Institut Universitaire de Technologie Nice Côte d'Azur

Département Réseaux et Télécommunications

TP n°3

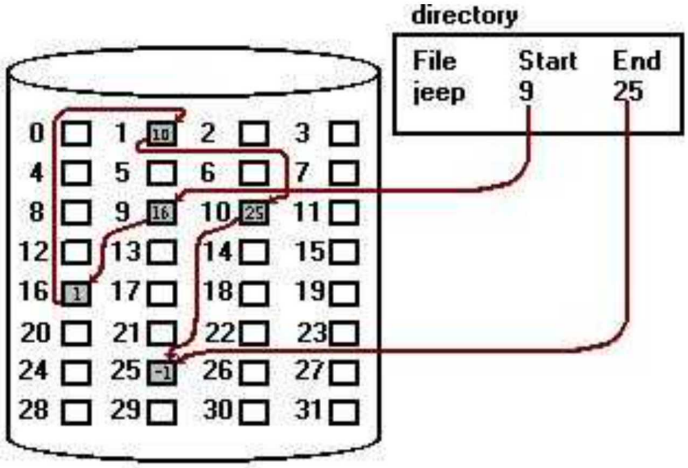
21/03/22

Noyau, Système de fichiers, droits utilisateurs

Utilisez createvm pour créer une machine Debian7 et connectez vous à la machine avec le nom d'utilisateur rt (mot de passe rt). Vous pouvez passer root dans un terminal en tapant la commande su (=superuser) et en mettant le mot de passe rt.

Vous devez rendre un rapport par binôme à la fin de chaque séance.

I. Qu'est-ce qu'un système de fichier ?



Un système de fichier doit être installer sur tout périphérique de masse (disque, clé USB). Ces périphériques stockent des blocs, typiquement de 512 octets. On rencontre aussi de plus en fréquemment des disques durs AF (Advanced Format) avec des blocs de 4096 octets . Vos fichiers correspondront à un ou plusieurs blocs disséminés sur le disque (voir figure ci-dessus).

Un système de fichier permet (citation wikipedia.fr) : « [...] l'accès au contenu des fichiers stockés (l'ouverture du fichier, son enregistrement sa copie ou son déplacement dans un second emplacement, ou sa suppression) à partir de leur chemin d'accès, formé d'un nom précédé d'une liste de répertoires imbriqués. » Le système de fichier va notamment stocker (sur le disque aussi, dans une partie protégée) la correspondance entre répertoire et fichiers et fichiers et blocs. Suivant la façon dont cela est fait, on va avoir différents types de systèmes de fichiers, par exemple, sous linux, ext2, ext3 ou ntfs (pour être compatible avec Microsoft dont c'est le système par défaut).

Vous pouvez visualiser l'arbre des fichiers/répertoires avec la commande tree qu'il faut au préalable installer avec la commande (il faut être root!) apt-get install tree

Placez vous dans votre répertoire de base puis visualisez l'arbre local. Reportez dans votre rapport ce que vous avez trouvé avec la commande tree puis tree -a.

Faites de mêmes dans le répertoire père de votre répertoire courant (quelle commande avez-vous tapée pour vous déplacez?)

Placez vous maintenant à la racine de l'arbre global (cd /). Faites un listing complet avec la commande

`tree`

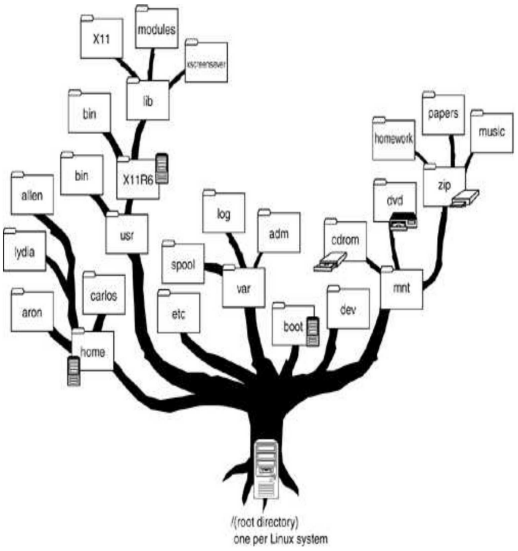
Combien y-a-t-il de fichiers et répertoires ? (tree le donne à la fin)

On va compter le nombre de répertoires à chaque niveau de l'arbre avec la commande

`tree -d -L n` où n vaut 1,2,3, etc et détermine la valeur où l'on coupe l'arborescence.

