Les fichiers réguliers

Julien Baste

IUT de Lille

Séance 02

2025/2026

Fichiers réguliers

Un fichier régulier est un fichier... qui n'a rien de particulier !

- qui n'est donc ni un catalogue, ni un lien, ni un fichier spécial, ni un tube, ni une socket.
- ça peut être un fichier texte, une image, un document pdf, etc.

Quelques commandes utiles concernant les fichiers :

- `cat` → affiche le contenu d'un fichier
- `file` → donne la convention de structure du fichier
- `stat` → afficher les caractéristiques de base d'un fichier
- `touch` → crée un fichier vide
- `diff` → donne la différence entre le contenu de deux fichiers
- `nano` → permet d'éditer un fichier texte

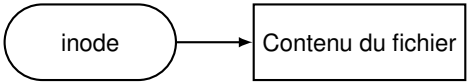
Les informations de `stat` : L'Inode

```
[baste@orkad ~]$ stat Mon_super_fichier
  File: Mon_super_fichier
  Size: 43          Blocks: 8          IO Block: 4096   regular file
Device: 253,1  Inode: 30816038    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   baste)   Gid: ( 1000/   baste)
Access: 2025-09-02 10:59:39.496740600 +0200
Modify: 2025-09-02 10:59:39.495740600 +0200
Change: 2025-09-02 10:59:39.495740600 +0200
 Birth: 2025-09-02 10:59:39.495740600 +0200
```

Toutes ces informations constituent l'inode associé au fichier `Mon_super_fichier`.

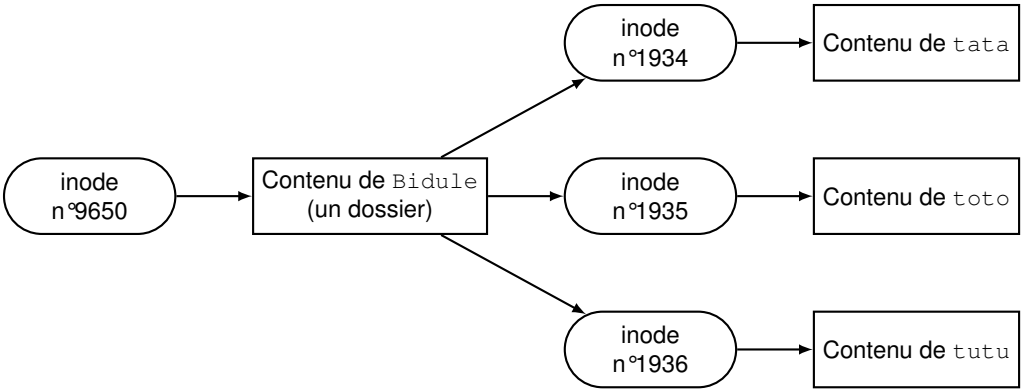
- L'inode est l'ensemble des informations servant à définir ou décrire un fichier.
- Chaque fichier possède un et un seul inode.
- Un inode est associé à un seul fichier
- Un inode identifie un fichier

En mémoire :



Inodes et dossier

Si le dossier `Bidule` contient les fichiers `tata`, `toto`, et `tutu`.



Les liens physiques

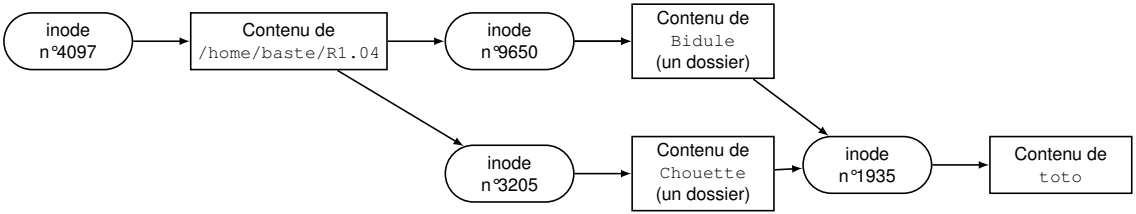
Le lien physique est le lien entre un dossier et les fichiers qu'ils contient.

Il est possible pour un fichier d'être dans plusieurs dossiers (d'avoir plusieurs liens physiques).

Pour cela il suffit de créer le lien : `ln` ⟨`cible`⟩ ⟨`destination`⟩

Ici `toto` existait dans le dossier `Bidule`, on veut qu'il soit aussi dans `Chouette` :

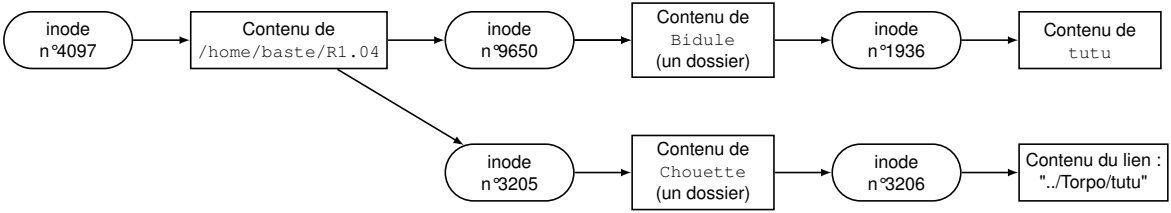
Depuis le dossier `Chouette` : `ln ../Bidule/toto ./`



Les liens symboliques

Un lien symbolique est un fichier (de type lien) qui contient le chemin et le nom d'un autre fichier.

Création depuis le dossier `Chouette` : `ln -s ../Bidule/tutu ./`



Manipulation du système de fichiers

Les commandes de manipulation des fichiers ont souvent la syntaxe suivante :

`commande` ⟨`source`⟩`...` ⟨`destination`⟩

où ⟨source⟩ et ⟨destination⟩ désigne chacun un chemin

- `cp` → copie un fichier :
  - Si ⟨`destination`⟩ existe : copie la source dans la destination, garde le même inode.
  - Si ⟨`destination`⟩ n'existe pas :
    - crée un nouveau fichier, son inode associé (ainsi que la création d'une entrée dans un dossier) et
    - copie la source dans ce nouveau fichier
- `mv` → déplace (et renomme) un fichier :
  - Conservation du même inode pour le fichier source
  - Supprime l'entrée dans le dossier source et crée une entre dans le dossier destination
  - Si ⟨`destination`⟩ existe : il est supprimé
- `ln` → surnomme un fichier :
  - création (ou modification) d'une nouvelle entrée dans un répertoire

Utilisateurs/Groupes

Unix est un système multi-utilisateurs. Les utilisateurs y sont rassemblés par groupe. Chaque utilisateur est donc identifié par le système par :

1. son *login* ..... au niveau noyau c'est un numéro unique : l'`uid`
2. son *groupe* ..... au niveau noyau c'est un numéro unique : le `gid`

Le système gère la correspondance entre identifiant symbolique et numérique via des fichiers textes :

- login et `uid` via le fichier `/etc/passwd`
- groupe et `gid` via le fichier `/etc/group`

Un utilisateur peut appartenir à plusieurs groupes, mais possède un groupe principal (spécifié dans le fichier `/etc/passwd`) dans lequel il est enregistré lors de chaque connexion.

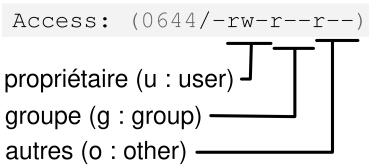
Les informations de `stat` : Les propriétaires

```
[baste@orkad ~]$ stat Mon_super_fichier
  File: Mon_super_fichier
  Size: 43          Blocks: 8          IO Block: 4096   regular file
Device: 253,1  Inode: 30816038    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   baste)   Gid: ( 1000/   baste)
Access: 2025-09-02 10:59:39.496740600 +0200
Modify: 2025-09-02 10:59:39.495740600 +0200
Change: 2025-09-02 10:59:39.495740600 +0200
 Birth: 2025-09-02 10:59:39.495740600 +0200
```

