Systèmes UNIX. Droits

Sergey Kirgizov

Merci Benoît !

Ce cours est fait en partie à partir du cours de Benoît Darties.
`https://benoit.darties.fr/`

Aujourd'hui

Droits d'accès et restrictions

Manipulation de fichiers

Fichiers, utilisateurs multiples, et configurations originales

Droits d'accès et restrictions

De la nécessité de droits d'accès ...

- Commandes vues jusqu'à présent : exécutées dans un système idéal, supposé sans restrictions.
- Sans restrictions, tout utilisateur pourrait naviguer n'importe où dans l'arborescence, créer et supprimer des fichiers où bon lui semble, ...
- Conséquence : accès à des informations personnelles, altération du système
- Besoin de mécanismes permettant de garantir la confidentialité de certaines données par rapport aux utilisateurs, ou de protéger certaines parties sensibles du système

Fichiers et propriétaire

Notion de propriétaire et groupe pour un fichier

- Un fichier, quelle que soit sa nature, possède un et un seul propriétaire[a]. Généralement : le créateur du fichier
- Il est également associé à un et un seul groupe d'utilisateurs. Généralement : le groupe par défaut du créateur du fichier
- Visualiser le propriétaire et le groupe d'un fichier : commande `ls` avec option `-l`

---

a. L'utilisation des ACL - Access Control List, n'est pas abordée

```
1 benoit$ ls -l fichier.jpg
2 -rw-r--r--  1 benoit  staff  6536142  6 jul 02:59 fichier.jpg
```

Fichiers et propriétaire

Classe d'utilisateurs

Pour un fichier, les utilisateurs sont partitionnés en trois classes :

- u : le propriétaire du fichier (user)
- g : les membres du groupe associé au fichier (group)
- o : tous les autres utilisateurs (others)

Les droits d'accès et d'utilisation d'un fichier (vus après) sont gérés de manière indépendante pour chacune de ces classes d'utilisateurs

Actions possibles sur fichiers

En fonction de son type, des droits sur un fichier vont différer :

Fichier régulier

- r : droit de lecture (read)
- w : droit d'écriture (write)
- x : droit d'exécution (execute) : n'a de sens que pour un programme ou un script

Fichier répertoire

- r : droit de lister le contenu du répertoire (read)
- w : droit d'écriture dans le répertoire (write)
  ajout / suppression de fichiers dans le répertoire
- x : droit de navigation et positionnement dans le répertoire

Actions possibles sur fichiers

En fonction de son type, des droits sur un fichier vont différer :

Fichier régulier

- r : droit de lecture (read)
- w : droit d'écriture (write)
- x : droit d'exécution (execute) : n'a de sens que pour un programme ou un script

Fichier répertoire

- r : droit de lister le contenu du répertoire (read)
- w : droit d'écriture dans le répertoire (write)
  ajout / suppression de fichiers dans le répertoire
- x : droit de navigation et positionnement dans le répertoire

Droits sur les fichiers

- 3 classes d'utilisateurs $\times$ 3 types de droits par utilisateur
- Soit 9 types droits (2^9 combinaisons) :

propriétaire			groupe			autres		
r	w	x	r	w	x	r	w	x

Exemples de droits possibles

- Fichier exécutable protégé en écriture

r	-	x	r	-	x	r	-	x

- Fichier accessible et exécutable pour le propriétaire seulement

r	w	x	-	-	-	-	-	-

- Fichier non exécutable éditable seulement par son propriétaire, et qui ne peut pas être lu par les autres.

r	w	-	r	-	-	-	-	-

Droits sur les fichiers

- 3 classes d'utilisateurs $\times$ 3 types de droits par utilisateur
- Soit 9 types droits (2^9 combinaisons) :

propriétaire			groupe			autres		
r	w	x	r	w	x	r	w	x

Exemples de droits possibles

- Fichier exécutable protégé en écriture

r	-	x	r	-	x	r	-	x

- Fichier accessible et exécutable pour le propriétaire seulement

r	w	x	-	-	-	-	-	-

- Fichier non exécutable éditable seulement par son propriétaire, et qui ne peut pas être lu par les autres.

r	w	-	r	-	-	-	-	-

Visualisation des droits : commande `ls`

- Visualiser les droits actuellement appliqués sur un fichier : commande `ls` avec option `-l`

```
1 benoit$ ls -l fichier.jpg
2 -rw-r--r--  1 benoit  staff  6536142  6 jul 02:59 fichier.jpg
```

Modifier les droits : commande `chmod`

- Commande `chmod` (*change file mode*) appliquée sur fichier
  - Specificateur de droits construit à partir de symboles
  - u : user, g : group, o : others, a : all (par défaut : all)
  - = : affectation, + : ajout, - suppression
  - r : lecture, w : écriture, x : exécution / navigation

Exemples

- `chmod u+rwx file.txt`
- `chmod ug=rw file.txt`
- `chmod -w file.txt`
- `chmod go+rx-w file.txt`
- `chmod u=rwx,go=rw file.txt`
- `chmod u=rw,go=u-w file.txt`

Visualisation des droits : commande `ls`

- Visualiser les droits actuellement appliqués sur un fichier : commande `ls` avec option `-l`

```
1 benoit$ ls -l fichier.jpg
2 -rw-r--r--  1 benoit  staff  6536142  6 jul 02:59 fichier.jpg
```

Modifier les droits : commande `chmod`

- Commande `chmod` (*change file mode*) appliquée sur fichier
  - Specificateur de droits construit à partir de symboles
  - u : user, g : group, o : others, a : all (par défaut : all)
  - = : affectation, + : ajout, - suppression
  - r : lecture, w : écriture, x : exécution / navigation

Exemples

- `chmod u+rwx file.txt`
- `chmod ug=rw file.txt`
- `chmod -w file.txt`
- `chmod go+rx-w file.txt`
- `chmod u=rwx,go=rw file.txt`
- `chmod u=rw,go=u-w file.txt`

Visualisation des droits : commande `ls`

- Visualiser les droits actuellement appliqués sur un fichier : commande `ls` avec option `-l`

```
1 benoit$ ls -l fichier.jpg
2 -rw-r--r--  1 benoit  staff  6536142  6 jul 02:59 fichier.jpg
```

Modifier les droits : commande `chmod`

- Commande `chmod` (*change file mode*) appliquée sur fichier
  - Specificateur de droits construit à partir de symboles
  - u : user, g : group, o : others, a : all (par défaut : all)
  - = : affectation, + : ajout, - suppression
  - r : lecture, w : écriture, x : exécution / navigation

Exemples

- `chmod u+rwx file.txt`
- `chmod ug=rw file.txt`
- `chmod -w file.txt`
- `chmod go+rx-w file.txt`
- `chmod u=rwx,go=rw file.txt`
- `chmod u=rw,go=u-w file.txt`

