

Mémo Linux

v 3.2



Mémo Linux de [Dr Michaël GUEDJ](#) est mis à disposition selon les termes de la [licence Creative Commons Attribution 4.0 International](#).
Fondé(e) sur une œuvre à https://github.com/michaelguedj/ens_scripts_systemes.

Commandes Linux

Commandes de bases

<code>mkdir toto</code>	→ créer le répertoire <code>toto</code>
<code>cd toto</code>	→ entre dans le répertoire <code>toto</code>
<code>cd ..</code>	→ entre dans le répertoire parant
<code>cd ~</code>	→ entre dans le répertoire d'accueil
<code>touch a b c</code>	→ créer les fichiers <code>a</code> , <code>b</code> et <code>c</code>
<code>echo "blabla" > toto</code>	→ créer le fichier <code>toto</code> (s'il n'existe pas) contenant le texte <code>blabla</code>
<code>ls</code>	→ affiche le contenu du répertoire courant
<code>ls *.py</code>	→ idem mais n'affiche que les fichiers d'extension <code>.py</code>
<code>ls -a</code>	→ idem que <code>ls</code> + affiche les éléments caches
<code>ls -l</code>	→ idem que <code>ls</code> + affiche les droits : propriétaire + groupe + autre
<code>ls toto/</code>	→ affiche le contenu de <code>toto/</code>
<code>rmdir toto/</code>	→ efface <code>toto/</code> s'il est vide
<code>rm -r toto/</code>	→ efface le répertoire <code>toto/</code>
<code>rm -r *</code>	→ efface le répertoire courant
<code>rm a.txt</code>	→ efface le fichier <code>a.txt</code>
<code>cp a.txt b.txt</code>	→ copie <code>a.txt</code> sous le nom <code>b.txt</code>
<code>cp -r a/ b/</code>	→ copie <code>a/</code> sous le nom <code>b/</code>
<code>mv a.txt b/</code>	→ déplace <code>a.txt</code> dans <code>b/</code>
<code>mv a/ b/</code>	→ déplace <code>a/</code> dans <code>b/</code> (on suppose que <code>b/</code> existe)
<code>mv a.txt b.txt</code>	→ renomme <code>a.txt</code> en <code>b.txt</code>
<code>mv a/ b/</code>	→ renomme <code>a/</code> en <code>b/</code> (ici on suppose que <code>b/</code> n'existe pas)
<code>cat a.txt</code>	→ affiche le contenu du fichier <code>a.txt</code>
<code>tree</code>	→ affiche l'arborescence de racine le répertoire courant (c'est-à-dire le répertoire <code>"."</code>)
<code>tree toto/</code>	→ affiche l'arborescence de racine <code>toto/</code>
<code>vi toto.txt</code>	→ édite le fichier <code>toto.txt</code> avec <code>vi</code>

Gestion des paquets

`apt-get update` → met à jour la liste des paquets disponibles à partir des sources du fichier `/etc/apt/sources.list`
`apt-get upgrade` → remplace chaque paquet installé par la dernière version disponible
`apt-get dist-upgrade` → remplace chaque paquet installé par la dernière version disponible, installe les paquets supplémentaires nécessaires et supprime les paquets devenus inutiles
`dpkg --status tree` → le paquet `tree` est-il installé ?
`apt-cache search web browser` → recherche d'un navigateur
`apt-cache search tree` → recherche du paquet `tree`
`apt-cache search tree | grep tree` → idem mais n'affiche que les lignes contenant le mot « tree »
`apt-cache search tree | grep ^tree` → idem mais n'affiche que les lignes commençant (^) par le mot « tree »
`apt-cache search tree | grep tree$` → idem mais n'affiche que les lignes terminant (\$) par le mot « tree »

Unix tools

grep

`grep "mot" toto.txt` → affiche les lignes contenant « mot » dans `toto.txt`
`grep "^mot" toto.txt` → affiche les lignes commençant par « mot »
`grep "mot$" toto.txt` → affiche les lignes terminant par « mot »
`grep "mot1\|mot2" toto.txt` → affiche les lignes contenant « mot1 » ou « mot2 »
`grep "mot[12]" toto.txt` → idem que précédemment
`grep "[1-8]" toto.txt` → affiche les lignes contenant un nombre compris entre 0 et 8
`grep [a-zA-Z] toto.txt` → affiche les lignes contenant un caractère alphabétique
`grep -r "mot" dossier/` → affiche les lignes contenant « mot » dans l'arborescence partant de `dossier/`

diverses options de `grep` :

