

03B : Fichiers, liens et permissions

420-3A6-ST : Programmation système

Dany Gagnon

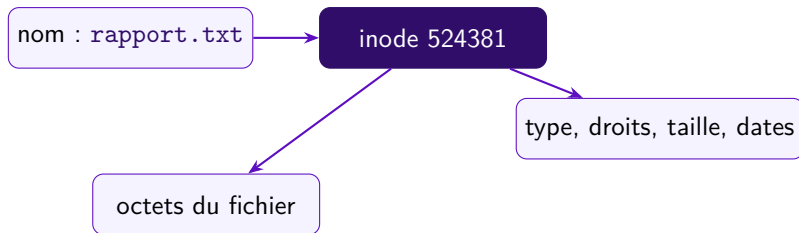
Cégep de Saint-Jean-sur-Richelieu

Mercredi 9 septembre 2026

Ce que vous saurez expliquer

- Ce que partagent deux liens physiques et ce que contient un lien symbolique.
- Pourquoi un fichier lisible peut rester inaccessible.
- Quels droits modifier pour une opération précise.
- Comment vérifier une hypothèse avec `stat`, `id` et `chmod`.

Un répertoire associe un nom à un inode



Le nom se trouve dans le répertoire. L'inode porte l'identité et les métadonnées du fichier.

Observer l'identité avec ls et stat

```
$ ls -li rapport.txt  
524381 -rw-r--r-- 1 dany dany 18 ... rapport.txt
```

```
$ stat rapport.txt  
File: rapport.txt  
Size: 18          Blocks: 8          IO Block: 4096 regular file  
Device: ...      Inode: 524381    Links: 1  
Access: (0644/-rw-r--r--)  Uid: (...)  Gid: (...)
```

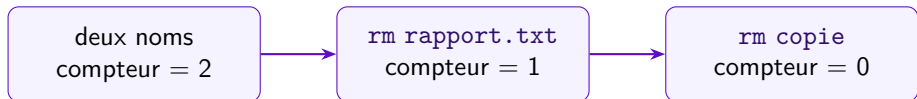
`ls -li` montre le numéro d'inode. `stat` rassemble les métadonnées sans afficher le contenu.

Un lien physique ajoute un nom



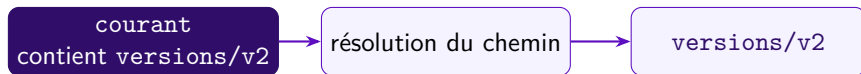
Aucun des deux noms n'est « l'original ». Ils désignent la même identité.

Supprimer un nom ne supprime pas toujours le fichier



Les données peuvent être libérées quand le dernier lien disparaît et qu'aucun processus ne garde le fichier ouvert.

Un lien symbolique contient un chemin



Le lien symbolique possède son propre inode. Quand on le suit, le noyau résout le chemin qu'il contient.

Lien absolu ou relatif ?

Chemin relatif

Interprété depuis le répertoire qui contient le lien.

Il survit souvent mieux au déplacement de toute l'arborescence.

Chemin absolu

Commence par / et vise un endroit précis du système.

Il devient faux si l'arborescence est déplacée ailleurs.

Le choix dépend de ce qui doit pouvoir être déplacé.

Comparer les deux liens dans un dossier neuf

```
ln -s rapport.txt raccourci
ls -li rapport.txt copie.txt raccourci
readlink raccourci
rm rapport.txt
cat copie.txt
cat raccourci
```

Le lien physique reste utilisable ; le lien symbolique pointe maintenant vers un chemin absent. Un lien physique reste dans le même système de fichiers.

