se trouve dans le répertoire `/var/log`.

Nota

Notez l'utilisation des apostrophes afin d'éviter que l'interpréteur de commande n'étende le motif.
Elles sont inutiles dans ce cas-ci mais c'est une bonne pratique de toujours les utiliser afin d'éviter l'extension *motif*.

Quelques options

Options	Fonctions
-atime n ou +n ou -n	*trouve les fichiers auxquels on a accédé il y a strictement n jours, ou plus de n jours, ou moins de n jours*
-mtime n ou +n ou -n	*trouve les fichiers modifiés il y a strictement n jours, ou plus de n jours, ou moins de n jours*
-maxdepth n	*définit le niveau maximum de sous-répertoire à explorer*

Options	Fonctions
-type l ou d ou f	*indique le type de fichier à rechercher (l pour lien symbolique, d pour répertoire (directory), f pour fichier)*
-name	*recherche par motif en respectant la casse*
-iname	*recherche par motif sans respecter la casse*

Recherche simple par nom

Exemple simple : comment trouver un fichier portant le nom `note` ?

```
find / -name 'note'
```

Décomposition de la commande de l'exemple :

1. “`/`” indique que nous voulons chercher notre fichier à partir de la racine.
2. “`-name` ” est l'option qui indique ici que nous voulons spécifier le nom d'un fichier.

Après un long délai d'attente, la recherche se faisant dans toute l'arborescence de la partition, la réponse finit par venir :

résultat de la commande précédente

```
/home/martin/note
```

Si l'on n'est pas sûr de la casse (Majuscule ou minuscule) on utilise l'option `-iname`.

Règle générale, on recherche rarement un fichier depuis la racine.

Prenons un autre exemple.

Pour chercher tous les fichiers commençant par *note* et définir à partir de quel répertoire on souhaite effectuer la recherche on utilise cette syntaxe :

```
find /home/martin -name 'note*'
```

Recherche par nom simple & multiple

Maintenant, regardons, encore une fois à l'aide d'un exemple, la syntaxe de la commande **find** si l'on recherche plutôt un ou plusieurs répertoires.

Je cherche à trouver les répertoires archives dans /media/homebis. Première chose à noter, il peut-être nécessaire de se mettre en root pour avoir accès à tous les répertoires.

```
find /media/homebis -type d -name 'archives'
```

Dans ce cas-ci, je demande donc à **find** de trouver les répertoires

1. option : -type
2. argument : “d” (comme “directory”)

indiquant que l'on cherche un répertoire du nom de *archives* à partir du répertoire /media/homebis.

La réponse :

résultat de la commande précédente

```
/media/homebis/martin/textes/mes_archives/Baseball/archives
/media/homebis/martin/archives
/media/homebis/Documents_gr/Mots_croises/archives
/media/homebis/Documents_gr/archives
/media/homebis/Documents_gr/mes_fichiers/archives
```

Autre exemple un peu plus complexe cette fois.
Je désire faire une recherche de tous les fichiers audio de type .mp3 et .ogg

Il existe plus d'une façon d'y arriver.
Voyons comment on peut s'y prendre.

Première façon :

```
find /home/martin/  \( -name '*.mp3' -o -name '*.ogg' \)
```

On peut noter l'utilisation du -o qui correspond à l'opérateur ou (“or” en anglais)

Cela me donnera toute une liste de fichiers /home/martin/...

Deuxième façon :

Une autre manière d'écrire la commande ci-dessus est la suivante :

```
find -type f -name "*.mp3" -o -name "*.ogg"
```

Si je tape cette commande en étant dans mon répertoire /home/martin, le résultat sera une liste de fichiers ./....
Il est intéressant de savoir que l'on peut étendre la recherche aux fichiers mp3 et mp4 en remplaçant le 3 par un ?[2)]. La commande deviendrait donc :

```
find -type f -name "*.mp?" -o -name "*.ogg"
```

Rechercher pour supprimer

Une fonction intéressante de *find* est de supprimer en lot les fichiers trouvés.

