Chapitre 3
find, la commande de recherche de fichiers

1. Description

La commande *find* permet la recherche de fichiers ou de répertoires dans une ou plusieurs arborescences. Les critères de recherche sont multiples : ils peuvent porter sur les attributs des fichiers, c'est-à-dire les informations stockées dans l'i-node associé au fichier, ou sur le nom du fichier.

La syntaxe de la commande *find* de la FSF (*Free Software Foundation*) disponible sur GNU/Linux a été considérablement enrichie et légèrement assouplie depuis la création de la commande : le nombre d'options de *find* a plus que triplé depuis UNIX version 7, et certains arguments sont devenus facultatifs. La commande permet maintenant l'utilisation d'*expressions régulières étendues*, portant sur le nom long, et plus seulement sur le nom court comme dans la commande d'origine.

Les fonctionnalités de *find* qui nous intéressent dans le cadre de cet ouvrage sont celles relatives à la recherche de noms de fichiers ou répertoires car elles utilisent les expressions régulières.

Nous allons toutefois effectuer un bref rappel du fonctionnement de cette commande et de ses principales options.

Remarque

Toutes les options et tous les tests présentés ici ne sont pas systématiquement disponibles dans l'ensemble des commandes find : leur existence dépend de la version du système ou du paquet utilisé. Veuillez vous référer au manuel de la commande pour connaître les options et tests dont vous disposez sur le système cible pour lequel vous écrivez les lignes de commandes.

2. Principe de fonctionnement

La commande *find* recherche dans une ou plusieurs arborescences données des fichiers ou répertoires ayant des caractéristiques particulières qui lui sont spécifiées au moyen d'options suivies d'arguments indiquant les valeurs recherchées. Elle peut effectuer des actions spécifiées, comme afficher le nom des fichiers trouvés ou exécuter une commande à laquelle *find* peut passer le nom du fichier trouvé.

Elle effectue un parcours récursif à partir des différents répertoires qui lui sont spécifiés, chacun étant considéré comme la racine d'une arborescence à partir de laquelle la commande *find* va commencer une recherche.

Pour chaque nom de fichier trouvé lors du parcours des arborescences, *find* évalue une expression booléenne constituée de critères de recherches ou d'actions. Les éléments de l'expression booléenne sont évalués de gauche à droite jusqu'à ce que son résultat soit connu.

Pour mémoire, en notant **||** le OU logique, et **&&** le ET logique, on connaît les résultats suivants de l'algèbre booléenne :

– (*VRAI* **||** *condition*) est vrai quelle que soit la valeur de vérité de *condition*,

– (*FAUX* **&&** *condition*) est faux quelle que soit la valeur de vérité de *condition*.

Par conséquent, dans ces deux cas, il est inutile d'évaluer l'expression *condition* pour connaître le résultat de l'expression booléenne.

Cette propriété est importante pour la compréhension du fonctionnement de *find*, en particulier dans le cas d'exécution d'actions, éventuellement multiples.

Les options de *find* peuvent être combinées sous forme d'une expression utilisant des OU (*-o*), éventuellement groupées avec des parenthèses, les ET entre les options étant implicites.

La valeur booléenne de chaque option peut être inversée en la faisant précéder d'un point d'exclamation (**!**), ce qui revient pour les tests à inverser leur signification.

Une option peut spécifier :

– un mode particulier de fonctionnement,
– un critère de recherche, utilisant un mot-clé pour désigner l'attribut concerné du fichier (au sens large), et une valeur correspondant à la valeur recherchée,
– une action à effectuer, intégrée à la commande *find*, ou externe et lancée via le système sous forme de processus.

Remarque

*Dans la terminologie UNIX, les termes **fichier au sens large** désignent un fichier régulier, un répertoire, un point d'accès aux périphériques de type bloc ou caractère, une FIFO, une socket, etc.*

Syntaxe

```
find liste_répertoires [-option ...]
```

Remarque

Les options historiques de la commande find ne sont pas les classiques options à une lettre de la majorité des commandes UNIX : ce sont des chaînes de plusieurs lettres précédées d'un tiret. Comme la commande dd, la commande find possède une syntaxe atypique par rapport aux autres commandes système.

