# ln

- Objet: commande ln
- Niveau requis :
  débutant, avisé
- Commentaires : *Permet de créer des liens physiques ou symboliques*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !
  - La commande chevron ">"
  - La commande ls
  - La commande rm
  - Les droits
  - La commande d'édition nano
- Suivi :
  à-compléter
  à-tester
  - Création par smolski le 21/06/2010
  - Testé par .... le ....
- Commentaires sur le forum : C'est ici[1)]

## Introduction

La commande `ln` établit un lien symbolique ou un lien physique entre 2 fichiers.

Son utilisation correspond à une redirection **active** entre deux *fichiers* désignés.

Les modifications opérées *sur/dans* l'un sont visibles immédiatement depuis l'un ou l'autre des fichiers liés. Autrement dit, un lien est un type spécial de fichier qui permet à plusieurs noms de fichiers de faire référence au même contenu sur un disque.

## Synopsis

```
ln [option] <cible> <lien>
```

**Lien symbolique :**

```
ln -s <fichier_cible> <fichier_symbole>
ln -s <répertoire_cible> <répertoire_symbole>
ln -s <point_cible> <point_symbole>
```

**Lien physique :**

```
ln <fichier_cible> <fichier_lié>
```

**ATTENTION :** Un lien physique à la particularité de devoir se situer sur le même système de fichier que sa cible !

# Description

On distingue 2 sortes de lien : les liens durs et les liens symboliques.

#### Les liens symboliques (symlink)

Un lien symbolique est constitué d'un réel fichier de petite taille ; il contient le nom du fichier auquel il correspond. Par conséquent, la suppression du fichier d'origine, c'est-à-dire la cible du lien, rendra le lien symbolique inutilisable puisqu'il ne correspondra plus à un fichier valide ; le lien sera alors brisé.

#### Les liens physiques ou durs (hardlink)

Un lien dur associe deux ou plusieurs fichiers à un même espace sur le disque tout en préservant leurs indépendances lors de modifications de leurs contenus. De même, la suppression d'un fichier n'affectera pas l'autre.

**Nota :**
Il semble que le système de fichier *privateur* `ntfs` ne soit pas capable de gérer des `hardlink` (ni des `symlink`) avec lui-même, sinon d'une manière incomplète et *très très à la con* en fait.

*Merci de ces remarques habiles au concert de* **Mahoru`Tsunemi** *et de* **captnfab** *sur l'IRC.*

# Illustration pratique

Afin de bien comprendre les principes que nous avons décrits ci-haut, nous allons illustrer tout cela à l'aide de quelques exemples simples.

Débutant - À savoir :

- La commande chevron ">"
- La commande LS
- La commande RM
- Les Droits
- La commande d'édition NANO

*Et oui, tout ça cher Débutant ! Il reste que la plupart de ces commandes sont les commandes en ligne principales usuelles à utiliser dans un terminal.*

## Créer les répertoires et fichiers tests que nous allons utiliser

Dans `/home/votre_user/`, créez un répertoire nommé *test_lien* avec mkdir et s'y positionner avec

cd :

```
cd ~
```

```
mkdir test_lien
```

```
cd test_lien/
```

```
 test_lien$
```

Créer dans ce répertoire un fichiers vide, *test1.txt*, avec touch, puis ajoutez-y le texte *Bonjour toto* avec la commande echo.

```
test_lien$ touch test1.txt
```

```
test_lien$ echo Bonjour Jojo > test1.txt
```

On vérifie avec la commande cat que notre texte est bien écrit dans le fichier test1.txt :

```
test_lien$ cat test1.txt
```

retour de la commande

```
Bonjour Jojo
```

## Créer un lien symbolique (symlink) :

Situation de départ :

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
-rw-r--r-- 1 martin martin 13 déc 14 22:02 test1.txt
```

On crée le lien symbolique :

```
test_lien$ ln -s test1.txt lien1.txt
```

Ce qui nous fait maintenant :

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
lrwxrwxrwx 1 martin martin  9 déc 14 22:05 lien1.txt -> test1.txt
-rw-r--r-- 1 martin martin 13 déc 14 22:02 test1.txt
```

Attardons-nous un peu sur la ligne : **l**rwxrwxrwx 1 martin martin **9** déc 14 22:05 lien1.txt **→** test1.txt

1. **l** indique un lien symbolique sur ce fichier
2. **9** indique l'occupation réelle du fichier sur le disque
3. **→ test1.txt** indique le fichier pointé par le lien symbolique

Précisons aussi que :

- Un *fichier symbole* créé avec la commande `ln` ne contient pas l'occupation disque affichée qui est contenue dans le fichier cible.
- Les modifications qui sont portées dans le contenu apparent d'un fichier symbole seront portés uniquement dans le fichier cible.
- La destruction totale d'un fichier symbole ne détruit rien dans le fichier cible.