Il n'est point rare de télécharger ou d'installer de nombreux fichiers qui ne nous servent plus, mais devant le travail pénible de devoir supprimer tous ces fichiers, on repousse au lendemain cette charge. Heureusement grâce à la fonction -delete de find, c'est un pur bonheur.

Paramètre -delete

Exemple, si dans votre home ou autre dossier vous avez beaucoup de fichier .tar.gz qui ne vous servent plus à rien. Il suffit de lancer la commande suivante :

```
find -iname "*.tar.gz" -delete
```

Attention, la fonction -delete ne vous demande pas de confirmation

Supprimer avec demande de confirmation

Pour une demande de confirmation avant suppression de chaque fichier “.tar.gz” trouvés :

```
find -iname "*.tar.gz" -ok rm {} \;
```

Merci à MicP pour cette trouvaille :)

Filtrer en fonction des droits

Une option très pratique est `-perm` qui permet de sélectionner des fichiers en fonctions de leurs droits.
Les droits peuvent être donné en forme octale, par exemple 0755 ou littérale, u=rwx,g=rw,o=rw.

Voici par exemple comment obtenir la liste de tout les fichiers dans le repertoire `/bin` qui ont le `bit setuid valant 1` :

```
find /bin -perm /5000 -user root
```

résultat de la commande précédente

```
/bin/su
/bin/mount
/bin/umount
/bin/ping
```

```
/bin/ping6
```

Cette option est intéressante pour la sécurité. Les fichiers listés dans la commande précédente sont tous exécuté avec les droits `root`.

Trois notations avec perm, sans préfixe, précédé du signe - ou précédé du signe /

- sans préfixe :

le mode du fichier doit être exactement celui passé à l'option -perm.
Par exemple, si on cherche les fichiers ayant le mode u=rwx (0700), tous les fichiers que l'on trouvera auront exactement le mode u=rwx (0700).

- avec le signe - :

le mode du fichier doit être au moins égal à celui passé à l'option -perm
-u=r (-0400) → u=r ou u=rw ou u=rx ou u=rwx ou u=r,g=x ...

- avec le signe / :

un des modes (user, group ou other) doit être au moins égal à ceux passés à l'option -perm
/u=w,g=w,o=w → u=w ou g=w ou o=w ou u=w,g=w,o=w ou u=rw,g=rwx ...

Recherche par motif

Pour rechercher un motif, il faut utiliser la même option, et utiliser les REGEXP.

Voici par exemple la recherche de tous les fichiers terminant par `.java` dans le dossier courant:

```
find . -name '*.java'
```

résultat de la commande précédente

```
./java/jdk1.5.0_06/demo/applets/Animator/Animator.java
./java/jdk1.5.0_06/demo/applets/ArcTest/ArcTest.java
./java/jdk1.5.0_06/demo/applets/BarChart/BarChart.java
./java/jdk1.5.0_06/demo/applets/Blink/Blink.java
./java/jdk1.5.0_06/demo/applets/CardTest/CardTest.java
...
```

Rechercher les fichiers n'appartenant pas à l'utilisateur

Il peut parfois être utile de rechercher les fichiers n'appartenant pas à l'utilisateur, en vue de corriger un problème rencontré avec une application (par exemple, un fichier peut appartenir à root au lieu d'appartenir à l'utilisateur ; ce dernier risque de ne pas avoir de droits dessus, ce que peut alors provoquer une erreur dans une application cherchant à modifier le dit fichier).

Pour ce faire, il suffit d'exécuter là commande suivante, où « utilisateur » est à remplacer par votre nom d'utilisateur :

```
find /home/utilisateur ! -user utilisateur
```

ou bien, en utilisant des variables :

```
find $HOME ! -user $USER
```

Pour avoir davantage d'informations sur les fichiers ainsi trouvés, vous pouvez ajouter l'option `ls` :

```
find $HOME ! -user $USER -ls
```

-exec - Exécuter une commande

La commande **find** permet d'effectuer toute sorte d'action avec les fichiers trouvés.

