

## Reprenez le contrôle à l'aide de Linux !

🕒 30 heures 📊 Facile

Mis à jour le 29/06/2021



## Le contenu de ce cours n'est plus à jour

Nous avons archivé ce cours et n'actualiserons plus son contenu.

Accédez au contenu le plus récent en découvrant ce cours :



## SYSTÈMES &amp; RÉSEAUX

## Initiez-vous à Linux

📊 Easy 🕒 8 heures

Dans ce cours débutant, découvrez Linux : un système d'exploitation gratuit et fascinant qui vous donnera un contrôle sans précédent sur votre ordinateur ! Créé par des passionnés d'informatique, Linux est un vecteur important de la philosophie du libre et l'alternative parfaite à Windows ou macOS.

VOIR LE NOUVEAU COURS

## Manipuler les fichiers

Après avoir vu comment étaient organisés les fichiers sous Linux, nous allons apprendre à les manipuler !

Par exemple, comment faire pour afficher le contenu d'un fichier ?

Comment le déplacer, le copier, le supprimer ?

C'est donc un chapitre à la fois simple et riche qui vous attend, tout au long duquel vous allez apprendre beaucoup de nouvelles commandes basiques de Linux qu'il vous faut connaître absolument !

## cat &amp; less : afficher un fichier

Nous allons d'abord voir comment afficher le contenu d'un fichier. Il y a en gros deux commandes basiques sous Linux qui permettent de faire cela :

- `cat` ;
- `less` .

Aucune de ces commandes ne permet d'éditer un fichier, elles permettent juste de le **voir**. Nous étudierons l'édition plus tard, ça mérite au moins un chapitre entier.

Mais... pourquoi deux commandes pour afficher un fichier ? Une seule n'aurait pas suffi ?

En fait, chacune a ses spécificités ! Nous allons les voir dans le détail.

Pour nos exemples, nous allons travailler sur un fichier qui existe déjà : `syslog` . Il se trouve dans le dossier `/var/log` . Commencez par vous y rendre :

```
mateo21@mateo21-desktop:~$ cd /var/log
```

Ce dossier contient plusieurs fichiers de **log**, c'est-à-dire des fichiers qui gardent une trace de l'activité de votre ordinateur. Vous pouvez en faire la liste si vous le voulez, en tapant `ls` :

```
mateo21@mateo21-desktop:/var/log$ ls
acpid          daemon.log.0      kern.log.0        scrollkeeper.log.2
acpid.1.gz     daemon.log.1.gz   kern.log.1.gz     syslog
acpid.2.gz     daemon.log.2.gz   kern.log.2.gz     syslog.0
acpid.3.gz     daemon.log.3.gz   kern.log.3.gz     syslog.1.gz
acpid.4.gz     debug             lastlog           syslog.2.gz
apparmor       debug.0           lpr.log           syslog.3.gz
apport.log     debug.1.gz        mail.err          syslog.4.gz
apport.log.1   debug.2.gz        mail.info         syslog.5.gz
apport.log.2   debug.3.gz        mail.log          syslog.6.gz
apport.log.3   dist-upgrade      mail.warn         udev
apport.log.4   dmesg             messages          unattended-upgrades
apport.log.5   dmesg.0           messages.0        user.log
apt            dmesg.1.gz        messages.1.gz     user.log.0
auth.log       dmesg.2.gz        messages.2.gz     user.log.1.gz
auth.log.0     dmesg.3.gz        messages.3.gz     user.log.2.gz
auth.log.1.gz  dmesg.4.gz        news              user.log.3.gz
auth.log.2.gz  dpkg.log           popularity-contest uucp.log
auth.log.3.gz  dpkg.log.1        popularity-contest.0 wtmp
bittorrent     dpkg.log.2.gz     popularity-contest.1.gz wtmp.1
boot           faillog            popularity-contest.2.gz wvdialconf.log
bootstrap.log  fontconfig.log    popularity-contest.3.gz Xorg.0.log
btmtp          fsck               pycentral.log     Xorg.0.log.old
btmtp.1        gdm               samba
cups           installer          scrollkeeper.log
daemon.log     kern.log           scrollkeeper.log.1
```

Le fichier sur lequel nous allons travailler, `syslog` , contient des informations de log de ce qui s'est passé récemment sur l'ensemble de votre ordinateur.

Vous noterez qu'il est fréquent de voir des fichiers sans extension sous Linux. Notre fichier s'appelle `syslog` tout court, et non pas `syslog.txt` ou `syslog.log` comme on pourrait avoir l'habitude de le voir sous Windows. Un fichier sans extension peut être ouvert et lu sans aucun problème comme n'importe quel autre fichier.

`cat` : afficher tout le fichierLa commande `cat` permet d'afficher tout le contenu d'un fichier dans la console d'un coup.Il vous suffit d'indiquer en paramètre le nom du fichier que vous voulez afficher, en l'occurrence `syslog` :

```
mateo21@mateo21-desktop:/var/log$ cat syslog
Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Job 'cron.daily' terminated
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)
Nov 14 00:44:25 mateo21-desktop NetworkManager: <info> eth1: link timed out.
Nov 14 00:44:51 mateo21-desktop NetworkManager: <info> eth1: link timed out.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <debug> [1104907508.332093] nm_device_002_11_wireless_get_activation_ap(): Forcing AP 'WIFI'
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> User Switch: /org/freedesktop/NetworkManager/Devices/eth1 / WIFI
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Deactivating device eth1.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): cancelling...
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1) cancellation handler scheduled...
```

```
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): waiting for device to cancel activation.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): cancellation handled.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): cancelled.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> SUP: sending command 'DISABLE_NETWORK 0'
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> SUP: response was 'OK'
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> SUP: sending command 'AP_SCAN 0'
Nov 14 00:45:16 mateo21-desktop NetworkManager: nm_act_request_get_ap: assertion 'req != NULL' failed
Nov 14 00:45:16 mateo21-desktop NetworkManager: nm_act_request_get_stage: assertion 'req != NULL' failed
Nov 14 00:45:16 mateo21-desktop NetworkManager: <info> eth1: link timed out.
Nov 14 00:45:21 mateo21-desktop NetworkManager: nm_act_request_get_ap: assertion 'req != NULL' failed
Nov 14 00:45:21 mateo21-desktop NetworkManager: ap_is_auth_required: assertion 'ap != NULL' failed
Nov 14 00:45:21 mateo21-desktop NetworkManager: <info> Activation (eth1/wireless): association took too long (>120s), asking for new key.
Nov 14 00:45:21 mateo21-desktop NetworkManager: nm_dbus_get_user_key_for_network assertion 'req != NULL' failed
Nov 14 00:47:45 mateo21-desktop init: tty4 main process (4517) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty6 main process (4518) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty2 main process (4520) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty3 main process (4522) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty1 main process (4524) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty6 main process (4525) killed by TERM signal
Nov 14 00:47:46 mateo21-desktop avahi-daemon[5390]: Got SIGTERM, quitting.
Nov 14 00:47:48 mateo21-desktop exiting on signal 15
Nov 14 00:48:42 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
...

