

Cours des Systèmes d'Exploitation LINUX

ENSAM – Casablanca
2022-2023

Chapitre 3

Gestion des utilisateurs et groupes sous Unix

Introduction

Linux étant multiutilisateur, les personnes doivent être identifiées afin d'assurer la confidentialité. Chaque personne doit avoir un compte utilisateur pour pouvoir accéder au système.

Pour identifier tous ces utilisateurs au niveau du système, un numéro unique leur sera attribué : le **UID** (**Eng** : *User's ID*). Le propriétaire d'un fichier est déterminé par ce numéro. Ces utilisateurs seront aussi dotés d'un nom d'utilisateur unique (*login*) et d'un mot de passe (*password*) pour qu'ils puissent s'authentifier lors de leur connexion au système.

Types de comptes

Les comptes utilisateur ne sont pas tous égaux sur Linux. On distingue trois types :

- ❑ **Super-utilisateur (root)**
- ❑ **Comptes systèmes**
- ❑ **Comptes ordinaires**

Super-utilisateur (root)

Super-utilisateur (root): c'est l'utilisateur le plus important du système du point de vue de l'administration. Il n'est pas concerné par les droits d'accès aux fichiers. Son **UID** égal à **0** (zéro) lui confère sa spécificité. Ce super-utilisateur aura donc à sa charge les tâches d'administration du système.

Comptes systèmes

On trouve sur le système toute une série de comptes qui ne sont pas affectés à des personnes (*bin, daemon, sync, apache...*). Ceux-ci servent à faciliter la gestion des droits d'accès de certaines application et démons. Ainsi en lançant le serveur Web sous l'identité du compte "apache", on pourra aisément limiter ses droits d'accès à certains fichiers. Les **UID** compris entre **1** et **999** sont généralement utilisés pour ces comptes

D'une façon générale, on ne lance jamais un service exposé aux attaques réseau comme un **serveur Web (Apache)** sous l'identité de **root**. Sinon, quelqu'un de mal intentionné qui réussirait à exploiter une faille de sécurité du logiciel obtiendrait automatiquement les droits d'administration. Toujours pour des raisons de sécurité, on fera on sorte que personne ne puisse se connecter à la machine à partir de l'un de ces comptes.

Comptes ordinaires

Tous les autres comptes utilisateur sont associés à des personnes; leur vocation est de permettre à des utilisateurs standard de se connecter. L'UID d'un utilisateur sera un nombre supérieur ou égal à 1000.

Administration du système

Nous allons nous intéresser aux commandes nécessaires pour :

- ❑ Créer des comptes utilisateurs
- ❑ De les affectées à des groupes
- ❑ Puis de définir les droits sur les fichiers

Ceci étant un travail d'administration du système, un utilisateur ordinaire ne peut pas accéder à ces droits d'administration.

Pour cela, nous aurons souvent à utiliser la commande **sudo** (**Eng** - *Substitute User DO*) qui permet à un utilisateur d'exécuter des commandes qui ne peuvent être utilisées que par le super-utilisateur. Elle s'utilise comme suit :

\$ sudo commande

Le mot de passe de votre compte est alors demandé afin que le système vérifie votre identité