Les options de cette commande étant relativement nombreuses, en particulier pour les versions récentes de *find*, nous n'en détaillerons que quelques-unes dans cet ouvrage : majoritairement celles d'UNIX v7 qui sont présentes dans toutes les versions de la commande *find*, en ce qui concerne la gestion des attributs de l'i-node, ainsi que celles qui sont utilisées dans les exemples et les exercices, et bien sûr toutes celles qui sont relatives aux expressions régulières.

3. Les options liées à l'i-node

Les options liées à l'i-node (c'est-à-dire au descripteur de fichier) permettent de spécifier des critères relatifs aux éléments de l'i-node du fichier, par exemple :

– le type,
– le mode,
– le nombre de liens,
– l'UID (*User ID*) du propriétaire,
– le GID (*Group ID*) du propriétaire,
– la taille,
– le numéro d'i-node,
– les dates (de modification, d'accès, de changement de l'i-node),
– etc.

Ces options sont des mots-clés précédés d'un unique tiret (*-type*, *-perm*, *-uid*, *-user*, *-size*, ...) suivies de la valeur recherchée. Dans le cas où la valeur en question est numérique, il est possible de la faire précéder d'un signe + ou d'un signe - pour indiquer que la valeur doit être respectivement supérieure ou inférieure (strictement) à la valeur qui suit.

3.1 Option -type

Cette option retourne *VRAI* si le type du fichier correspond au type spécifié par le caractère *c* qui suit l'option *-type*, avec les significations suivantes :

– **f** pour un fichier régulier.
– **d** pour un répertoire (directory).
– **b** pour un périphérique de type bloc.
– **c** pour un périphérique de type caractère.
– **p** pour une FIFO (pipe nommé).
– **l** pour un lien symbolique.
– **s** pour une socket (réseau).

Syntaxe

```
-type c
```

Exemple

```
find /dev -type p
```

Cette commande recherche sous le répertoire **/dev** tous les fichiers de type FIFO (**-type p**), et affiche leur nom long absolu puisque le nom du répertoire spécifié est un nom absolu. L'action par défaut est l'affichage du nom du fichier trouvé.

3.2 Option -perm

Cette option retourne *VRAI* si le mode du fichier correspond au mode d'accès spécifié en octal.

Syntaxe

```
-perm mode_octal
```

Exemple

```
find /bin /usr/bin /sbin /usr/sbin ! -type l -perm -02 -ls
```

Cette commande recherche sous **/bin**, **/usr/bin**, **/sbin** et **/usr/sbin** tous les fichiers qui ne sont pas des liens symboliques (**! -type l**), dont le mode est au moins 002 (**-perm -02**), c'est-à-dire permettant l'écriture à tout le monde, et les affiche au format de la commande *ls -dils* (**-ls**).

3.3 Option -links

Cette option retourne *VRAI* si le nombre de liens du fichier correspond au nombre *n* spécifié.

Syntaxe

```
-links n
```

Exemple

```
find /bin -type f -links +2 -ls
```

Cette commande recherche sous le répertoire **/bin** tous les fichiers réguliers (**-type f**) dont le nombre de liens est strictement supérieur à 2 (**-links +2**), et les affiche au format de la commande *ls*.

3.4 Option -user

Cette option retourne *VRAI* si le nom du propriétaire du fichier correspond au nom spécifié.

Syntaxe

```
-user nom_d'utilisateur
```

Exemple

```
find /home -type f -user nobody -ls
```

Cette commande recherche sous le répertoire **/home** tous les fichiers réguliers (**-type f**) appartenant à l'utilisateur nobody (**-user nobody**) et les affiche au format de la commande *ls -dils* (**-ls**).

3.5 Option -uid

Cette option retourne *VRAI* si l'identité numérique du propriétaire du fichier correspond au nombre *n* spécifié.

Syntaxe

```
-uid n
```

Exemple

```
find /dev ! -uid 0 -ls
```

Cette commande recherche sous le répertoire **/dev** tous les fichiers (au sens large) dont le propriétaire a une UID différente de 0 (**! -uid 0**) et les affiche au format *ls -dils* (**-ls**).