- i → ne pas tenir compte de la casse (majuscules / minuscules)
- n → connaître les numéros des lignes
- v → inverser la recherche : ignorer un mot

find

`find dossier/ -name "toto"`
 → recherche dans l'arborescence `dossier/` les fichiers et répertoires portant le nom « toto »

`find dossier/ -name "toto*"`
 → idem mais le nom à rechercher est « toto » suivi de "n'importe quoi"

```
find dossier/ -name "*toto*"
```

→ idem mais le nom à rechercher est "n'importe quoi" suivit de « toto »
suivit de "n'importe quoi"

```
find dossier/ -name "toto????"
```

→ idem mais le nom à rechercher est « toto » suivit de 4 caractères
(?=1 caractère quelconque)

```
find dossier/ -iname "toto"
```

→ recherche dans l'arborescence dossier/ les fichiers et répertoires portant
le nom « toto » sans tenir compte de la casse

wc

```
wc -l toto.txt
```

→ nombre de lignes de toto.txt

```
wc -w toto.txt
```

→ nombre de mots de toto.txt

```
wc -m toto.txt
```

→ nombre de caractères de toto.txt

sed

```
sed s/bonjour/bonsoir/ toto.txt
```

→ substitue la première occurrence de « bonjour » par « bonsoir »
pour toutes les lignes de toto.txt

```
sed s/bonjour/bonsoir/g toto.txt
```

→ substitue toutes les occurrences de « bonjour » par « bonsoir »
pour toutes les lignes de toto.txt

```
sed -i s/bonjour/bonsoir/ toto.txt
```

→ l'option « -i » permet d'effectuer la substitution sur le fichier toto.txt

head et tail

```
head toto.txt → 10 premières lignes de toto.txt
```

```
tail toto.txt → 10 dernières lignes de toto.txt
```

Exemples de combinaisons avec le « pipe »

```
ps ax | grep firefox
```

```
cat toto.txt | wc -l
```

```
ls * > toto.txt
```

```
cat toto.txt | less
```

```
cat toto.txt | grep ^blabla
```

→ afficher uniquement les lignes commençant par « blabla »

```
sed s/bonjour/bonsoir/g toto.txt | grep bonsoir > selection_substituee.txt
```

Commandes vi/Vim

ESC → mode commande

i → insertion

a → insertion "after"

:w → sauvegarde (*write*)

:q → quitter

:wq → quitter en sauvent

:q! → quitter sans sauver

u → *undo*

CTR+r → *redo*

v → sélectionne

y → copie

p → colle (*paste*)

d → couper

Script Bash

\$ (cmde) → évalue la commande et affiche son résultat.

\$ `cmde` → idem.

exemples :

\$ ((expression)) → évalue l'expression arithmétique et affiche le résultat.

```
nom=toto
echo $toto
```

1.sh

```
#!/bin/bash

for i in $( ls ); do
    echo item: $i
done
```

Script Bash uniligne :

```
for i in $( ls ); do echo $i; done
```

2.sh

```
#!/bin/bash

for i in {1..50}
do
    mkdir dossier$i
done
```

3.sh

```
#!/bin/bash

mkdir dossier
cd dossier

for i in {1..50}
do
    echo "blabla $i blabla" > fichier_$i.txt
done
```

4.sh

```
#!/bin/bash

read "votre nom : " nom

if [ $nom = "Toto" ]
then
    echo "Bonjour Toto !"
elif [ $nom = "Bobo" ]
then
    echo "Bonjour Bobo !"
elif [ $nom = "Gogo" ]
then
    echo "Bonjour Gogo !"
else
    echo "Bonjour Mr. X !"
fi
```

Droits

Les permissions peuvent être :

- r → permission en lecture.
- w → permission en écriture.
- x → permission d'exécution pour un fichier, permission d'entrer dans un répertoire.

chmod

- u → *user* (propriétaire)
- g → *group* (groupe)
- o → *other* (autres)

- + → "ajouter le droit"
- → "supprimer le droit"
- (-R pour affecter récursivement)

Chiffres correspondants aux droits recherchés

- pour l'utilisateur :
 - droits d'accès en lecture : 400
 - droits d'accès en écriture : 200
 - droits d'accès en exécution : 100
- pour le groupe :
 - droits d'accès en lecture : 40
 - droits d'accès en écriture : 20
 - droits d'accès en exécution : 10
- pour les autres :
 - droits d'accès en lecture : 4
 - droits d'accès en écriture : 2
 - droits d'accès en exécution : 1
- on additionne ensuite les droits pour chacun

ls -l → Les droits s'affichent pour l'utilisateur, le groupe et les autres.