Reportez les valeurs aux niveaux 1,2,3,4 et 5. Quelle est la fraction du total des répertoires obtenus au niveau 5?

Nous allons examiner dans la suite de ce TP différents systèmes de fichiers qui sont présents sur votre machine. Une caractéristique importante de Linux est que si il y a plusieurs systèmes de fichiers simultanément, par exemple votre disque local, une clef usb et un disque partagé en réseau, ils sont TOUS mis dans la même arborescence. Par exemple, l'arborescence du système de fichiers d'un CD/ROM peut être attachée sous /mnt/cdrom comme montré ci-dessous.



II. Système de fichiers local

Le système de fichiers par défaut en linux est ext4 mais il en supporte beaucoup plus. Avant de visualiser des éléments sur le système de fichiers ext2 et 3, nous allons examiner d'autres systèmes

de fichiers qui nous seront très utiles dans la suite ou qui permettront de modifier la configuration du noyau en cours de fonctionnement.

1. Les pseudo systèmes de fichiers

a) Le répertoire `/proc`

Un système de fichiers très utile est « `/proc` » qui est « monté » par défaut en linux et qui permet d'accéder facilement aux paramètres de fonctionnement du noyau tant en lecture qu'en écriture.

Cela permet de « dialoguer » avec le noyau. **Initialement, il contenait les informations sur les processus actifs, d'où son nom proc. Il a été entendu pour permettre plus d'interactions avec le noyau.** On peut par exemple dire au noyau que l'on veut que la machine devienne un routeur (si elle est connectée à plusieurs réseaux) en écrivant la valeur +1 dans le fichier /proc/sys/net/ipv4/ip_forward, avec la commande :

echo 1 > /proc/sys/net/ipv4/ip_forward).

Nous allons explorer le répertoire/proc qui nous donne plein d'informations sur les processus, les systèmes de fichiers, les périphériques, etc.

La commande `ls /proc` permet de voir toute l'arborescence des processus identifiés par pid (pid=process id, vu au TP précédent) et les variables du noyau. Tapez dans un terminal la commande

```
sleep 1000 &
```

Le '&' met la commande en tâche de fond et retourne le PID du processus (chiffre de droite). Trouvez le répertoire /proc/<PID> (remplacer <PID> par le numéro) et indiquez les fichiers qui contiennent le statut (status en anglais) ainsi que la commande associée à ce processus. Expliquez le statut (en vous souvenant du TP précédent) ?

Comment fonctionne d'après-vous la commande top vu ce que l'on vient de dévouvrir dans /proc?

Que nous indiquent les commandes suivantes (seulement les informations importantes … et compréhensibles) ?

```
cat /proc/cpuinfo
cat /proc/meminfo
```

Les systèmes de fichiers supportés par le noyau sont listés dans le fichier /proc/filesystems.

Les périphériques sont supportés par des pilotes associés au noyau (les drivers), c'est-à-dire des programmes (on parle de modules) qui s'ajoutent au noyau.

Listez les modules avec

```
lsmod
```

Ajouter les modules vfat et cifs (2 types de système de fichier) avec les commandes :

```
sudo modprobe vfat
sudo modprobe cifs
```

Montrez que ces modules ont bien été ajoutés. Comment avez-vous faits (pour vérifier)?

Retirez le module cifs avec la commande

```
rmmod cifs
```

Montrez que ce module a bien été retiré.

2. Le système de fichiers normaux locaux

Mise en œuvre de plusieurs systèmes de fichiers au moyen de `loop`.

a) Examen d'un partitionnement

Les systèmes de fichiers locaux (souvenez-vous : il y aun arbre global auquel sont attachés tous les sous-arbres) sont appelés des partitions.
Actuellement combien de partitions sont montées ? Utilisez la commande :

```
mount
```

Expliquer complètement ce que nous dit la ligne sur sda1 en utilisant le savoir acquis au premier TP et à celui-ci.

b) L'option loop

Afin de manipuler des systèmes de fichiers, sans risquer de compromettre le système sur lequel vous travaillez, nous allons utiliser le périphérique « `loop` » qui permet d'utiliser un fichier normal comme une partition (un système de fichier sur un disque externe). Il s'agit bien d'une image disque : nous pourrons manipuler n'importe quel type de système de fichier.