## Modification du contenu des fichiers liés

```
test_lien$ echo Comment vas-tu ? >> lien1.txt
```

```
test_lien$ cat lien1.txt
```

retour de la commande

```
Bonjour toto
Comment vas-tu ?
```

```
test_lien$ cat test1.txt
```

retour de la commande

```
Bonjour toto
Comment vas-tu ?
```

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
lrwxrwxrwx 1 martin martin  9 déc 14 22:05 lien1.txt -> test1.txt
-rw-r--r-- 1 martin martin 30 déc 14 22:08 test1.txt
```

Nous pouvons voir que seul le fichier cible *test1.txt* s'est trouvé modifié, passant de **13** à **30** octets d'occupation !

**Modifier directement le fichier cible donnera le même résultat :**

1. nul pour le fichier symbole
2. effectif pour le fichier cible

### Occupation concrète sur le disque

À l'aide de la commande `ls -li`, nous pouvons voir l'occupation disque générée par un lien *symlink* en constatant les chiffres indiquant les index respectifs des inodes de chacun des fichiers liés :

```
martin@madebian:~/test_lien$ ls -li
```

retour de la commande

```
total 4
8667148 lrwxrwxrwx 1 martin martin  9 déc 14 22:05 lien1.txt ->
test1.txt
8667139 -rw-r--r-- 1 martin martin 30 déc 14 22:08 test1.txt
```

Cela implique qu'il y a bien deux occupations différentes sur le disque, avec une occupation qui restera toujours fixe et minimale pour le fichier symbole. 😀

## Modification par effacement du fichier symbole

L'effacement du fichier symbole `lien1.txt` ne détruit ni le contenu, ni l'apparence du fichier cible `test1.txt` :

```
test_lien$ rm lien1.txt
```

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
-rw-r--r-- 1 martin martin 30 déc 14 22:08 test1.txt
```

## Modification du nom des fichiers liés :

Recréons le même fichier symbole *lien1.txt* :

```
test_lien$ ln -s test1.txt lien1.txt
```

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
lrwxrwxrwx 1 martin martin  9 déc 14 22:13 lien1.txt -> test1.txt
-rw-r--r-- 1 martin martin 30 déc 14 22:08 test1.txt
```

### Modifier le nom du fichier symbole

Modifions le nom du fichier symbole lien1.txt à l'aide de la commande mv :

```
test_lien$ mv lien1.txt lien2.txt
```

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
lrwxrwxrwx 1 martin martin  9 déc 14 22:13 lien2.txt -> test1.txt
-rw-r--r-- 1 martin martin 30 déc 14 22:08 test1.txt
```

Tout reste fonctionnel et en place. 😀


Pour poursuivre cette illustration dans la clarté, remettons le nommage du fichier symbole `lien2.txt` en `lien1.txt`.

```
test_lien$ mv lien2.txt lien1.txt
```

### Modifier le nom du fichier cible

Modifions le nom du fichier cible `test1.txt` :

```
test_lien$ mv test1.txt test2.txt
```

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
```

```
lrwxrwxrwx 1 martin martin  9 déc 14 22:13 lien1.txt -> test1.txt
-rw-r--r-- 1 martin martin 30 déc 14 22:08 test2.txt
```

*Patatras !* Nous voyons alors que le fichier cible `lien1.txt` se met en carafe - il devient écrit en rouge dans le terminal bash) dès le *rafraîchissement* du terminal ouvert.

Il suffit :

1. de renommer le fichier cible `test2.txt` en `test1.txt`,

~~- ou bien de recréer le fichier symbole `lien1.txt` en le pointant sur le fichier cible `test2.txt` par reformulation de la commande : test_lien$ ln -s test2.txt lien1.txt~~

Et tout redevient effectif.

Pour la suite du tuto, choisir de remettre le fichier en test1.txt

```
test_lien$ mv test2.txt test1.txt
```

## Modifications des droits des fichiers liés :

Depuis le début de cette illustration, nous voyons que les droits restent immuablement complets pour le fichier symbole `lien1.txt`.

```
 rwx rwx rwx
```

Essayons de les modifier.
Modifions la propriété du fichier symbole `lien1.txt` pour que root devienne le propriétaire.

### Modification des droits d'un fichier symbole

Situation de départ :

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
lrwxrwxrwx 1 martin martin  9 déc 14 23:00 lien1.txt -> test1.txt
-rw-r--r-- 1 martin martin 13 déc 14 22:59 test1.txt
```

En terminal root, changez les droits sur le fichier avec la commande chmod :

```
test_lien$ chmod 700 lien1.txt
```

On obtient alors :

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
lrwxrwxrwx 1 martin martin  9 déc 14 23:00 lien1.txt -> test1.txt
-rwx------ 1 martin martin 13 déc 14 22:59 test1.txt
```

Nous voyons que le fichier symbole `lien1.txt` n'est pas affecté par cette modifications des droits :

```
lrwxrwxrwx 1 martin martin  9 déc 14 23:00 lien1.txt -> test1.txt
```

et que le fichier cible `test1.txt` s'en trouve directement rectifié par cette commande exécuter sur le fichier symbole `lien1.txt` :