```

Comme le fichier est très gros (il fait plusieurs centaines de lignes), je n'ai pas copié tout ce qui s'est affiché dans ma console. Ne vous étonnez pas si vous voyez tout s'afficher d'un coup : c'est normal, c'est le but. La commande `cat` vous envoie tout le fichier à la figure. Elle est plus adaptée lorsque l'on travaille sur de petits fichiers que sur des gros, car dans un cas comme celui-là, on n'a pas le temps de lire tout ce qui s'affiche à l'écran.

Il y a peu de paramètres vraiment intéressants à utiliser avec la commande `cat`, car c'est une commande somme toute très basique. On notera quand même le paramètre `-n` qui permet d'afficher les numéros de ligne :

```
mateo21@mateo21-desktop:/var/log$ cat -n syslog
1 Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
2 Nov 14 00:44:23 mateo21-desktop anacron[6725]: Job 'cron.daily' terminated
3 Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)
4 Nov 14 00:44:25 mateo21-desktop NetworkManager: <info> eth1: link timed out.
5 Nov 14 00:44:51 mateo21-desktop NetworkManager: <info> eth1: link timed out.
6 Nov 14 00:45:08 mateo21-desktop NetworkManager: <debug> [1194997588.332093]
...

```

### `less` : afficher le fichier page par page

La commande `cat` est rapide. Trop rapide. Tout le fichier est lu et affiché d'un coup dans la console, ce qui fait que l'on n'a pas le temps de le lire s'il est très gros.

C'est là qu'une autre commande comme `less` devient vraiment indispensable. La grosse différence entre `less` et `cat`, c'est que `less` affiche progressivement le contenu du fichier, page par page. Ça vous laisse le temps de le lire dans la console. :)

Notez qu'il existe aussi une commande très proche : `more`. Pour faire simple, la différence entre `more` et `less`, c'est que `more` est vieux et possède peu de fonctionnalités, tandis que `less` est beaucoup plus puissant et rapide. Bref, utilisez `less`, mais si vous voyez un jour quelqu'un utiliser `more`, ne soyez pas surpris.

Comment ça marche ? Eh bien la commande est très simple : `less nomdufichier`.

```
mateo21@mateo21-desktop:/var/log$ less syslog
Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Job 'cron.daily' terminated
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)
Nov 14 00:44:25 mateo21-desktop NetworkManager: <info> eth1: link timed out.
Nov 14 00:44:51 mateo21-desktop NetworkManager: <info> eth1: link timed out.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <debug> [1194997588.332093] nm_device_802_11_wireless_get_activation_ap(): Forcing AP 'WIFI'
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> User Switch: /org/freedesktop/NetworkManager/Devices/eth1 / WIFI
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Deactivating device eth1.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): cancelling...
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): cancellation handler scheduled...
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): waiting for device to cancel activation.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): cancellation handled.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): cancelled.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> SUP: sending command 'DISABLE_NETWORK 0'

```

Ce qui est intéressant pour nous ici, c'est que la commande `less` a arrêté la lecture du fichier au bout de quelques lignes (la taille d'un écran de console). Cela vous laisse le temps de lire le début du fichier.

On n'a lu pour le moment que les toutes premières lignes du fichier.

Et comment lire la suite ?

Il y a quelques raccourcis clavier à connaître. :)

### Les raccourcis basiques indispensables

Commençons par les quelques raccourcis clavier les plus indispensables, à connaître absolument.

- `Espace` : affiche la suite du fichier. La touche `Espace` fait défiler le fichier vers le bas d'un « écran » de console. C'est celle que j'utilise le plus souvent.
- `Entrée` : affiche la ligne suivante. Cela permet donc de faire défiler le fichier vers le bas ligne par ligne.

Vous pouvez aussi utiliser la touche `Flèche vers le bas`.

- `d` : affiche les onze lignes suivantes (soit une moitié d'écran). C'est un peu l'intermédiaire entre `Espace` (tout un écran) et `Entrée` (une seule ligne).
- `b` : retourne en arrière d'un écran.
- Vous pouvez aussi appuyer sur la touche `Page Up`.
- `y` : retourne d'une ligne en arrière.
- Vous pouvez aussi appuyer sur la touche `Flèche vers le haut`.
- `u` : retourne en arrière d'une moitié d'écran (onze lignes).
- `q` : arrête la lecture du fichier. Cela met fin à la commande `less`.

La casse des caractères est importante. Ainsi, si je vous dis qu'il faut appuyer sur la touche « d », ce n'est pas un « D » majuscule (si vous essayez, vous verrez que ça ne fonctionne pas). Sous Linux, on fait souvent la différence entre majuscules et minuscules : souvenez-vous-en !