Administration du système

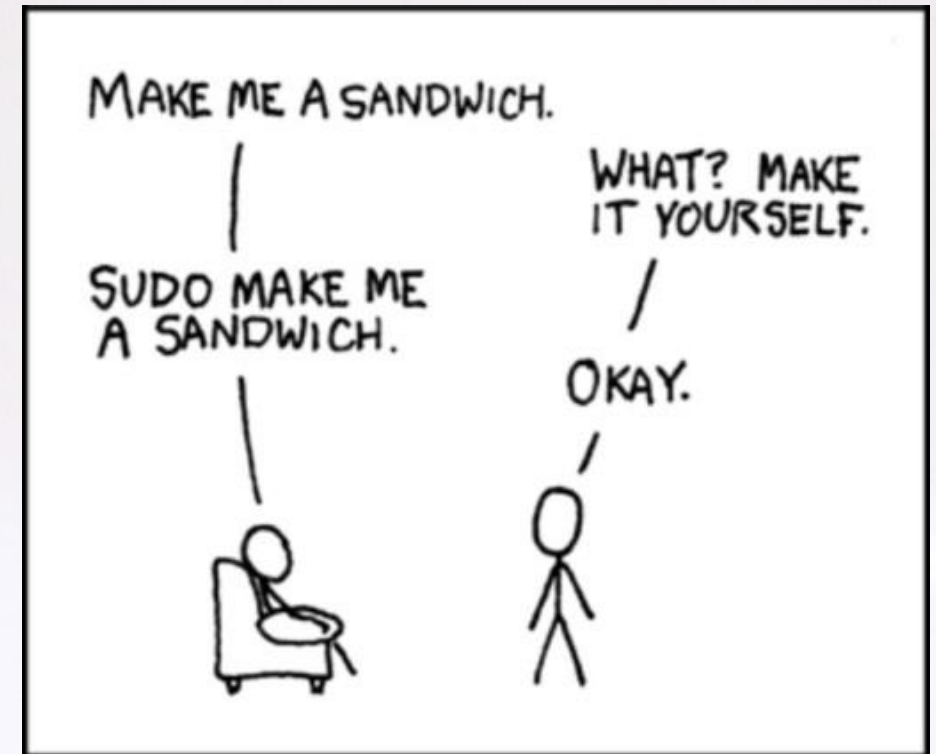
Ceci dit : On pourrait être tenté d'utiliser le super utilisateur (root) comme une session, afin de ne pas avoir à changer d'utilisateur lorsqu'il faut reconfigurer le système.

C'est possible (et nous verrons comment...) Mais

Il ne faut pas faire cela !!!

En effet, avec le super utilisateur, une mauvaise manipulation peut causer la perte irréversible de tout ou partie de vos données ou rendre la machine inutilisable !

Mais avec la substitution d'utilisateur par **sudo** c'est plutôt une manière de vous dire et de vous rappeler : **Est ce que vous êtes sûr que vous voulez faire ceci?!**



Administration du système

Ouvrir un terminal en mode root

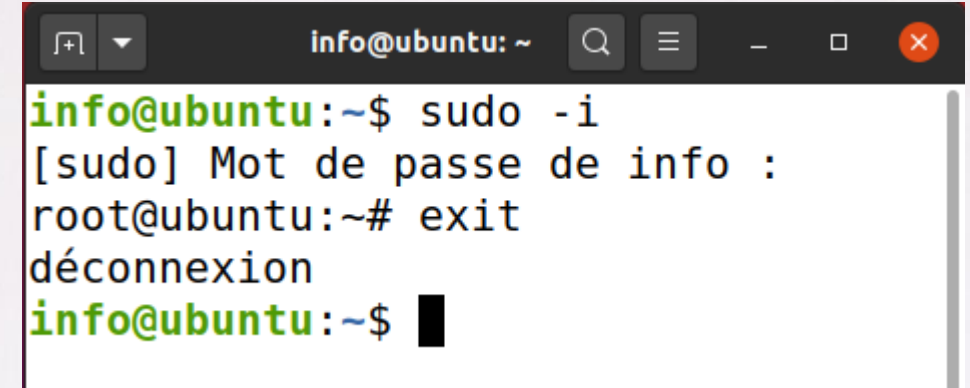
Utiliser **sudo** pour exécuter une seule commande ne cause pas un réel désagrément, mais il peut être désagréable de l'utiliser pour exécuter une longue procédure nécessitant plusieurs interventions en mode super-utilisateur (**root**). L'ouverture d'un terminal en mode **root** permet d'éviter d'avoir à appeler **sudo** à chacune des étapes de cette procédure, sans avoir à activer l'accès au compte d'utilisateur **root**. L'inconvénient de cette méthode est qu'aucune trace des actions posées n'est inscrite dans le journal de **sudo** (sinon l'ouverture du terminal **root** lui-même). Il est déconseillé d'ouvrir un terminal **root**.

