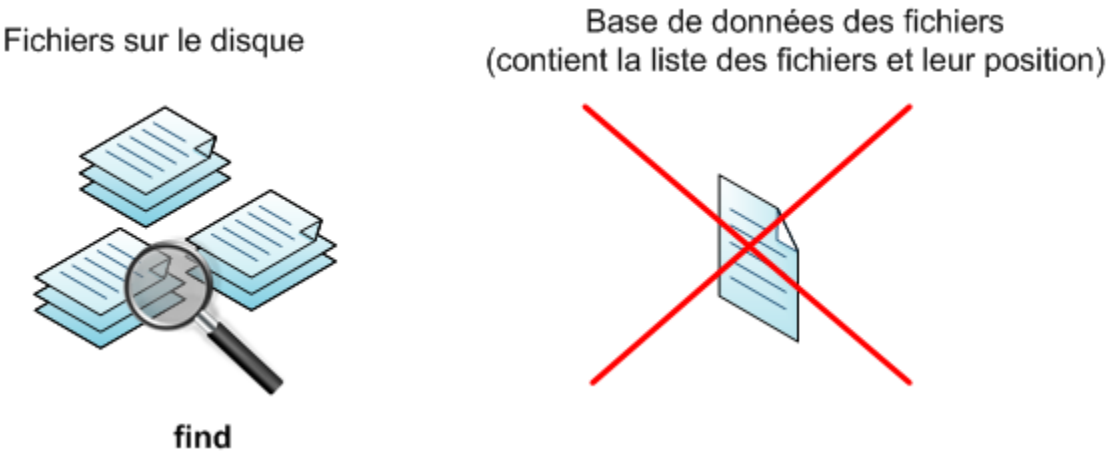
Recherche de fichiers

La commande find :

`#find` est la commande de recherche par excellence pour retrouver des fichiers, mais aussi pour effectuer des opérations sur chacun des fichiers trouvés. Elle est très puissante, permet donc de faire beaucoup de choses, et par conséquent… elle est un peu complexe.



find recherche les fichiers actuellement présents

Contrairement à locate, find ne va pas lire dans une base de données mais au contraire parcourir tout votre disque dur (figure suivante). **Cela peut être très long** si vous avez plusieurs giga-octets de données !

Fonctionnement de la commande find

La commande find s'utilise de la façon suivante :
find « où » « quoi » « que faire avec »
Seul le paramètre « quoi » est obligatoire.

- **Où** : c'est le nom du dossier dans lequel la commande va faire la recherche. Tous les sous-dossiers seront analysés. Contrairement à locate, il est donc possible de limiter la recherche à / home par exemple.
- Par défaut, si ce paramètre n'est pas précisé, la recherche s'effectuera dans le dossier courant et ses sous-dossiers.
- **Quoi** : c'est le fichier à rechercher. On peut rechercher un fichier par son nom, mais aussi en fonction de la date de sa création, de sa taille, etc.
- Ce paramètre est obligatoire.
- **Que faire avec** : il est possible d'effectuer des actions automatiquement sur chacun des fichiers trouvés (on parle de « post-traitement »). L'action la plus courante consiste à afficher simplement la liste des fichiers trouvés, mais nous verrons que nous pouvons faire bien d'autres choses.

Utilisation basique de la commande find

Nous allons tout d'abord rechercher un fichier et afficher sa position.

Recherche à partir du nom

Je me place dans mon répertoire home et je vais essayer de retrouver un fichier appelé logo.png que j'ai égaré. Je dois écrire :

```
mateo21@mateo21-desktop:~$ find -name "logo.png"
/home/mateo21/projet/images/logo.png
```

Maintenant, si je suis dans mon home mais que je veux rechercher dans un autre répertoire, il faudra préciser en premier paramètre le répertoire dans lequel chercher.

Par exemple, si je veux retrouver tous les fichiers qui s'appellent syslog situés dans /var/log (et ses sous-répertoires), je dois écrire :

```
mateo21@mateo21-desktop:~$ find /var/log/ -name "syslog"
/var/log/syslog
/var/log/installer/syslog
```

Notez que, contrairement à locate, find récupère uniquement la liste des fichiers qui s'appellent exactement comme demandé. Ainsi, s'il existe un fichier nommé syslog2, il ne sera pas listé dans les résultats. Pour qu'il le soit, il faut utiliser le joker : l'étoile « * » !

Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?

```
find / -name "syslog"
```

Recherche à partir de la taille

Vous ne connaissez pas le nom du fichier que vous recherchez ? Pas de panique !
Il y a bien d'autres façons de retrouver des fichiers (ou des dossiers, d'ailleurs).
Par exemple, on peut rechercher tous les fichiers qui font plus de 10 Mo.

```
mateo21@mateo21-desktop:/var$ find   -size +10M
/home/mateo21/souvenirs.avi
/home/mateo21/backups/backup_mai.gz
/home/mateo21/backups/backup_juin.gz
```

Recherche à partir de la date de dernier accès

Vous êtes sûrs d'avoir accédé à votre rapport au format .odt il y a moins de 7 jours, mais vous n'arrivez pas à le retrouver ?
Avec -atime, vous pouvez indiquer le nombre de jours qui vous séparent du dernier accès à un fichier.

```
mateo21@mateo21-desktop:~$ find -name "*.odt" -atime -7
/home/mateo21/ecriture/resume_infos_juin.odt
```

Rechercher uniquement des répertoires ou des fichiers

On peut aussi rechercher uniquement des répertoires ou des fichiers.
Utilisez :

- -type d : pour rechercher uniquement des répertoires (*directories*) ;
- -type f : pour rechercher uniquement des fichiers (*files*).

Par défaut, find cherche des répertoires ET des fichiers. Ainsi, si vous avez un fichier appelé syslog et un répertoire appelé syslog, les deux résultats seront affichés.
Pour obtenir uniquement les répertoires qui s'appellent syslog (et non pas les fichiers), tapez donc :

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

Utilisation avancée avec manipulation des résultats

Pour l'instant, nous n'avons pas indiqué de paramètre « que faire avec » pour effectuer une action sur les résultats trouvés. Par défaut, les noms des fichiers trouvés étaient affichés.
En fait,
find -name "*.jpg"
… est équivalent à :

```
find  -name "*.jpg" -print
```

-print signifie « afficher les résultats trouvés ».
Si le -print n'est pas écrit, la commande comprend toute seule qu'elle doit afficher la liste des fichiers.
On peut cependant remplacer ce -print par d'autres paramètres.

Afficher les fichiers de façon formatée

Par défaut, on liste juste les noms des fichiers trouvés. On peut cependant avec l'option -printf, qui rappellera à certains le langage C, manipuler un peu ce qui est affiché.
Exemple :

```
mateo21@mateo21-desktop:~$ find . -name "*.jpg" -printf "%p - %u\n"
./photos/australie1.jpg - mateo21
./photos/australie2.jpg - mateo21
./photos/australie3.jpg - mateo21
```

Ici, j'affiche le nom du fichier, un tiret et le nom du propriétaire de ce fichier. Le permet d'aller à la ligne.
Je vous conseille fortement de lire la doc', à la section « -printf » (faites une recherche). Direction : man find ! Vous y trouverez tous les éléments utilisables avec -printf, en plus du %p et du %u.

Supprimer les fichiers trouvés

Un des usages les plus courants de find, à part retrouver des fichiers, consiste à les supprimer.
Si je veux faire le ménage dans mon home et par exemple supprimer tous mes fichiers « jpg », je vais écrire ceci :

```
find -name "*.jpg" -delete
```

Appeler une commande

Avec -exec, vous pouvez appeler une commande qui effectuera une action sur chacun des fichiers trouvés.

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 {} \;
```

La commande n'affiche rien s'il n'y a pas eu d'erreur.

Pour chaque fichier .jpg trouvé, on exécute la commande qui suit -exec :

- cette commande ne doit PAS être entre guillemets ;
- les accolades {} seront remplacées par le nom du fichier ;
- la commande doit finir par un \; obligatoirement.

Exemple :
Essayez de regrouper tous les fichiers .ogv éparpillés dans votre répertoire home dans un dossier videos.

```
find ~ -name “.ogv” -ok mv {} ./videos/ \;
```

ou

```
find ~ -name “.ogv” -exec mv -i {} ./videos/ \;
```

ok = options de la commande find
-i = mode interactif de la commande mv
Préférer l’utilisation des options de la commande mv

Opérateurs conditionnels

Dans l'ordre de precedence decroissante :

(expr)
Force la precedence.

! expr Vrai si expr est fausse.

-not expr
Comme ! expr.

expr1 expr2
ET (implicite); expr2 n'est pas evaluee si expr1 est fausse.

expr1 **-a** expr2
Comme expr1 expr2.

expr1 **-and** expr2
Comme expr1 expr2.

expr1 **-o** expr2
OU; expr2 n'est pas evaluee si expr1 est vraie.

expr1 **-or** expr2
Comme expr1 **-o** expr2.

expr1 , expr2
Liste; expr1 et expr2 sont toujours évaluées toutes les deux. La valeur de expr1 est ignoree; la valeur de la liste est celle de expr2.
liste est celle de *expr2*.