Un exemple guidé

```
1  benoit$ ls -l file.txt
2  -rw-rw-rw-  1 benoit  staff  0  7 jul 01:43 file.txt
3
4  benoit$ chmod u+x file.txt
5  benoit$ ls -l file.txt
6  -rwxrw-rw-  1 benoit  staff  0  7 jul 01:43 file.txt
7
8  benoit$ chmod go-r file.txt
9  benoit$ ls -l file.txt
10 -rwx-w--w-  1 benoit  staff  0  7 jul 01:43 file.txt
11
12 benoit$ chmod u=rx file.txt
13 benoit$ ls -l file.txt
14 -r-x-w--w-  1 benoit  staff  0  7 jul 01:43 file.txt
15
16 benoit$ chmod g=u,o-w+r file.txt
17 benoit$ ls -l file.txt
18 -r-xr-xr--  1 benoit  staff  0  7 jul 01:43 file.txt
19
20 benoit$ chmod +rwx,go-w file.txt
21 benoit$ ls -l file.txt
22 -rwxr-xr-x  1 benoit  staff  0  7 jul 01:43 file.txt
```

Droits sur les fichiers

- Peut être vu comme trois mot binaires de 3 chiffres chacun :
  - un chiffre de valeur 1 représente une autorisation
  - un chiffre de valeur 0 représente une interdiction
- Chaque mot binaire peut se transformer en une valeur décimale

exemple :

propriétaire			groupe			autres			
r	w	x	r	-	x	r	-	-	notation en lettres
1	1	1	1	0	1	1	0	0	notation binaire
$\times$ 2^2	$\times$ 2^1	$\times$ 2^0	$\times$ 2^2	$\times$ 2^1	$\times$ 2^0	$\times$ 2^2	$\times$ 2^1	$\times$ 2^0	puissance de deux
4	2	1	4	0	1	4	0	0	
7			5			4			notation décimale

Droits sur les fichiers

- Peut être vu comme trois mot binaires de 3 chiffres chacun :
  - un chiffre de valeur 1 représente une autorisation
  - un chiffre de valeur 0 représente une interdiction
- Chaque mot binaire peut se transformer en une valeur décimale

exemple :

propriétaire			groupe			autres			
r	w	x	r	-	x	r	-	-	notation en lettres
1	1	1	1	0	1	1	0	0	notation binaire
$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	
2^2	2^1	2^0	2^2	2^1	2^0	2^2	2^1	2^0	puissance de deux
4	2	1	4	0	1	4	0	0	
7			5			4			notation décimale

Question :

Transformez cette notation à lettres en notation décimale

propriétaire			groupe			autres			
r	-	x	r	w	x	-	-	x	notation en lettres
$\times 2^2$	$\times 2^1$	$\times 2^0$	$\times 2^2$	$\times 2^1$	$\times 2^0$	$\times 2^2$	$\times 2^1$	$\times 2^0$	notation binaire puissance de deux
									notation décimale

Question :

Transformez cette notation décimale en notation à lettres

propriétaire			groupe			autres			
3			4			6			notation décimale
									notation en lettres

Un exemple guidé

```
1  benoit$ ls -l file.txt
2  -rw-rw-rw-  1 benoit  staff  0  7 jul 01:43 file.txt
3
4  benoit$ chmod 600 file.txt
5  benoit$ ls -l file.txt
6  -rw-------  1 benoit  staff  0  7 jul 01:43 file.txt
7
8  benoit$ chmod 742 file.txt
9  benoit$ ls -l file.txt
10 -rwxr---w-  1 benoit  staff  0  7 jul 01:43 file.txt
11
12 benoit$ chmod 145 file.txt
13 benoit$ ls -l file.txt
14 ---xr--r-x  1 benoit  staff  0  7 jul 01:43 file.txt
15
16 benoit$ chmod 557 file.txt
17 benoit$ ls -l file.txt
18 -r-xr-xrwx  1 benoit  staff  0  7 jul 01:43 file.txt
```

Questions

Créez les fichiers suivants et donnez leurs les droits correspondants :

1. `Salameche.txt` avec les droits `rwxrwxrwx`
2. `Bulbizarre.txt` avec les droits `r-xr-xr--`
3. `Dracofeu.txt` avec les droits `---rw-r-`
4. `Pokedex` (répertoire) avec les droits `---rw-r-x`

Réponse :

Questions

Créez les fichiers suivants et donnez leurs les droits correspondants :

1. `Salameche.txt` avec les droits `rwxrwxrwx`
2. `Bulbizarre.txt` avec les droits `r-xr-xr--`
3. `Dracofeu.txt` avec les droits `---rw-r-`
4. `Pokedex` (répertoire) avec les droits `---rw-r-x`

Réponse :

Questions

Créez les fichiers suivants et donnez leurs les droits correspondants :

1. `Salameche.txt` avec les droits `rwxrwxrwx`
2. `Bulbizarre.txt` avec les droits `r-xr-xr--`
3. `Dracofeu.txt` avec les droits `---rw-r-`
4. `Pokedex` (répertoire) avec les droits `---rw-r-x`

Réponse :

1. `touch Salameche.txt; chmod 777 Salameche.txt`

Questions

Créez les fichiers suivants et donnez leurs les droits correspondants :

1. `Salameche.txt` avec les droits `rwxrwxrwx`
2. `Bulbizarre.txt` avec les droits `r-xr-xr--`
3. `Dracofeu.txt` avec les droits `---rw-r-`
4. `Pokedex` (répertoire) avec les droits `---rw-r-x`

Réponse :

1. `touch Salameche.txt; chmod 777 Salameche.txt`
2. `touch Bulbizarre.txt; chmod 550 Bulbizarre.txt`

Questions

Créez les fichiers suivants et donnez leurs les droits correspondants :

1. `Salameche.txt` avec les droits `rwxrwxrwx`
2. `Bulbizarre.txt` avec les droits `r-xr-xr--`
3. `Dracofeu.txt` avec les droits `---rw-r-`
4. `Pokedex` (répertoire) avec les droits `---rw-r-x`

Réponse :

1. `touch Salameche.txt; chmod 777 Salameche.txt`
2. `touch Bulbizarre.txt; chmod 550 Bulbizarre.txt`
3. `touch Dracofeu.txt; chmod 064 Dracofeu.txt`

Questions

Créez les fichiers suivants et donnez leurs les droits correspondants :

1. `Salameche.txt` avec les droits `rwxrwxrwx`
2. `Bulbizarre.txt` avec les droits `r-xr-xr--`
3. `Dracofeu.txt` avec les droits `---rw-r-`
4. `Pokedex` (répertoire) avec les droits `---rw-r-x`

Réponse :

1. `touch Salameche.txt; chmod 777 Salameche.txt`
2. `touch Bulbizarre.txt; chmod 550 Bulbizarre.txt`
3. `touch Dracofeu.txt; chmod 064 Dracofeu.txt`
4. `mkdir Pokedex; chmod 065 Pokedex`

Droits d'accès spéciaux

Remarque

- Il existe d'autres types de droits, représentés par une quatrième valeur octale
- setUID (+s), setGID (+s) , sticky Bit (+t)
- abordés plus tard dans ce cours (chaque chose en son temps)

Droits D'accès spéciaux

setUID : set User ID on execution : (+s)

- sur fichier régulier : si exécutable, s'exécutera avec les droits du propriétaire, et non de celui qui qui lancera le programme

setGID : set Group ID on execution : (+s)

- sur fichier régulier : si exécutable, s'exécutera avec les droits du groupe, et non de celui qui qui lancera le programme
- sur répertoire : les nouveaux fichiers créés dans ce répertoire hériteront du groupe du répertoire