Administration du système

Ouvrir un terminal en mode root

Pour vous servir d'un terminal root :

- Ouvrez une fenêtre de terminal ;
- Saisissez la commande suivante :
utilisateur@ordinateur:~\$ sudo -i
- Saisissez votre mot de passe à l'invite de saisie de mot de passe ;
- Exécutez votre série de commandes d'administration ;
- Fermez la session root :
root@ordinateur:~# exit ou Ctrl+D



```
info@ubuntu: ~  
info@ubuntu:~$ sudo -i  
[sudo] Mot de passe de info :  
root@ubuntu:~# exit  
déconnexion  
info@ubuntu:~$
```


Création d'un compte utilisateur

La commande **useradd** permet d'ajouter un nouvel utilisateur sans spécifier aucune information sur ce compte (UID, mot de passe,...)

Syntaxe :

useradd [options] login

Le login définis le nom du compte à créer.

Exemple d'un simple utilisation :



```
info@ubuntu: ~  
info@ubuntu:~$ sudo useradd Ensamiste  
[sudo] Mot de passe de info :  
info@ubuntu:~$
```

En principe tout c'est bien passé, nous avons créé un compte
Mais malheureusement nous avons oublié exprès de le configurer

Le fichier de gestion des utilisateurs

Pour comprendre la configuration d'un compte utilisateur, nous allons nous intéresser à la façon dont Unix gère ces comptes. Or, tout est fichier en Unix.

Le premier fichier que nous allons voir, est le fichier `/etc/passwd` qui contient les informations relatives aux comptes (login, interpréteur de commande, ...). C'est ce fichier que le système consulte lorsque vous vous connectez à votre compte en tapant votre identifiant et mot de passe. Si ce que vous avez tapé n'existe pas dans ce fichier alors vous ne pourrez pas vous connecter. Ce fichier contient des champs de texte séparés par **:** et qui respecte le format suivant:

```
nom_du_compte : mot_de_passe : numero_utilisateur :  
numero_de_groupe : commentaire : répertoire :  
programme_de_demarrage
```

Le fichier de gestion des utilisateurs

La signification des champs :

- ❑ **nom du compte** = identifiant de l'utilisateur
- ❑ **mot de passe** = mot de passe de l'utilisateur. Celui-ci est codé, il est inutile de l'éditer.
- ❑ **numéro utilisateur** = le UID. Cet identifiant est unique. Les valeurs supérieures à 1000 sont pour les comptes utilisateurs (par conséquent il sera visible dans la table d'utilisateurs).
- ❑ **numéro de groupe** = un entier qui identifie le groupe de l'utilisateur. C'est un identifiant unique appelé GID (Groupe Identifier).
- ❑ **commentaire** = des informations sur l'utilisateur.
- ❑ **répertoire** = le répertoire dans lequel se retrouve l'utilisateur après s'être connecté (son homedir).
- ❑ **commande** = le shell par défaut qui sera associé à ce compte.

Le fichier de gestion des utilisateurs

Ainsi, si nous explorons le contenu de ce fichier, nous trouverons pas mal d'informations sur les différents comptes.

Par exemple la première et les deux dernières lignes:

```
root:x:0:0:root:/root:/bin/bash
```

...

```
info:x:1000:1000:linux2020,,,:/home/smi:/bin/bash
```

```
Ensamiste:x:1001:1001::/home/Ensamiste:/bin/sh
```

Déjà nous remarquons les informations qui manquent au compte nouvellement créé **Ensamiste**.

Et si nous explorant le répertoire /home nous ne trouverons pas le dossier personnel du compte **Ensamiste**.

Options de la commande d'ajout d'utilisateur

Création d'un compte avec les options de configuration :

```
# useradd -u 2040 -g sigma -G UH2M -c "compte  
configurer" -e 2021-07-01 -s /bin/bash -d  
/home/Ensamiste2 Ensamiste2
```

-u : pour spécifier manuellement le UID du compte.