Si on tape `Espace`, on avance donc d'un écran dans le fichier :

```
Nov 14 00:47:45 mateo21-desktop init: tty4 main process (4517) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty5 main process (4518) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty2 main process (4520) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty3 main process (4522) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty1 main process (4524) killed by TERM signal
Nov 14 00:47:45 mateo21-desktop init: tty6 main process (4525) killed by TERM signal
Nov 14 00:47:46 mateo21-desktop avahi-daemon[5390]: Got SIGTERM, quitting.
Nov 14 00:47:48 mateo21-desktop exiting on signal 15
Nov 14 00:48:42 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:48:42 mateo21-desktop kernel: Inspecting /boot/System.map-2.6.22-14-generic
Nov 14 00:48:42 mateo21-desktop kernel: Loaded 25445 symbols from /boot/System.map-2.6.22-14-generic.
Nov 14 00:48:42 mateo21-desktop kernel: Symbols match kernel version 2.6.22.
Nov 14 00:48:42 mateo21-desktop kernel: No module symbols loaded - kernel modules not enabled.
Nov 14 00:48:42 mateo21-desktop kernel: [ 0.000000] Linux version 2.6.22-14-generic (build@palmer) (gcc version 4.1.3 20070929 (prerelease)
:

```

Quelques raccourcis plus avancés

Ce ne sont pas des raccourcis que l'on utilise tous les jours, mais ça vaut le coup de savoir qu'ils existent. :)

- `=` : indique où vous en êtes dans le fichier (numéro des lignes affichées et pourcentage).
- `h` : affiche l'aide (toutes les commandes que je vous apprendis ici, je les tire de là). Tapez `q` pour sortir de l'aide.
- `/` : tapez `/` suivi du texte que vous recherchez pour lancer le mode recherche. Faites `Entrée` pour valider. Pour ceux qui savent s'en servir, sachez que les expressions régulières sont acceptées. Je ne vais pas vous faire un cours sur les expressions régulières ici, ce serait trop long, mais il y en a dans mon cours sur le PHP [Concevez votre site Web avec PHP et MySQL](#) dans la même collection.
- `n` : après avoir fait une recherche avec `/` , la touche `n` vous permet d'aller à la prochaine occurrence de votre recherche. C'est un peu comme si vous cliquiez sur le bouton « Résultat suivant ».
- `N` : pareil que `n` , mais pour revenir en arrière.

Comme vous le voyez, la commande `!less` est très riche. On peut utiliser beaucoup de touches différentes pour se déplacer dans le fichier. Prenez le temps de vous familiariser avec : c'est un peu perturbant au début, mais lorsque vous aurez appris à vous en servir, vous aurez déjà fait un grand pas en avant... et puis ça vous sera très utile plus tard, croyez-moi. :)

head & tail : afficher le début et la fin d'un fichier

Quoi ? Encore des commandes pour lire un fichier ?

Eh oui.  
Et figurez-vous que celles-là aussi valent le coup d'être connues. Comme quoi on en fait des commandes, rien que pour lire un fichier !  
Ces deux commandes sont un peu à l'opposé l'une de l'autre : la première permet d'afficher le début du fichier, la seconde permet d'afficher la fin.

head : afficher le début du fichier

La commande `head` (« tête » en anglais) affiche seulement les premières lignes du fichier. Elle ne permet pas de se déplacer dans le fichier comme `!less` , mais juste de récupérer les premières lignes.

```
mateo21@mateo21-desktop:/var/log$ head syslog
Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Job 'cron.daily' terminated
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)
Nov 14 00:44:25 mateo21-desktop NetworkManager: <info> eth1: link timed out.
Nov 14 00:44:51 mateo21-desktop NetworkManager: <info> eth1: link timed out.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <debug> [1194997508.332093] nm_device_802_11_wireless_get_activation_ap(): Forcing AP 'WIFI'
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> User Switch: /org/freedesktop/NetworkManager/Devices/eth1 / WIFI
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Deactivating device eth1.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1): cancelling...
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info> Activation (eth1) cancellation handler scheduled...
```

Si vous avez juste besoin de récupérer les premières lignes d'un fichier, `head` est donc la commande qu'il vous faut. Simple, net, efficace.

Comment ? Vous voulez des paramètres ?  
Je n'en ai pas beaucoup à vous offrir, mais celui-là au moins est à connaître : `-n` , suivi d'un nombre. Il permet d'afficher le nombre de lignes que vous voulez. Par exemple, si vous ne voulez que les trois premières lignes, tapez :

```
mateo21@mateo21-desktop:/var/log$ head -n 3 syslog
Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Job 'cron.daily' terminated
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)
```

Et voilà le travail !

tail : afficher la fin du fichier

Très intéressante aussi (voire même plus), la commande `tail` vous renvoie la fin du fichier, donc les dernières lignes.

```
mateo21@mateo21-desktop:/var/log$ tail syslog
Nov 14 22:42:10 mateo21-desktop NetworkManager: <info> Clearing nscd hosts cache.
Nov 14 22:42:10 mateo21-desktop NetworkManager: <WARN> nm_spawn_process(): nm_spawn_process('/usr/sbin/nscd -i hosts'): could not spawn process. (Failed to execute child process "/usr/sbin/nscd" (No such file or directory))
Nov 14 22:42:10 mateo21-desktop NetworkManager: <info> Activation (eth1) Finish handler scheduled.
Nov 14 22:42:10 mateo21-desktop NetworkManager: <info> Activation (eth1) Stage 5 of 5 (IP Configure Commit) complete.
Nov 14 22:42:10 mateo21-desktop NetworkManager: <info> Activation (eth1) successful, device activated.
Nov 14 22:41:57 mateo21-desktop ntpdate[8422]: step time server 91.189.94.4 offset -13.401451 sec
Nov 14 22:41:59 mateo21-desktop avahi-daemon[5385]: Registering new address record for fe80::219:d2ff:fe61:900a on eth1.*.
Nov 14 22:42:08 mateo21-desktop kernel: [ 7870.160000] eth1: no IPv6 routers present
Nov 14 23:11:26 mateo21-desktop -- MARK --
Nov 14 23:17:01 mateo21-desktop /USR/SBIN/CRON[8515]: (root) CMD ( cd / && run-parts --report /etc/cron.hourly)
```

On peut là encore utiliser `-n` suivi d'un nombre pour afficher les \$X\$ dernières lignes :

```
mateo21@mateo21-desktop:/var/log$ tail -n 3 syslog
Nov 14 22:42:08 mateo21-desktop kernel: [ 7870.160000] eth1: no IPv6 routers present
Nov 14 23:11:26 mateo21-desktop -- MARK --
Nov 14 23:17:01 mateo21-desktop /USR/SBIN/CRON[8515]: (root) CMD ( cd / && run-parts --report /etc/cron.hourly)
```

Mais ce n'est pas tout ! Il y a un autre paramètre à côté duquel vous ne pouvez pas passer : `-f` (f pour *follow*, « suivre » en anglais).