Sticky bit

- sur fichier exécutable : généralement pas d'effet
- sur répertoire : les nouveaux fichiers créés dans ce répertoire ne pourront être supprimés que par leur propriétaire

Manipulation de fichiers

Droits et navigation

Droits nécessaires pour naviguer dans une arborescence

- Droits de navigation (+x) sur tous les répertoires ascendants (situés entre la racine et le répertoire visé)
- Droit de navigation (+x) sur le répertoire visé
- Si un des répertoires sans droit de navigation : accès refusé

Droits nécessaires pour lister le contenu d'un répertoire

- droit navigation (+x) sur tous les répertoires ascendants
- droit de lecture (+r) sur le répertoire visé

Un exemple guidé

```
1  benoit$ pwd
2  /Users/benoit
3
4  benoit$ ls -l
5  drwxr-xr-x  2 benoit  staff   68  20 jul 12:14  monRep
6  drwxr-xr-x  2 benoit  staff 2414  15 jul 12:07  Bureau
7  drwxr-xr-x  2 benoit  staff 2414  15 jul 12:07  MP3
8
9  benoit$ cd monRep/
10
11 Galactica:monRep benoit$ pwd
12 /Users/benoit/monRep
13
14 Galactica:monRep benoit$ cd ..
15
16 benoit$ chmod -x monRep
17
18 benoit$ cd monRep/
19 bash: cd: monRep/: Permission denied
20
21 benoit$ chmod u+x monRep/
22
23 benoit$ cd monRep/
24
25 Galactica:monRep benoit$ pwd
26 /Users/benoit/monRep
```

Créer un fichier

Créer un nouveau fichier (régulier ou répertoire)

D'une manière générale, créer un nouveau fichier revient à :

- ajouter une entrée dans la table d'attributs des fichiers,
- ajouter un lien physique de cette nouvelle entrée vers le répertoire dans lequel sera contenu le fichier

Droits nécessaires :

- Navigation (+x) jusqu'au répertoire contenant le fichier
- Ecriture (+w) sur le répertoire qui contiendra le fichier

Créer un nouveau fichier régulier : commande `touch`

- usage : `touch` *`nomfichier`*
- crée le fichier *nomfichier* s'il n'existait pas
- autre usage (originel) : change la date de dernier accès et modification de *nomfichier*

Créer un nouveau répertoire : commande `mkdir`

- usage : `mkdir [-p]` *`nomRep`*
- Crée un répertoire nommé *nomRep* dans le répertoire de travail
- avec option -p, crée l'ensemble des sous-répertoire si *nomRep* est une arborescence de répertoires

Créer un nouveau fichier régulier : commande `touch`

- usage : `touch` *`nomfichier`*
- crée le fichier *nomfichier* s'il n'existait pas
- autre usage (originel) : change la date de dernier accès et modification de *nomfichier*

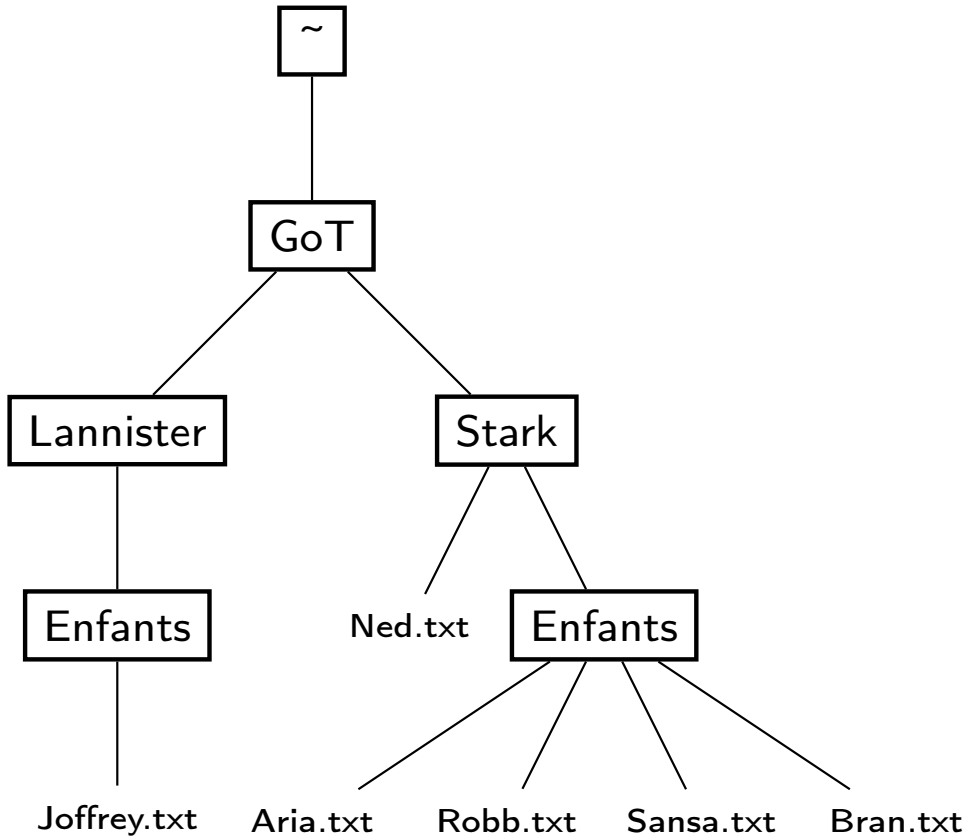
Créer un nouveau répertoire : commande `mkdir`

- usage : `mkdir [-p]` *`nomRep`*
- Crée un répertoire nommé *nomRep* dans le répertoire de travail
- avec option -p, crée l'ensemble des sous-répertoire si *nomRep* est une arborescence de répertoires

Un exemple guidé

```
1  benoit$ mkdir GoT
2  benoit$ cd GoT/
3  Galactica:GoT benoit$ pwd
4  /Users/benoit/GoT
5
6  Galactica:GoT benoit$ mkdir Stark
7  Galactica:GoT benoit$ cd Stark/
8  Galactica:Enfants benoit$ pwd
9  /Users/benoit/GoT/Stark
10 Galactica:Stark benoit$ touch Ned.txt
11
12 Galactica:Stark benoit$ mkdir Enfants
13 Galactica:Stark benoit$ cd Enfants/
14 Galactica:Enfants benoit$ touch Robb.txt
15 Galactica:Enfants benoit$ cd ..
16
17 Galactica:Stark benoit$ pwd
18 /Users/benoit/GoT/Stark
19 Galactica:Stark benoit$ touch ./Enfants/Aria.txt
20 Galactica:Stark benoit$ touch ./Enfants/Bran.txt ./Enfants/Sansa.txt
21
22 Galactica:Stark benoit$ cd Enfants/
23 Galactica:Enfants benoit$ ls
24 Aria.txt  Bran.txt  Robb.txt  Sansa.txt
25
26 Galactica:Enfants benoit$ cd ~
27 benoit$ pwd
28 /Users/benoit
29 benoit$ mkdir -p ./GoT/Lannister/Enfants
30 benoit$ touch ./GoT/Lannister/Enfants/Joffrey.txt
```

Résultat de l'arborescence créée



Lecture et modification de fichiers réguliers

Droits nécessaires pour lire le contenu d'un fichier régulier

- Navigation (+x) jusqu'au répertoire contenant le fichier
- Droits de lecture (+r) sur le fichier

Droits nécessaires pour modifier le contenu d'un fichier régulier

On suppose que le fichier existe déjà

- Navigation (+x) jusqu'au répertoire contenant le fichier
- Droits d'écriture (+w) sur le fichier

Si le fichier n'existe pas, il faut d'abord le créer : cf droits création nouveau fichier.