-g : pour spécifier le groupe par défaut.

-G : pour spécifier les groupes secondaires

-c : pour affecter un commentaire (nom exacte, adresse Email...)

-e : pour spécifier une date d'expiration de ce compte (à partir de laquelle, le compte ne sera plus accessible)

-s : pour indiquer le shell par défaut

-d : pour spécifier le répertoire personnel

-m : pour que le dossier personnel soit créé

Modifier les propriétés d'un compte `-usermod`

Pour modifier les propriétés d'un compte déjà créé, on peut utiliser la commande **usermod** dont le fonctionnement est très similaire à *useradd*.

Nous effectuons quelques modifications sur le premier compte **Ensamiste** :

```
sudo usermod -c "c'est le compte de Ensamiste" -s /bin/bash Ensamiste
```

Si on visualise encore une fois le fichier **/etc/passwd**, que effectivement la définition du compte **Ensamiste** a changer:

```
smil:x:1001:1001:c'est le compte de smil :/home/Ensamiste : /bin/bash
```



Changer les options par défaut –useradd

Le second fichier que nous allons étudier est le fichier qui permet de changer les options par défaut de la commande **useradd**.

Nous pouvons faire cela en modifiant directement le fichier **/etc/default/useradd** ou bien en utilisant la commande **useradd** avec l'option **-D**.

Pour afficher les options par défaut :
useradd -D

Changement des options par défaut :

- ☐ pour changer le répertoire home
 - **useradd -D -b /home_2**
- ☐ pour changer le groupe par défaut
 - **useradd -D -g dev**
- ☐ pour changer le shell par défaut
 - **useradd -D -s /bin/csh**



```
info@ubuntu:~$ useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/sh
SKEL=/etc/skel
CREATE_MAIL_SPOOL=no
info@ubuntu:~$
```

Le fichier des propriétés cachés –shadow

Le 3^{ème} fichier concernant toujours l'administration des utilisateurs Unix, est : **/etc/shadow**. C'est un fichier sensible (sudo) qui contient les mots de passe et d'autres informations sur les comptes utilisateurs. Essayant de détailler son contenu :

```
root:!:16719:0:99999:7:::
```

```
daemon*:15630:0:99999:7:::
```

...

```
info:$1$Pv$Omhf8yD/Pg3sViEtYQtHj/:16719:0:99999:7:::
```

```
Ensamiste:!:18546:0:99999:7:::
```



Le fichier des propriétés cachés –shadow

Comme avec le fichier *passwd*, chaque champ dans le fichier **shadow** est aussi séparé par deux points “:”, et on trouve les 9 champs suivants:

1. Nom d'utilisateur, il doit être présent dans le fichier */etc/passwd*.
2. Mot de passe crypté de 13 caractères.
 - Une entrée nulle (::) indique qu'un mot de passe n'est pas demandé pour entrer dans le système.
 - Une entrée (:!) indique que le mot de passe n'a jamais été initialiser, en conséquence le compte n'est pas encore activé.
 - Une entrée (:*) indique que le compte n'aura pas de mot de passe et il n'y aura pas la possibilité d'y accéder.
3. Le nombre de jours depuis le 1er Janvier 1970 jusqu'au jour dus dernier changement du mot de passe.

Le fichier des propriétés cachés –shadow

4. Le nombre de jours avant que le mot de passe ne puisse être changé (un 0 indique qu'il peut être changé à n'importe quel moment).
5. Le nombre de jours après lesquels le mot de passe *doit* être changé (99999 indique que l'utilisateur peut garder son mot de passe inchangé pendant beaucoup, beaucoup d'années)
6. Le nombre de jours pour avertir l'utilisateur qu'un mot de passe ne va plus être valable (7 pour une semaine entière)
7. Le nombre de jours avant de désactiver le compte après expiration du mot de passe
8. Le nombre de jours depuis le 1er Janvier 1970 pendant lesquels un compte a été désactivé
9. Un champ réservé pour une utilisation future possible

