RÉFÉRENCE RAPIDE FIND

Recherche de fichiers par nom, type, taille, date, permissions et actions

Recherche de base

Utiliser find

```
find .                              # list all files recursively
find /var/log                       # search from specific path
find . -name "*.txt"                # find by name
find / -name "config" 2>/dev/null   # suppress permission errors
find dir1 dir2 -name "*.py"         # search multiple directories
```

Syntaxe générale

`find [path...] [tests] [actions]`	Forme générale — chemins, puis tests, puis actions
`path`	Répertoire de départ (défaut : répertoire courant)
`test`	Condition pour filtrer les fichiers (`-name`, `-type`, etc.)
`action`	Ce qu'il faut faire avec les correspondances (`-print`, `-exec`, etc.)
`Action par défaut`	`-print` si aucune action n'est spécifiée

Patterns de noms

Correspondance par nom

```
find . -name "*.log"                # case-sensitive glob
find . -iname "readme*"             # case-insensitive glob
find . -name "*.py" -o -name "*.js"  # OR: Python or JS files
find . -path "*/src/*.ts"           # match against full path
find . -regex '.*\.\(py\|js\)'    # POSIX regex on full path
```

Tests de nom

`-name pattern`	Le nom de base correspond au glob shell (sensible à la casse)
`-iname pattern`	Le nom de base correspond au glob (insensible à la casse)
`-path pattern`	Le chemin complet correspond au glob shell
`-ipath pattern`	Le chemin complet correspond au glob (insensible à la casse)
`-regex pattern`	Le chemin complet correspond à l'expression régulière
`-iregex pattern`	Le chemin complet correspond à la regex (insensible à la casse)

Filtres de type

Filtrer par type

```
find . -type f                      # regular files only
find . -type d                      # directories only
find . -type l                      # symbolic links
find . -type f -name "*.sh"         # combine type + name
```

Types de fichiers

`-type f`	Fichier ordinaire
`-type d`	Répertoire
`-type l`	Lien symbolique
`-type b`	Périphérique bloc
`-type c`	Périphérique caractère
`-type p`	Tube nommé (FIFO)
`-type s`	Socket
`-empty`	Fichier ou répertoire vide

Taille et date

Exemples de taille et date

```
find . -size +100M                  # larger than 100 MB
find . -size -1k                    # smaller than 1 KB
find . -mtime -7                    # modified in last 7 days
find . -mmin -30                    # modified in last 30 minutes
find . -newer reference.txt         # newer than reference file
```

Tests de taille et date

`-size +/-Nc`	Taille en octets (c), kilooctets (k), mégaoctets (M), gigaoctets (G)
`-mtime +/-N`	Modifié il y a N*24 heures (+plus ancien, -plus récent)
`-atime +/-N`	Accédé il y a N*24 heures
`-ctime +/-N`	Statut changé il y a N*24 heures
`-mmin +/-N`	Modifié il y a N minutes
`-newer file`	Modifié plus récemment que file
`-newermt date`	Modifié après la chaîne de date (GNU)

Permissions

Exemples de permissions

```
find . -perm 644                    # exact permissions: rw-r--r--
find . -perm -u+x                   # user has execute bit set
find . -perm /o+w                   # others have write (any match)
find . -user root                   # owned by root
find . -group www-data -type f      # owned by group
```

Tests de permissions

`-perm mode`	Correspondance exacte des permissions
`-perm -mode`	Tous les bits spécifiés sont définis
`-perm /mode`	Au moins un des bits spécifiés est défini
`-user name`	Appartenant à l'utilisateur (nom ou UID)
`-group name`	Appartenant au groupe (nom ou GID)
`-nouser`	Aucun utilisateur correspondant dans /etc/passwd
`-nogroup`	Aucun groupe correspondant dans /etc/group

Actions

Exemples d'actions

```
find . -name "*.log" -print         # print paths (default)
find . -name "*.tmp" -delete        # delete matching files
find . -type f -ls                  # detailed listing
find . -name "*.txt" -print0        # null-delimited output
find . -type f -printf "%p %s\n"    # custom format (GNU)
```