Remarque

Droits similaire pour lire / modifier un fichier de type répertoire

Copier le contenu d'un fichier

Copier le contenu d'un fichier source vers un fichier destination

- Deux cas de figure : le fichier destination existe ou n'existe pas
- Lecture du contenu du fichier source
- Ecriture du contenu vers le fichier destination
- Généralement, le contenu du fichier de destination est écrasé
- Après copie, les deux fichiers auront le même contenu
- Mais seront deux fichiers bien distincts :
  - Deux entrées différentes dans la table d'attributs des fichiers
  - Deux inodes différents
  - Blocs utilisés par les fichiers bien distincts
  - La modification a postériori de l'un n'entrainera pas la modification de l'autre

Droits nécessaires

Si le fichier destination existe

- Droits de navigation (+x) jusqu'au fichier source
- Droits de navigation (+x) jusqu'au fichier destination
- Droits de lecture (+r) sur le fichier source
- Droits d'écriture (+w) sur le fichier destination

Si le fichier destination n'existe pas

- Copie du fichier source dans un autre répertoire
- Equivalent à création d'un nouveau fichier destination, et copie d'un fichier source vers un fichier destination existant
  - Droits de navigation (+x) jusqu'au fichier source
  - Droits de lecture (+r) sur le fichier source
  - Droits de navigation (+x) jusqu'au répertoire de destination
  - Droits d'écriture (+w) sur le répertoire de destination

Rappels : questions ouvertes

Vu la méthode de codage de l'arborescence :

- Un fichier de type fichier régulier pourrait-il apparaître dans plusieurs répertoires quelconques ?
- Un fichier de type répertoire pourrait-il apparaître dans plusieurs répertoires quelconques ?

Conceptuellement : rien ne l'empêche !

- Il suffit de pouvoir ajouter, dans la liste des entrées du répertoire *destination*, une entrée dont l'inode correspond à un fichier *source* qui était déjà présent dans un autre répertoire
- Dit autrement : de créer un lien physique du fichier *source* vers le répertoire *destination*

Création de liens physiques

Dans la pratique : liens physiques sur fichiers réguliers

- On peut créer des liens physiques sur des fichiers réguliers vers n'importe quel répertoire, si les droits le permettent
- Traduction : un fichier peut appartenir à plusieurs répertoires !
- Droits nécessaires :
  - navigation (+x) jusqu'au répertoire contenant le fichier source
  - navigation et écriture (+wx) sur le répertoire de destination

Dans la pratique : liens physiques sur répertoires restreints

- Un répertoire peut avoir plusieurs liens physiques (il en a déjà 2 de base : un dans lui même, un dans son répertoire parent)
- MAIS pas possible de créer ses propres liens physiques entre répertoires : interdit par l'OS
- Risque de briser la structure arborescente

Création de liens physiques

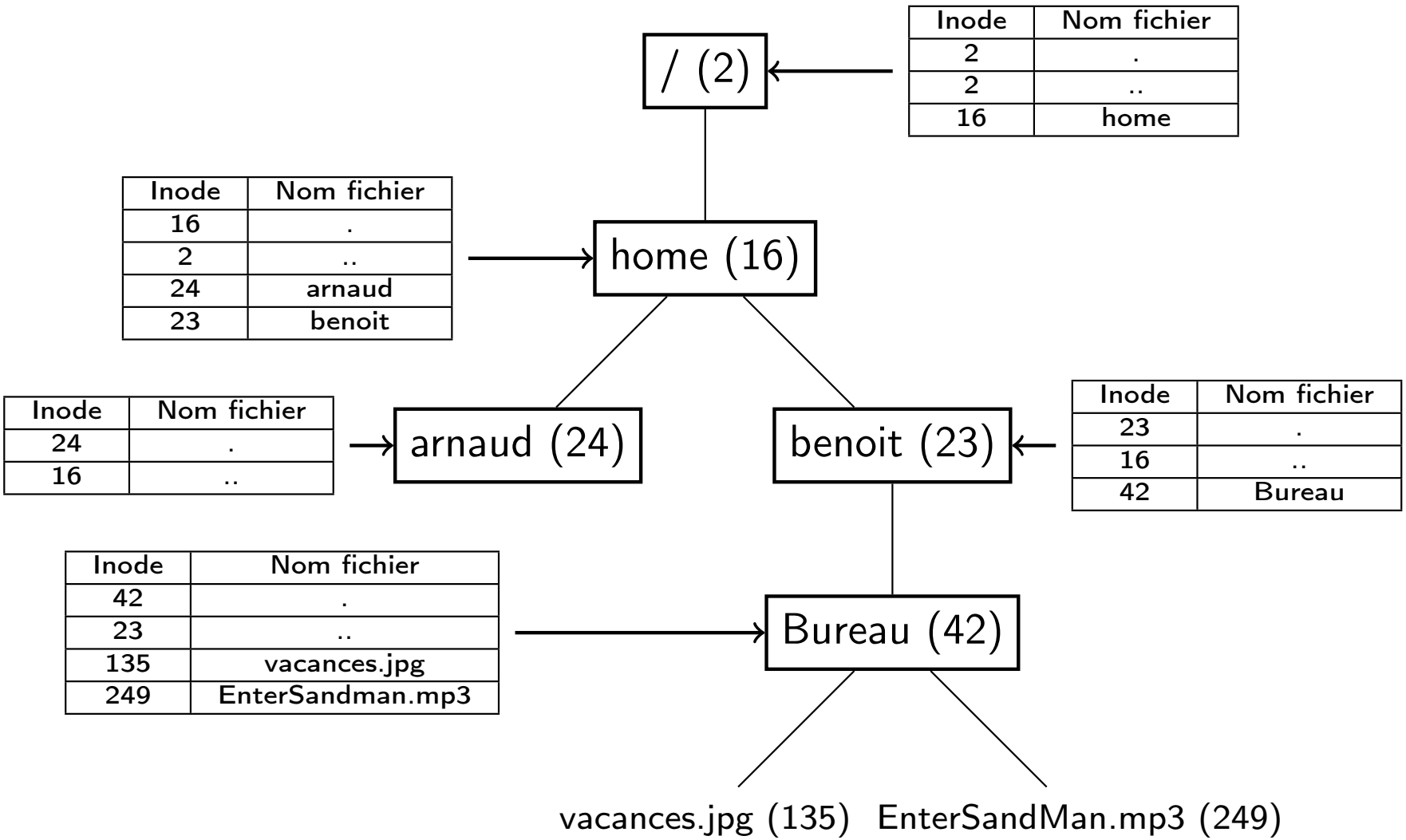
Dans la pratique : liens physiques sur fichiers réguliers

- On peut créer des liens physiques sur des fichiers réguliers vers n'importe quel répertoire, si les droits le permettent
- Traduction : un fichier peut appartenir à plusieurs répertoires !
- Droits nécessaires :
  - navigation (+x) jusqu'au répertoire contenant le fichier source
  - navigation et écriture (+wx) sur le répertoire de destination