Changer de mot de passe utilisateur -passwd

La commande **passwd** permet de changer le mot de passe d'un utilisateur. L'utilisateur peut changer son mot de passe personnel. Alors que, seul l'administrateur peut changer le mot de passe d'un autre.

```
info@ubuntu: ~  
info@ubuntu:~$ sudo passwd Ensamiste  
[sudo] Mot de passe de info :  
Nouveau mot de passe :  
Retapez le nouveau mot de passe :  
passwd : le mot de passe a été mis à jour avec succès  
info@ubuntu:~$
```



Revoyant l'effet de cette modification dans le fichier **/etc/shadow** :

```
Ensamiste:$6$YfrGPY$SW$soeegfaoVcd1/AeGswS8Q31YsyN5DBghqxA1/wH  
3Oe1LXr7ZX6M90bWINvvsP15YM1oa09A9zO7PCLMgRb92E/:16734:0:99999  
:7:::
```

À partir de là le compte **Ensamiste** est **Activé**

La suppression d'un utilisateur –userdel

La commande qui permet de supprimer un utilisateur :

userdel Ensamiste

Si on désire aussi supprimer son dossier personnel :

userdel -r Ensamiste

La commande précédente supprime l'utilisateur **Ensamiste** ainsi que son répertoire personnel, cependant un problème demeure: les fichiers appartenant à smi1 et qui se trouvent en dehors du répertoire personnel ne sont pas supprimés. La commande suivante permet de les trouver et de les supprimer à partir du **UID** de l'utilisateur (on suppose que le **UID** de l'exemple **Ensamiste** est **1001**)

```
find / -type f -uid 1001 -print -exec rm {} \;
```

NB: les détails sur le fonctionnement de la commande find seront donnés ultérieurement



Les groupes : Définitions

Un groupe est un ensemble d'utilisateurs pouvant partager des fichiers et des ressources système. Par exemple, des utilisateurs qui travaillent sur le même projet peuvent former une équipe. Une tel équipe est traditionnellement connu comme **groupe UNIX**.

Chaque groupe doit disposer d'un nom, d'un ID (le **GID**) et d'une liste des noms d'utilisateur appartenant au groupe. Un GID identifie le groupe en interne sur le système.

Les groupes : Types

Les deux types de groupes auxquels un utilisateur peut appartenir sont les suivants :

- ❑ **Groupe principal** : groupe assigné par le système d'exploitation aux fichiers créés par l'utilisateur. Chaque utilisateur doit appartenir à un groupe principal (par défaut le nom de l'utilisateur est aussi le nom de son groupe principal lors de sa création).
- ❑ **Groupes secondaires** : groupes auxquels un utilisateur peut appartenir. Les utilisateurs peuvent appartenir à un nombre maximal de 15 groupes secondaires.

Pour cela, il existe un fichier qui comporte les noms des groupes existants dans votre système.

Les groupes : Fichiers

Le 4^{ème} fichier à examiner est le fichier qui gère les groupes : **/etc/group**. En visualisant son contenu, nous pouvons distinguer le nom et le GID des groupes présents sur notre système. En particulier :

```
sudo:x:27:info
```

```
info:x:1000:
```

```
Ensamiste:x:1001:
```

Remarquant (à partir du fichier **passwd**) que le compte utilisateur principale (ici : **info**) a pour groupe principal : **info GID-1000**. Cependant, il appartient à d'autres groupes secondaire comme par exemple **sudo**, et par conséquent il détient les droits d'administration. Nous pouvons changer cela par l'ajout de l'utilisateur **Ensamiste** au groupe **sudo** (**sudo:x:27:info,Ensamiste**). Cependant, il est toujours **préférable** de changer les groupes secondaires en utilisant la commande :

```
sudo usermod -a -G sudo Ensamiste
```

Les groupes : Fichiers

D'autres fichiers sont liés aux opérations d'administrations des utilisateurs et des groupes (comme `/etc/gshadow`, `./etc/sudoers...`)

Il possible d'avoir plus d'informations sur ces fichiers de configuration on utilisant **man** section 5 du fichier.