Tout d'abord créons un fichier de 100 Mo.

```
dd if=/dev/zero of=limage.dd count=200k
```

Le bloc de données utilisé par dd est de 512 octets et il en en crée count.
Montrons que le résultat est cohérent avec la commande

```
ls -l limage.dd
```

On formate le fichier en ext2, c'est-à-dire que l'on traite le fichier précédent comme si c'était un disque vierge sur lequel il faut installer un système de fichier.

```
Sudo mkfs.ext2 limage.dd
```

Le programme qui formate (mkfs.ext2) constate qu'il ne s'agit pas d'une partition. Il veut savoir si vous le faites exprès. Répondez « oui » !

En regardant le résultat de la commande mkfs.ext2, quelle est la taille de bloc choisi par mkfs.ext2 et quelle est le nombre de bloc (sur la ligne avec inode et blocs) ?

Maintenant on va « monter » le système de fichiers créé dans `limage.dd`. Il faut être `root`.

```
mount -v limage.dd /mnt
```

Si ça ne marche pas (car mount voit bien que c'est un fichier que vous lui donnez et non un périphérique externe), tapez :

```
mount -v -o loop limage.dd /mnt
```

Cette fois-ci c'est bon ? Que vous retourne :

```
mount
```

Placez-vous dans /mnt et demandez des informations sur la partition courante avec la commande qui indique la partition courante et sa taille :

```
df -kh .
```

Replacez-vous sur votre répertoire personnel et exécutez la même commande. Qu'obtenez-vous cette fois-ci ? Vous devez montrer que la partition courante et la taille sont différentes.

Vous pouvez maintenant copier des fichiers dans l'image disque (elle a été montée en `rw`) ou les effacer. Par exemple, placez-vous dans /mnt et créez un fichier vide qui s'appelle toto avec la commande touch

```
touch toto
```

Ne pas oublier de « démonter » le fichier.

```
umount /mnt
```

Attention, la commande précédente ne fonctionne que si vous n'êtes pas placé dans /mnt (sinon, vous êtes en train de scier la branche sur laquelle vous êtes assis).

Placez vous à nouveau dans /mnt et tapez la commande

```
df -kh .
```

Qu'obtenez-vous? Expliquez.

c) Ext2, ext3, ext4

- ***La journalisation***

Ext4 est maintenant devenu le système de fichiers par défaut en linux. En effet, il est complètement compatible avec ext2 et il est même possible de transformer une partition ext2 en ext3 à la volée sans redémarrer. C'est ce que nous allons faire en ajoutant la journalisation du système. La journalisation veut dire qu'avant chaque rafales d'écriture (on écrit toujours plusieurs blocs d'un coup), le pilote va commencer par noter quels blocs il veut modifier, comme une liste de travail. Puis il commence son travail d'écriture et à chaque fois qu'il a écrit un bloc, il le note soigneusement. Cela permet de reconstruire l'état du système simplement si l'ordinateur s'arrête de manière non désirée (plantage, coupure électrique).

Ajoutons la journalisation :

```
sudo tune2fs -j limage.dd
```

On remonte l'image disque. En quel type de système de fichier l'image est-elle montée ?

- ***Le montage public***

Le simple utilisateur ne peut pas utiliser, normalement, la commande `mount`. Pour que cela lui soit permis, il faut ajouter (en tant que root) une entrée dans le fichier `/etc/fstab` avec l'option `user`. Pour notre cs, limage.dd, il faut donc ajouter la ligne :

```
/home/rt/limage.dd  /mnt ext3 loop,user 0 0
```

Puis on tape la commande

```
mount /mnt
```

Pour tester cette option, vérifiez qu'avec elle, il est possible à l'utilisateur `rt` de monter l'`image.dd` alors que sans elle il ne le peut pas. Reportez dans votre rapport les résultats (message d'erreur notamment).

d) Et les systèmes de fichiers `iso`

On va fabriquer un système de fichiers `iso` pour graver un CD.