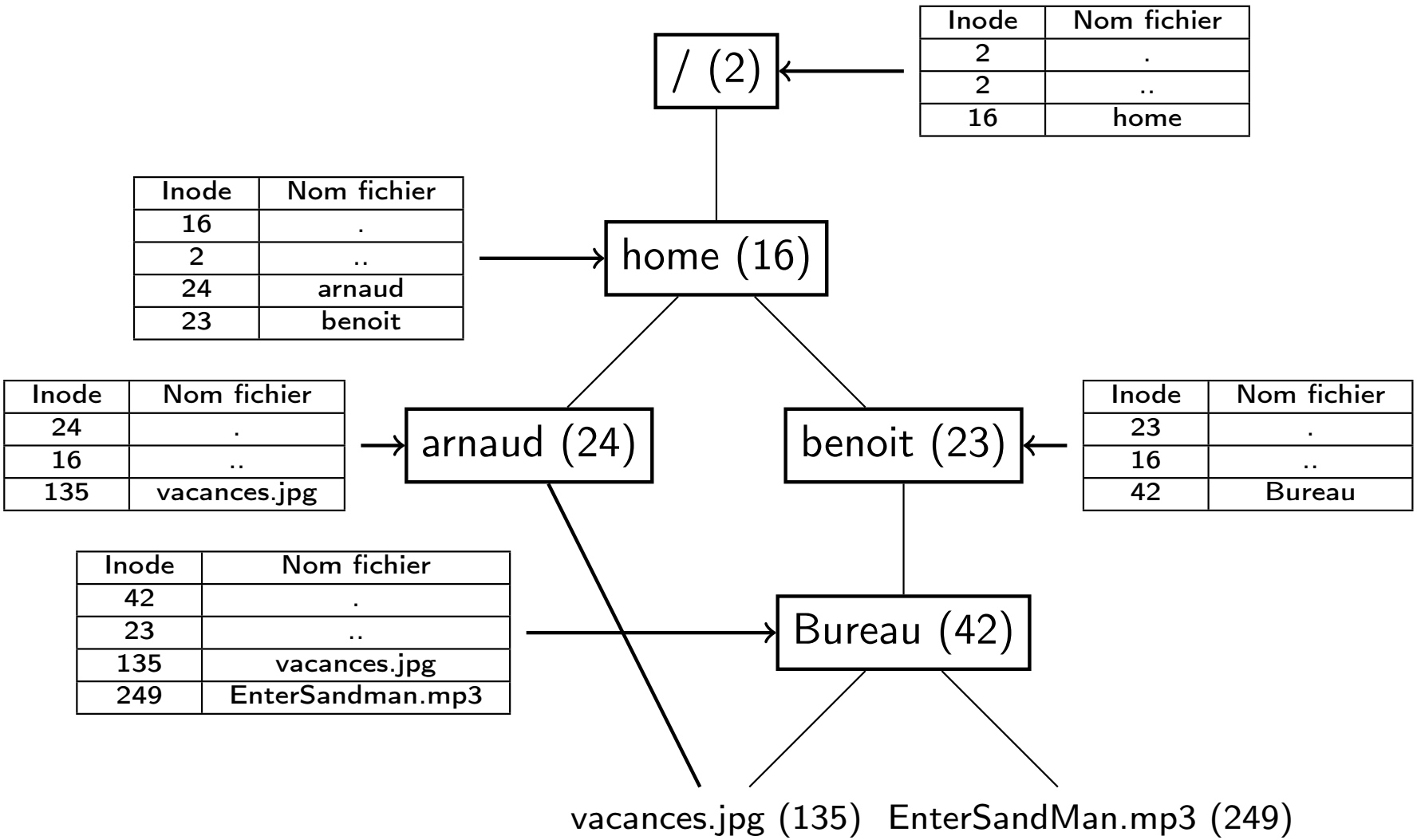
Dans la pratique : liens physiques sur répertoires restreints

- Un répertoire peut avoir plusieurs liens physiques (il en a déjà 2 de base : un dans lui même, un dans son répertoire parent)
- MAIS pas possible de créer ses propres liens physiques entre répertoires : interdit par l'OS
- Risque de briser la structure arborescente

Création d'un lien physique sur fichier régulier



Création d'un lien physique sur fichier régulier



Création d'un lien physique sur fichier régulier

Commande `ln` (ou `link`)

- usage : `ln` *`source destination`*

Fichier avant création d'un lien (options -l et -i) :

```
1 ls -li /home/benoit/Bureau/vacances.jpg
2 135 -rw-r--r--  1 benoit  staff  132796 21 jul 16:08 vacances.jpg
```

Créeons un lien physique dans le répertoire /home/arnaud/

```
1 ln /home/benoit/Bureau/vacances.jpg /home/arnaud/
```

Le fichier est alors présent dans les deux répertoires :

```
1 ls -li /home/benoit/Bureau/vacances.jpg
2 135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg
3
4 ls -li /home/arnaud/vacances.jpg
5 135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg
```

Création d'un lien physique sur fichier régulier

Commande `ln` (ou `link`)

- usage : `ln` *`source destination`*

Fichier avant création d'un lien (options -l et -i) :

```
1 ls -li /home/benoit/Bureau/vacances.jpg
2 135 -rw-r--r--  1 benoit  staff  132796 21 jul 16:08 vacances.jpg
```

Créeons un lien physique dans le répertoire /home/arnaud/

```
1 ln /home/benoit/Bureau/vacances.jpg /home/arnaud/
```

Le fichier est alors présent dans les deux répertoires :

```
1 ls -li /home/benoit/Bureau/vacances.jpg
2 135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg
3
4 ls -li /home/arnaud/vacances.jpg
5 135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg
```

Création d'un lien physique sur fichier régulier

Commande `ln` (ou `link`)

- usage : `ln` *`source destination`*

Fichier avant création d'un lien (options -l et -i) :

```
1 ls -li /home/benoit/Bureau/vacances.jpg
2 135 -rw-r--r--  1 benoit  staff  132796 21 jul 16:08 vacances.jpg
```

Créeons un lien physique dans le répertoire /home/arnaud/

```
1 ln /home/benoit/Bureau/vacances.jpg /home/arnaud/
```

Le fichier est alors présent dans les deux répertoires :

```
1 ls -li /home/benoit/Bureau/vacances.jpg
2 135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg
3
4 ls -li /home/arnaud/vacances.jpg
5 135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg
```

Création d'un lien physique sur fichier régulier

Commande `ln` (ou `link`)

- usage : `ln` *`source destination`*

Fichier avant création d'un lien (options -l et -i) :

```
ls -li /home/benoit/Bureau/vacances.jpg
135 -rw-r--r--  1 benoit  staff  132796 21 jul 16:08 vacances.jpg
```

Créeons un lien physique dans le répertoire /home/arnaud/

```
ln /home/benoit/Bureau/vacances.jpg /home/arnaud/
```

Le fichier est alors présent dans les deux répertoires :

```
ls -li /home/benoit/Bureau/vacances.jpg
135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg

ls -li /home/arnaud/vacances.jpg
135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg
```

Création d'un lien physique sur fichier régulier

Quelques observations

- Les deux fichiers sont identiques.
- En fait, il n'y a qu'un seul fichier, et deux liens physiques
- On peut même changer le nom d'un lien sans affecter l'autre !
- Un fichier, avec 2 noms différents