Observer son identité avec id

```
$ id
uid=1000(dany) gid=1000(dany) \
groups=1000(dany),27(sudo),1001(developpement)
```

```
$ id -u
```

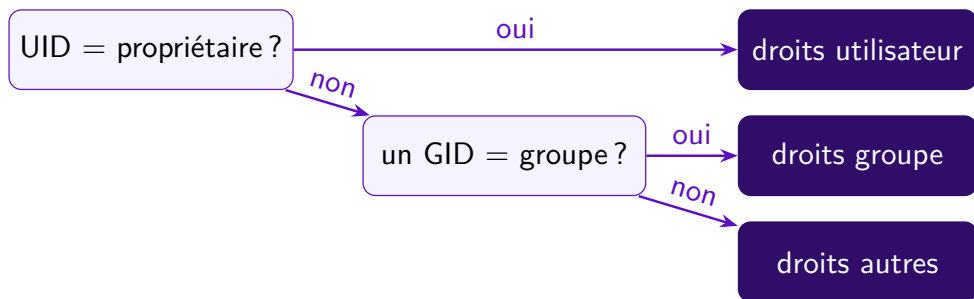
```
1000
```

```
$ id -Gn
```

```
dany sudo developpement
```

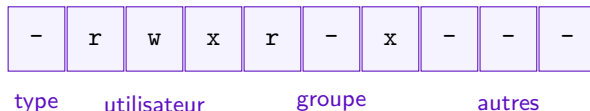
L'appartenance à un groupe permet de partager des fichiers sans leur donner accès à tous les utilisateurs.

Le noyau choisit une seule classe



Les classes ne s'additionnent pas. Une fois la classe choisie, seuls ses trois bits sont examinés.

Découper -rwxr-x--



Chaque classe reçoit trois positions : lire **r**, écrire **w**, exécuter ou traverser **x**.

Sur un fichier régulier

Droit	Permet	Ne garantit pas
r	lire les octets	modifier le contenu
w	modifier les octets	supprimer le nom
x	demander son exécution	qu'il s'agit d'un programme valide

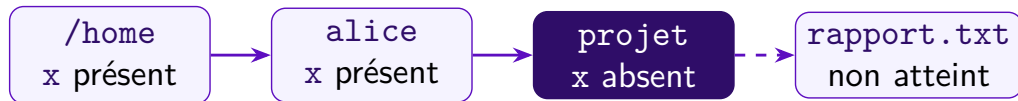
La suppression d'un nom modifie le répertoire qui le contient. Elle ne dépend donc pas directement du droit **w** du fichier.

Sur un répertoire

Droit	Permet
<code>r</code>	lire la liste des noms
<code>w</code>	ajouter ou retirer des entrées
<code>x</code>	traverser le répertoire et atteindre une entrée connue

Pour créer ou supprimer une entrée, il faut habituellement `w` et `x` sur le répertoire.

Chaque répertoire du chemin doit être traversable



Un seul répertoire non traversable suffit à bloquer la résolution du chemin.

Notation symbolique de chmod

Une modification symbolique indique :

classe + opération + droits

Classe	Opération	Droits
u, g, o, a	+, -, =	r, w, x

Exemples : `chmod u+x script.sh`, `chmod go-r secret.txt`, `chmod g=rw rapport.txt`.

Notation octale

Chaque droit correspond à un bit : $r=4$, $w=2$, $x=1$.

Valeur	Droits	Valeur	Droits
0	--	4	r-
1	-x	5	r-x
2	-w-	6	rw-
3	-wx	7	rwX

`chmod 640 rapport.txt` signifie `rw-r---`.

Corriger un accès refusé

```
mkdir coffre  
printf 'message\n' > coffre/message.txt  
chmod 600 coffre  
ls coffre  
cat coffre/message.txt  
chmod u+x coffre  
cat coffre/message.txt
```

Faites cet essai avec votre utilisateur ordinaire. Le bit `x` du répertoire permet d'atteindre le fichier.

Pourquoi un fichier neuf n'est pas exécutable

Plusieurs outils demandent au plus :

- 0666 pour un fichier régulier, soit lecture et écriture ;
- 0777 pour un répertoire, car la traversée doit être possible.

Avec `umask 0022` :

$$0666 \ \& \sim 0022 = 0644 \qquad 0777 \ \& \sim 0022 = 0755$$

Le masque retire des droits. Il n'ajoute jamais le bit d'exécution.