Exemple 1 : droit "ugo"

chmod +x toto
→ rends exécutable toto

chmod ug+x toto
→ rends exécutable toto pour l'utilisateur et le groupe

Exemple 2 : droits avec chiffres

Les droits rwxr-xr-x pour toto.txt équivalent à :

- 400+200+100=700 pour l'utilisateur
- 40+10=50 pour le groupe
- 4+1=5 pour les autres

Soit au total 700+50+5=755 → chmod 755 toto.txt

chown → changer le propriétaire d'un fichier (-R changement récursif)

`chown toto fichier` → « toto » est propriétaire de « fichier »

chgrp → changer le groupe propriétaire d'un fichier (-R changement récursif)

`chgrp toto fichier` → « toto » est le "groupe" de « fichier »

Gestion des utilisateurs

`/etc/passwd`

→ tout ce qui concerne la gestion et l'authentification des utilisateurs

`/etc/group`

→ la gestion des groupes

`/etc/shadow`

→ Les mots de passe cryptés sont souvent placés dans ce fichier,
par sécurité lisible seulement par root.

Structure de `/etc/passwd`

Ce fichier comprend 7 champs, séparés par le symbole « : »

- nom de connexion (encore appelé nom d'utilisateur ou login)
- ancienne place du mot de passe crypté
- numéro d'utilisateur **uid**, sa valeur est le véritable identifiant pour le système Linux ;
l'uid de root est 0,
le système attribue conventionnellement un uid à partir de 500 aux comptes créés.
- numéro de groupe **gid**, dans lequel se trouve l'utilisateur par défaut ; le gid de root est 0,
les groupes d'utilisateurs au-delà de 500
- nom complet, il peut être suivi d'une liste de renseignements personnels
- rép. personnel (c'est également le rép. de connexion)
- shell, interpréteur de commandes (par défaut `/bin/bash`)

Structure de `/etc/group`

Ce fichier comprend 4 champs, séparés par le symbole « : »

- nom du groupe
- x pour remplacer un mot de passe non attribué maintenant
- numéro de groupe, c-à-d l'identifiant **gid**
- la liste des membres du groupe

`useradd, usermod, userdel`

→ gestion des comptes utilisateur

`groupadd, groupmod, groupdel`

→ gestion des groupes

`passwd`

→ changer le mot de passe d'un utilisateur

useradd → outils de création d'un compte d'utilisateur

`useradd -g group1 toto`

→ Créer « toto » de groupe primaire « group1 »

`useradd -G group1 toto`

→ Créer « toto » de groupe secondaire « group1 »

`useradd -G group1,group2 toto`

→ Créer « toto » de groupe secondaire « group1 » et « group2 »

usermod → modifier un utilisateur

« `-l` » → renomme l'utilisateur

« `-g` » → change de groupe

`usermod -g group1 toto`

→ Modification du groupe primaire d'un utilisateur

`usermod -a -G group1 toto`

→ Ajout d'un groupe secondaire a un utilisateur existant

L'arborescence des fichiers – Debian

Filesystem Hierarchy Standard (« norme de la hiérarchie des systèmes de fichiers ») définit l'arborescence et le contenu des principaux répertoires des systèmes de fichiers des systèmes d'exploitation GNU/Linux et de la plupart des systèmes Unix.

Répertoire Contenu

<code>bin</code>	Binaires (exécutables) des commandes essentielles.
<code>boot</code>	Fichiers statiques pour le programme d'amorçage.
<code>dev</code>	Fichiers des pilotes de périphériques.
<code>etc</code>	Configuration système propre a la machine.
<code>home</code>	Répertoires personnels des utilisateurs.
<code>lib</code>	Bibliothèques partagées et modules noyaux essentiels.
<code>media</code>	Points de montage pour les supports amovibles.
<code>mnt</code>	Point de montage pour les montages temporaires.
<code>proc</code>	Répertoire virtuel pour les informations système (noyaux 2.4 et 2.6).
<code>root</code>	Répertoire personnel de l'utilisateur <i>root</i> .
<code>sbin</code>	Exécutables système essentiels.
<code>sys</code>	Répertoire virtuel pour les informations système (noyaux 2.6).
<code>tmp</code>	Fichiers temporaires.
<code>usr</code>	Hiérarchie secondaire.
<code>var</code>	Données variables.
<code>srv</code>	Données pour les services fournis par le système.
<code>opt</code>	Répertoire pour d'autres logiciels.

Raccourcis du terminal Bash

CTR-a	→ début de ligne
CTR-e	→ fin de ligne
CTR-l	→ efface la console
CTR-k	→ efface à droite