Visualisation du nombre de liens physiques

- sur le résultat de `ls -li` de l'exemple précédent, le chiffre après les droits est passé de 1 à 2 : c'est le nombre de liens physiques du fichier.
- On peut vérifier sur les répertoires que le nombre de liens physiques correspond aux observations précédentes, à savoir : nombre de sous-répertoires + 2

Création d'un lien physique sur fichier régulier

Quelques observations

- Les deux fichiers sont identiques.
- En fait, il n'y a qu'un seul fichier, et deux liens physiques
- On peut même changer le nom d'un lien sans affecter l'autre !
- Un fichier, avec 2 noms différents

Visualisation du nombre de liens physiques

- sur le résultat de `ls -li` de l'exemple précédent, le chiffre après les droits est passé de 1 à 2 : c'est le nombre de liens physiques du fichier.
- On peut vérifier sur les répertoires que le nombre de liens physiques correspond aux observations précédentes, à savoir : nombre de sous-répertoires + 2

Supprimer un fichier

Suppression d'un fichier ... ou d'un lien ?

- **On ne supprime pas vraiment des fichiers comme on pourrait le comprendre**
- On se contente de supprimer des liens physiques et de remettre à disposition (libérer) les blocs qui étaient utilisés

Conséquences

- Le contenu d'un fichier supprimé n'est jamais vraiment perdu juste après la suppression du fichier
- Il reste disponible jusqu'à ce que les blocs qui étaient utilisés soient réaffectés à d'autres, que leur contenu soit écrasé
- Logiciels de récupération de fichiers : scan des blocs du disque
- VS effacement en mode sécurisé : réécriture sur les blocs

Supprimer un fichier

Suppression d'un fichier ... ou d'un lien ?

- **On ne supprime pas vraiment des fichiers comme on pourrait le comprendre**
- On se contente de supprimer des liens physiques et de remettre à disposition (libérer) les blocs qui étaient utilisés

Conséquences

- Le contenu d'un fichier supprimé n'est jamais vraiment perdu juste après la suppression du fichier
- Il reste disponible jusqu'à ce que les blocs qui étaient utilisés soient réaffectés à d'autres, que leur contenu soit écrasé
- Logiciels de récupération de fichiers : scan des blocs du disque
- VS effacement en mode sécurisé : réécriture sur les blocs

Droits nécessaires à la suppression

Droits nécessaires à la suppression d'un fichier

Droits identiques à la création d'un fichier dans un répertoire !

- Navigation (+x) jusqu'au répertoire contenant le fichier à supprimer
- Ecriture (+w) sur le répertoire qui contient le fichier

Si le fichier est un répertoire, il doit être vide avant de pouvoir être supprimé ! $\rightarrow$ suppression de chacun de ses éléments.

Remarque

- Aucun droit n'est nécessaire sur le fichier lui-même !
- Droits nécessaires sur le répertoire le contenant uniquement
- Il est possible de supprimer des fichiers qui ne nous appartiennent pas, et sur lesquels l'on a aucun drooit !

Droits nécessaires à la suppression

Droits nécessaires à la suppression d'un fichier

Droits identiques à la création d'un fichier dans un répertoire !

- Navigation (+x) jusqu'au répertoire contenant le fichier à supprimer
- Ecriture (+w) sur le répertoire qui contient le fichier

Si le fichier est un répertoire, il doit être vide avant de pouvoir être supprimé ! $\rightarrow$ suppression de chacun de ses éléments.

Remarque

- Aucun droit n'est nécessaire sur le fichier lui-même !
- Droits nécessaires sur le répertoire le contenant uniquement
- Il est possible de supprimer des fichiers qui ne nous appartiennent pas, et sur lesquels l'on a aucun drooit !

Supprimer un fichier régulier

Commande originelle : `unlink`

- usage : `unlink` *`nomFichier`*
- Supprime le lien physique référencé par *`nomFichier`*

En reprenant l'exemple précédent :

```
1  $ ls -li /home/arnaud/vacances.jpg
2  135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg
3
4  $ ls -li /home/benoit/Bureau/vacances.jpg
5  135 -rw-r--r--  2 benoit  staff  132796 21 jul 16:08 vacances.jpg
6
7  $ unlink /home/benoit/Desktop/vacances.jpg
8
9  $ ls -li /home/arnaud/vacances.jpg
10 135 -rw-r--r--  1 benoit  staff  132796 21 jul 16:08 vacances.jpg
11
12 $ ls -li /home/benoit/Bureau/vacances.jpg
13 ls :  /home/benoit/Bureau/vacances.jpg: No such file or directory
```

Libération des blocs utilisés

Lors de l'appel à la commande `unlink`

- Suppression d'un lien physique
- Le nombre de liens physiques de l'inode est décrémenté dans la table d'attributs des fichiers
- Si ce nombre atteint 0 :
  - Aucun autre lien physique sur ce fichier
  - Les blocs utilisés pour ce fichier sont libérés
  - L'entrée correspondant à ce fichier est supprimée de la table d'attributs des fichiers
- Sinon : on ne fait rien de plus

Supprimer un fichier répertoire

Commande originelle : `rmdir`

- usage : `rmdir` *nomRep*
- Supprime le lien physique référencé par *nomRep*
- Le répertoire doit être vide
- Si le répertoire n'est pas vide : supprimer chacun de ses fichiers réguliers, et pour chaque sous-répertoire faire la procédure de suppression de répertoire.
- Les liens physiques sont automatiquement gérés ici

```
1  benoit$ mkdir Series
2  benoit$ mkdir Series/TheWalkingDead
3
4  benoit$ rmdir Series
5  rmdir: Series: Directory not empty
6
7  benoit$ rmdir Series/TheWalkingDead
8  benoit$ rmdir Series
```

Supprimer un fichier

La commande ultime : `rm`

- usage : `rm [-fr]` *nomRep, nomFichier*
- Supprime fichiers réguliers, répertoires, même non vides
- Commande `très` dangereuse (!)
- Pas de retour possible sans l'aide d'utilitaires spécialisés
- Tellement dangereuse que demande de confirmation si le fichier à supprimer n'a pas les droits d'écriture
- option -f (force) : enlève la demande de confirmation
- option -r (recursive) : active la récursivité pour les répertoires. Si le répertoire n'est pas vide, application de la commande dans chacun de ses sous-répertoires.

Suppression du répertoire GoT et de tout son contenu :

```
1 benoit$ rm -fr ./GoT
```

Une commande dangereuse

Contexte

l'utilisateur veut supprimer tout un album de Kyo contenu dans le sous-répertoire `MP3/Kyo/300Lesions` de son répertoire personnel.

Il voulait taper :

```
1 $ cd ~
2 $ rm -fr ./MP3/Kyo/300Lesions
```

Mais il tape :

```
1 $ cd ~
2 $ rm -fr . /MP3/Kyo/300Lesions
```

et le message d'erreur suivant apparait (?)

```
1 rm: /MP3/Kyo/300Lesions: No such file or directory
```

Bref : il vient de supprimer tout son répertoire personnel...

Une commande dangereuse

Contexte

l'utilisateur veut supprimer tout un album de Kyo contenu dans le sous-répertoire `MP3/Kyo/300Lesions` de son répertoire personnel.