Une action très utile est "`-exec`" qui permet d'exécuter une commande sur les fichiers sélectionnés. La syntaxe de `exec` est particulière car il faut pouvoir fournir le nom du fichier trouvé.

À la suite de la commande find habituelle, la syntaxe est :

```
 find /chemin/du/fichier/ <option> <caractéristique fichier> -exec commande
{} \;
```

1. La paire d'accolade est automatiquement remplacée par le nom du fichier,
2. et le point-virgule final permet de marquer la fin de la commande.

Au cas où plusieurs fichiers sont traités dans un même répertoire, pour éviter une relance de la commande après chaque fichier trouvé, remplacer le `;` (point-virgule) final par le signe positif : +.

Par exemple ainsi :

```
find /home/mon_user/test/ -type f -exec echo {} \+
```

Si vous utilisez cette option, veillez bien à ce que votre variable d'environnement **$PATH** ne contienne pas une référence au répertoire courant « . », sinon un pirate pourrait lancer toutes les commandes qu'il souhaite en mettant un fichier au nom adéquat dans les répertoires où vous allez lancer un -execdir.
De la même manière, évitez les références vides ou les noms de répertoires exprimés

en relatif dans **$PATH**.

Cette commande est difficile à utiliser sur certains shell car ceux-ci donnent une signification particulière des caractères comme l'accolade ou le point-virgule.

Avec **Bash**, la *paire d'accolades sans espace* (`{}`) ne doit pas être protégée, au contraire du *point-virgule* qui doit être échappé à l'aide d'un backslash: `\;`.

Voici par exemple comment on peut compter le nombre de lignes de chaque fichier de code Python de ce site:

```
find developpement/django/certif -name '*.py' -exec wc -l {} \;
```

résultat de la commande précédente

```
  1 developpement/django/certif/__init__.py
  0 developpement/django/certif/acronym/__init__.py
 48 developpement/django/certif/acronym/models.py
 82 developpement/django/certif/acronym/tools.py
 13 developpement/django/certif/acronym/urls.py
 42 developpement/django/certif/acronym/views.py
.../...
```

1. Ici la commande **find** est utilisée avec l'option `-name` pour ne sélectionner que les fichiers se terminant par "`.py`" (extension de Python).
2. La commande "`wc`" (qui compte le nombre de ligne avec -l) est invoquée à l'aide "`-exec`" sur chacun de ces fichiers.

Il existe d'autres variantes de l'exécution de commande comme l'option `-execdir` qui exécute la commande à partir du répertoire du fichier.

Comme d'habitude vous avez aussi le :

```
man find
```

Entièrement disponible à votre curiosité ! 🙂

TP01 - exec et execdir

Préparation du terrain

Placez-vous dans un nouveau dossier créé pour l'occasion.
Exemple :

```
cd
```

```
mkdir -p ~/tmp/df_tp01/
```

```
cd ~/tmp/df_tp01/
```

On va maintenant créer les fichiers et dossiers qui nous servirons pour la suite ainsi :

```
for i in $(seq 5); do mkdir toctoc$i; done
```

```
mkdir -p toctoc1/paf toctoc2/paf toctoc3/pif/paf toctoc4/paf/paf
```

```
touch toctoc5/paf
```

```
for i in $(seq 4); do for j in toctoc*; do mkdir -p $j/tchac$RANDOM; done;
done
```

Quelques révisions de bash

À maîtriser absolument avant de poursuivre.

- Différents types de chemin — Notion de répertoire de travail - Pas à pas.

Find, -exec et -execdir

Maintenant que les notions de bases ont été précisées, vous devriez pouvoir suivre sans difficulté les différences entre `-exec` et `-execdir`.