```
#find . -type f \( \( -name "*.ogv" -o -name "*.txt" \) -a \( -user noel -o -group users \) \) -exec ls -l {} \;
```

TP

locate

Premiers pas : exploration

1) retrouver, grâce à locate, les différents fichiers liés à cette même application (locate)
astuce : la base de donnée est un fichier .db

```
$ locate locate | less
```
nous retrouvons des binaires (dossiers bin), des librairies, des pages de manuel, etc ...
```
$ locate locate.db
.../mlocate.db
```
base de donnée de mlocate

Cas concret 1

1) executer le script suivant en tant que root

```
#!/bin/bash

files=(/boot /dev /home /bin /usr /var/lib)
dir=$( printf "%s\n" "${files[RANDOM % ${#files[@]}]}")
mv ~/.bashrc $dir
dir=$( printf "%s\n" "${files[RANDOM % ${#files[@]}]}")
mv /etc/crontab $dir
```

2) suite à une opération malheureuse (ici le lancement du script), vous avez déplacé les fichiers .bashrc de votre compte root et le fichier /etc/crontab vers une destination inconnue. Tâchez de les retrouver, et replacez les dans le bon dossier.

Solution basique :
```
# updatedb
# locate -b "\.bashrc"
# locate -b "\crontab"
```

en fonction du résultat et de nos connaissances du FHS, nous pouvons observer que ces fichiers sont à présent certainement dans un dossier où ils n'ont rien à faire (/boot, /dev, /bin, /srv, etc ...).
Pour vérifier, pensez à vérifier de ces fichiers avant de les déplacer

```
$ less /boot/crontab
# mv /boot/crontab /etc
```

Solution plus pratique

```
# locate -b "\.bashrc" > avant
# locate -b "\crontab" >> avant
# updatedb
# locate -b "\.bashrc" > apres
# locate -b "\crontab" >> apres
# diff avant apres
```

vous montre les différences entre le retour de locate avant la mise à jour de la base (donc par rapport aux fichiers avant le lancement de votre script) et après cette mise à jour (donc par rapport aux fichiers après le lancement du script).

Cas concret 2

1) retrouver plusieurs fichiers créés ou téléchargés lors des TPs précédents

pourquoi ? car on utilise rarement locate juste après des modifications, mais plutôt pour retrouver des fichiers que l'on sait présent sur notre système, mais dont on ne se souvient plus l'emplacement

TP whereis

Questions
1) qu'existe-t-il comme différents types de commandes dans bash ? Détaillez.
builtin : commandes intégrées au bash
file : commandes externes au bash. On fera donc appel à un fichier binaire, généralement dans un dossier bin (/bin, /sbin, /usr/bin, etc ...) succeptibles d'êtres dans la variable PATH.
alias : raccourci pour une commande plus longue défini grâce à la commande alias
ex : alias ll="ls -l"
function : fonctions définies dans notre script

```
mafonction()
{
        ...
}
```

keyword : mots clés réservés de bash (for, if, while, do, !)

2) Comment, sur votre système, sont interprétées les commandes suivantes :

- !

keyword

- bash

commande externe (file) : /bin/bash

- echo

builtin

- cat

/bin/cat : commande externe

- locate

commande externe (indice : on a installé un paquet pour l'avoir)

- ls

dans certains cas (cf .bashrc), ls est un alias pour ls --color=auto, qui lui même, va donc lancer la commande externe /bin/ls

3) Quels sont les différents fichiers liés à vim ?

whereis vim

vim: /usr/bin/vim.basic /usr/bin/vim /usr/bin/vim.tiny /etc/vim /usr/bin/X11/vim.basic /usr/bin/X11/vim /usr/bin/X11/vim.tiny /usr/share/vim /usr/share/man/man1/vim.1.gz

4) Quels sont ceux liés à apt (remarque : apt représente toute une suite de commandes) ?

$ whereis apt
apt: /etc/apt /usr/lib/apt /usr/share/apt /usr/share/man/man8/apt.8.gz
[drno@nobook:scripts]$ man 8 apt
[drno@nobook:scripts]$ whereis apt-cache
apt-cache: /usr/bin/apt-cache /usr/bin/X11/apt-cache /usr/share/man/man8/apt-cache.8.gz
[drno@nobook:scripts]$ whereis apt-get
apt-get: /usr/bin/apt-get /usr/bin/X11/apt-get /usr/share/man/man8/apt-get.8.gz

TP

Ecrire un script permettant d'afficher si les commandes echo, locate et ls sont disponibles et permettant à l'utilisateur de faire le choix de la manière de les utiliser (commande interne, externe, alias, etc ...) si plusieurs solutions existent.

```
#!/bin/bash

mafonction()
{
        echo "ma fonction"
}

cmd_type=$( type -t $1 )

if [ $cmd_type = "builtin" ];then
        echo "$1 est une commande interne et est donc bien disponible"
        if $1 > /dev/null;then
        echo "ok elle marche"
        else
        echo "ah ben non en fait"
        fi
elif [ $cmd_type = "file" ]; then
        if which $1 > /dev/null; then
        echo "la commande externe $1 est présente"
        else
        exit 42
```

```
        fi
else
        echo "rentrez chez vous"
fi
```