Il voulait taper :

```
1 $ cd ~
2 $ rm -fr ./MP3/Kyo/300Lesions
```

Mais il tape :

```
1 $ cd ~
2 $ rm -fr . /MP3/Kyo/300Lesions
```

et le message d'erreur suivant apparait (?)

```
1 rm: /MP3/Kyo/300Lesions: No such file or directory
```

Bref : il vient de supprimer tout son répertoire personnel...

Une commande dangereuse

Contexte

l'utilisateur veut supprimer tout un album de Kyo contenu dans le sous-répertoire `MP3/Kyo/300Lesions` de son répertoire personnel.

Il voulait taper :

```
1 $ cd ~
2 $ rm -fr ./MP3/Kyo/300Lesions
```

Mais il tape :

```
1 $ cd ~
2 $ rm -fr . /MP3/Kyo/300Lesions
```

et le message d'erreur suivant apparait (?)

```
1 rm: /MP3/Kyo/300Lesions: No such file or directory
```

Bref : il vient de supprimer tout son répertoire personnel...

Une commande dangereuse

Contexte

l'utilisateur veut supprimer tout un album de Kyo contenu dans le sous-répertoire `MP3/Kyo/300Lesions` de son répertoire personnel.

Il voulait taper :

```
1 $ cd ~
2 $ rm -fr ./MP3/Kyo/300Lesions
```

Mais il tape :

```
1 $ cd ~
2 $ rm -fr . /MP3/Kyo/300Lesions
```

et le message d'erreur suivant apparait (?)

```
1 rm: /MP3/Kyo/300Lesions: No such file or directory
```

Bref : il vient de supprimer tout son répertoire personnel...

Déplacer un fichier

Déplacer un fichier d'un répertoire source vers un répertoire destination : finalement très facile

Revient à créer un nouveau lien physique vers le répertoire de destination, et à supprimer le lien physique du répertoire source

Droits nécessaires au déplacement

Pour la création du nouveau lien physique

- Navigation (+x) vers le répertoire de destination
- Ecriture (+w) sur le répertoire de destination

Pour la suppression du lien physique de départ

- Navigation (+x) vers le répertoire source
- Ecriture (+w) sur le répertoire source

Concept identique que le fichier à déplacer soit régulier ou répertoire

Déplacer un fichier / répertoire

La commande `mv`

- usage : `mv` *`source, destination`*
- Déplace le fichier / répertoire *source* dans le répertoire *destination* qui doit déjà exister
- *destination* peut également un nom de fichier régulier, auquel cas le fichier *source* sera renommé

Déplacement du fichier vacances.jpg à la racine du répertoire personnel

```
1 benoit$ mv ./Bureau/vacances.jpg ./
```

Puis renommage du fichier vacances.jpg en photo.jpg

```
1 benoit$ mv vacances.jpg photo.jpg
```

Finalement on remet tout comme avant :

```
1 benoit$ mv photo.jpg ./Bureau/vacances.jpg
```

Déplacer un fichier / répertoire

La commande `mv`

- usage : `mv` *`source, destination`*
- Déplace le fichier / répertoire *source* dans le répertoire *destination* qui doit déjà exister
- *destination* peut également un nom de fichier régulier, auquel cas le fichier *source* sera renommé

Déplacement du fichier vacances.jpg à la racine du répertoire personnel

```
1 benoit$ mv ./Bureau/vacances.jpg ./
```

Puis renommage du fichier vacances.jpg en photo.jpg

```
1 benoit$ mv vacances.jpg photo.jpg
```

Finalement on remet tout comme avant :

```
1 benoit$ mv photo.jpg ./Bureau/vacances.jpg
```

Déplacer un fichier / répertoire

La commande `mv`

- usage : `mv` *`source, destination`*
- Déplace le fichier / répertoire *source* dans le répertoire *destination* qui doit déjà exister
- *destination* peut également un nom de fichier régulier, auquel cas le fichier *source* sera renommé

Déplacement du fichier vacances.jpg à la racine du répertoire personnel

```
1 benoit$ mv ./Bureau/vacances.jpg ./
```

Puis renommage du fichier vacances.jpg en photo.jpg

```
1 benoit$ mv vacances.jpg photo.jpg
```

Finalement on remet tout comme avant :

```
1 benoit$ mv photo.jpg ./Bureau/vacances.jpg
```

Déplacer un fichier / répertoire

La commande `mv`

- usage : `mv` *`source, destination`*
- Déplace le fichier / répertoire *source* dans le répertoire *destination* qui doit déjà exister
- *destination* peut également un nom de fichier régulier, auquel cas le fichier *source* sera renommé

Déplacement du fichier vacances.jpg à la racine du répertoire personnel

```
1 benoit$ mv ./Bureau/vacances.jpg ./
```

Puis renommage du fichier vacances.jpg en photo.jpg

```
1 benoit$ mv vacances.jpg photo.jpg
```

Finalement on remet tout comme avant :

```
1 benoit$ mv photo.jpg ./Bureau/vacances.jpg
```

Fichiers, utilisateurs multiples, et configurations originales

Change le propriétaire d'un fichier

Qui peut changer le propriétaire d'un fichier ?

- Un administrateur peut changer le propriétaire et le groupe associé a un fichier
- Utilisateur appartenant aux groupes A, B et C peut changer le groupe d'un fichier sii le fichier lui appartient. Il est capable de mettre A, B ou C.

Commandes

- Permet de changer le propriétaire ou le groupe associé à un ou plusieurs fichiers
- `chown [-R] proprio[:groupe] fichier fichier2`
- `chgrp [-R] groupe fichier fichier2 ...`
- option -R pour aciver la récursivité pour les répertoires

Propriétaire à la création d'un fichier

Rappel

- Lorsqu'un utilisateur crée un nouveau fichier, ce fichier prend comme propriétaire l'identité de son créateur
- La création de ce fichier requiert la navigation (+x) vers le répertoire qui va contenir ce fichier, ainsi que l'accès en écriture (+w) sur ce répertoire. Et rien d'autre
- Il est donc possible de créer des fichiers et répertoires dans des répertoires dont on n'est pas le propriétaire
- Pour la suppression d'un fichier : droits identiques

Ceci peut emmener à des situations originales ...

On peut créer des situations originales

- Suppression de fichiers dont n'on est pas propriétaire et sur lesquels on a aucun droit.
- Création de sous-répertoires dans le répertoire d'un utilisateur, que l'utilisateur ne pourra pas supprimer

Utilisateur Arnaud :

```
1 Galactica:~ arnaud$ mkdir repPublic
2 Galactica:~ arnaud$ chmod 777 repPublic
```

Utilisateur Benoit :

```
1 Galactica:~ arnaud$ mkdir repPublic/MonRepPrive
2 Galactica:~ arnaud$ chmod 000  repPublic/MonRepPrive
```

Utilisateur Arnaud :

```
1 Galactica:~ arnaud$ rm -fr repPublic
2 rm repPublic: Permission denied
```

repPublic doit être vide $\Rightarrow$ il faut supprimer MonRepPrive $\Rightarrow$ Mais impossible de naviguer et de lister son contenu.