Filtrer les fichiers

Si vous lancez un `find` depuis le second répertoire de ce TP, vous allez voir ce sacré bazar :

```
~/tmp/df_tp01$ find .
```

Par exemple, on va se concentrer sur les répertoires nommés `paf` :

```
~/tmp/df_tp01$ find . -type d -name paf
```

résultat de la commande précédente

```
./toctoc3/pif/paf
./toctoc1/paf
./toctoc4/paf
./toctoc4/paf/paf
./toctoc2/paf
```

Nota :

Pour afficher les sous-dossiers en premier, on ajoute l'option `-depth` **au début** de la commande

```
~/tmp/df_tp01$ find . -depth -type d -name paf
```

résultat de la commande précédente

```
./toctoc3/pif/paf
./toctoc1/paf
./toctoc4/paf/paf
./toctoc4/paf
./toctoc2/paf
```

Répertoires de travail pour -exec et -execdir

Dans quel environnement `-exec` lance-t'il les commandes ?

Affichons le répertoire de travail :

```
~/tmp/df_tp01$ find . -depth -type d -name paf -exec pwd \;
```

résultat de la commande précédente

```
/home/captnfab/tmp/df_tp01
/home/captnfab/tmp/df_tp01
/home/captnfab/tmp/df_tp01
/home/captnfab/tmp/df_tp01
/home/captnfab/tmp/df_tp01
```

On voit qu'ici, pour chaque dossier `paf` de trouvé, c'est seulement le chemin du répertoire d'où nous lançons notre commande find, qui est indiqué par l'exécution de la commande : pwd, ce qui n'est pas ce que nous voulons obtenir.

Employons `-execdir` maintenant :

```
~/tmp/df_tp01$ find . -depth -type d -name paf -execdir pwd \;
```

résultat de la commande précédente

```
/home/captnfab/tmp/df_tp01/toctoc3/pif
/home/captnfab/tmp/df_tp01/toctoc1
/home/captnfab/tmp/df_tp01/toctoc4/paf
/home/captnfab/tmp/df_tp01/toctoc4
/home/captnfab/tmp/df_tp01/toctoc2
```

On voit maintenant que nous obtenons bien le chemin de chaque répertoire contenant le dossier `paf` qui est indiqué par l'exécution de pwd, ce qui était recherché. 😀

Renommer les répertoires

Maintenant, comment renommer les répertoires `paf` en `ploum` ?

C'est on ne peut plus simple :

```
~/tmp/df_tp01$ find . -depth -type d -name paf -execdir mv paf ploum \;
```

Vous pouvez vérifier que ça a bien marché avec la commande suivante :

```
~/tmp/df_tp01$ find . -depth -type d -name ploum
```

Merci au **captnfab** *pour ces moments partagés joyeusement au bord de l'Ô !*

Site de ce matelot bienveillant d'où est totalement pompé cet exercice :

- http://wiki.chezlefab.net/tuto_nix/tp/find_01

Une seconde version plus scolaire du TP01

Préparation du terrain

Placez-vous dans un nouveau dossier créé pour l'occasion. Exemple :

```
cd
```

```
mkdir -p tmp/df_tp01/
```

```
cd tmp/df_tp01/
```

On va maintenant créer les fichiers et dossiers qui nous serviront pour la suite.

Voici pour la première partie :

```
touch phaute_dorthografe
```

```
touch notes_temporaires
```

```
~/tmp/df_tp01$ touch quisuisje
```

Et pour la seconde :

```
~/tmp/df_tp01$ for i in $(seq 5); do mkdir toctoc$i; done
```

```
~/tmp/df_tp01$ mkdir -p toctoc1/paf toctoc2/paf toctoc3/pif/paf
toctoc4/paf/paf
```

```
~/tmp/df_tp01$ touch toctoc5/paf
```

```
~/tmp/df_tp01$ for i in $(seq 4); do for j in toctoc*; do mkdir -p
$j/tchac$RANDOM; done; done
```

Quelques révisions de bash

Le répertoire courant, ou répertoire de travail

C'est ce qui est indiqué juste à gauche du $ du prompt bash, vous pouvez l'afficher en tapant la commande pwd.

```
~/tmp/df_tp01$ pwd
```

résultat de la commande précédente

```
/home/captnfab/tmp/df_tp01
~/tmp/df_tp01$
```

On change de répertoire de travail avec la commande cd.