```
rwx------ 1 martin martin 13 déc 14 22:59 test1.txt
```

## Créer un lien physique (hardlink) :

Un lien physique est la création d'un fichier à l'identique de celui qu'il pointe.

Mais outre d'évoluer à l'identique, comme avec un lien symbolique, les deux fichiers se nourrissent mutuellement et concrètement de toutes les modifications apportées à l'un ou à l'autre.

**ATTENTION :** Un lien physique à la particularité de devoir se situer sur le même système de fichier que sa cible !

## Créer un lien physique

Commençons par supprimer les fichiers liés symboliquement et créer un fichier de départ vide, test2, :

```
test_lien$ rm *1.txt
```

retour de la commande

```
touch test2.txt
```

Au départ, nous avons donc la situation suivante :

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 0
-rw-r--r-- 1 martin martin 0 déc 15 00:02 test2.txt
```

Pour créer un lien physique, on utilise la commande `ln` sans l'option `-s`.

```
martin@madebian:~/test_lien$ ln test2.txt lien2.txt
```

```
martin@madebian:~/test_lien$ ls -lv
```

retour de la commande

```
total 0
-rw-r--r-- 2 martin martin 0 déc 15 00:02 lien2.txt
-rw-r--r-- 2 martin martin 0 déc 15 00:02 test2.txt
```

Ah ! Plus de lettre “`l`” ni de “*flèches*” pour indiquer le lien physique.

Nous pouvons toutefois distinguer un changement dans le listage des droits des fichiers liés physiquement :

Le chiffre 1 devient 2 dans la ligne du fichier cible test2.txt :

Pareillement dans la ligne du fichier symbole physique lien2.txt créé :

```
-rw-r--r-- **2** martin martin 0 déc 15 00:02 lien2.txt
```

Une autre différence est le partage des droits qui sont là tout à fait identiques entre les fichiers liés physiquement.

## Modifions le contenu des fichiers liés physiquement

### Ajout

Ajoutons du contenu dans l'un puis un second ajout dans l'autre des fichiers liés *physiquement* en vérifiant les contenus à chaque fois :

```
martin@madebian:~/test_lien$ echo Très bien titi ! > lien2.txt
```

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 8
-rw-r--r-- 2 martin martin 18 déc 15 13:49 lien2.txt
-rw-r--r-- 2 martin martin 18 déc 15 13:49 test2.txt
```

```
martin@madebian:~/test_lien$ cat test2.txt
```

retour de la commande

```
Très bien titi !
```

```
martin@madebian:~/test_lien$ echo Et toi Jojo ? >> test2.txt
```

```
martin@madebian:~/test_lien$ cat lien2.txt
```

retour de la commande

```
Très bien titi !
Et toi Jojo ?
```

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 8
-rw-r--r-- 2 martin martin 32 déc 15 13:52 lien2.txt
-rw-r--r-- 2 martin martin 32 déc 15 13:52 test2.txt
```

Toutes les modifications, ajoutées ou retranchées dans chacun des fichiers agira de même dans l'autre.

### Occupation concrète sur le disque

À l'aide de la commande `ls -li`, nous pouvons voir l'occupation disque générée par un lien *hardlink* en constatant les chiffres indiquant les index respectifs des inodes de chacun des fichiers liés :

```
martin@madebian:~/test_lien$ ls -li
```

retour de la commande

```
total 8
8667149 -rw-r--r-- 2 martin martin 32 déc 15 13:52 lien2.txt
8667149 -rw-r--r-- 2 martin martin 32 déc 15 13:52 test2.txt
```

Ah ! Le même index d'inode pour les deux fichiers !

Cela implique qu'il n'y a pas deux occupations différentes sur le disque mais une seule occupation vers laquelle pointe les deux fichiers liés en même temps ! 😀

*On va pas s'gêner avec pour les gonfler de données car cela ne doublera pas l'occupation physique du disque !* 😀

#### Suppression

Suppression du fichier cible `test2.txt` :

```
test_lien$ rm test2.txt
```

```
martin@madebian:~/test_lien$ ls -l
```

retour de la commande

```
total 4
-rw-r--r-- 1 martin martin 32 déc 15 13:52 lien2.txt
```

Comme dit le captnfab dans sa relecture :
Il est à remarquer que le chiffre 2 est passé à 1 car il n'y a plus de second fichier lié.

Chacun des fichiers liés par un lien physique survit nominalement et physiquement à l'effacement de l'autre.

Pour supprimer définitivement les deux fichiers et leurs contenus, nous devons les effacer tous deux.

```
test_lien$ rm *2.txt
```

```
test_lien$ ls -l
```

retour de la commande

```
total 0
```

*Voilà pour la commande* `ln` *dans tous ses états !*

*Merci au* **captnfab** *pour sa bienveillante et rigolote attention !* 😀

## Conclusion

Les liens sont utiles si vous souhaitez qu'un fichier apparaisse dans plusieurs répertoires, ou sous un nom différent.

Si le fichier a une assez grande taille vous pouvez envisager, au lieu de copier dans un répertoire donné, de créer un lien réduisant ainsi l'utilisation d'espace disque.

Autre point très intéressant, créer des liens, au lieu de copier les fichiers, assure que toute modification dans un fichier se retrouvera bien dans les « copies » dispersées un peu partout.

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !