Fichiers, systèmes de fichiers

Ivan Noyer

Crédits

- Un cours du site Malekal

Crédits

- Un cours du site Malekal
- Un cours de Eric Thirion ici

Introduction

- La RAM est *volatile*. Mais les codes et les données des programmes stockés dans la mémoire centrale (et en particulier ceux du systèmes d'exploitation) ont besoin d'être conservé au delà de l'arrept de l'ordinateur.

Introduction

- La RAM est *volatile*. Mais les codes et les données des programmes stockés dans la mémoire centrale (et en particulier ceux du systèmes d'exploitation) ont besoin d'être conservé au delà de l'arrept de l'ordinateur.
- C'est le rôle du *système de gestion de fichiers* qui assure la conservation des données sur un support de masse *non volatile* (disque dur, DVD, clé USB...).

Introduction

- La RAM est *volatile*. Mais les codes et les données des programmes stockés dans la mémoire centrale (et en particulier ceux du systèmes d'exploitation) ont besoin d'être conservé au delà de l'arrept de l'ordinateur.
- C'est le rôle du *système de gestion de fichiers* qui assure la conservation des données sur un support de masse *non volatile* (disque dur, DVD, clé USB...).
- Le système d'exploitation offre à l'utilisateur une interface de stockage indépendante des propriétés physiques des supports de conservation. Son unité de base est le *fichier*.

Introduction

- La RAM est *volatile*. Mais les codes et les données des programmes stockés dans la mémoire centrale (et en particulier ceux du systèmes d'exploitation) ont besoin d'être conservé au delà de l'arrept de l'ordinateur.
- C'est le rôle du *système de gestion de fichiers* qui assure la conservation des données sur un support de masse *non volatile* (disque dur, DVD, clé USB...).
- Le système d'exploitation offre à l'utilisateur une interface de stockage indépendante des propriétés physiques des supports de conservation. Son unité de base est le *fichier*.
- Le concept de fichier présente deux niveaux :

Introduction

- La RAM est *volatile*. Mais les codes et les données des programmes stockés dans la mémoire centrale (et en particulier ceux du systèmes d'exploitation) ont besoin d'être conservé au delà de l'arrept de l'ordinateur.
- C'est le rôle du *système de gestion de fichiers* qui assure la conservation des données sur un support de masse *non volatile* (disque dur, DVD, clé USB...).
- Le système d'exploitation offre à l'utilisateur une interface de stockage indépendante des propriétés physiques des supports de conservation. Son unité de base est le *fichier*.
- Le concept de fichier présente deux niveaux :
  - le *fichier logique* représente les données incluses dans le fichier telles qu'elles sont vues par l'utilisateur ;

Introduction

- La RAM est *volatile*. Mais les codes et les données des programmes stockés dans la mémoire centrale (et en particulier ceux du systèmes d'exploitation) ont besoin d'être conservé au delà de l'arrept de l'ordinateur.
- C'est le rôle du *système de gestion de fichiers* qui assure la conservation des données sur un support de masse *non volatile* (disque dur, DVD, clé USB...).
- Le système d'exploitation offre à l'utilisateur une interface de stockage indépendante des propriétés physiques des supports de conservation. Son unité de base est le *fichier*.
- Le concept de fichier présente deux niveaux :
  - le *fichier logique* représente les données incluses dans le fichier telles qu'elles sont vues par l'utilisateur ;
  - le *fichier physique* représente le fichier tel qu'il est alloué physiquement sur la mémoire de masse.

Introduction

- La RAM est *volatile*. Mais les codes et les données des programmes stockés dans la mémoire centrale (et en particulier ceux du systèmes d'exploitation) ont besoin d'être conservé au delà de l'arrept de l'ordinateur.
- C'est le rôle du *système de gestion de fichiers* qui assure la conservation des données sur un support de masse *non volatile* (disque dur, DVD, clé USB...).
- Le système d'exploitation offre à l'utilisateur une interface de stockage indépendante des propriétés physiques des supports de conservation. Son unité de base est le *fichier*.
- Le concept de fichier présente deux niveaux :
  - le *fichier logique* représente les données incluses dans le fichier telles qu'elles sont vues par l'utilisateur ;
  - le *fichier physique* représente le fichier tel qu'il est alloué physiquement sur la mémoire de masse.
- Le système d'exploitation gère ces deux niveaux et assure la correspondance entre eux via la notion de *répertoire*.

Qu'est-ce qu'un fichier

- Un *fichier* est un ensemble de données numériques réunies sous un même nom, enregistré sur un suport de stockage permanent appelé *mémoire de masse* (ex : disque dur, clé USB, mémoire flash etc...).

Qu'est-ce qu'un fichier

- Un *fichier* est un ensemble de données numériques réunies sous un même nom, enregistré sur un suport de stockage permanent appelé *mémoire de masse* (ex : disque dur, clé USB, mémoire flash etc...).
- Afin de faciliter leur organisation, les fichiers sont disposés dans des *systèmes de fichiers* qui permettent de les placer dans des emplacements appelés *répertoires* eux-même organisés selon le même principe. On obtient alors une hierarchie *arborescente*.

Qu'est-ce qu'un fichier

- Un *fichier* est un ensemble de données numériques réunies sous un même nom, enregistré sur un suport de stockage permanent appelé *mémoire de masse* (ex : disque dur, clé USB, mémoire flash etc...).
- Afin de faciliter leur organisation, les fichiers sont disposés dans des *systèmes de fichiers* qui permettent de les placer dans des emplacements appelés *répertoires* eux-même organisés selon le même principe. On obtient alors une hierarchie *arborescente*.
- Un fichier possède toujours (au moins) un *nom* qui sert à désigner le contenu pour l'utilisateur humain et à y accéder. Le système, lui, utilise plutôt un numéro d'*inode*.

Qu'est-ce qu'un fichier

- Un *fichier* est un ensemble de données numériques réunies sous un même nom, enregistré sur un suport de stockage permanent appelé *mémoire de masse* (ex : disque dur, clé USB, mémoire flash etc...).
- Afin de faciliter leur organisation, les fichiers sont disposés dans des *systèmes de fichiers* qui permettent de les placer dans des emplacements appelés *répertoires* eux-même organisés selon le même principe. On obtient alors une hierarchie *arborescente*.
- Un fichier possède toujours (au moins) un *nom* qui sert à désigner le contenu pour l'utilisateur humain et à y accéder. Le système, lui, utilise plutôt un numéro d'*inode*.
- Chaque fichier possède en outre des *métadonnées* (des informations sur le fichier lui-même) comme sa longueur, son auteur, les personnes autorisées à le manipuler ou la date de dernière modification.

Fichier logique

Correspond à la vue que l'utilisateur a de la conservation de ces données :

- C'est un type de données standard défini dans les langages de programmation que l'on peut *créer*, *ouvrir*, *fermer*, *détruire*. Les opérations de créations et de fermeture effectuent la liaison du fichier logique avec le fichier physique. Les opérations de destruction et de fermeture rompent cette liaison.
- C'est également un ensemble d'*enregistrements* (ou *articles*) sur le modèle des *structures* en C.

Opérations

Les langages de programmation permettent de :

- créer des fichiers

Opérations

Les langages de programmation permettent de :

- créer des fichiers
- enregistrer des informations dans un fichier (c'est l'accès *en écriture*)

Opérations

Les langages de programmation permettent de :

- créer des fichiers
- enregistrer des informations dans un fichier (c'est l'accès *en écriture*)
- lire des informations dans un fichier (c'est l'accès *en lecture*).

Opérations

Les langages de programmation permettent de :

- créer des fichiers
- enregistrer des informations dans un fichier (c'est l'accès *en écriture*)
- lire des informations dans un fichier (c'est l'accès *en lecture*).
- Que ce soit en lecture ou en écriture, il y a deux façons d'accéder au contenu d'un fichier

Opérations

Les langages de programmation permettent de :

- créer des fichiers
- enregistrer des informations dans un fichier (c'est l'accès *en écriture*)
- lire des informations dans un fichier (c'est l'accès *en lecture*).
- Que ce soit en lecture ou en écriture, il y a deux façons d'accéder au contenu d'un fichier
  - accès *séquentiel*

Opérations

Les langages de programmation permettent de :

- créer des fichiers
- enregistrer des informations dans un fichier (c'est l'accès *en écriture*)
- lire des informations dans un fichier (c'est l'accès *en lecture*).
- Que ce soit en lecture ou en écriture, il y a deux façons d'accéder au contenu d'un fichier
  - accès *séquentiel*
  - et accès *direct*

Séquence de manipulation des fichiers

1. Ouvrir le fichier : on informe le système que l'on a l'intention d'accèder au fichier. C'est ici que peuvent survenir les *erreurs d'accès*.

Séquence de manipulation des fichiers

1. Ouvrir le fichier : on informe le système que l'on a l'intention d'accèder au fichier. C'est ici que peuvent survenir les *erreurs d'accès*.
2. Lire des informations / Ecrire des informations

Séquence de manipulation des fichiers

1. Ouvrir le fichier : on informe le système que l'on a l'intention d'accèder au fichier. C'est ici que peuvent survenir les *erreurs d'accès*.
2. Lire des informations / Ecrire des informations
3. Fermer le fichier. On informe le système que le programme a fini d'utiliser le fichier. Un autre programme ou un autre utilisateur pourra donc y accéder à son tour.

Séquence de manipulation des fichiers

1. Ouvrir le fichier : on informe le système que l'on a l'intention d'accèder au fichier. C'est ici que peuvent survenir les *erreurs d'accès*.
2. Lire des informations / Ecrire des informations
3. Fermer le fichier. On informe le système que le programme a fini d'utiliser le fichier. Un autre programme ou un autre utilisateur pourra donc y accéder à son tour.
4. Quand on accède en écriture à un fichier qui n'existe pas, le système le crée. En revanche l'accès en lecture à un fichier qui n'existe pas soulève une erreur.

Erreurs d'accès

Le système d'exploitation peut refuser l'accès à un fichier pour diverses raisons :

- accès en lecture à un fichier qui n'existe pas,

Cette liste n'est pas exhaustive.

Erreurs d'accès

Le système d'exploitation peut refuser l'accès à un fichier pour diverses raisons :

- accès en lecture à un fichier qui n'existe pas,
- accès en lecture sur un fichier dont on n'a pas les droits (ce fichier appartient à un autre utilisateur)

Cette liste n'est pas exhaustive.

Erreurs d'accès

Le système d'exploitation peut refuser l'accès à un fichier pour diverses raisons :

- accès en lecture à un fichier qui n'existe pas,
- accès en lecture sur un fichier dont on n'a pas les droits (ce fichier appartient à un autre utilisateur)
- accès en écriture sur un fichier alors qu'on n'a pas le droit d'écriture dessus.

Cette liste n'est pas exhaustive.

Erreurs d'accès

Le système d'exploitation peut refuser l'accès à un fichier pour diverses raisons :

- accès en lecture à un fichier qui n'existe pas,
- accès en lecture sur un fichier dont on n'a pas les droits (ce fichier appartient à un autre utilisateur)
- accès en écriture sur un fichier alors qu'on n'a pas le droit d'écriture dessus.
- accès en écriture sur un fichier alors qu'un autre utilisateur est en train de l'exploiter (en lecture ou écriture),

Cette liste n'est pas exhaustive.

Erreurs d'accès

Le système d'exploitation peut refuser l'accès à un fichier pour diverses raisons :

- accès en lecture à un fichier qui n'existe pas,
- accès en lecture sur un fichier dont on n'a pas les droits (ce fichier appartient à un autre utilisateur)
- accès en écriture sur un fichier alors qu'on n'a pas le droit d'écriture dessus.
- accès en écriture sur un fichier alors qu'un autre utilisateur est en train de l'exploiter (en lecture ou écriture),
- accès en écriture sur un fichier alors que le périphérique qui le contient est plein.

Cette liste n'est pas exhaustive.

Erreurs d'accès

Le système d'exploitation peut refuser l'accès à un fichier pour diverses raisons :

- accès en lecture à un fichier qui n'existe pas,
- accès en lecture sur un fichier dont on n'a pas les droits (ce fichier appartient à un autre utilisateur)
- accès en écriture sur un fichier alors qu'on n'a pas le droit d'écriture dessus.
- accès en écriture sur un fichier alors qu'un autre utilisateur est en train de l'exploiter (en lecture ou écriture),
- accès en écriture sur un fichier alors que le périphérique qui le contient est plein.
- quand on essaye d'accèder à une ligne après la dernière ligne du fichier.

Cette liste n'est pas exhaustive.

Accès séquentiel

Dans l'accès séquentiel, une opération de lecture ou d'écriture se fait toujours juste après la dernière partie du fichier lue ou écrite (voir figure 1).

- Les fichiers texte n'autorisent que ce type d'accès.

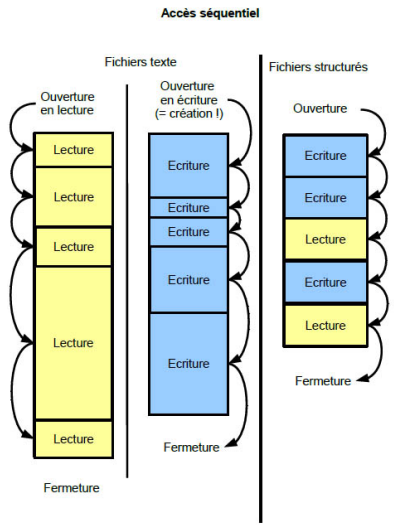


FIGURE 1 – Accès séquentiel : on accède au 53ème élément après le 52ème. (d'après E. Thirion)

Accès séquentiel

Dans l'accès séquentiel, une opération de lecture ou d'écriture se fait toujours juste après la dernière partie du fichier lue ou écrite (voir figure 1).

- Les fichiers texte n'autorisent que ce type d'accès.
- Les fichiers structurés autorisent les deux types d'accès (séquentiel ou direct).

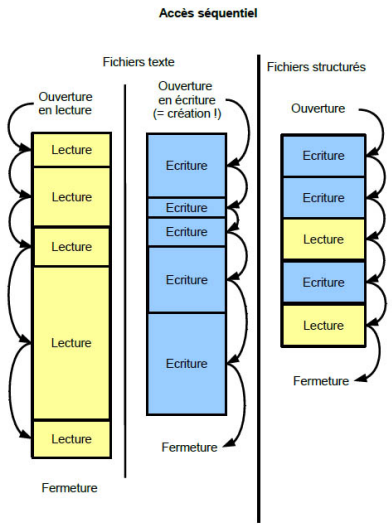


FIGURE 1 – Accès séquentiel : on accède au 53ème élément après le 52ème. (d'après E. Thirion)

Accès direct

- L'accès direct n'est possible que pour les fichiers dits *structurés* : ils sont définis comme une suite de blocs de taille fixe que l'on appelle *enregistrements*. (voir figure 3).

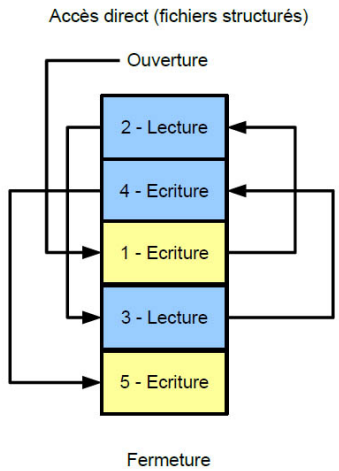


FIGURE 2 – Accès direct : on accède au 53ème enregistrement directement sans passer par les précédents. (d'après E. Thirion)

Accès direct

- L'accès direct n'est possible que pour les fichiers dits *structurés* : ils sont définis comme une suite de blocs de taille fixe que l'on appelle *enregistrements*. (voir figure 3).
- Le terme « accès direct » signifie qu'il est possible d'accéder directement à un enregistrement à partir de sa position dans le fichier : par exemple lire (ou modifier) le 53ème enregistrement du fichier ou le modifier sans avoir lu ou modifié ceux qui précèdent.

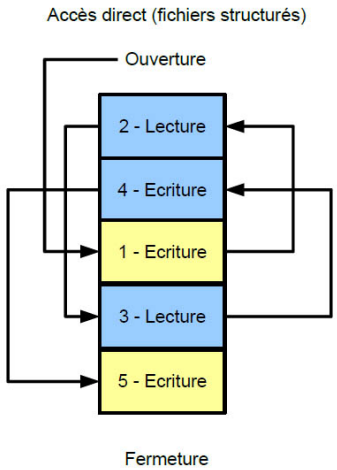


FIGURE 2 – Accès direct : on accède au 53ème enregistrement directement sans passer par les précédents. (d'après E. Thirion)

Manipuler un fichier texte

- L'accès se fait en lecture ou en écriture mais (en général) pas les deux. Deux options pour le mode écriture : écriture au début (donc avec écrasement du contenu) ou bien, à la fin, en conservant le contenu précédent (dans ce cas, on parle de mode *ajout*).

Manipuler un fichier texte

- L'accès se fait en lecture ou en écriture mais (en général) pas les deux. Deux options pour le mode écriture : écriture au début (donc avec écrasement du contenu) ou bien, à la fin, en conservant le contenu précédent (dans ce cas, on parle de mode *ajout*).
- On accède aux informations séquentiellement, le « grain » de la séquence étant soit la ligne soit le caractère. On indique ici ce qu'il se passe dans le mode *par ligne* :

Manipuler un fichier texte

- L'accès se fait en lecture ou en écriture mais (en général) pas les deux. Deux options pour le mode écriture : écriture au début (donc avec écrasement du contenu) ou bien, à la fin, en conservant le contenu précédent (dans ce cas, on parle de mode *ajout*).
- On accède aux informations séquentiellement, le « grain » de la séquence étant soit la ligne soit le caractère. On indique ici ce qu'il se passe dans le mode *par ligne* :
  - Après l'ouverture du fichier en lecture on accède à sa première ligne. Le contenu d'une ligne est recopié dans une variable de type *chaîne de caractères* pour un usage immédiat ou ultérieur. Dès qu'une ligne i est lue, un « curseur » se déplace dans le fichier pour accéder à la ligne suivante.

Manipuler un fichier texte

- L'accès se fait en lecture ou en écriture mais (en général) pas les deux. Deux options pour le mode écriture : écriture au début (donc avec écrasement du contenu) ou bien, à la fin, en conservant le contenu précédent (dans ce cas, on parle de mode *ajout*).
- On accède aux informations séquentiellement, le « grain » de la séquence étant soit la ligne soit le caractère. On indique ici ce qu'il se passe dans le mode *par ligne* :
  - Après l'ouverture du fichier en lecture on accède à sa première ligne. Le contenu d'une ligne est recopié dans une variable de type *chaîne de caractères* pour un usage immédiat ou ultérieur. Dès qu'une ligne i est lue, un « curseur » se déplace dans le fichier pour accéder à la ligne suivante.
  - Après l'ouverture en écriture, on accède aussi à la première ligne. On peut alors écrire dans cette ligne une chaîne de caractère (par exemple, le contenu d'une variable). Puis on passe à la seconde ligne si on décide d'en ajouter une etc.

Présentation

- Une *partition* est une section d'un support de stockage (disque dur, SSD, carte-mémoire...).

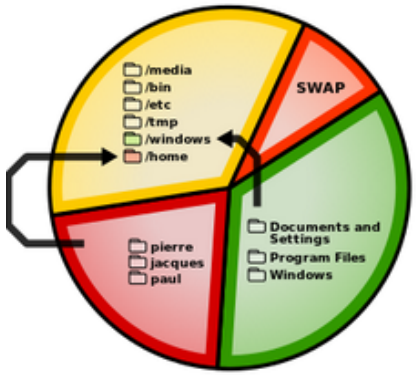


FIGURE 3 – Exemple schématique de partitionnement d'un support mixte Linux/Windows, avec des liens entre les partitions (Wikipedia)

Présentation

- Une *partition* est une section d'un support de stockage (disque dur, SSD, carte-mémoire...).
- Le partitionnement est l'opération qui consiste à diviser ce support en partitions dans lesquelles le système d'exploitation peut gérer les informations de manière séparée, généralement en y créant un système de fichiers, une manière d'organiser l'espace disponible.

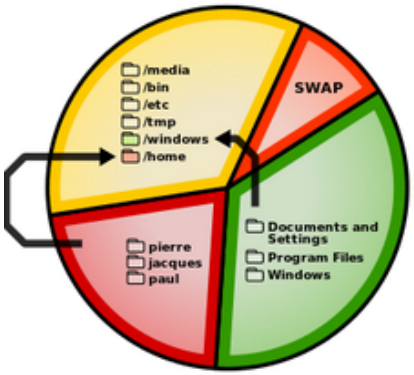


FIGURE 3 – Exemple schématique de partitionnement d'un support mixte Linux/Windows, avec des liens entre les partitions (Wikipedia)

Présentation

- Une *partition* est une section d'un support de stockage (disque dur, SSD, carte-mémoire...).
- Le partitionnement est l'opération qui consiste à diviser ce support en partitions dans lesquelles le système d'exploitation peut gérer les informations de manière séparée, généralement en y créant un système de fichiers, une manière d'organiser l'espace disponible.
- Faux ami : le sens mathématique est différent.

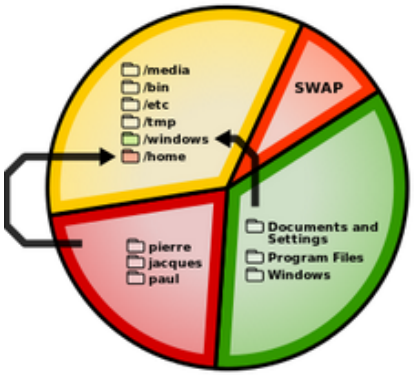


FIGURE 3 – Exemple schématique de partitionnement d'un support mixte Linux/Windows, avec des liens entre les partitions (Wikipedia)

Périphériques

- Un *périphérique* est un matériel (physique) connecté à l'unité centrale d'un ordinateur : disque, souris, écran, réseau...
  Les périphériques sont repérables par un nom dans l'arborescence (sous /**dev**).

Périphériques

- Un *périphérique* est un matériel (physique) connecté à l'unité centrale d'un ordinateur : disque, souris, écran, réseau...
  Les périphériques sont repérables par un nom dans l'arborescence (sous /**dev**).
- Un *pilote de périphériques* ou *driver* est une fonction du système d'exploitation permettant de manipuler une catégorie de périphériques via les opérations classiques autorisées par les *inodes* : **open, read, write, close**.

Périphériques

- Un *périphérique* est un matériel (physique) connecté à l'unité centrale d'un ordinateur : disque, souris, écran, réseau...
  Les périphériques sont repérables par un nom dans l'arborescence (sous /**dev**).
- Un *pilote de périphériques* ou *driver* est une fonction du système d'exploitation permettant de manipuler une catégorie de périphériques via les opérations classiques autorisées par les *inodes* : **open, read, write, close**.
- Un *pseudo-périphérique* est une entrée gérée comme un périphérique bien que non associée à un élément physique.

Périphériques

- Un *périphérique* est un matériel (physique) connecté à l'unité centrale d'un ordinateur : disque, souris, écran, réseau...
  Les périphériques sont repérables par un nom dans l'arborescence (sous /**dev**).
- Un *pilote de périphériques* ou *driver* est une fonction du système d'exploitation permettant de manipuler une catégorie de périphériques via les opérations classiques autorisées par les *inodes* : **open, read, write, close**.
- Un *pseudo-périphérique* est une entrée gérée comme un périphérique bien que non associée à un élément physique.
  - Utilisation 1 : repérage de périphériques dits virtuels, *i.e.* des parties de périphériques (physiques), par exemple : les écrans virtuels de l'écran (physique), partitions (logiques) du disque (physique)...

Périphériques

- Un *périphérique* est un matériel (physique) connecté à l'unité centrale d'un ordinateur : disque, souris, écran, réseau...
  Les périphériques sont repérables par un nom dans l'arborescence (sous /**dev**).
- Un *pilote de périphériques* ou *driver* est une fonction du système d'exploitation permettant de manipuler une catégorie de périphériques via les opérations classiques autorisées par les *inodes* : **open, read, write, close**.
- Un *pseudo-périphérique* est une entrée gérée comme un périphérique bien que non associée à un élément physique.
  - Utilisation 1 : repérage de périphériques dits virtuels, *i.e.* des parties de périphériques (physiques), par exemple : les écrans virtuels de l'écran (physique), partitions (logiques) du disque (physique)...
  - Utilisation 2 : mise en évidence de fonctionnalités du système d'exploitation dans l'arborescence, par exemple : la « poubelle » qui a pour nom /**dev/null**.

Périphériques

- Un *périphérique* est un matériel (physique) connecté à l'unité centrale d'un ordinateur : disque, souris, écran, réseau...
  Les périphériques sont repérables par un nom dans l'arborescence (sous /**dev**).
- Un *pilote de périphériques* ou *driver* est une fonction du système d'exploitation permettant de manipuler une catégorie de périphériques via les opérations classiques autorisées par les *inodes* : **open, read, write, close**.
- Un *pseudo-périphérique* est une entrée gérée comme un périphérique bien que non associée à un élément physique.
  - Utilisation 1 : repérage de périphériques dits virtuels, *i.e.* des parties de périphériques (physiques), par exemple : les écrans virtuels de l'écran (physique), partitions (logiques) du disque (physique)...
  - Utilisation 2 : mise en évidence de fonctionnalités du système d'exploitation dans l'arborescence, par exemple : la « poubelle » qui a pour nom /**dev/null**.
- Pour l'OS, périphériques et pseudo-périphériques sont des fichiers.

Connaître les systèmes de fichiers montés

- Il suffit d'entrer la commande **mount**.

Connaître les systèmes de fichiers montés

- Il suffit d'entrer la commande **mount**.
- Par exemple :

```
$ mount
...
/dev/sdc1 on /media/ivan/KINGSTON type vfat (...)
...
```

Connaître les systèmes de fichiers montés

- Il suffit d'entrer la commande **mount**.
- Par exemple :

```
$ mount
...
/dev/sdc1 on /media/ivan/KINGSTON type vfat (...)
...
```

- On constate que ma clé usb est repérée par **sdc1**, qu'elle est montée sur **/media/ivan/KINGSTOM** et que son système de fichier est VFAT

udev

- Le démon `udev` est chargé de gérer les *device-nodes* : pseudos-fichiers stockés normalement dans /**dev** chargés de représenter les périphériques.

udev

- Le démon `udev` est chargé de gérer les *device-nodes* : pseudos-fichiers stockés normalement dans /**dev** chargés de représenter les périphériques.
- Ce sont en réalité des points d'entrées vers le noyau caractérisés par un type d'accès (bloc ou char) et deux nombres, un majeur et un mineur :

udev

- Le démon `udev` est chargé de gérer les *device-nodes* : pseudos-fichiers stockés normalement dans /**dev** chargés de représenter les périphériques.
- Ce sont en réalité des points d'entrées vers le noyau caractérisés par un type d'accès (bloc ou char) et deux nombres, un majeur et un mineur :
  - le type définit le mode d'accès (donc par bloc ou par caractère) au périphérique

udev

- Le démon `udev` est chargé de gérer les *device-nodes* : pseudos-fichiers stockés normalement dans /**dev** chargés de représenter les périphériques.
- Ce sont en réalité des points d'entrées vers le noyau caractérisés par un type d'accès (bloc ou char) et deux nombres, un majeur et un mineur :
  - le type définit le mode d'accès (donc par bloc ou par caractère) au périphérique
  - le majeur permet au noyau de connaître le driver qui gère le périphérique

udev

- Le démon `udev` est chargé de gérer les *device-nodes* : pseudos-fichiers stockés normalement dans /**dev** chargés de représenter les périphériques.
- Ce sont en réalité des points d'entrées vers le noyau caractérisés par un type d'accès (bloc ou char) et deux nombres, un majeur et un mineur :
  - le type définit le mode d'accès (donc par bloc ou par caractère) au périphérique
  - le majeur permet au noyau de connaître le driver qui gère le périphérique
  - le mineur permet au driver de savoir quel périphérique parmi ceux qu'il gère (il peut y en avoir plusieurs) est utilisé.

Majeurs et mineurs

- La commande **ls -l** nous donne les informations sur le type d'accès et les majeurs et mineurs (entre autres)

Majeurs et mineurs

- La commande **ls -l** nous donne les informations sur le type d'accès et les majeurs et mineurs (entre autres)
- J'interroge ci-dessous le système à propos de mon disque dur NVMe :

```
$ ls -l /dev/nvme0n1
brw-rw---- 1 root disk 259, 0 août 19 09:57 /dev/nvme0n1
```

Majeurs et mineurs

- La commande **ls -l** nous donne les informations sur le type d'accès et les majeurs et mineurs (entre autres)
- J'interroge ci-dessous le système à propos de mon disque dur NVMe :

```
$ ls -l /dev/nvme0n1
brw-rw---- 1 root disk 259, 0 août 19 09:57 /dev/nvme0n1
```

  - le **b** en début de ligne indique qu'il s'agit d'un périphérique avec accès par bloc.

Majeurs et mineurs

- La commande **ls -l** nous donne les informations sur le type d'accès et les majeurs et mineurs (entre autres)
- J'interroge ci-dessous le système à propos de mon disque dur NVMe :

```
$ ls -l /dev/nvme0n1
brw-rw---- 1 root disk 259, 0 août 19 09:57 /dev/nvme0n1
```

  - le **b** en début de ligne indique qu'il s'agit d'un périphérique avec accès par bloc.
  - le majeur est 259, il indique quel driver est utilisé

Majeurs et mineurs

- La commande **ls -l** nous donne les informations sur le type d'accès et les majeurs et mineurs (entre autres)
- J'interroge ci-dessous le système à propos de mon disque dur NVMe :

```
$ ls -l /dev/nvme0n1
brw-rw---- 1 root disk 259, 0 août 19 09:57 /dev/nvme0n1
```

  - le **b** en début de ligne indique qu'il s'agit d'un périphérique avec accès par bloc.
  - le majeur est 259, il indique quel driver est utilisé
  - le mineur est 0 : il indique quel périphérique est ici géré par ce driver.

Majeurs et mineurs

Exemple d'un préiphérique avec accès séquentiel

- J'interroge ci-dessous le système à propos de ma souris :

```
ls -l /dev/input/mouse1
crw-rw---- 1 root input 13, 33 janv. 11 17:25 /dev/input/mouse1
```

Majeurs et mineurs

Exemple d'un préiphérique avec accès séquentiel

- J'interroge ci-dessous le système à propos de ma souris :

```
ls -l /dev/input/mouse1
crw-rw---- 1 root input 13, 33 janv. 11 17:25 /dev/input/mouse1
```

- le **c** en début de ligne indique qu'il s'agit d'un périphérique avec accès par octet (ou par caractère).

Majeurs et mineurs

Exemple d'un préiphérique avec accès séquentiel

- J'interroge ci-dessous le système à propos de ma souris :

```
ls -l /dev/input/mouse1
crw-rw---- 1 root input 13, 33 janv. 11 17:25 /dev/input/mouse1
```

- le **c** en début de ligne indique qu'il s'agit d'un périphérique avec accès par octet (ou par caractère).
- le majeur est 13, il indique quel driver est utilisé

Majeurs et mineurs

Exemple d'un préiphérique avec accès séquentiel

- J'interroge ci-dessous le système à propos de ma souris :

```
ls -l /dev/input/mouse1
crw-rw---- 1 root input 13, 33 janv. 11 17:25 /dev/input/mouse1
```

- le **c** en début de ligne indique qu'il s'agit d'un périphérique avec accès par octet (ou par caractère).
- le majeur est 13, il indique quel driver est utilisé
- le mineur est 33 : il indique quel périphérique est ici géré par ce driver.

Généralités

Sous LINUX, considérons un fichier :

- Il est toujours désigné par un nom

Généralités

Sous LINUX, considérons un fichier :

- Il est toujours désigné par un nom
- Il possède un unique *inode* (certaines informations concernant le fichier). C'est une structure créée au même moment que le fichier afin de contenir ses informations fondamentales. Tous les inodes sont conservés dans une table, et sont identifiés par un *INumber* (numéro d'index ou d'inode).

Généralités

Sous LINUX, considérons un fichier :

- Il est toujours désigné par un nom
- Il possède un unique *inode* (certaines informations concernant le fichier). C'est une structure créée au même moment que le fichier afin de contenir ses informations fondamentales. Tous les inodes sont conservés dans une table, et sont identifiés par un *INumber* (numéro d'index ou d'inode).
- Le système fournit à propos de ce fichier des primitives permettant de :

Généralités

Sous LINUX, considérons un fichier :

- Il est toujours désigné par un nom
- Il possède un unique *inode* (certaines informations concernant le fichier). C'est une structure créée au même moment que le fichier afin de contenir ses informations fondamentales. Tous les inodes sont conservés dans une table, et sont identifiés par un *INumber* (numéro d'index ou d'inode).
- Le système fournit à propos de ce fichier des primitives permettant de :
  - l'ouvrir

Généralités

Sous LINUX, considérons un fichier :

- Il est toujours désigné par un nom
- Il possède un unique *inode* (certaines informations concernant le fichier). C'est une structure créée au même moment que le fichier afin de contenir ses informations fondamentales. Tous les inodes sont conservés dans une table, et sont identifiés par un *INumber* (numéro d'index ou d'inode).
- Le système fournit à propos de ce fichier des primitives permettant de :
  - l'ouvrir
  - le fermer

Généralités

Sous LINUX, considérons un fichier :

- Il est toujours désigné par un nom
- Il possède un unique *inode* (certaines informations concernant le fichier). C'est une structure créée au même moment que le fichier afin de contenir ses informations fondamentales. Tous les inodes sont conservés dans une table, et sont identifiés par un *INumber* (numéro d'index ou d'inode).
- Le système fournit à propos de ce fichier des primitives permettant de :
  - l'ouvrir
  - le fermer
  - le lire

Généralités

Sous LINUX, considérons un fichier :

- Il est toujours désigné par un nom
- Il possède un unique *inode* (certaines informations concernant le fichier). C'est une structure créée au même moment que le fichier afin de contenir ses informations fondamentales. Tous les inodes sont conservés dans une table, et sont identifiés par un *INumber* (numéro d'index ou d'inode).
- Le système fournit à propos de ce fichier des primitives permettant de :
  - l'ouvrir
  - le fermer
  - le lire
  - le modifier

Les types de fichiers

Pour chaque type de fichier, on donne également la convention d'affichage utilisée par la commande **ls -l nomDuFichier**

ordinaire (on dit aussi régulier ou normal). (**-**)

Les types de fichiers

Pour chaque type de fichier, on donne également la convention d'affichage utilisée par la commande **ls -l nomDuFichier**

ordinaire (on dit aussi régulier ou normal). (**-**)

répertoire ils contiennent seulement la liste des noms des fichiers qui y sont stockés ainsi que leurs numéros d'inode. (**d**)

Les types de fichiers

Pour chaque type de fichier, on donne également la convention d'affichage utilisée par la commande **ls -l nomDuFichier**

ordinaire (on dit aussi régulier ou normal). (**-**)

répertoire ils contiennent seulement la liste des noms des fichiers qui y sont stockés ainsi que leurs numéros d'inode. (**d**)

lien symbolique (**l**)

Les types de fichiers

Pour chaque type de fichier, on donne également la convention d'affichage utilisée par la commande **ls -l nomDuFichier**

ordinaire (on dit aussi régulier ou normal). (**-**)

répertoire ils contiennent seulement la liste des noms des fichiers qui y sont stockés ainsi que leurs numéros d'inode. (**d**)

lien symbolique (**l**)

Les types de fichiers

Pour chaque type de fichier, on donne également la convention d'affichage utilisée par la commande **ls -l nomDuFichier**

ordinaire (on dit aussi régulier ou normal). (**-**)

répertoire ils contiennent seulement la liste des noms des fichiers qui y sont stockés ainsi que leurs numéros d'inode. (**d**)

lien symbolique (**l**)

pseudo-fichier
- Périphériques :

Les types de fichiers

Pour chaque type de fichier, on donne également la convention d'affichage utilisée par la commande **ls -l nomDuFichier**

ordinaire (on dit aussi régulier ou normal). (**-**)

répertoire ils contiennent seulement la liste des noms des fichiers qui y sont stockés ainsi que leurs numéros d'inode. (**d**)

lien symbolique (**l**)

pseudo-fichier
- Périphériques :
  - avec accès par caractères (**c**). Exemple : souris, clavier.

Les types de fichiers

Pour chaque type de fichier, on donne également la convention d'affichage utilisée par la commande **ls -l nomDuFichier**

ordinaire (on dit aussi régulier ou normal). (**-**)

répertoire ils contiennent seulement la liste des noms des fichiers qui y sont stockés ainsi que leurs numéros d'inode. (**d**)

lien symbolique (**l**)

pseudo-fichier
- Périphériques :
  - avec accès par caractères (**c**). Exemple : souris, clavier.
  - avec accès par blocs (**b**). Les périphériques blocs (disque dur, lecteurs de DVD) gèrent une certaine quantité d'information par groupe de bloc.

Les types de fichiers

Pour chaque type de fichier, on donne également la convention d'affichage utilisée par la commande **ls -l nomDuFichier**

ordinaire (on dit aussi régulier ou normal). (**-**)

répertoire ils contiennent seulement la liste des noms des fichiers qui y sont stockés ainsi que leurs numéros d'inode. (**d**)

lien symbolique (**l**)

pseudo-fichier
- Périphériques :
  - avec accès par caractères (**c**). Exemple : souris, clavier.
  - avec accès par blocs (**b**). Les périphériques blocs (disque dur, lecteurs de DVD) gèrent une certaine quantité d'information par groupe de bloc.
- dispositif de communication (**p**) pour les canaux de communication des *pipes*.

Les types de fichiers

Pour chaque type de fichier, on donne également la convention d'affichage utilisée par la commande **ls -l nomDuFichier**

ordinaire (on dit aussi régulier ou normal). (**-**)

répertoire ils contiennent seulement la liste des noms des fichiers qui y sont stockés ainsi que leurs numéros d'inode. (**d**)

lien symbolique (**l**)

pseudo-fichier
- Périphériques :
  - avec accès par caractères (**c**). Exemple : souris, clavier.
  - avec accès par blocs (**b**). Les périphériques blocs (disque dur, lecteurs de DVD) gèrent une certaine quantité d'information par groupe de bloc.
- dispositif de communication (**p**) pour les canaux de communication des *pipes*.
- les sockets (**s**) (réseaux)

Périphériques

- Dans **/dev/input** on trouve la plupart des périphériques d'entrées sur l'ordinateur (mais pas tous -par ex. : les microphones-).

Périphériques

- Dans **/dev/input** on trouve la plupart des périphériques d'entrées sur l'ordinateur (mais pas tous -par ex. : les microphones-).
- Ci-dessous, les **event** représentent des évènements en rapport avec le clavier (même en rapport lointain, ma webcam est considérée comme faisant partie du clavier !) et **mouse**...

```
$ ls -l /dev/input/
crw-rw---- 1 root input 13,74 août 19 09:57 event10
crw-rw---- 1 root input 13,75 août 19 09:57 event11
...
crw-rw---- 1 root input 13,32 août 19 09:57 mouse0
crw-rw---- 1 root input 13,33 août 19 09:57 mouse1
```

Périphériques

- Dans **/dev/input** on trouve la plupart des périphériques d'entrées sur l'ordinateur (mais pas tous -par ex. : les microphones-).
- Ci-dessous, les **event** représentent des évènements en rapport avec le clavier (même en rapport lointain, ma webcam est considérée comme faisant partie du clavier !) et **mouse**...

```
      $ ls -l /dev/input/
crw-rw---- 1 root input 13,74 août 19 09:57 event10
crw-rw---- 1 root input 13,75 août 19 09:57 event11
...
crw-rw---- 1 root input 13,32 août 19 09:57 mouse0
crw-rw---- 1 root input 13,33 août 19 09:57 mouse1
```

- La souris est bien un périphérique en mode caractère (la première lettre de la ligne). L'évènement 11 est curieusement associé à ma caméra (comme on peut le voir en entrant **xinput list**)

Bloc

- Un *bloc* (ou en anglais cluster) est la plus petite unité de stockage du système de fichiers d'une mémoire de masse. Le choix de la taille de bloc est effectué lors du formatage, et influe sur les performances et sur la capacité utile.

Bloc

- Un *bloc* (ou en anglais cluster) est la plus petite unité de stockage du système de fichiers d'une mémoire de masse. Le choix de la taille de bloc est effectué lors du formatage, et influe sur les performances et sur la capacité utile.
- Le *bloc logique* est la plus petite unité transférable (en lecture/écriture) du disque. Sa taille peut être différente de celle d'un bloc physique.

Bloc

- Un *bloc* (ou en anglais cluster) est la plus petite unité de stockage du système de fichiers d'une mémoire de masse. Le choix de la taille de bloc est effectué lors du formatage, et influe sur les performances et sur la capacité utile.
- Le *bloc logique* est la plus petite unité transférable (en lecture/écriture) du disque. Sa taille peut être différente de celle d'un bloc physique.
- Un *système de fichiers* est une séquence de blocs logiques décomposées hiérarchiquement comme suit :

Bloc

- Un *bloc* (ou en anglais cluster) est la plus petite unité de stockage du système de fichiers d'une mémoire de masse. Le choix de la taille de bloc est effectué lors du formatage, et influe sur les performances et sur la capacité utile.
- Le *bloc logique* est la plus petite unité transférable (en lecture/écriture) du disque. Sa taille peut être différente de celle d'un bloc physique.
- Un *système de fichiers* est une séquence de blocs logiques décomposées hiérarchiquement comme suit :

  bloc amorce : contient le chargeur primaire (ou secteur de boot) ;

Bloc

- Un *bloc* (ou en anglais cluster) est la plus petite unité de stockage du système de fichiers d'une mémoire de masse. Le choix de la taille de bloc est effectué lors du formatage, et influe sur les performances et sur la capacité utile.
- Le *bloc logique* est la plus petite unité transférable (en lecture/écriture) du disque. Sa taille peut être différente de celle d'un bloc physique.
- Un *système de fichiers* est une séquence de blocs logiques décomposées hiérarchiquement comme suit :

  bloc amorce : contient le chargeur primaire (ou secteur de boot) ;

  superbloc : informations sur le système de fichiers ;

Bloc

- Un *bloc* (ou en anglais cluster) est la plus petite unité de stockage du système de fichiers d'une mémoire de masse. Le choix de la taille de bloc est effectué lors du formatage, et influe sur les performances et sur la capacité utile.
- Le *bloc logique* est la plus petite unité transférable (en lecture/écriture) du disque. Sa taille peut être différente de celle d'un bloc physique.
- Un *système de fichiers* est une séquence de blocs logiques décomposées hiérarchiquement comme suit :

  bloc amorce : contient le chargeur primaire (ou secteur de boot) ;
  superbloc : informations sur le système de fichiers ;
  table des inodes un inode contient des informations sur un fichier ;

Bloc

- Un *bloc* (ou en anglais cluster) est la plus petite unité de stockage du système de fichiers d'une mémoire de masse. Le choix de la taille de bloc est effectué lors du formatage, et influe sur les performances et sur la capacité utile.
- Le *bloc logique* est la plus petite unité transférable (en lecture/écriture) du disque. Sa taille peut être différente de celle d'un bloc physique.
- Un *système de fichiers* est une séquence de blocs logiques décomposées hiérarchiquement comme suit :

  bloc amorce : contient le chargeur primaire (ou secteur de boot) ;
  superbloc : informations sur le système de fichiers ;
  table des inodes un inode contient des informations sur un fichier ;
  blocs de données le contenu d'un même fichier est réparti sur un ou plusieurs blocs.

Organisation du super-bloc

Il contient les informations suivantes

- nombre total de blocs de la partition ;

Organisation du super-bloc

Il contient les informations suivantes

- nombre total de blocs de la partition ;
- nombre et bitmap de blocs libres : pour un disque de n blocs, le *bitmap* est un tableau de n bits tel que le bit x est à zéro si le bloc x est occupé et à 1 sinon.

Organisation du super-bloc

Il contient les informations suivantes

- nombre total de blocs de la partition ;
- nombre et bitmap de blocs libres : pour un disque de n blocs, le *bitmap* est un tableau de n bits tel que le bit x est à zéro si le bloc x est occupé et à 1 sinon.
- nombre total d'inodes ;

Organisation du super-bloc

Il contient les informations suivantes

- nombre total de blocs de la partition ;
- nombre et bitmap de blocs libres : pour un disque de n blocs, le *bitmap* est un tableau de n bits tel que le bit x est à zéro si le bloc x est occupé et à 1 sinon.
- nombre total d'inodes ;
- nombre d'inodes libres ;

Organisation du super-bloc

Il contient les informations suivantes

- nombre total de blocs de la partition ;
- nombre et bitmap de blocs libres : pour un disque de n blocs, le *bitmap* est un tableau de n bits tel que le bit x est à zéro si le bloc x est occupé et à 1 sinon.
- nombre total d'inodes ;
- nombre d'inodes libres ;
- tête de la liste chaînée des inodes libres.

Vue d'un disque logique

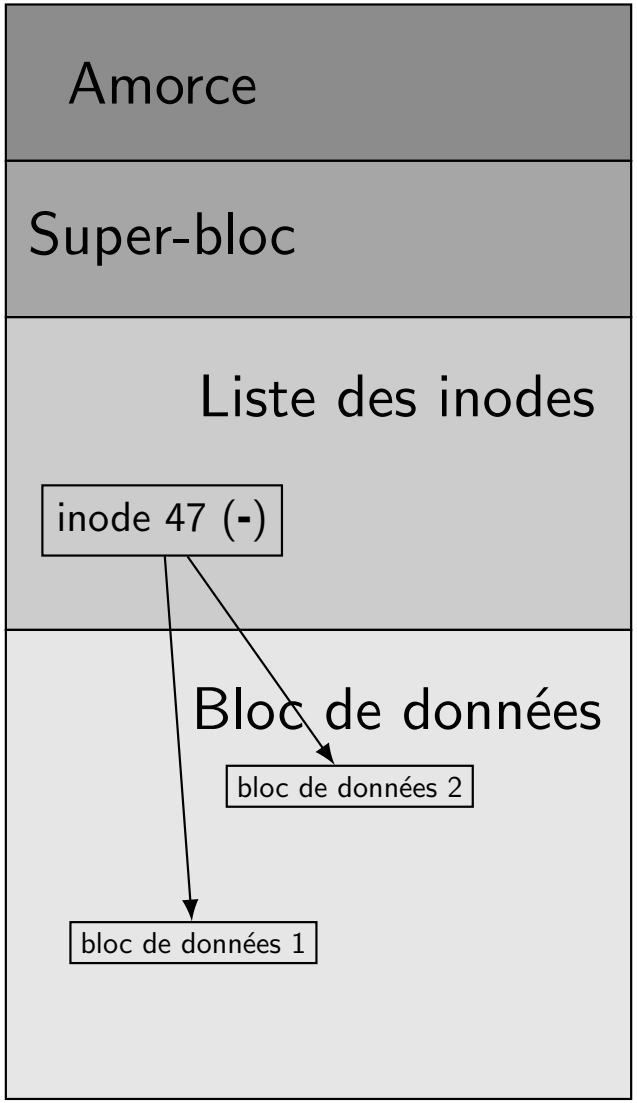


FIGURE 4 – Un disque logique

Inodes

- Un *nœud d'index* ou inode (contraction de l'anglais index et node) est une structure de données contenant des informations à propos d'un fichier ou répertoire.

Inodes

- Un *nœud d'index* ou inode (contraction de l'anglais index et node) est une structure de données contenant des informations à propos d'un fichier ou répertoire.
- À chaque fichier correspond un numéro d'inode (i-number) dans le système de fichiers dans lequel il réside, unique dans la partition sur laquelle est située le système de fichiers.

Inodes

- Un *nœud d'index* ou inode (contraction de l'anglais index et node) est une structure de données contenant des informations à propos d'un fichier ou répertoire.
- À chaque fichier correspond un numéro d'inode (i-number) dans le système de fichiers dans lequel il réside, unique dans la partition sur laquelle est située le système de fichiers.
- Chaque fichier a un seul inode mais peut avoir plusieurs noms (lesquels font référence au même inode). Un nom de fichier est aussi appelé un *lien*.

Inodes

- Un *nœud d'index* ou inode (contraction de l'anglais index et node) est une structure de données contenant des informations à propos d'un fichier ou répertoire.
- À chaque fichier correspond un numéro d'inode (i-number) dans le système de fichiers dans lequel il réside, unique dans la partition sur laquelle est située le système de fichiers.
- Chaque fichier a un seul inode mais peut avoir plusieurs noms (lesquels font référence au même inode). Un nom de fichier est aussi appelé un *lien*.
- Les inodes contiennent toutes les informations sur les fichiers à part le ou les noms.

Inodes

- Un *nœud d'index* ou inode (contraction de l'anglais index et node) est une structure de données contenant des informations à propos d'un fichier ou répertoire.
- À chaque fichier correspond un numéro d'inode (i-number) dans le système de fichiers dans lequel il réside, unique dans la partition sur laquelle est située le système de fichiers.
- Chaque fichier a un seul inode mais peut avoir plusieurs noms (lesquels font référence au même inode). Un nom de fichier est aussi appelé un *lien*.
- Les inodes contiennent toutes les informations sur les fichiers à part le ou les noms.
- Les inodes sont créés en même temps que le système de fichier auxquels ils appartiennent. Leur nombre est donc fixe et dépend de la taille de la partition à laquelle est associé le système de fichier.

Connaître l'inode d'un fichier

Pour connaître le numéro d'inode d'un fichier :

```
$ ls -i asup # num d'inode du fichier asup
10224747 asup
```

Nombre d'inodes utilisées

On a des informations sur les inodes utilisés par les partitions avec **df -i** :

```
$ df -i
Sys. de fichiers Inœuds IUtil. ILibre IUti% Monté sur
udev 2016836 580 2016256 1% /dev
tmpfs 2032479 1076 2031403 1% /run
/dev/nvme0n1p5 31227904 736385 30491519 3% /
tmpfs 2032479 1 2032478 1% /dev/shm
tmpfs 2032479 5 2032474 1% /run/lock
tmpfs 2032479 18 2032461 1% /sys/fs/cgroup
/dev/loop1 10803 10803 0 100% /snap/core18/2074
/dev/loop2 293 293 0 100% /snap/discord/122
```

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)
- droits d'accès ou *mode* (**-rw-rw-r–**)

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)
- droits d'accès ou *mode* (**-rw-rw-r–**)
- nombres de liens physiques sur le fichier

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)
- droits d'accès ou *mode* (**-rw-rw-r–**)
- nombres de liens physiques sur le fichier
- numéro de l'utilisateur et du groupe auquel appartient le fichier.

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)
- droits d'accès ou *mode* (**-rw-rw-r–**)
- nombres de liens physiques sur le fichier
- numéro de l'utilisateur et du groupe auquel appartient le fichier.
- Taille en byte du fichier

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)
- droits d'accès ou *mode* (**-rw-rw-r–**)
- nombres de liens physiques sur le fichier
- numéro de l'utilisateur et du groupe auquel appartient le fichier.
- Taille en byte du fichier
- Deux tableaux contenant un total de 15 ≪ adresses disques ≫ (ce sont des pointeurs) (15 pour le standard **ext2**). 12 adresses directes, une adresse indirecte à un niveau d'indirection, une adresse indirecte à deux niveaux d'indirection et une adresse indirecte à trois niveaux.

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)
- droits d'accès ou *mode* (**-rw-rw-r–**)
- nombres de liens physiques sur le fichier
- numéro de l'utilisateur et du groupe auquel appartient le fichier.
- Taille en byte du fichier
- Deux tableaux contenant un total de 15 ≪ adresses disques ≫ (ce sont des pointeurs) (15 pour le standard **ext2**). 12 adresses directes, une adresse indirecte à un niveau d'indirection, une adresse indirecte à deux niveaux d'indirection et une adresse indirecte à trois niveaux.
- Informations de dates :

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)
- droits d'accès ou *mode* (**-rw-rw-r–**)
- nombres de liens physiques sur le fichier
- numéro de l'utilisateur et du groupe auquel appartient le fichier.
- Taille en byte du fichier
- Deux tableaux contenant un total de 15 ≪ adresses disques ≫ (ce sont des pointeurs) (15 pour le standard **ext2**). 12 adresses directes, une adresse indirecte à un niveau d'indirection, une adresse indirecte à deux niveaux d'indirection et une adresse indirecte à trois niveaux.
- Informations de dates :
  - date et heure (**ctime**) de dernière modification de l'inode (**ls -lc**)

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)
- droits d'accès ou *mode* (**-rw-rw-r–**)
- nombres de liens physiques sur le fichier
- numéro de l'utilisateur et du groupe auquel appartient le fichier.
- Taille en byte du fichier
- Deux tableaux contenant un total de 15 « adresses disques » (ce sont des pointeurs) (15 pour le standard **ext2**). 12 adresses directes, une adresse indirecte à un niveau d'indirection, une adresse indirecte à deux niveaux d'indirection et une adresse indirecte à trois niveaux.
- Informations de dates :
  - date et heure (**ctime**) de dernière modification de l'inode (**ls -lc**)
  - dernière date et heure (**mtime**) auxquelles le fichier a été modifié (**ls -l**)

Champs d'un inode (format **ext2**)

Les inodes sont tous de même taille et contiennent les informations suivantes :

- type du fichier (**- d l c p b**...)
- droits d'accès ou *mode* (**-rw-rw-r–**)
- nombres de liens physiques sur le fichier
- numéro de l'utilisateur et du groupe auquel appartient le fichier.
- Taille en byte du fichier
- Deux tableaux contenant un total de 15 « adresses disques » (ce sont des pointeurs) (15 pour le standard **ext2**). 12 adresses directes, une adresse indirecte à un niveau d'indirection, une adresse indirecte à deux niveaux d'indirection et une adresse indirecte à trois niveaux.
- Informations de dates :
  - date et heure (**ctime**) de dernière modification de l'inode (**ls -lc**)
  - dernière date et heure (**mtime**) auxquelles le fichier a été modifié (**ls -l**)
  - date et heure (**atime**) de création (**ls -lu**)

Exemple

```
UFS$ ls -lu slides_ufs.tex
-rw-rw-r-- 1 ivan ivan 42531 janv. 9 2023 slides_ufs.tex
UFS$ ls -lc slides_ufs.tex
-rw-rw-r-- 1 ivan ivan 42531 janv. 17 10:43 slides_ufs.tex
$ ls -l slides_ufs.tex
-rw-rw-r-- 1 ivan ivan 42531 janv. 17 10:43 slides_ufs.tex
```

Vue schématique pour le format ext4

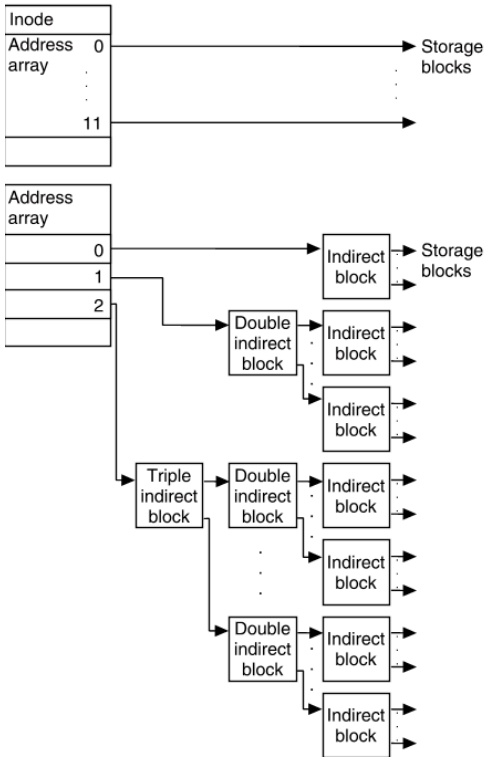


FIGURE 5 – Une inode avec ses liens directs et les indirections de niveau 1,2,3 (cf SCO Group)

Taille maximale d'un fichier régulier

En exercice

Exemple de contenu d'un répertoire

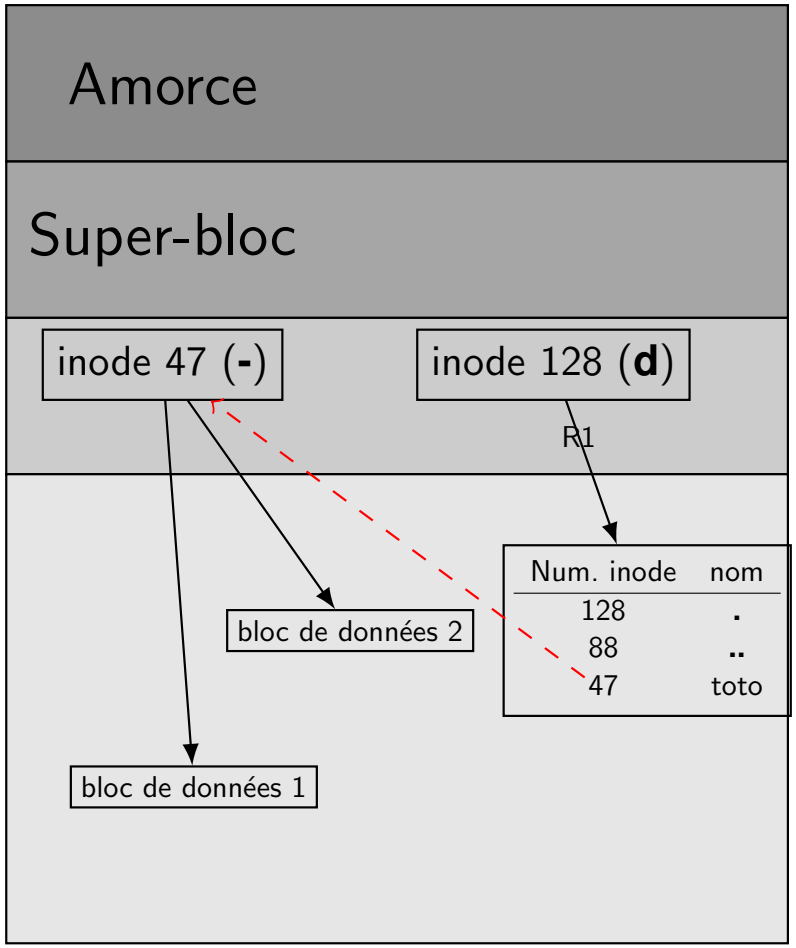


FIGURE 6 – Un répertoire contenant un fichier toto

Copier

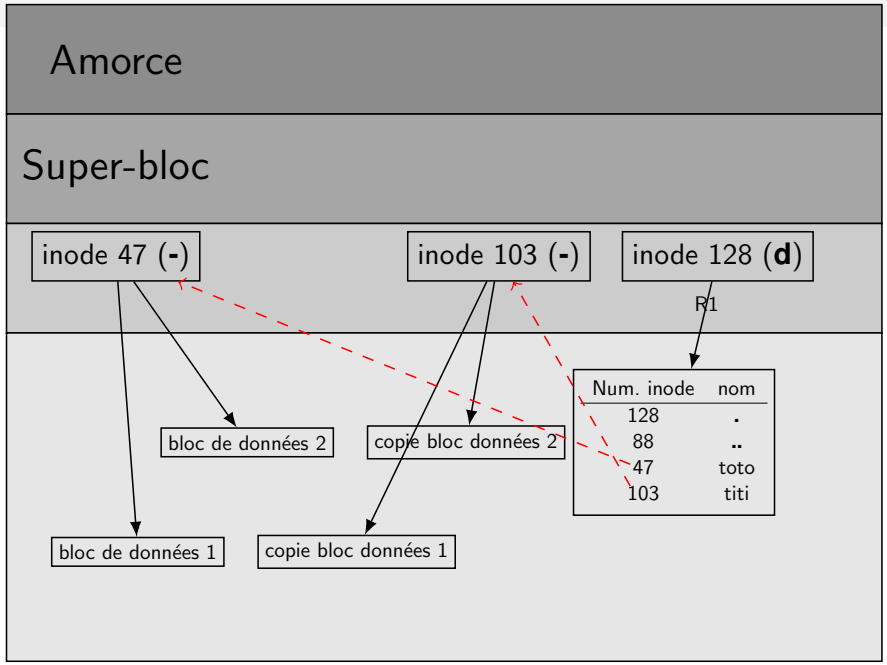


FIGURE 7 –

cp toto titi duplique les données de **toto**. Un inode autre que celui de **toto** pointe sur les données dupliquées.

Renommage ou déplacement

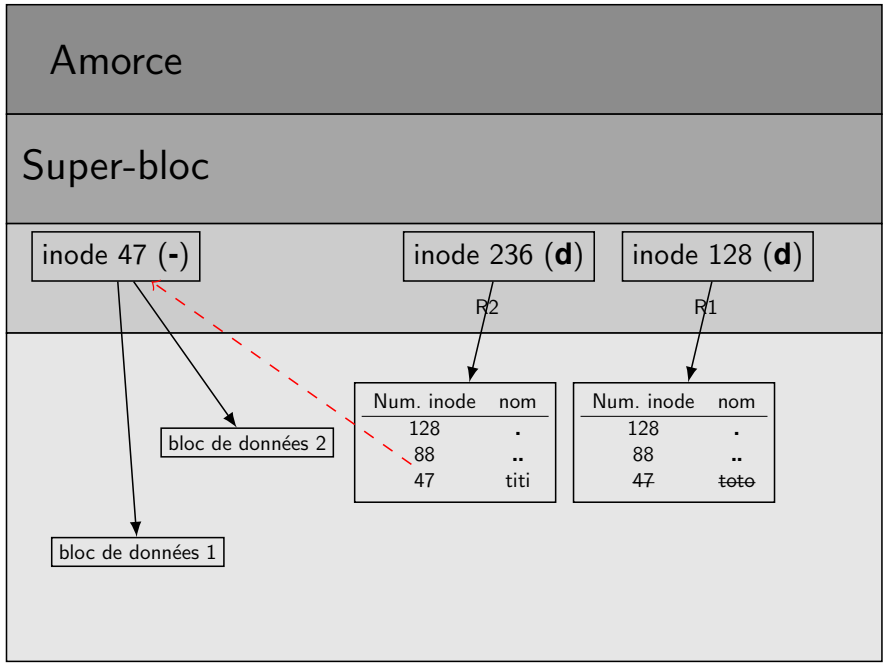


FIGURE 8 – Déplacement **mv toto R2/titi** : par rapport à la figure 6, **titi** de **R2** (inode 236) est l'ancien **toto** de **R1** (inode 128)

Lien physique

- Un *lien physique* est une référence directe à un fichier via son inode.

Lien physique

- Un *lien physique* est une référence directe à un fichier via son inode.
- Avantage : On peut changer le contenu ou la localisation du fichier original, le lien physique restera valide.

Lien physique

- Un *lien physique* est une référence directe à un fichier via son inode.
- Avantage : On peut changer le contenu ou la localisation du fichier original, le lien physique restera valide.
- Inconvénient : un lien physique référence un numéro d'inode ce qui impose de rester dans la même partition (en effet : deux disques logiques différents peuvent avoir chacun une inode 47).

Lien physique

- Un *lien physique* est une référence directe à un fichier via son inode.
- Avantage : On peut changer le contenu ou la localisation du fichier original, le lien physique restera valide.
- Inconvénient : un lien physique référence un numéro d'inode ce qui impose de rester dans la même partition (en effet : deux disques logiques différents peuvent avoir chacun une inode 47).
- Inconvénient : on ne peut pas faire un lien physique avec un répertoire.

Lien physique

- Un *lien physique* est une référence directe à un fichier via son inode.
- Avantage : On peut changer le contenu ou la localisation du fichier original, le lien physique restera valide.
- Inconvénient : un lien physique référence un numéro d'inode ce qui impose de rester dans la même partition (en effet : deux disques logiques différents peuvent avoir chacun une inode 47).
- Inconvénient : on ne peut pas faire un lien physique avec un répertoire.
- syntaxe : **ln toto ../R2/titi**.

Lien physique

- Un *lien physique* est une référence directe à un fichier via son inode.
- Avantage : On peut changer le contenu ou la localisation du fichier original, le lien physique restera valide.
- Inconvénient : un lien physique référence un numéro d'inode ce qui impose de rester dans la même partition (en effet : deux disques logiques différents peuvent avoir chacun une inode 47).
- Inconvénient : on ne peut pas faire un lien physique avec un répertoire.
- syntaxe : **ln toto ../R2/titi**.
- Je peux changer **toto** de place (dans la même partition), le lien entre **toto** et **titi** ne disparaît pas.

Lien physique

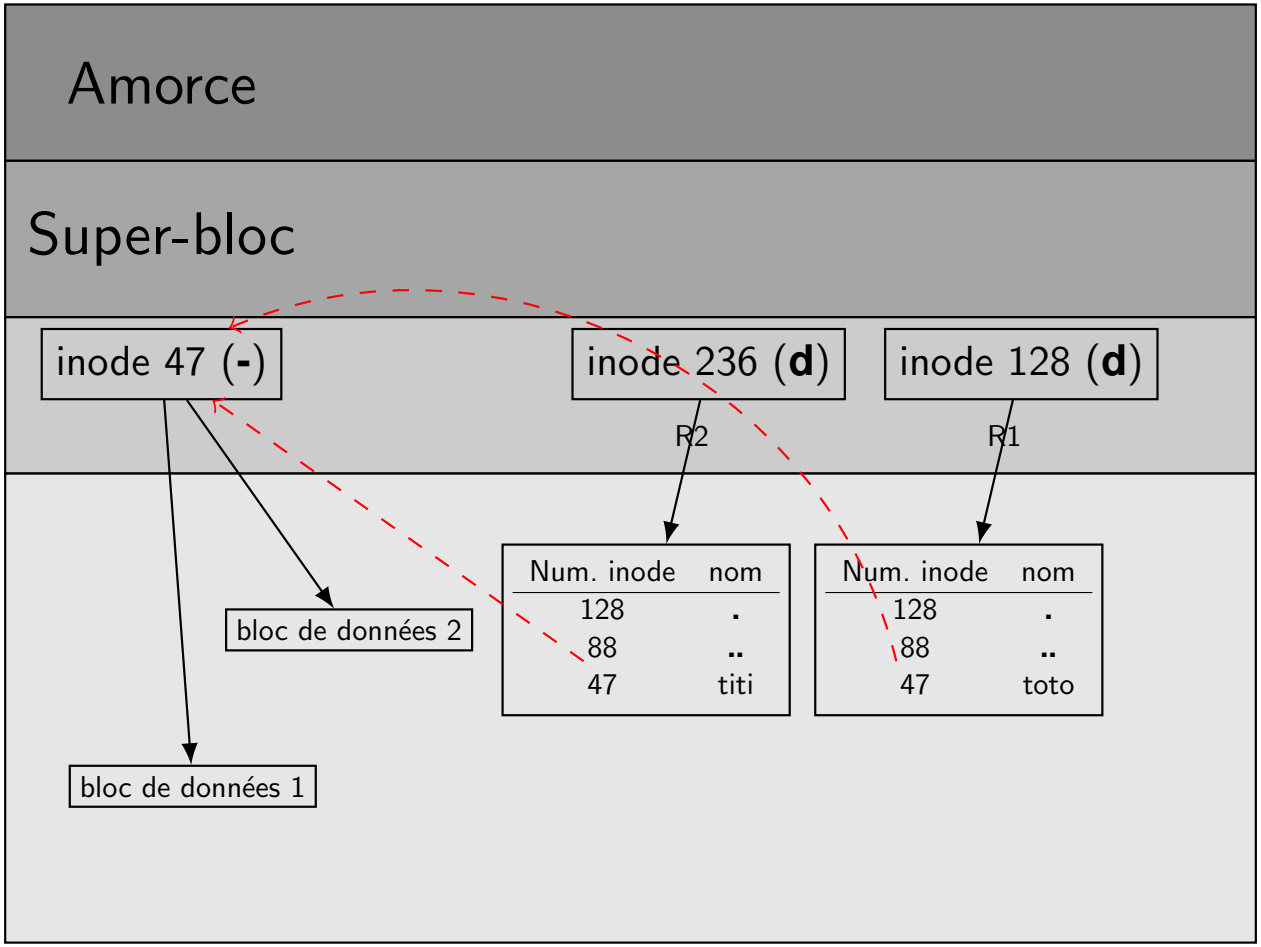


FIGURE 9 – Lien physique : Dans le répertoire courant, on entre **ln toto R2/titi** : **titi** et **toto** pointent vers le même inode. Pas de duplication des données.

Lien physique

- Situation initiale : un fichier **toto** de 121 octets et 1 seul alias.

```
$ ls -ali toto
2373803 -rw-rw-r-- 1 ivan ivan 121 janv. 11 22:03 toto
```

Lien physique

- Situation initiale : un fichier **toto** de 121 octets et 1 seul alias.

```
$ ls -ali toto
2373803 -rw-rw-r-- 1 ivan ivan 121 janv. 11 22:03 toto
```

- Création d'un lien symbolique :

```
$ ln toto titi
$ ls -ali toto
2373803 -rw-rw-r-- 2 ivan ivan 121 janv. 11 22:03 toto
(base) ivan@fixe:~/.../UFS$ ls -ali titi
2373803 -rw-rw-r-- 2 ivan ivan 121 janv. 11 22:03 titi
```

Le nombre d'alias de **toto** est maintenant 2.

Lien physique

- Situation initiale : un fichier **toto** de 121 octets et 1 seul alias.

```
$ ls -ali toto
2373803 -rw-rw-r-- 1 ivan ivan 121 janv. 11 22:03 toto
```

- Création d'un lien symbolique :

```
$ ln toto titi
$ ls -ali toto
2373803 -rw-rw-r-- 2 ivan ivan 121 janv. 11 22:03 toto
(base) ivan@fixe:~/.../UFS$ ls -ali titi
2373803 -rw-rw-r-- 2 ivan ivan 121 janv. 11 22:03 titi
```

  Le nombre d'alias de **toto** est maintenant 2.

- **toto** et **titi** partagent le même inode 2373803.

Suppression et inode

En supprimant un fichier (**rm** pour remove), tout ce qui se passe est la suppression de l'un des noms pointant vers un numéro d'inode spécifique. Les données resteront jusqu'à la suppression de tous les noms associés au même numéro d'inode. Les systèmes Linux se mettent à jour sans nécessiter de redémarrage du système en grande partie à cause du fonctionnement des inodes.

- Dans la situation de la figure 41, entrons **rm toto** : l'inode de **toto** n'est pas libéré puisque le lien de **titi** existe toujours. Voir figure 10.

Suppression et inode

En supprimant un fichier (**rm** pour remove), tout ce qui se passe est la suppression de l'un des noms pointant vers un numéro d'inode spécifique. Les données resteront jusqu'à la suppression de tous les noms associés au même numéro d'inode. Les systèmes Linux se mettent à jour sans nécessiter de redémarrage du système en grande partie à cause du fonctionnement des inodes.

- Dans la situation de la figure 41, entrons **rm toto** : l'inode de **toto** n'est pas libéré puisque le lien de **titi** existe toujours. Voir figure 10.
- Entrons **rm titi**. Les blocs de données et l'inode correspondant sont libérés : le système pourra les attribuer à d'autres fichiers. Voir figure 11.

Suppression et inode

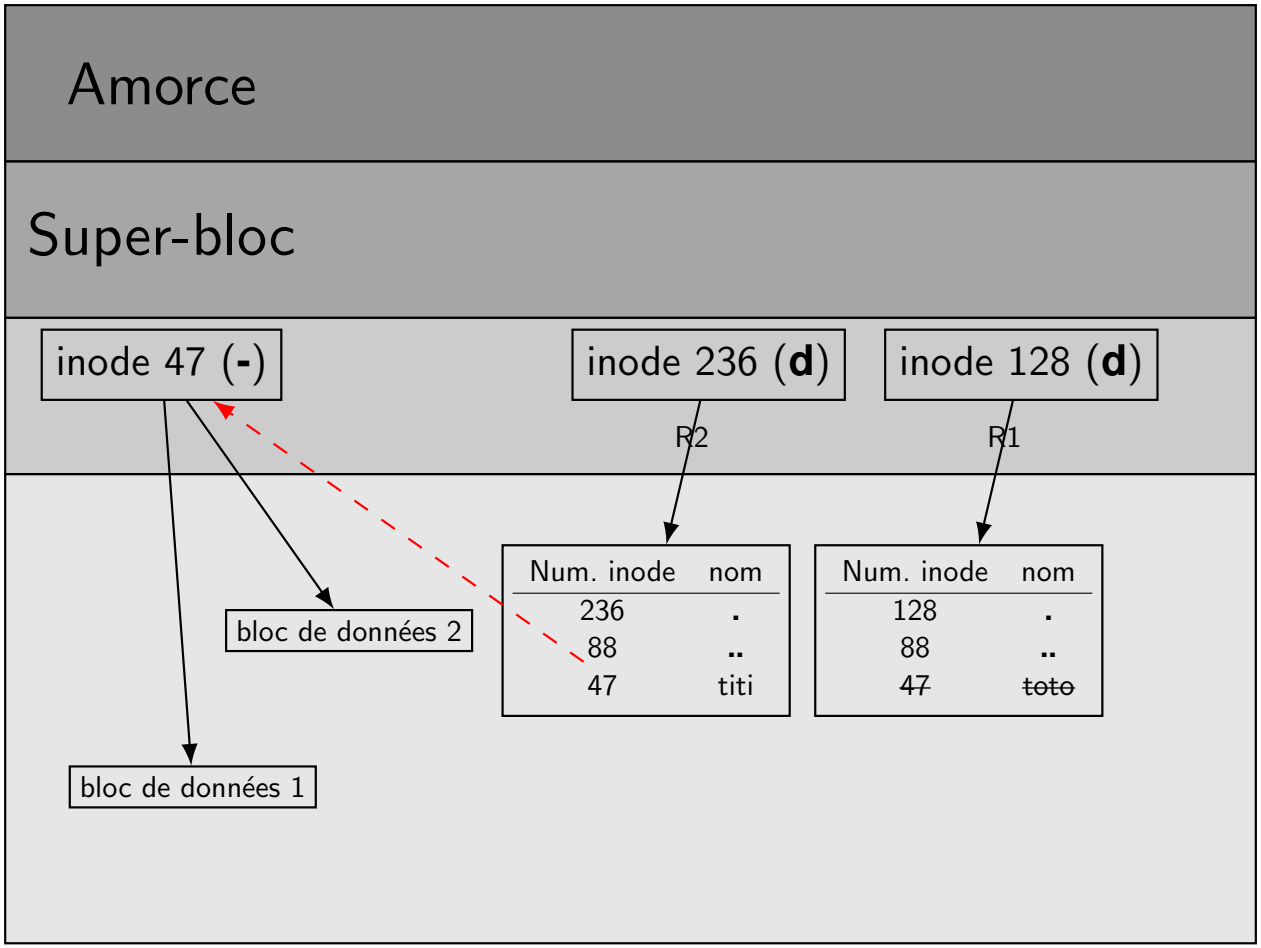


FIGURE 10 – Suppression **rm toto** : l'inode 47 et ses données ne sont pas libérés

Suppression et inode

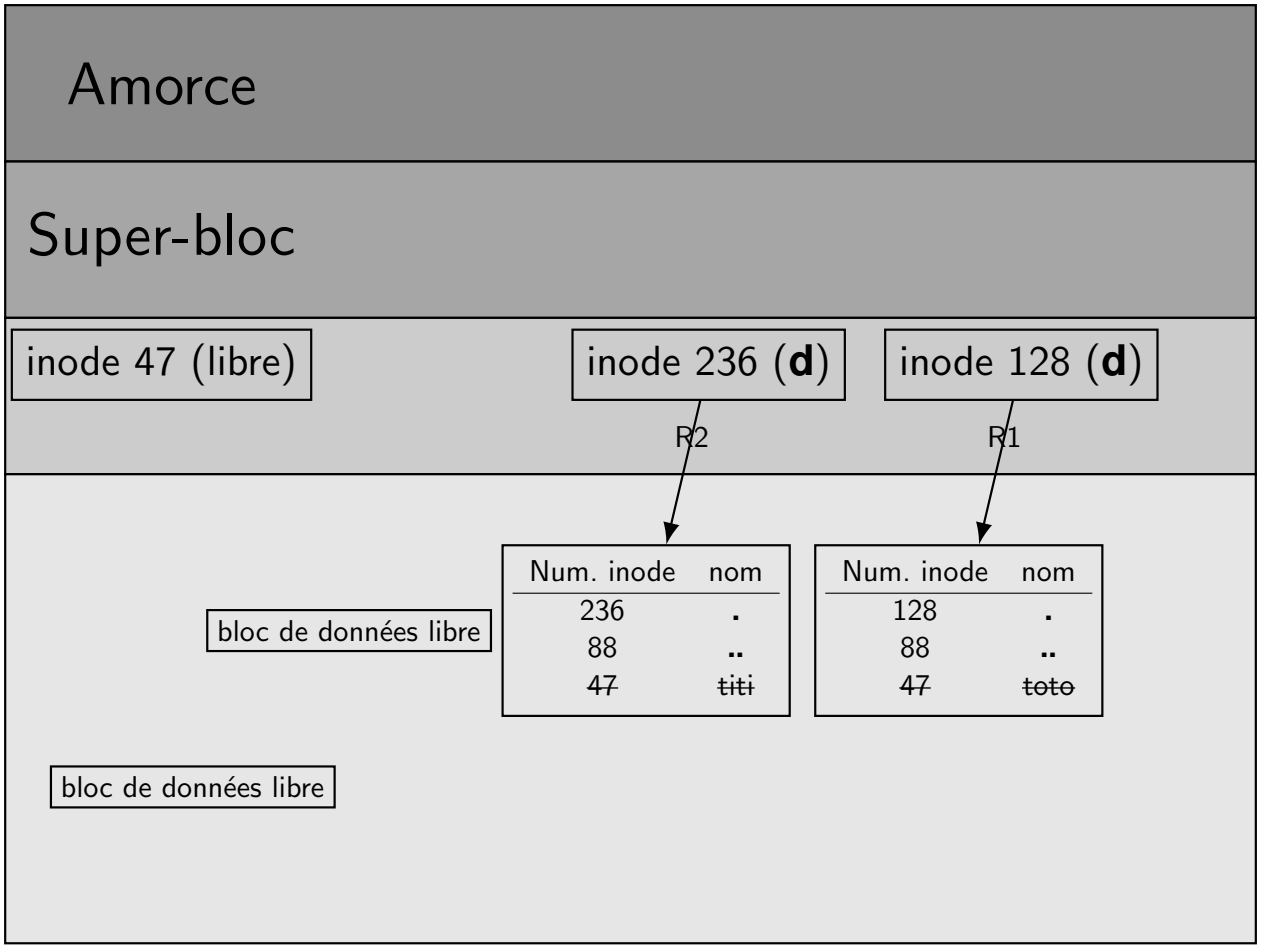


FIGURE 11 – Suppression **rm titi** : l'inode 47 et ses données sont libérées

Lien symbolique

- Un *lien symbolique* est une référence à un nom (*i.e.* un chemin absolu ou relatif). Tandis qu'un lien physique réfère plutôt un inode.

Lien symbolique

- Un *lien symbolique* est une référence à un nom (*i.e.* un chemin absolu ou relatif). Tandis qu'un lien physique réfère plutôt un inode.
- Avec un lien symbolique on peut traverser les partitions.

Lien symbolique

- Un *lien symbolique* est une référence à un nom (*i.e.* un chemin absolu ou relatif). Tandis qu'un lien physique réfère plutôt un inode.
- Avec un lien symbolique on peut traverser les partitions.
- Création avec l'option **s** de **ln** :
  ln -s /path/to/original symlink. Par exemple **ln -s R1/toto titi** crée un lien symbolique entre **titi** du répertoire courant et **toto** du répertoire **R1**.

Lien symbolique

- Un *lien symbolique* est une référence à un nom (*i.e.* un chemin absolu ou relatif). Tandis qu'un lien physique réfère plutôt un inode.
- Avec un lien symbolique on peut traverser les partitions.
- Création avec l'option **s** de **ln** :
  ln -s /path/to/original symlink. Par exemple **ln -s R1/toto titi** crée un lien symbolique entre **titi** du répertoire courant et **toto** du répertoire **R1**.
- Par défaut, le lien symbolique doit être créé dans le répertoire courant. Si on veut spécifier aussi un chemin relatif pour le lien symbolique, utiliser l'option **r** :

```
-r, --relative
              create symbolic links relative to link location
```

ln -sr R1/toto R2/titi. Voir figure 12.

Lien symbolique

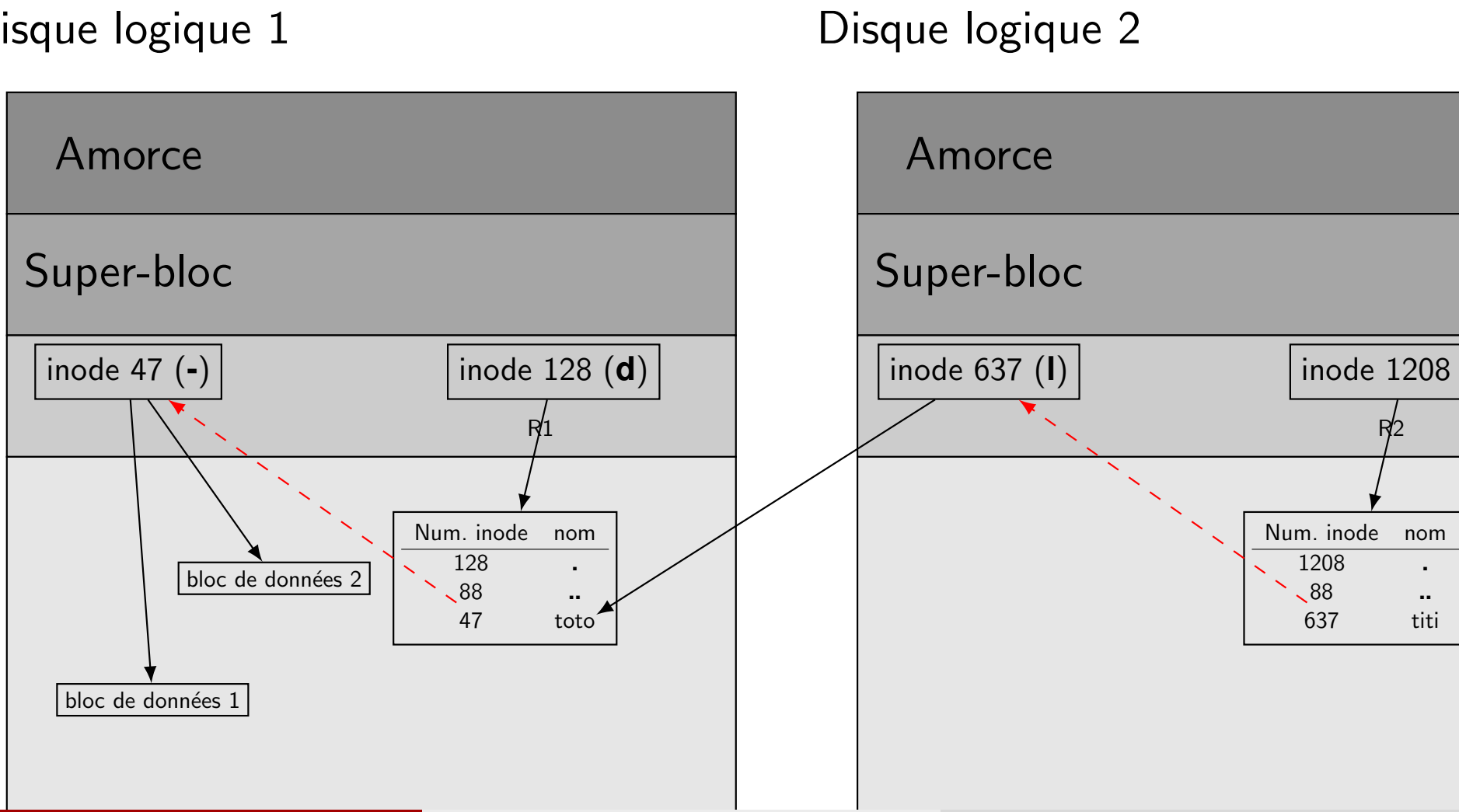
- Un *lien symbolique* est une référence à un nom (*i.e.* un chemin absolu ou relatif). Tandis qu'un lien physique réfère plutôt un inode.
- Avec un lien symbolique on peut traverser les partitions.
- Création avec l'option **s** de **ln** :
  ln -s /path/to/original symlink. Par exemple **ln -s R1/toto titi** crée un lien symbolique entre **titi** du répertoire courant et **toto** du répertoire **R1**.
- Par défaut, le lien symbolique doit être créé dans le répertoire courant. Si on veut spécifier aussi un chemin relatif pour le lien symbolique, utiliser l'option **r** :

```
-r, --relative
              create symbolic links relative to link location
```

  ln -sr R1/toto R2/titi. Voir figure 12.
- Si on déplace ou renomme la source, le lien est cassé.

Lien symbolique

Lien symbolique **ln -sr R1/toto R2/titi**



Lien symbolique

```
$ stat toto
  Fichier: toto
   Taille: 121  Blocs: 8 Blocs d'E/S: 4096 fichier
...
$ ln -s toto lstoto
$ ls -al toto lstoto
lrwxrwxrwx 1 ivan ivan 4 janv. 17 11:33 lstoto -> toto
-rw-rw-r-- 2 ivan ivan 121 janv. 11 2023 toto
$ stat lstoto
Fichier: lstoto -> toto
   Taille: 4  Blocs: 0 Blocs d'E/S: 4096 lien symbolique
```

- La taille de **toto** est de 121 bytes (121 carcatères), celle du lien symbolique de 4.

Lien symbolique

```
$ stat toto
  Fichier: toto
   Taille: 121  Blocs: 8 Blocs d'E/S: 4096 fichier
...
$ ln -s toto lstoto
$ ls -al toto lstoto
lrwxrwxrwx 1 ivan ivan 4 janv. 17 11:33 lstoto -> toto
-rw-rw-r-- 2 ivan ivan 121 janv. 11 2023 toto
$ stat lstoto
Fichier: lstoto -> toto
   Taille: 4  Blocs: 0 Blocs d'E/S: 4096 lien symbolique
```

- La taille de **toto** est de 121 bytes (121 carcatères), celle du lien symbolique de 4.
- On constate que la première lettre du retour de **ls** est **l**. Et qu'un lien symbolique possède tous les droits par défaut.

chown

exemple avec un script qui affiche coucou
+ export path