Chaque fichier est possédé par deux entités, un utilisateur et un groupe :

- `uid` → identifiant (et nom) de l'utilisateur propriétaire
- `gid` → identifiant (et nom) du groupe propriétaire

Les informations de `stat` : Les droits d'accès



Chaque fichier possède des droits d'utilisation applicables :

1. à son propriétaire
2. aux utilisateurs appartenant à son groupe
3. aux utilisateurs n'appartenant pas à son groupe

Pour chacune de ces trois catégories, il existe trois types de droits :

1. lecture (r) :
   - autorise la lecture du contenu du fichier
2. écriture (w) :
   - autorise la modification du contenu du fichier
3. exécution/franchissement (x) :
   - autorise l'exécution d'un fichier régulier,
   - permet de traverser un répertoire

Le mode d'utilisation d'un fichier est l'ensemble de ses droits d'accès.

La commande `chmod` permet au propriétaire d'un fichier de modifier son mode d'utilisation.

La syntaxe de `chmod` est la suivante :

`chmod` ⟨`mode`⟩ ⟨`fichiers`⟩

Le mode peut être précisé de deux manières :

- via la spécification des modifications à effectuer sur le mode courant :
  → forme **symbolique**.
- via la spécification complète du nouveau mode :
  → forme **numérique octale** (base 8)

`chmod` (forme symbolique)

Les modifications à effectuer sur le mode courant sont spécifiées par un code dont la syntaxe est :

⟨personne⟩⟨action⟩⟨accès⟩

⟨personne⟩	
`u`	propriétaire
`g`	groupe
`o`	autres
`a`	tous

⟨action⟩	
`+`	ajouter
`-`	enlever
`=`	initialiser

⟨accès⟩	
`r`	lecture
`w`	écriture
`x`	exécution/franchissement

- Il ne peut y avoir qu'une action par code
- Plusieurs modifications peuvent être spécifiées si elles sont séparées les unes des autres par des virgules «`,`».

Exemple : `chmod u-r Mon_super_fichier` va retirer à l'utilisateur les droits de lecture.

chmod (forme numérique octale)

Les différentes combinaisons de droits d'accès peuvent être représentées par :

symbolique	binaire	octal
`---`	000	0
`--x`	001	1
`-w-`	010	2
`-wx`	011	3
`r--`	100	4
`r-x`	101	5
`rw-`	110	6
`rwx`	111	7

Le mode d'un fichier peut alors être spécifié par un nombre en base 8, dont les chiffres représentent, de gauche à droite, les droits d'accès pour :

1. le propriétaire du fichier
2. les membres du groupe du fichier
3. les autres utilisateurs

`chmod` (forme numérique octale) : exemples

755 représente les droits `rwx r-x r-x`

777 représente les droits `rwx rwx rwx`

644 représente les droits `rw- r- r-`

Lorsqu'un programme crée un fichier, il spécifie les droits d'accès qu'il demande pour ce fichier.

Certains des droits demandés seront accordés d'autres seront refusés en fonction d'un *masque de protection*.

La commande `umask` permet :

- de connaître la valeur du masque si elle est utilisée sans argument
- de modifier la valeur du masque si elle est utilisée avec un argument

Dans tous les cas elle utilise des masques sous forme numérique octale.

Les droits accordés sont déterminés en retirant aux droits demandés les droits spécifiés par le masque.

Pour les répertoires :

droits demandés :	`rwx rwx rwx`	777
- masque :	`-- -w- rwx`	027
droits accordés :	`rwx r-x --`	750

Pour les fichiers ordinaires :

droits demandés :	`rw- rw- rw-`	666
- masque :	`-- -w- -w-`	022
droits accordés :	`rw- r- r-`	644

C’est tout pour aujourd’hui