Ce paramètre magique ordonne à `tail` de « suivre » la fin du fichier au fur et à mesure de son évolution.  
C'est extrêmement utile pour suivre un fichier de log qui évolue souvent. Vous pouvez tester sur `syslog` par exemple :

```
mateo21@mateo21-desktop:/var/log$ tail -f syslog
Nov 14 23:17:01 mateo21-desktop -- MARK --
Nov 14 23:17:01 mateo21-desktop /USR/SBIN/CRON[8515]: (root) CMD ( cd / && run-parts --report /etc/cron.hourly)
Nov 14 23:27:52 mateo21-desktop kernel: [10614.344000] ata2.00: exception Emask 0x0 SAct 0x0 SErr 0x0 action 0x2 frozen
Nov 14 23:27:52 mateo21-desktop kernel: [10614.344000] ata2.00: cmd a0/00:00:00:00:20/00:00:00:00:00/a0 tag 0 cdb 0x0 data 0
Nov 14 23:27:52 mateo21-desktop kernel: [10614.344000] res 40/00:03:00:00:00/00:00:00:00/a0 Emask 0x4 (timeout)
Nov 14 23:27:57 mateo21-desktop kernel: [10619.388000] ata2: port is slow to respond, please be patient (Status 0x0d)
Nov 14 23:28:02 mateo21-desktop kernel: [10624.392000] ata2: device not ready (errno=-16), forcing hardreset
Nov 14 23:28:02 mateo21-desktop kernel: [10624.392000] ata2: soft resetting port
Nov 14 23:28:02 mateo21-desktop kernel: [10624.928000] ata2.00: configured for UDMA/33
Nov 14 23:28:02 mateo21-desktop kernel: [10624.928000] ata2: EH complete
```

Le problème de `syslog` c'est qu'il n'évolue pas forcément toutes les secondes. Mais si vous êtes patients et que vous regardez votre console, vous devriez le voir écrire de nouvelles lignes sous vos yeux au bout d'un moment.

Faites `Ctrl + C` ( `Ctrl` et `C` en même temps) pour arrêter la commande `tail` .

À connaître : la combinaison de touches `Ctrl + C` est utilisable dans la plupart des programmes console pour demander leur arrêt. C'est un peu l'équivalent du `Alt + F4` de Windows.

Pour tout vous dire, `tail -f` est une de mes commandes préférées sous Linux. C'est un bon moyen de surveiller ce qui se passe en temps réel sur un ordinateur (si vous êtes assez rapides pour suivre).

Par exemple, les logs Apache du Site du Zéro permettent de voir en temps réel qui se connecte sur le site, avec quelle IP, quel fichier a été chargé, à quelle heure, etc.  
Aux heures d'affluence du site, ce fichier évolue tellement vite qu'il est pratiquement impossible de le suivre pour un humain.

Je vous ai fait une petite vidéo d'un `tail -f` en action pour que vous vous rendiez compte de la chose. Elle est [accessible par téléchargement](#) (580 Ko) :

Notez que par défaut, `tail -f` recherche les nouveaux changements dans le fichier toutes les secondes. Si vous voulez, vous pouvez rajouter le paramètre `-s` suivi d'un nombre. Par exemple, `tail -f -s 3 syslog` recherchera les changements toutes les trois secondes (plutôt que toutes les secondes). Les nombres décimaux sont acceptés, à condition d'utiliser le point « . » à la place de la virgule.

touch & mkdir : créer des fichiers et dossiers

Assez lu de fichiers, maintenant voyons voir comment on les crée !

Nous allons d'abord voir comment créer un fichier, puis comment créer un dossier, car ce n'est pas la même commande...

touch : créer un fichier

En fait, il n'existe aucune commande spécialement faite pour créer un fichier vide sous Linux (ce n'est pas très utile). En général, on se contente d'ouvrir un éditeur de texte et d'enregistrer, ce qui provoque la création d'un fichier comme sous Windows.

La commande `touch` est à la base faite pour modifier la date de dernière modification d'un fichier. D'où son nom : on « touche » le fichier pour faire croire à l'ordinateur qu'on vient de le modifier alors que l'on n'a rien changé. Ça peut se révéler utile dans certains cas précis qu'on ne verra pas ici.

L'intérêt de `touch` pour nous dans ce chapitre, c'est que si le fichier n'existe pas, il sera créé ! On peut donc **aussi** utiliser `touch` pour créer des fichiers, même s'il n'a pas vraiment été fait pour ça à la base.

La commande attend un paramètre : le nom du fichier à créer.

Commencez par vous rendre dans votre dossier personnel ; ce n'est pas une bonne idée de mettre le bazar dans `/var/log`, le dossier personnel est là pour ça.

Si vous vous souvenez bien, il suffit de taper `cd` :

```
mateo21@mateo21-desktop:~/var/log$ cd
mateo21@mateo21-desktop:~$
```

Pour le moment, mon dossier personnel ne contient que des sous-dossiers :

```
mateo21@mateo21-desktop:~$ ls -F
Desktop/  Exemples/ Images/  Modèles/  Public/  Vidéos/
Documents/ images/  log/     Musique/  tutos/
```

Créons un fichier appelé `fichierbidon` :

```
mateo21@mateo21-desktop:~$ touch fichierbidon
mateo21@mateo21-desktop:~$ ls -F
Desktop/  Exemples/ Images/  log/     Musique/  tutos/
Documents/ fichierbidon Images/  Modèles/  Public/  Vidéos/
```


La commande `ls -F` que j'ai tapée ensuite le montre, un fichier appelé `fichierbidon` (sans extension) a été créé. Bien entendu, vous pouvez créer un fichier de l'extension que vous voulez :

```
mateo21@mateo21-desktop:~$ touch autrefichierbidon.txt
mateo21@mateo21-desktop:~$ ls -F
Desktop/  Exemples/ Images/  Musique/  Vidéos/
autrefichierbidon.txt Exemples/ log/     Public/
Documents/ fichierbidon log/     Public/
            images/  Modèles/  tutos/
```

Autre information intéressante : vous pouvez créer plusieurs fichiers en une seule commande. Il vous suffit de les lister l'un après l'autre, séparés par des espaces.