On débute par un format `iso` rudimentaire. Il contient l'arborescence `/etc`.

```
genisoimage -iso-level 1 -o etc.iso /etc
```

(si genisoimage non installé: apt install genisoimage)

Montons cette image

```
mount -v -t iso9660 -o loop etc.iso /mnt
ls -l /mnt/*
```

On constate que les fichiers ont un format 8.3;v : 8 caractères maximum pour le nom, 3 pour l'extension. On ne retrouve pas vraiment le contenu des répertoires du disque.

On peut ajouter les options `-J` pour avoir les noms de fichiers longs et en unicode, `-R` pour une description *RockRidge* des fichiers qui permet de gérer les droits type unix (voir ci-dessous).

III. Gestion des droits Unix : le 4ème champ

La gestion des droits en unix est assez rudimentaire. On ne distingue que le propriétaire, le groupe et les autres, pour un accès en lecture, écriture et exécution. Cela se fait avec 3 champ. Il existe un 4ème champ qui permet d'utiliser le sticky bit et l'extended id bit.

Dans le premier TP nous avons examiné les droits d'accès classiques avec la commande `chmod` en mode symbolique. Ici nous allons rappeler les résultats en mode numérique et les étendre.

Code	*Effet*
0001	Permet l'exécution par les autres
0002	Permet l'écriture par les autres
0004	Permet la lecture par les autres
0010	Permet l'exécution par les membres du groupe
0020	Permet l'écriture par les membres du groupe
0040	Permet la lecture par les membres du groupe
0100	Permet l'exécution par le propriétaire
0200	Permet l'écriture par le propriétaire
0400	Permet la lecture par le propriétaire
1000	Positionne le « `sticky bit` »
2000	Utilise le `gid` (identification du groupe) lors de l'exécution
4000	Utilise `l'uid` (identification du propriétaire) lors de l'exécution

Ces valeurs ont un effet légèrement différents si elles sont appliquées à des répertoires.
Le code complet s'écrit comme la somme de tous les bits positionnés.

a) Le sticky bit

Le `sticky bit` appliqué à un répertoire permet de le partager. Tous les fichiers créés ne peuvent être effacés que par le propriétaire et l'administrateur. C'est le régime utilisé pour le répertoire `/tmp`.

Créer un utilisateur toto avec la commande `adduser`.

On réutilise fichier précédent `limage.dd` formaté en ext3 : le monter sur `/mnt`.

Aller dans ce répertoire et en fabriquer un nouveau de nom `tmp`.

Avec `chmod` lui donner les droits 1777. Constater les modifications sur ce répertoire.

Dans un terminal supplémentaire se loger sous le nom rt, et dans un autre sous le nom toto.

Aller dans le répertoire `/mnt/tmp` et créer des fichiers sous les deux identités. Bien que le répertoire soit modifiable par tout le monde, pouvez vous effacer les fichiers qui ne vous appartiennent pas ?

Après avoir modifié les droits de `/mnt/tmp` en 777 : pouvez-vous détruire ce qui ne vous appartient pas dans ce répertoire ?

b) Le eid

On va essayer de pirater la machine.

Démontez l'image :

```
umount /mnt
```

Remontez la manuellement :

```
mount -v -o loop limage.dd /mnt
```

En tant que root ou avec un sudo, copiez le fichier `/bin/sh` dans le répertoire `/mnt`. (avec `limage.dd` montée)

Passez le fichier `/mnt/sh` en 4755

Sous le login rt, exécutez ce fichier.

Que ce passe-t-il ? Créez un fichier là où vous êtes : à qui appartient-il ?

Vous pouvez vérifier en demandant quel est votre nom d'utilisateur actuel avec la commande

```
whoami
```

Le fichier `limage.dd` aurait pu venir de n'importe où, il n'est pas contrôlé par l'administrateur : il est très dangereux de monter ce type de pseudo périphériques sans précaution.

Démontons le fichier, puis remontons le avec des options supplémentaires normales : on interdit les suid, les fichiers dev et pourquoi pas exécutables.

```
mount limage.dd -o loop,remount,nosuid,nodev /mnt
```

Si vous réexécutez le fichier `/mnt/sh` que se passe-t-il ? Que vous donne whoami ?

Attention que si l'on autorise le montage par un simple utilisateur de périphériques (clé USB) ou autre fichier venu d'ailleurs sans interdire le suid, vous avez constaté qu'il est très aisé de passer administrateur.