Fichier	Description	Plus d'informations
<code>/etc/passwd</code>	Information sur les comptes utilisateurs	<code>man 5 passwd</code>
<code>/etc/shadow</code>	information cachée sur les comptes utilisateurs	<code>man 5 shadow</code>
<code>/etc/group</code>	Défini les groupes auxquels les utilisateurs appartiennent	<code>man 5 group</code>
<code>/etc/gshadow</code>	Information cachée sur les groupes	<code>man 5 gshadow</code>
<code>/etc/sudoers</code>	Liste de qui peut lancer quoi avec <code>sudo</code>	<code>man 5 sudoers</code>

Les groupes : Commandes

- ❑ Pour connaître les groupes d'un utilisateur à l'aide de l'une des commandes :

groups nom_utilisateur ou
id nom_utilisateur

- ❑ Pour créer un nouveau groupe :

groupadd nom_groupe

- ❑ Pour ajouter un utilisateur à un groupe :

gpasswd -a nom_utilisateur nom_groupe

- ❑ Il est possible de rajouter plusieurs groupes à un utilisateur:

usermod -aG grp1,grp2 nom_utilisateur

- ❑ Pour effacer un groupe :

groupdel nom_groupe



Les groupes : Commandes

```
info@ubuntu: ~  
info@ubuntu:~$ groups info  
info : info adm cdrom sudo dip plugdev lpadmin lxd sambashare  
info@ubuntu:~$ id info  
uid=1000(info) gid=1000(info) groupes=1000(info),4(adm),24(cdrom),27(sudo),  
,30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)  
info@ubuntu:~$ sudo groupadd TP  
info@ubuntu:~$ sudo gpasswd -a info TP  
Ajout de l'utilisateur info au groupe TP  
info@ubuntu:~$ id info  
uid=1000(info) gid=1000(info) groupes=1000(info),4(adm),24(cdrom),27(sudo),  
,30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),1002(TP)  
info@ubuntu:~$ sudo groupdel TP  
info@ubuntu:~$ id info  
uid=1000(info) gid=1000(info) groupes=1000(info),4(adm),24(cdrom),27(sudo),  
,30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)  
info@ubuntu:~$
```

Les droits d'accès

Linux possède des mécanismes permettant au propriétaire d'un fichier d'en protéger le contenu, mais aussi de faciliter le partage et le travail en groupes. C'est le but principal de la gestion des **droits d'accès** (**Eng** - **permissions**) à un fichier (ou un répertoire) qui sont appliqués par son propriétaire.

Les droits d'accès d'un fichier sont la première ligne de défense dans la sécurité d'un système Unix. Les modes de base des droits d'accès Unix sont : lire, écrire et exécuter, comme il est décrit dans le tableau suivant :

Les droits d'accès – Types

Droits d'accès	Sur les répertoires	Sur les fichiers
Lire (read) (r)	Autorisation de voir le contenu d'un répertoire ou les sous-répertoires.	Autorisation de voir le contenu du fichier.
Écrie (write) (w)	Autorisation de créer, modifier, supprimer les fichiers ou les sous-répertoires.	Autorisation aux entités d'ajouter, de modifier, de supprimer le contenu d'un fichier.
Executer (execute)(x)	Autorisation d'accéder au répertoire	Permettre d'exécuter le fichier
(-)	Pas d'autorisation	Pas d'autorisation

Les droits d'accès – Symbolisation

Dans l'exemple `mon_fichier.txt`, les droits sont affichés de la sorte :

- ❑ Les 3 premiers caractères sont les droits du propriétaire du fichier (*user Unix u*):
`rw-`
- ❑ Les trois suivants sont les droits des utilisateurs qui appartiennent au groupe (*group g*):
`rw-`
- ❑ Les trois derniers sont les droits des autres utilisateurs (*others o*) :
`r--`