Ainsi, on aurait pu créer nos deux fichiers comme ceci :

```
touch fichierbidon autrefichierbidon.txt
```

 Et si je veux que mon fichier contienne un espace, je fais comment ?

Entourez-le de guillemets !

```
touch "Fichier bidon"
```

`mkdir` : créer un dossier

La commande `mkdir`, elle, est faite pour créer un dossier. Elle fonctionne de la même manière que `touch`.

```
mkdir mondossier
```

On peut créer deux dossiers (ou plus !) en même temps en les séparant là aussi par des espaces :

```
mkdir mondossier autredossier
```

Si vous faites un `ls`, vous verrez que les dossiers ont bien été créés. :-)

Il y a un paramètre utile avec `mkdir` : `-p`. Il sert à créer tous les dossiers intermédiaires. Par exemple :

```
mkdir -p animaux/vertèbres/chat
```

... créera le dossier `animaux`, puis à l'intérieur le sous-dossier `vertèbres`, puis à l'intérieur encore le sous-dossier `chat` !

## cp & mv : copier et déplacer un fichier

Parmi les opérations de base que l'on veut pouvoir faire avec les fichiers, il y a la copie et le déplacement de fichier. C'est un peu le genre de chose que l'on fait tous les jours, il est donc important de savoir s'en servir.

`cp` : copier un fichier

La commande `cp` (abréviation de `CoPy`, « copier » en anglais) vous permet comme son nom l'indique de copier un fichier... mais aussi de copier plusieurs fichiers à la fois, et même de copier des dossiers !


Si on essayait de copier le fichier `fichierbidon` qu'on a créé tout à l'heure ?

Ça fonctionnait comme ceci :

```
cp fichierbidon fichiercopie
```

Le premier paramètre est le nom du fichier à copier, le second le nom de la copie du fichier à créer.

En faisant cela, on aura donc deux fichiers identiques dans le même répertoire : `fichierbidon` et `fichiercopie`.

 N'oubliez pas d'utiliser l'autocomplétion avec la touche `Tab` ! Lorsque vous avez écrit `cp fic`, tapez `Tab`, et `fichierbidon` devrait se compléter tout seul !


### Copier dans un autre dossier

On n'est pas obligé de copier le fichier dans le même dossier, bien sûr. On peut très bien utiliser le système de répertoires relatifs et absolus qu'on a vu dans le chapitre précédent.

Par exemple, si je veux copier `fichierbidon` dans le sous-dossier `mondossier` que j'ai créé tout à l'heure :

```
cp fichierbidon mondossier/
```

Le fichier `fichierbidon` sera copié dans `mondossier` sous le même nom.

 Notez que mettre le `/` à la fin n'est pas obligatoire. Si vous le voyez là, c'est parce que l'autocomplétion me l'a automatiquement ajouté lorsque j'ai appuyé sur `Tab`. Ehhh oui, je suis tellement flemmard que je n'écris même pas `mondossier` en entier, j'écris juste `mon` suivi de `Tab`, et hop c'est écrit en entier ! Ça va beaucoup plus vite lorsqu'on prend ce réflexe.

Si vous voulez copier `fichierbidon` dans `mondossier` sous un autre nom, faites comme ceci :

```
cp fichierbidon mondossier/fichiercopie
```

Avec cette commande, on aura créé une copie de `fichierbidon` dans `mondossier` sous le nom `fichiercopie` !

Enfin là, j'utilise des répertoires relatifs, mais je peux aussi écrire un répertoire en absolu :

```
cp fichierbidon /var/log/
```

... copiera `fichierbidon` dans le dossier `/var/log`.

### Copier des dossiers

Avec l'option `-R` (un « R » majuscule !), vous pouvez copier un dossier, ainsi que tous les sous-dossiers et fichiers qu'il contient !

Tout à l'heure, on a créé un dossier `animaux` qui contenait un autre dossier `vertèbres`, qui lui-même contenait le dossier `chat`. Si vous tapez cette commande :

```
cp -R animaux autresanimaux
```

... cela aura pour effet de copier `animaux` ainsi que tous ses sous-dossiers sous le nom `autresanimaux`.

Faites des `ls` après pour vérifier que les sous-dossiers sont bien là et que je ne vous mène pas en bateau !

Utiliser le `joker` \*

Le symbole `*` est appelé *joker*, ou encore *wildcard* en anglais sous Linux.  
Il vous permet de copier par exemple tous les fichiers image `.jpg` dans un sous-dossier :

```
cp *.jpg mondossier/
```

Vous pouvez aussi vous en servir pour copier tous les fichiers dont le nom commence par « so » :

```
cp so* mondossier/
```

Le `joker` est un atout très puissant, n'hésitez pas à l'utiliser !  
C'est avec des outils comme le `joker` que la console deviendra pour vous progressivement plus puissante que l'explorateur de fichiers que vous manipulez à la souris.

`mv` : déplacer un fichier

Très proche de `cp`, la commande `mv` (*MoVe*, « déplacer » en anglais) a en fait deux utilités :

- déplacer un fichier (ou un dossier) ;
- renommer un fichier (ou un dossier).

Vous allez comprendre pourquoi.

Déplacer un fichier

La commande `mv` s'utilise pratiquement comme `cp` :

```
mv fichierbidon mondossier/
```

Au lieu de copier `fichierbidon` dans `mondossier` comme on l'a fait tout à l'heure, ici on a juste déplacé le fichier. Il n'existe plus dans son dossier d'origine.