3.6 Option -group

Cette option retourne *VRAI* si le nom du groupe propriétaire du fichier correspond au nom spécifié.

Syntaxe

```
-group nom_de_groupe
```

Exemple

```
find /etc -group shadow -ls
```

Cette commande recherche sur **/etc** tous les fichiers (au sens large) dont le groupe propriétaire est shadow (**-group shadow**) et les affiche au format *ls -dils* (**-ls**).

3.7 Option -size

Cette option retourne *VRAI* si la taille du fichier correspond au nombre *n* de blocs spécifié. Le nombre peut être suivi d'un suffixe indiquant l'unité choisie pour exprimer la taille, comme **b** (blocs de 512 octets), **c** (caractères), **w** (pour words : mots de 2 octets), **k** (kilo-octets : 1024 octets), **M** (méga-octets : 1024 * 1024 = 1048576 octets), **G** (giga-octets = 1024 * 1024 * 1024 = 1073741824 octets), etc.

Syntaxe

```
-size n
```

Exemple

```
find /bin -size +1M -ls
```

Cette commande recherche sous **/bin** tous les fichiers (au sens large) dont la taille est strictement supérieure à 1 Mo (**-size +1M**) et les affiche au format *ls -dils* (**-ls**).

3.8 Option -inum

Cette option retourne *VRAI* si le numéro d'i-node du fichier correspond au nombre spécifié.

Syntaxe

```
-inum n
```

Exemple

```
find / -inum 3 -ls
```

Cette commande recherche à partir de la racine du système de fichiers (/) tous les fichiers (au sens large) dont le numéro d'i-node a la valeur 3 (**-inum 3**) et les affiche au format *ls -dils* (c'est-à-dire avec le numéro d'i-node).

3.9 Option -atime

Cette option retourne *VRAI* si la date d'accès du fichier correspond à la date courante diminuée de *n* jours (*n* fois 24h), c'est-à-dire si le fichier a été accédé il y a *n* jours.

Syntaxe

```
-atime n
```

Exemple

```
find /bin -type f -atime 0 -print | xargs ls -ltur
```

Cette commande recherche sous **/bin** tous les fichiers réguliers (**-type f**) dont la date d'accès est inférieure de moins de 24h à la date courante (**-atime 0**) puis passe leur nom (**-print**) à la commande **xargs** qui les affiche au format de la commande **ls -lutr**.

Remarque

L'intérêt d'envoyer le nom des fichiers trouvés à la commande xargs via un tube de communication est d'avoir des colonnes correctement alignées grâce à l'exécution d'un unique processus ls -lutr.

3.10 Option -mtime

Cette option retourne *VRAI* si la date de modification du fichier correspond à la date courante diminuée de *n* jours (*n* fois 24h), c'est-à-dire si le fichier a été modifié il y a *n* jours.

Syntaxe

```
-mtime n
```

Exemple

```
find /bin -mtime -60 -ls
```

Cette commande recherche sous **/bin** tous les fichiers au sens large dont la date de modification est strictement inférieure de moins de 60 jours à la date courante (**-mtime -60**) et les affiche au format de la commande *ls*.

3.11 Option -exec

Cette option retourne *VRAI* si la commande exécutée retourne un code retour (exit code) valant 0. La fin de la commande doit impérativement être matérialisée par un **;** indiquant la fin de la commande, et donc de celle des arguments de l'option *-exec*. Parmi les arguments de cette commande, il est possible de spécifier le nom du fichier trouvé en utilisant des accolades ouvrantes et fermantes **{}**. Le nom de la commande à exécuter et ses arguments doivent être passés sous forme de chaînes indépendantes les unes des autres, et non sous forme de chaînes concaténées : *find* ne fait pas d'analyse lexicale de la commande avant d'exécuter le programme spécifié.

Syntaxe

```
-exec commande [arguments] ;
```

Remarque

*Ainsi que cela a été spécifié dans le chapitre relatif aux shells, le caractère **;** étant un métacaractère, il est indispensable qu'il soit privé de sa signification de séparateur de commande shell, soit en le faisant précéder d'un antislash, soit en l'encadrant entre des quotes simples ou doubles.*