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ ls -l
total 4
-rw-rw-r-- 1 info info    0 oct.  12 01:02 mon_fichier.txt
drwxrwxr-x 2 info info 4096 oct.  12 01:02 mon_repertoire
info@ubuntu:~/Documents$
```


Changement des droits d'accès –chmod

Pour modifier les droits d'accès d'un fichier ou d'un répertoire, nous utilisons la commande **chmod** (**Eng** - *change mode*).

Il existe deux façons d'utiliser **chmod**:

- ❑ mode symbolique
- ❑ mode absolue

Les droits d'accès appliqués avec **chmod** en mode symbolique utilisation le familier rwx format et sont plus faciles à comprendre par la plupart des nouveaux utilisateurs.

Les droits d'accès appliqués avec **chmod** en mode absolu nécessite une représentation numérique de ces droits, ce qui est considéré comme plus efficace du point de vue système.

Changement des droits d'accès –chmod

❑ chmod en mode symbolique

Opérateur chmod	Signification	Exemple	Résultat
+	Ajouter les droits désignés à un fichier ou répertoire	chmod o+wx mon_fichier.txt	Ajout des droits de modification et d'exécution au autres utilisateurs
-	Supprimer les droits désignés à un fichier ou répertoire	chmod u-x mon_fichier.txt	Supprime le droit d'exécuter ce fichier pour le propriétaire
=	Attribuer exactement ces droits	chmod g=r-x mon_fichier.txt	Donne exactement les droits de lire et d'exécuter pour les utilisateurs du groupe

Changement des droits d'accès –chmod

❑ chmod en mode symbolique

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ ls -l mon_fichier.txt
-rw-rw-r-- 1 info info 0 oct. 12 01:02 mon_fichier.txt
info@ubuntu:~/Documents$ chmod o+wx mon_fichier.txt
info@ubuntu:~/Documents$ ls -l mon_fichier.txt
-rw-rw-rwx 1 info info 0 oct. 12 01:02 mon_fichier.txt
info@ubuntu:~/Documents$ chmod u-x mon_fichier.txt
info@ubuntu:~/Documents$ ls -l mon_fichier.txt
-rw-rw-rwx 1 info info 0 oct. 12 01:02 mon_fichier.txt
info@ubuntu:~/Documents$ chmod g=rx mon_fichier.txt
info@ubuntu:~/Documents$ ls -l mon_fichier.txt
-rw-r-xrwx 1 info info 0 oct. 12 01:02 mon_fichier.txt
info@ubuntu:~/Documents$
```

NB : il est possible de regrouper toutes ces modifications en une seule commande :
chmod o+wx,u-x,g=rx mon_fichier.txt

Changement des droits d'accès –chmod

❑ chmod en mode absolu

La seconde façon d'attribuer les droits d'accès par chmod, consiste à utiliser des nombres pour chaque ensemble de droits. Nous pouvons calculer ces nombres si on retient la manière simple de transformer un nombre en base binaire vers une base décimale. Ainsi si nous avons ces règles simples à retenir:

- ❑ (r, w, ou x) est représenté par 1. (-) est représenté par 0
- ❑ Nous supposant que les 3 bits obtenus sont en binaire, puis nous calculons le nombre équivalent en décimale.

Exemples :

--- → 000 → 0

r-x → 101 → 5

rwX → 111 → 7

Changement des droits d'accès -chmod

❑ chmod en mode absolu

Par exemple si nous voulons que les droits d'un certain fichier deviennent **rwxr-xr--** :

rwx r-x r-- → **111 101 100** → **7 5 4 = 754**

Ainsi nous utilisons chmod en mode absolu comme suivant :

chmod 754 mon_fichier.txt

Exemples :