Vous pouvez déplacer des dossiers aussi simplement :

```
mv animaux/ mondossier/
```

... déplacera le dossier `animaux` (et tous ses sous-dossiers) dans `mondossier`.

Vous pouvez aussi utiliser les *jokers* :

```
mv *.jpg mondossier/
```

Renommer un fichier

La commande `mv` permet de faire quelque chose d'assez étonnant : renommer un fichier. En effet, il n'existe pas de commande spéciale pour renommer un fichier en console sous Linux, c'est la commande `mv` qui est utilisée pour ça.

Par exemple :

```
mv fichierbidon superfichier
```

... renommra `fichierbidon` en `superfichier`. Après cette commande, `fichierbidon` n'existe plus, il a été renommé.


Déplacer et renommer un fichier à la fois

Vous pouvez aussi déplacer `fichierbidon` dans `mondossier` tout en lui affectant un nouveau nom :

```
mv fichierbidon mondossier/superfichier
```

Et voilà le travail !

Je vous conseille **fortement** de vous entraîner à utiliser `cp` et `mv` dans tous les sens : avec ou sans `joker`, en déplaçant, renommant des dossiers, en déplaçant / renommant à la fois, en utilisant des chemins relatifs et absolus, etc.  
C'est assez intuitif normalement, mais il faut pratiquer et pas seulement se contenter de lire ce que j'écris pour que ça rentre.



N'oubliez pas d'utiliser l'autocomplétion de fichiers et dossiers avec la touche `Tab` : si vous ne le faites pas dès maintenant, vous perdrez du temps et vous trouverez la console nulle alors que vous devriez la trouver géniale.  
Autre chose : le symbole `..` signifie « dossier précédent », et `.` signifie « dossier dans lequel je me trouve ». Vous pourriez en avoir besoin lorsque vous copiez ou déplacez un fichier.

Si vous avez la tête qui tourne à force de copier et déplacer des fichiers dans des dossiers, c'est normal. Ça commence à devenir un beau bazar dans vos dossiers d'ailleurs, non ?  
Il est temps de faire un peu de ménage avec la commande permettant de **supprimer** : `rm` !

rm : supprimer des fichiers et dossiers ▼

On attaque la commande qui fâche : `rm`.  
Pourquoi est-ce qu'elle fâche ? Parce qu'il n'existe pas de corbeille dans la console de Linux : le fichier est directement supprimé sans possibilité de récupération !

`rm` : supprimer un fichier

La commande `rm` (pour *ReMove*, « supprimer » en anglais) peut supprimer un fichier, plusieurs fichiers, des dossiers, voire même votre ordinateur entier si vous le voulez.

Il faut donc l'utiliser avec précaution. Commençons par des choses simples, supprimons ce `fichierbidon` :

```
rm fichierbidon
```

Normalement, on ne vous demande pas de confirmation, on ne vous affiche rien. Le fichier est supprimé sans autre forme d'avertissement. Brutal, hein ?

Vous pouvez aussi supprimer plusieurs fichiers en séparant leurs noms par des espaces :

```
rm fichierbidon fichiercopie
```

`-i` : demander confirmation

La commande `-i` permet de vous demander une confirmation pour chacun des fichiers :

```
mateo21@mateo21-desktop:~$ rm -i fichierbidon
rm: détruire fichier régulier vide `fichierbidon'?
```

Lorsqu'on vous demande une confirmation de type oui/non comme ici, vous devez répondre par une lettre :

- `o` : signifie « Oui ». Sur certains systèmes anglais, il faudra peut-être utiliser `y` de *Yes* ;
- `n` : signifie « Non ».

Tapez ensuite sur `Entrée` pour valider.

`-f` : forcer la suppression, quoi qu'il arrive

`-f`, c'est un peu le contraire de `-i` : c'est le mode des gros bourrins.  
Ce paramètre force la suppression, ne demande pas de confirmation, même s'il y a un problème potentiel.

En raison des risques que cela comporte, utilisez-le aussi rarement que possible.

```
rm -f fichierbidon
```

`-v` : dis-moi ce que tu fais, petit cachotier

Le paramètre `-v` (*Verbose*, verbeux en anglais, c'est-à-dire « parler beaucoup ») est un paramètre que l'on retrouve dans beaucoup de commandes sous Linux. Il permet de demander à la commande de dire ce qu'elle est en train de faire.

Comme vous l'avez vu, par défaut la commande `rm` est silencieuse. Si vous supprimez de très nombreux fichiers, ça peut prendre du temps. Pour éviter que vous vous impatientiez, pensez à utiliser `-v` :

```
mateo21@mateo21-desktop:~$ rm -v fichierbidon fichiercopie
détruit `fichierbidon'
détruit `fichiercopie'
```

Vous voyez au fur et à mesure de l'avancement ce qui est en train d'être fait. Très pratique !

`-r` : **supprimer un dossier et son contenu**

Le paramètre `-r` peut être utilisé pour supprimer un dossier (au lieu d'un fichier) ainsi que tout ce qu'il contient : fichiers et dossiers !

C'est un paramètre assez dangereux, faites donc bien attention de l'utiliser sur un dossier dont vous ne voulez vraiment plus, car tout va disparaître à l'intérieur :

```
rm -r animaux/
```

... supprime le dossier `animaux` ainsi que tout ce qu'il contenait (sous-dossiers `vertèbres` et `chat`).

Notez qu'il existe aussi la commande `rmdir`. La grosse différence avec `rm -r`, c'est que `rmdir` ne peut supprimer un dossier que s'il est vide ! Il faudra y avoir fait le ménage auparavant.

### `rm` et le joker de la mort (qui tue)

Bon, vous êtes grands, je crois que le moment est venu de vous révéler un terrible secret : les enfants ne naissent pas dans les choux.

Euh pardon, je voulais dire : la commande `rm` est vraiment dangereuse. Très dangereuse. Vous pouvez potentiellement bousiller tout votre système avec !

