

# La commande find

Parmi toutes les commandes Unix, **find** est sans aucun doute l'une des plus puissantes, des moins comprises et des plus effrayantes. Pourtant, une fois maitrisee, elle se revele extremement utile et c'est pourquoi nous allons nous pencher sur elle ce mois-ci. Le but de cet article n'est pas de paraphaser le man, mais d'illustrer a travers quelques exemples le fonctionnement et les possibilites de find.

Le find dont vous disposez sous Linux est celui de GNU qui differe des autres versions sur plusieurs points (notamment il se revele plus souple, comme d'habitude!). Nous allons parler ici de find en general et les exemples marchent sur tous les find (en principe). Parfois, certaines options seront inutiles avec le find de GNU, mais autant les apprendre, ca vous servira le jour ou vous vous retrouverez avec une autre version (genre 10000F HT par utilisateur et par poste, pourquoi pas?)

---

Comme son nom ne l'indique pas, le but de **find** n'est pas de chercher un fichier dans l'arborescence (bien qu'il puisse le faire!). Cette commande est en fait beaucoup plus generale et versatile que ca. Ce que fait **find**, c'est en realite de parcourir une arborescence et appliquer un traitement quelconque a certains fichiers. Il peut donc aussi bien imprimer votre arborescence que rechercher un fichier particulier ou compiler tous les \*.c sur votre disque qui appartiennent a l'utilisateur izard, dont la taille fait 342K et dont le chemin d'acces ne contient pas la lettre p.

Les options de la ligne de commande de **find** forment un veritable langage de programmation qui permet de controler tout ceci. Elles peuvent etre classees dans trois ensembles:

- Les actions: ces options declenchent une operation qui s'applique sur le nom de fichier que **find** est actuellement en train de lire
- Les conditions: elles definissent des clauses logiques sur les noms des fichiers. Lorsque ces conditions sont evaluees a vrai, les actions sont appliquees
- Les options: elles modifient le fonctionnement de find

Le man de find contient une liste exhaustive de toutes les commandes, je vous invite donc a le lire. On en verra egalement quelques unes dans nos exemples.

---

Commençons par le plus simple. Lorsqu'on appelle **find**, le premier paramètre doit être un nom de répertoire, les paramètres suivants sont les commandes sus-citées. Par exemple, la commande `-print` a pour effet d'afficher le nom du fichier en cours de traitement. Tapez donc:

```
find . -print
```

Le répertoire `.` est le répertoire courant, cette commande a donc pour effet de lister toute l'arborescence depuis le point où vous vous trouvez. Essayez également:

```
find / -print
```

et tout le contenu de vos périphériques montés défile devant vos yeux!

Maintenant, prenons un exemple pratique. Supposons que vous voulez chercher le fichier `XF86Config`. Nous allons utiliser pour cela la condition `-name` suivie d'une expression régulière qui ordonne à **find** de ne prendre en compte que les fichiers dont le nom est reconnu par cette expression. Dans notre cas, cela donne:

```
find / -name "XF86Config" -print
```

Observez les parenthèses autour de l'expression régulière: elles sont là pour empêcher le shell d'évaluer cette expression (il faut toujours se souvenir que sous Unix, les expressions régulières sont parsees par le shell et non par le programme appelé comme sous DOS, enfin on parlera de ça une autre fois).

Avec cet ordre, **find** cherchera le fichier dans tous les répertoires montés. Il est donc utile de connaître l'option `-xdev` qui empêche **find** de chercher ailleurs que dans la partition contenant le répertoire de départ.

---

Vous connaissez maintenant le principe de **find** et vous n'avez qu'à consulter le man pour connaître toutes les commandes, options et conditions. Toutefois, il y a une commande particulière dont nous allons encore parler: il s'agit de `-exec`. Cette commande permet de lancer pour chaque fichier un autre programme. C'est très puissant, mais il faut bien comprendre comment ça marche.

Lorsque **find** rencontre un `-exec`, il considère tout ce qui suit comme une ligne de commandes, jusqu'au caractère `;`. Exemple:

```
find / -name "*.rc" -exec echo "J'ai trouve" \;
```

Deux remarques à propos de cette commande: premièrement, j'attire votre attention sur le `\` qui précède le `;`. En effet, le shell considère le point-virgule comme un séparateur de commandes et sans le `\`, il tenterait d'exécuter `find / -name "*.rc" -exec echo "J'ai trouve"` (donc sans le `;` fermant `-exec`) suivi d'une commande vide (celle qui suit le `;`).

Deuxièmement, cette commande ne fait qu'afficher "J'ai trouve" un certain nombre de fois: en effet, à chaque fois qu'il rencontre un fichier `.rc`, **find** appelle la commande `echo "J'ai trouve"`.

Pour que tout ceci ait un intérêt, il faut pouvoir transmettre à la commande lancée par `-exec` le nom du fichier en cours d'évaluation. **find** le fait en remplaçant dans la commande appelée la chaîne de caractères `{ }` par le nom du fichier. En clair, pour copier tous les fichiers de votre

disque commençant par a sous /tmp, tapez par exemple:

```
find / -name "a*" -exec cp -f -r {} /tmp \; -print
```

Voilà, vous pouvez maintenant parcourir votre disque dans tous les sens et automatiser tout ce que vous voudrez. Avant de nous quitter, je vous suggère d'ouvrir le man de **find** et regarder toutes ses commandes, puis tenter de résoudre les deux petits exercices suivants:

1. Ecrivez la commande réalisant l'exemple donné en introduction (vous savez, compiler tous les \*.c tels que...)
2. Utilisez find pour effacer tous les fichiers de sauvegarde (ceux qui se terminent par ~ ou %)

Reponses:

1. `find / -name "*.c" -user izard -size 342k -regex "[^p]+" -exec gcc -c {} \;`
2. `find / -name "*[~%]" -exec rm -f {} \;`