Référence des actions

`-print`	Afficher le chemin (délimité par des sauts de ligne)
`-print0`	Afficher le chemin (délimité par des nulls, sûr pour xargs)
`-ls`	Afficher les détails du fichier (comme `ls -dils`)
`-delete`	Supprimer les fichiers correspondants (implique `-depth`)
`-printf format`	Format de sortie personnalisé (GNU) : `%p` chemin, `%s` taille, `%t` date
`-fprint file`	Écrire les chemins dans un fichier
`-quit`	Quitter après la première correspondance

Combiner les tests

Opérateurs logiques

```
find . -name "*.py" -type f         # implicit AND
find . -name "*.py" -a -size +10k   # explicit AND
find . -name "*.py" -o -name "*.js" # OR
find . ! -name "*.log"              # NOT
find . \( -name "*.py" -o -name "*.js" \) -type f
```

Référence des opérateurs

`expr1 expr2 / expr1 -a expr2`	ET — les deux doivent être vrais (défaut)
`expr1 -o expr2`	OU — l'un ou l'autre doit être vrai
`! expr / -not expr`	NON — nier l'expression
`\( expr \)`	Grouper les expressions (échapper les parenthèses dans le shell)
`Ordre d'évaluation`	De gauche à droite ; `-a` lie plus fort que `-o`

Exec et suppression

Exemples avec exec

```
find . -name "*.sh" -exec chmod +x {} \;
find . -name "*.log" -exec rm {} +
find . -type f -exec grep -l "TODO" {} +
find . -name "*.bak" -ok rm {} \;   # prompt before each
find . -name "*.tmp" -print0 | xargs -0 rm
```

Référence exec

`-exec cmd {} \;`	Exécuter cmd une fois par fichier (`{}` = chemin du fichier)
`-exec cmd {} +`	Exécuter cmd avec plusieurs fichiers à la fois (plus rapide)
`-ok cmd {} \;`	Comme `-exec` mais demande confirmation
`-execdir cmd {} \;`	Exécuter cmd depuis le répertoire du fichier
`xargs -0`	Associer à `-print0` pour un traitement par lots sécurisé
`-delete`	Supprimer les fichiers ; traiter les plus profonds en premier

Profondeur et élagage

Exemples de profondeur et élagage

```
find . -maxdepth 1 -type f          # current dir only
find . -mindepth 2 -name "*.py"     # skip top-level
find . -name ".git" -prune -o -print # skip .git dirs
find . -depth -name "*.tmp" -delete  # process children first
```

Options de profondeur

`-maxdepth N`	Descendre au plus N niveaux (0 = chemin de départ seulement)
`-mindepth N`	Ne pas appliquer les tests aux niveaux inférieurs à N
`-depth`	Traiter le contenu du répertoire avant le répertoire lui-même
`-prune`	Ne pas descendre dans le répertoire correspondant
`-mount / -xdev`	Ne pas traverser les frontières du système de fichiers
`-follow / -L`	Suivre les liens symboliques

Patterns courants

Commandes en une ligne

```
find . -name "*.pyc" -delete            # clean Python bytecode
find . -type f -size 0 -delete          # remove empty files
find . -mtime +30 -name "*.log" -delete # purge old logs
find . -type f -name "*.md" | wc -l     # count Markdown files
find . -type d -empty -delete           # remove empty dirs
```

Recettes

`Trouver les fichiers les plus grands`	`find . -type f -printf '%s %p\n' | sort -rn | head`
`Trouver les doublons par nom`	`find . -type f | awk -F/ '{print $NF}' | sort | uniq -d`
`Renommer l'extension`	`find . -name '*.txt' -exec rename 's/.txt/.md/' {} +`
`Trouver les liens symboliques cassés`	`find . -xtype l`
`Archiver les fichiers récents`	`find . -mtime -7 -print0 | tar czf recent.tar.gz --null -T -`
`Chercher dans les fichiers de code`	`find . -name '*.py' -exec grep -l 'pattern' {} +`