Je vais vous montrer quelque chose d'horrible, d'interdit aux moins de 18 ans, bref vous m'avez compris, le truc à ne faire sous aucun prétexte, même pas en cauchemar.

```
NON NON NON NE FAITES JAMAIS CA !!! => rm -rf /*
```

Je me suis permis de mettre du texte avant pour vous éviter la tentation de recopier bêtement la commande pour « rigoler », pour « voir ce que ça fait ». Je vais vous l'expliquer dans le détail, parce que c'est quand même **l'erreur n°1** à ne pas faire sous Linux.

- `rm` : commande la suppression ;
- `-r` : supprime de manière récursive tous les fichiers et dossiers ;
- `-f` : force la suppression sans demander la moindre confirmation ;
- `/*` : supprime tous les fichiers et dossiers qui se trouvent à la racine ( `/` ) quel que soit leur nom (joker `*`).

En clair, cette commande supprime tout votre disque dur depuis la racine, sous-dossiers compris, et ne demande aucune confirmation. Aucune possibilité de récupération, votre PC est foutu. Vous êtes bons pour une réinstallation de Linux, **et aussi de Windows** si la partition de Windows était accessible depuis Linux.

Mais ils sont bêtes les gens qui ont créé cette commande ! Pourquoi autoriser de faire une chose aussi risquée ?

En fait, il y a plusieurs mécanismes de protection. On en apprendra plus dans le prochain chapitre (qui traitera des utilisateurs et de leurs droits).

Par exemple, les fichiers à la racine ne vous « appartiennent » pas, ils appartiennent au superutilisateur « root ». Moi je me suis loggé en tant que `mateo21`, je n'ai donc théoriquement pas le droit de supprimer ces fichiers. La suppression sera refusée.

Seulement, pour peu que vous soyez loggés en tant que « root » (on verra comment le faire dans le chapitre suivant), vous aurez le droit de le faire, et là **plus rien ne vous arrêtera** !

On apprendra plus tard comment utiliser les alias de commande pour éviter qu'une commande aussi dangereuse ne s'exécute. En attendant, ne jouez pas avec le feu, car vous y perdriez les mains, les pieds, la tête et tout ce qui va avec.

Le joker reste quand même très utile, mais lorsque vous l'utilisez avec `rm`, triplez d'attention.

Par exemple :

```
rm -rf *
```

... supprime tous les fichiers et sous-dossiers du dossier dans lequel je me trouve. Il m'arrive de l'utiliser, d'en avoir besoin, mais à chaque fois je fais très très attention à ce qu'il n'y ait plus rien dans ce dossier (et dans les sous-dossiers) qui m'intéresse.

Comme vous pouvez le voir, il n'y a qu'un seul caractère de différence (le `/` ) avec la commande de la mort que je vous ai montrée un peu plus haut.

Une erreur est vite arrivée. J'ignore combien de gens se sont pendus après avoir exécuté cette commande, mais ça méritait au moins un **GROS** avertissement !

## In : créer des liens entre fichiers

Bien qu'un peu moins courante, la commande `ln` vous sera certainement utile un jour ou l'autre. Elle permet de créer des liens entre des fichiers, c'est-à-dire (pour employer des mots que vous connaissez) qu'elle permet de **créer des raccourcis**.

Ces « raccourcis », qu'on appelle des **liens** sous Linux, sont un peu plus complexes que ceux que vous avez l'habitude de voir sous Windows. En effet, on peut créer deux types de liens :

- des liens **physiques** ;
- des liens **symboliques**.

Ces deux types ne fonctionnent pas de la même manière. Pour comprendre ce qui les différencie, il faut savoir comment un OS tel que Linux gère les fichiers sur le disque dur. Allons, allons, ne faites pas cette tête-là, un peu de théorie sur le fonctionnement des OS, c'est toujours très intéressant !-)

### Le stockage des fichiers

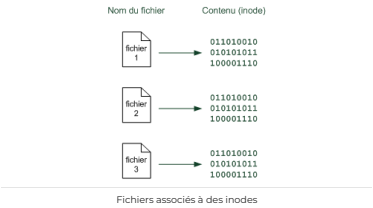
Sur le disque dur, chaque fichier est grosso-modo séparé en deux parties :

- son nom ;
- son contenu.

Vous avez bien lu : la liste des noms de fichiers est stockée à un autre endroit que leur contenu. Cette séparation aide Linux à s'organiser.

Je simplifie ici volontairement les choses. En pratique, c'est (toujours) un peu plus compliqué. Il y en a fait trois parties : le nom, les informations de gestion (droits d'accès) et le contenu. Mais nous allons faire simple car notre but est juste de comprendre l'idée générale du fonctionnement.

Chaque contenu de fichier se voit attribuer un numéro d'identification appelé **inode** (figure suivante). Chaque nom de fichier est donc associé à un inode (son contenu).



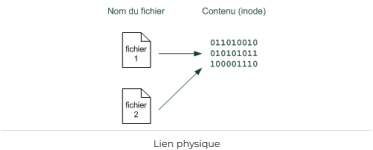
C'est tout ce que vous avez besoin de savoir pour comprendre la suite.

Nous allons maintenant découvrir comment créer des liens physiques puis des liens symboliques.

### Créer des liens physiques

Ce type de lien est plus rarement utilisé que le lien symbolique, mais il faut tout de même le connaître car il peut se révéler pratique.

Un lien physique permet d'avoir deux noms de fichiers qui partagent exactement le même contenu, c'est-à-dire le même inode (figure suivante).



Ainsi, que vous passiez par `fichier1` ou par `fichier2`, vous modifiez exactement le même contenu. En quelque sorte, **le fichier est le même**. On peut juste y accéder via deux noms de fichiers différents.

On ne peut pas créer de liens physiques sur des répertoires. Cela ne fonctionne qu'avec les fichiers. Il existe des options pour que ça fonctionne avec des répertoires, mais c'est un peu particulier et on n'en parlera pas. Pour faire un « raccourci » vers un répertoire, on préférera utiliser un lien symbolique.