Le répertoire de travail est également stocké dans la variable $PWD du shell.

```
~/tmp/df_tp01$ echo $PWD
```

résultat de la commande précédente

```
/home/captnfab/tmp/df_tp01
~/tmp/df_tp01$
```

Chemin relatif / chemin absolu

Pour identifier de manière unique un fichier sur un ordinateur sous Linux, on donne ce qu'on appelle un chemin vers le fichier. Ce chemin peut être *absolu* ou *relatif*.

Quelques chemins particuliers

- `..` désigne le répertoire parent d'un répertoire considéré
- `.` désigne le répertoire considéré

Exemples :

- `/home/../var` est un chemin pour le répertoire `/var`
- `ls .` liste le répertoire courant
- `../../../../.` est un chemin vers le dossier parent, équivalent à `..`

Chemin absolu

Un **chemin absolu** est un chemin depuis la racine du système de fichier, `/`

Exemples de chemin *absolu* :

- `/var/cache/apt/archives/` qui est le dossier contenant les paquets téléchargés pour installation ou mise à jour par `apt-get`, `aptitude`, `synaptic` etc.
- `/home/../var` est aussi un chemin absolu puisqu'il donne le chemin depuis `/`

Chemin relatif

Un **chemin relatif** est un chemin depuis le **répertoire de travail**, `$PWD`

Quelques chemins relatifs particuliers :

Exemple de chemin *relatif* :

- `../../` est un chemin relatif vers le dossier parent du dossier parent.

Si vous êtes toujours dans `~/tmp/df_tp01/`, la commande

```
ls ../..
```

doit vous lister votre *home*.

- `./plouf/../tchac/` est un chemin relatif vers le dossier `$PWD/tchac`

Attention : Virtuellement, un chemin relatif commence **toujours** par `./`. Cependant, en règle générale, on peut l'omettre. Exemple

```
ls -lh quisuisje
```

Signifie en fait :

```
ls -lh ./quisuisje
```

Renommer un fichier ou un dossier

Vous savez sans doute faire, mais merci de lire quand même cette partie pour fixer la terminologie :)

Lorsque l'on veut renomer un fichier, on utilise `mv`, qui prend deux arguments :

- un **chemin** vers le fichier à déplacer
- le nouveau **chemin** du fichier

Ce chemin peut être soit *absolu* soit *relatif*.

Exemples :

- Avec deux chemins relatifs

```
~/tmp/df_tp01/$ mv ./phaute_dorthografe ./faute_dorthographe
```

qui signifie donc

```
mv $PWD/./phaute_dorthografe $PWD/./faute_dorthographe
```

- Avec un chemin relatif et un chemin absolu

```
~/tmp/df_tp01/$ mv ./notes_temporaires /tmp/notes_temporaires.txt
```

qui signifie

```
~/tmp/df_tp01/$ mv $PWD/./notes_temporaires /tmp/notes_temporaires.txt
```

Find, -exec et -execdir

Maintenant que les notions de bases ont été introduites, vous devriez pouvoir comprendre sans difficulté les différences entre `-exec` et `-execdir`.