Observer l'effet dans un sous-shell

```
$ bash -c 'umask 0022; touch public; mkdir dossier-public'
$ bash -c 'umask 0077; touch prive; mkdir dossier-prive'
$ ls -ld public prive dossier-public dossier-prive
-rw----- ... prive
-rw-r--r-- ... public
drwx----- ... dossier-prive
drwxr-xr-x ... dossier-public
```

Le sous-shell évite de changer le masque de la session de travail actuelle.

Propriétaire et administration

- Le propriétaire peut modifier les droits avec `chmod`.
- `chgrp` choisit un groupe autorisé ; changer de propriétaire avec `chown` demande habituellement une autorité administrative.
- `sudo commande` utilise les droits permis par la configuration pour cette commande.
- Pour les essais de fichiers et de permissions, travaillez avec votre compte ordinaire dans un dossier temporaire.

Une archive transporte aussi les liens et les droits

- `tar` rassemble une arborescence dans un fichier ; `gzip` la compresse.
- L'archive garde les permissions Unix et le chemin des liens symboliques.
- Les liens physiques entre les fichiers inclus sont conservés.

Git transporte `remise.tar.gz` comme un fichier. Les propriétés du dossier restent à l'intérieur de l'archive.

Créer l'archive depuis le dépôt de l'exercice

Le travail terminé se trouve dans `terrain/atelier/`.

```
tar -czf remise.tar.gz -C terrain atelier
```

<code>-c</code>	créer une archive
<code>-z</code>	compresser avec gzip
<code>-f remise.tar.gz</code>	nommer le fichier de sortie
<code>-C terrain</code>	lire les fichiers depuis ce répertoire
<code>atelier</code>	dossier à inclure, avec son contenu

L'archive est créée dans le dépôt, à côté de `terrain/`.

Vérifier ce qui sera remis

```
tar -tvzf remise.tar.gz
```

`-t` liste sans extraire ; `-v` affiche les détails.

- Les chemins commencent par `atelier/`.
- Les lignes affichent les droits des fichiers et des répertoires.
- `courant -> rapport.txt` indique le lien symbolique relatif.
- `link to` signale un lien physique vers un autre fichier de l'archive.

Tester la remise dans un dossier neuf

```
mkdir verification
tar --same-permissions --no-same-owner \
    -xzf remise.tar.gz -C verification
ls -li verification/atelier/equipe/rapport.txt \
    verification/atelier/livraison/rapport-final.txt
readlink verification/atelier/equipe/courant
```

-x extrait; **-same-permissions** restaure les droits enregistrés. **-no-same-owner** garde votre compte comme propriétaire.

Les deux noms du rapport doivent partager un inode dans ce nouveau dossier. Son numéro peut différer de celui du terrain.

Remettre l'exercice 03 dans GitLab

- ❶ Compléter `observations.md` : manipulations guidées et questions de compréhension.
- ❷ Refaire `remise.tar.gz` après les dernières modifications et lancer `make test`.
- ❸ Depuis le dépôt de l'exercice :

```
git add remise.tar.gz observations.md
git commit -m "Remise de l'exercice 03"
git push
```

Vérifier dans GitLab la présence des deux fichiers et le résultat des tests. Remise : mercredi 16 septembre, 23 h 59.

Atelier : remettre un dossier partageable

- ① Créer un rapport, un lien physique et un lien symbolique relatif.
- ② Observer l'effet d'une modification, puis du retrait du premier nom.
- ③ Rendre un répertoire non traversable, expliquer l'échec et le corriger.
- ④ Comparer des créations avec les masques 0022 et 0077.

Pour chaque essai : prédiction, commande, observation, explication.

Relier ce qui vient d'être observé

- Le répertoire associe un nom à un inode ; les droits concernent l'objet.
- Deux liens physiques partagent donc les mêmes droits et les mêmes octets.
- Traverser un chemin et lire un fichier sont deux vérifications différentes.
- Le prochain cours utilisera Bash pour automatiser ces manipulations.