Utilisation du setUID / setGID sur les répertoires

setIUD et setGID sur des repertoires

- Appliqué sur un répertoire, le setUID / setGID permet de modifier le propriétaire / groupe par défaut à la création de tout fichier dans ce répertoire
- Permet d'imposer que le propriétaire / groupe du nouveau fichier créé soit le même que celui du répertoire dans lequel le fichier est créé

activer setUID et setGID : commande chmod

- setUID : `chmod u+s` *`repertoire`*
- setGID : `chmod g+s` *`repertoire`*
- En notation octale : ajout d'une quatrième valeur en début de droits : (4) pour setUID, (2) pour setGID

Propriétaire et exécutant d'un programme

Notion de propriétaire et exécutant d'un programme

- Un programme est un fichier avec les droits d'exécution
- Il possède donc un propriétaire et un groupe
- Quand un utilisateur exécute un programme, il crée un processus associé à ce programme

Remarque

Les droits ne sont (généralement) pas hérités du programme vers le processus ! Le processus lancé possède les mêmes droits que l'utilisateur qui est à l'origine de sa création, et pas ceux du propriétaire du programme

- Raisons évidentes de sécurité
- Autrement, dérives possibles

Propriétaire et exécutant d'un programme

Notion de propriétaire et exécutant d'un programme

- Un programme est un fichier avec les droits d'exécution
- Il possède donc un propriétaire et un groupe
- Quand un utilisateur exécute un programme, il crée un processus associé à ce programme

Remarque

Les droits ne sont (généralement) pas hérités du programme vers le processus ! Le processus lancé possède les mêmes droits que l'utilisateur qui est à l'origine de sa création, et pas ceux du propriétaire du programme

- Raisons évidentes de sécurité
- Autrement, dérives possibles

Utilisation du setUID sur les fichiers exécutables

setUID sur les fichiers exécutables

- Appliqué sur un fichier exécutable, le setUID permet de modifier les droits d'exécution d'un processus
- Permet d'imposer que les droits du processus soient ceux du propriétaire du programme, et pas ceux de l'utilisateur qui exécute le programme

activer setUID : commande chmod

- setUID : `chmod u+s` *`fichier`*
- En notation octale : ajout d'une quatrième valeur en début de droits : (4) pour setUID

Remarque : setGID inopérant sur les exécutables (plupart des shells)

Utilisation du setUID sur les fichiers exécutables

sUID est à la fois formidable et dangereux

- Permet pour un utilisateur / administrateur de donner ses droits à celui qui exécute un programme
- Si le programme n'a pas de faille : permet de donner l'accès à des ressources supplémentaires, tout en contrôlant que l'utilisateur ne fasse pas n'importe quoi
- Mais si le programme peut être détourné : dangereux
- Heureusement, on ne peut (en théorie) changer les droits que des fichiers qui nous appartiennent

Question :

Si le propriétaire du shell bash (fichier exécutable /bin/bash) est l'administrateur de la machine, quelle serait la conséquence d'ajouter le sUID au shell bash ?

Exemple d'attaque sur un système

À ne pas reproduire à l'ESIREM ! !

- Boot sur une clé live USB Linux
- Recherche du répertoire correspondant au volume Linux du disque dur (dans `/Volumes/(id disque)/`)
- Recherche du fichier `./bin/bash`
- Ajout du sUID avec `chmod +s`
- Rebooter normalement sur Linux
- Se logguer, lancer `/bin/bash -p`

Droits administrateurs sur la machine

Exemple d'attaque sur un système

Quelles protections contre cette attaque ?

- Empêcher le boot par clé USB (pédagogiquement difficile)
- Chiffrer les partitions
- Reinstaller périodiquement les systèmes d'exploitation : clonage des disques et réinstallation toutes les semaines

Le sticky bit (+t)

Etude de cas : le repertoire /tmp

- Répertoire /tmp : espace temporaire dans lequel tout le monde est autorisé à écrire et lire
- Le répertoire possède les droits en lecture écriture exécution pour toutes les classes d'utilisateurs
- Pourtant l'utilisateur arnaud n'arrive pas à supprimer le fichier régulier créé par l'utilisateur benoit !
- Raison : le sticky bit est activé sur /tmp

Notez la présence du (+t) :

```
1 Galactica:tmp benoit$ ls -ld /tmp
2 drwxrwxrwt  6 root  wheel  204 23 jul 18:06 /tmp
```

Le sticky bit (+t)

Intérêt du sticky bit

- Rendre un répertoire en mode "ajout seulement"
- Dit autrement : contrôler la suppression des fichiers
- Suppression d'un fichier dans un répertoire avec sticky bit : seulement le propriétaire du répertoire, l'administrateur, ou le propriétaire du fichier

activer le sticky bit : commande chmod

- `chmod +t` *`repertoire`*
- En notation octale : ajout d'une quatrième valeur en début de droits : (1) pour sticky bit

Droits sur les fichiers [update]

- Peut être vu comme ~~trois~~ quatre mot binaires de 3 chiffres :
  - un chiffre de valeur 1 représente une autorisation
  - un chiffre de valeur 0 représente une interdiction
- Chaque mot binaire peut se transformer en une valeur décimale

droits spéciaux			propriétaire			groupe			autres		
suid	gUID	sticky	r	w	x	r	w	x	r	w	x

exemple :

droits spéciaux			propriétaire			groupe			autres		
1	0	1	1	1	1	1	0	1	1	0	0
$\times 2^2$	$\times 2^1$	$\times 2^0$	$\times 2^2$	$\times 2^1$	$\times 2^0$	$\times 2^2$	$\times 2^1$	$\times 2^0$	$\times 2^2$	$\times 2^1$	$\times 2^0$
4	0	1	4	2	1	4	0	1	4	0	0
5			7			5			4		

Droits d'accès à la création

Quels sont les droits affectés aux fichiers lors de leur création ?

- Lorsqu'il n'y a pas de masque (?) les droits de création affectés aux fichiers sont les suivants :
  - Fichier régulier : 0666 en notation octale.

r	w	-	r	w	-	r	w	-

  - Répertoire : 0777 en notation octale.

r	w	x	r	w	x	r	w	x

- Aucun droit spécial (pas de sticky bit, de sUID, de gUID)
- Mais en pratique, présence d'un masque ...

Masque à la création des fichiers

Utilité du masque à la création des fichiers

- Possibilité de modifier les droits appliqués par défaut lors de la création d'un fichier, avec utilisation d'un masque
- Ne marche **que** lors de la création d'un nouveau fichier
- Pour modifier les droits des fichiers existants : cf `chmod`
- La création d'un masque ne permet **que de supprimer** des droits appliqués par défaut
- **Il n'est pas possible d'ajouter des droits par défaut**
- ex : un nouveau fichier créé **ne pourra pas** avoir les droits d'exécution, même en utilisant les masques

Masque à la création des fichiers

Créer un masque de droits : commande `umask`

- Usage : `umask` *`droits à supprimer`*
- Droits à supprimer en notation octale (identique `chmod`)
- S'applique sur les droits par défauts (0666 pour les fichiers réguliers, 0777 pour les répertoires), et supprime les droits donnés en paramètre de umask
- S'applique pour tous les nouveaux fichiers qui seront créés après exécution de la commande, pas pour ceux créés avant

Exemple d'utilisation de la commande

- `umask 0023`
- `umask 0000`

Questions ?