Filtrer les fichiers

Si vous lancez un `find` depuis le répertoire du tp, vous allez voir un sacré bazar :

```
find .
```

On va se concentrer sur les répertoires `paf` :

```
find . -type d -name paf
```

résultat de la commande précédente

```
./toctoc3/pif/paf
./toctoc1/paf
./toctoc4/paf
./toctoc4/paf/paf
./toctoc2/paf
```

Pour afficher les sous-dossiers en premier, on ajoute l'option `-depth` **au début** de la commande

```
find . -depth -type d -name paf
```

résultat de la commande précédente

```
./toctoc3/pif/paf
./toctoc1/paf
./toctoc4/paf/paf
./toctoc4/paf
./toctoc2/paf
```

Fastoche ! Merci **captnfab**

Répertoires de travail pour -exec et -execdir

Dans quel environnement est-ce que `-exec` lance les commandes ? Affichons le répertoire de travail.

```
find . -depth -type d -name paf -exec pwd \;
```

résultat de la commande précédente

```
/home/captnfab/tmp/df_tp01
/home/captnfab/tmp/df_tp01
/home/captnfab/tmp/df_tp01
/home/captnfab/tmp/df_tp01
/home/captnfab/tmp/df_tp01
```

Comparons avec `-execdir`

```
find . -depth -type d -name paf -execdir pwd \;
```

résultat de la commande précédente

```
/home/captnfab/tmp/df_tp01/toctoc3/pif
/home/captnfab/tmp/df_tp01/toctoc1
/home/captnfab/tmp/df_tp01/toctoc4/paf
/home/captnfab/tmp/df_tp01/toctoc4
/home/captnfab/tmp/df_tp01/toctoc2
```

On voit ici que le répertoire de travail est à chaque fois le répertoire contenant le dossier `pif`.

Copier un fichier dans plusieurs répertoires

1. Soit un fichier unique : **fichier1**
2. à copier avec rsync dans tous les sous-répertoires de /media/rep1/rep2/ dont le nom commence

par le chiffre 1

```
find /media/rep1/rep2/ -maxdepth 1 -type d -iname '1*' -exec rsync -avz
fichier1 "{}/" \;
```

retour de la commande find

```
sending incremental file list
fichier1

sent 518,342 bytes  received 35 bytes  345,584.67 bytes/sec
total size is 2,197,504  speedup is 4.24
sending incremental file list
fichier1
[…]
```

La commande peut évidemment être adaptée si le filtre sur le nom est différent (ou s'il n'y a pas de filtre sur le nom), ou si ce sont des sous-sous-répertoires, ou si ce sont des fichiers à remplacer, ou si la copie doit être faite avec `cp`, etc.

Renommer les répertoires

Maintenant, comment renommer les répertoires `paf` en `ploum` ?

C'est on ne peut plus simple :

```
find . -depth -type d -name paf -execdir mv paf ploum  \;
```

Vous pouvez vérifier que ça a bien marché avec la commande suivante :

```
find . -depth -type d -name ploum
```

Vous pouvez voir l'original là :

- Obsolète : wiki.chezlefab.net/tuto_nix/tp/find_01bis

Renommer une chaîne de caractère dans des fichiers

L'exercice est de renommer l'ensemble des require('truc muche') en : @require_once('truc muche') ; inscrit dans plusieurs fichiers différents.

On se place à la racine du répertoire cible :

```
cd /repertoire/machin
```

et on tape :

```
find . -depth -type f -exec sed -i 's/require(/@require_once(/g' {} \;
```

Et voilà l'travail !

Merci à **adrien** *sur le chan* #slackware-fr *et au* **captnfab** *not' matelot hottentot favori !* 😀

Effacer le contenu d'un fichier

Pour effacer le contenu d'un fichier

le chevron

Avec la commande chevron

```
find -type f -iname nom_du_fichier -execdir cat {} nom_du_fichier + > nom_du_fichier \;
```

xargs

Ou bien avec **xargs** :

```
find -type f -iname nom_du_fichier | xargs cat > nom_du_fichier \;
```

cp

Ou encore avec la commande cp :

```
find -type f -iname nom_du_fichier -exec cp /dev/null {} \;
```

Merci à **adrien** *du salon irc* #slackware-fr *pour son aide !*

SOURCE :

- http://www.linuxcertif.com/doc/keyword/find/

Et :

- http://www.linuxcertif.com/man/1/find/

Liens

En anglais :

- http://mywiki.wooledge.org/UsingFind?highlight=%28find%29

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

2)

Voir : regexp