Pour créer un lien physique, nous allons utiliser la commande `ln`. Je vous propose tout d'abord de créer un répertoire pour nos tests :

```
mkdir tests
cd tests
```

Une fois dans ce dossier, créez un fichier avec la commande `touch` par exemple :

```
touch fichier1
```

Nous voulons maintenant créer un lien physique : nous allons créer un `fichier2` qui partagera le même inode (le même contenu) que `fichier1`. Tapez :

```
ln fichier1 fichier2
```

Si vous listez les fichiers du répertoire, vous avez l'impression d'avoir deux fichiers différents :

```
mateo21@mateo21-desktop:~/tests$ ls -l
total 0
-rw-r--r-- 2 mateo21 mateo21 0 2008-07-31 13:55 fichier1
-rw-r--r-- 2 mateo21 mateo21 0 2008-07-31 13:55 fichier2
```

A priori, rien ne nous permet ici de deviner que ces fichiers modifient le même contenu. Le lien physique est donc un lien dur, pas évident à détecter au premier coup d'œil.

La seconde colonne de la liste (qui indique « 2 » pour chacun des fichiers) correspond au nombre de fichiers qui partagent le même inode. C'est le seul indice qui vous permet de savoir que quelqu'un a fait un lien physique, mais vous ne pouvez pas savoir lequel. Le seul moyen de vérifier que ces fichiers partagent le même contenu, c'est de faire `ls -i` pour afficher les numéros d'inode correspondants et de vérifier que ces deux fichiers sont associés au même inode. En temps normal, sur la plupart des fichiers la seconde colonne indique donc « 1 ». Si c'est un dossier, ce nombre indique en revanche le nombre de fichiers à l'intérieur.

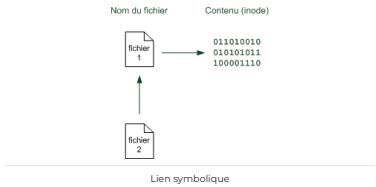
Si vous supprimez un des deux fichiers, l'autre fichier reste en place et le contenu sera toujours présent sur le disque. L'inode est supprimé uniquement quand plus aucun nom de fichier ne pointe dessus.

En clair, supprimez `fichier1` pour voir. Vous verrez que `fichier2` existe toujours et qu'il affiche toujours le même contenu. Il faut supprimer `fichier1` ET `fichier2` pour supprimer le contenu.

### Créer des liens symboliques

Les liens symboliques ressemblent plus aux « raccourcis » dont vous avez peut-être l'habitude sous Windows. La plupart du temps, on crée des liens symboliques sous Linux pour faire un raccourci, et non des liens physiques qui sont un peu particuliers.

Le principe du lien symbolique est que l'on crée un lien vers un autre nom de fichier. Cette fois, on pointe vers le nom de fichier et non vers l'inode directement (figure suivante).



Supprimez le `fichier2` que nous avons créé tout à l'heure (sous forme de lien physique) :

```
rm fichier2
```

Créons maintenant un nouveau `fichier2`, cette fois sous forme de lien symbolique. On utilise là encore la commande `ln`, mais avec le paramètre `-s` (s comme symbolique) :

```
ln -s fichier1 fichier2
```

Cette fois, la commande détaillée `ls -l` sera beaucoup plus précise :

```
mateo21@mateo21-desktop:~/tests$ ls -l
total 0
-rw-r--r-- 1 mateo21 mateo21 0 2008-07-31 13:55 fichier1
lrwxrwxrwx 1 mateo21 mateo21 8 2008-07-31 14:15 fichier2 -> fichier1
```

On note deux choses :

- la toute première lettre de la seconde ligne est un `l` (comme *link* c'est-à-dire lien) ;
- tout à la fin de la seconde ligne, une flèche montre clairement que `fichier2` pointe vers `fichier1`.

Bref, les liens symboliques sont beaucoup plus faciles à repérer que les liens physiques !

Ok, mais quelles différences à part ça ? Le résultat revient au même, non ? Qu'on ouvre `fichier1` ou `fichier2`, on éditera le même contenu au final !

Tout à fait. Il y a quand même quelques subtilités :

- par exemple, si vous supprimez `fichier2`, il ne se passe rien de mal. Par contre, si vous supprimez `fichier1`, `fichier2` pointera vers un fichier qui n'existe plus. Le lien symbolique sera cassé et ne servira donc plus à rien. On parle de « lien mort » ;
- d'autre part, l'avantage des liens symboliques est qu'ils fonctionnent aussi sur des répertoires, contrairement aux liens physiques.

### En résumé

- `cat` permet d'afficher tout le contenu d'un fichier, mais lorsque celui-ci est long, il est préférable d'utiliser `less` qui affiche le fichier page par page.
- On peut obtenir uniquement le début ou la fin d'un fichier avec `head` et `tail`. En utilisant `tail -f` on peut suivre l'évolution d'un fichier en temps réel, ce qui est utile sur les fichiers de log qui enregistrent l'activité du système.
- `mkdir` permet de créer un dossier, `touch` permet de créer un fichier vide.
- `cp` permet de copier un fichier ou un dossier, tandis que `mv` permet de les déplacer ou de les renommer.
- `rm` supprime un fichier. Il n'y a pas de corbeille en console, la suppression est définitive ; il faut donc être prudent.
- On peut créer des liens (raccourcis) vers des fichiers et dossiers à l'aide de la commande `ln`.

Que pensez-vous de ce cours ?

J'AI TERMINÉ CE CHAPITRE ET JE PASSE AU SUIVANT

[LA STRUCTURE DES DOSSIERS ET FICHIERS](#)

[LES UTILISATEURS ET LES DROITS](#)

**Le professeur**

**Mathieu Nebra**  
Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

**Découvrez aussi ce cours en...**

Livre

PDF

---

OPENCLASSROOMS

Qui sommes-nous ?

Financements

Expérience de formation

Forum


Blog 

Presse 

---

OPPORTUNITÉS

Nous rejoindre 

Devenir mentor 

Devenir coach carrière 

---

AIDE



FAQ

---

POUR LES ENTREPRISES

Former et recruter

---

EN PLUS

Boutique 

Mentions légales


Conditions générales d'utilisation

Politique de protection des données personnelles

Cookies

Accessibilité

---

 Français 