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ chmod 000 mon_fichier.txt
info@ubuntu:~/Documents$ ls -l mon_fichier.txt
----- 1 info info 0 oct. 12 01:02 mon_fichier.txt
info@ubuntu:~/Documents$ chmod 777 mon_fichier.txt
info@ubuntu:~/Documents$ ls -l mon_fichier.txt
-rwxrwxrwx 1 info info 0 oct. 12 01:02 mon_fichier.txt
info@ubuntu:~/Documents$ chmod 754 mon_fichier.txt
info@ubuntu:~/Documents$ ls -l mon_fichier.txt
-rwxr-xr-- 1 info info 0 oct. 12 01:02 mon_fichier.txt
info@ubuntu:~/Documents$
```

Droits attribués automatiquement à un fichier

Lorsqu'un nouveau fichier est créé, celui-ci obtient automatiquement certains droits. Ces derniers sont définis par défaut dans le fichier de paramétrage de la session : `/etc/pam.d/common-session`.

Il est possible de définir les droits aux fichiers et aux répertoires lors de leur création. Nous utilisons la notion de masque à travers la commande **umask** (**Eng** - *user file creation mode mask*, masque de création de fichier par l'utilisateur).

Droits attribués automatiquement à un fichier `-umask`

Premièrement pour savoir quel masque est utilisé par défaut, nous tapons :

`umask`

Si on veut afficher les droits par défaut en mode symbolique :

`umask -S`

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ umask
0002
info@ubuntu:~/Documents$ umask -S
u=rwx,g=rwx,o=rx
info@ubuntu:~/Documents$ touch f0; mkdir d0
info@ubuntu:~/Documents$ ls -l
total 8
drwxrwxr-x 2 info info 4096 oct. 12 01:10 d0
-rw-rw-r-- 1 info info 0 oct. 12 01:10 f0
```

Nous remarquons que le fichier créé obtient les droits par défaut : **`rw-rw-r--`**

Droits attribués automatiquement à un fichier -umask

❑ Règles de calcul pour les fichiers

Le principe de calcul est basé sur des notions d'algèbre booléen.

- Les droits initiaux à tout fichier est 666.
- Les droits par défaut = $666 \oplus \text{NON masque } 002$.

Explications :

Droits initiaux	Masque umask	NON masque
666	002	-
110110110	000000010	111111101

\oplus 110110110 ← Droits initiaux
111111101 ← Non masque

110110100
=664
=rw-rw-r-- ← Droits par défaut

Droits attribués automatiquement à un fichier –umask

❑ Règles de calcul pour les répertoires

Mêmes règles, sauf que Les droits initiaux à tout répertoire est 777.

Droits initiaux	Masque umask	NON masque
777	002	-
11111111	00000010	11111101

\oplus 11111111 ← Droits initiaux
11111101 ← Non masque

11111101
=775
=rwxrwxr-x ← Droits par défaut

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ umask
0002
info@ubuntu:~/Documents$ umask -S
u=rwx,g=rwx,o=rx
info@ubuntu:~/Documents$ touch f0; mkdir d0
info@ubuntu:~/Documents$ ls -l
total 8
drwxrwxr-x 2 info info 4096 oct. 12 01:10 d0
-rw-rw-r-- 1 info info 0 oct. 12 01:10 f0
```

Droits attribués automatiquement à un fichier –umask

❑ Changement des droits par défaut

Pour changer temporellement les droits par défaut, il faut changer le masque utilisé pour la génération de ces droits. La commande **umask** permet de définir un nouveau masque de génération des droits pour les nouveaux fichiers/dossiers créés :

umask num_masque

Par exemple : **umask 145**

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ umask 145
info@ubuntu:~/Documents$ touch f1; mkdir d1
info@ubuntu:~/Documents$ ls -l
total 12
drwxrwxr-x 2 info info 4096 oct. 12 01:10 d0
drw--wx-w- 2 info info 4096 oct. 12 01:13 d1
-rw-rw-r-- 1 info info 0 oct. 12 01:10 f0
-rw--w--w- 1 info info 0 oct. 12 01:13 f1
```

NB : la commande umask définit un masque temporel pour la session terminal ouverte. Une fois fermé, le masque reprend la valeur 022

Droits attribués automatiquement à un fichier -umask

❑ Changement des droits par défaut

Pour changer les droits par défaut d'une manière **permanente**. Il faut éditer le fichier caché **~/.profile** qui se trouve dans votre répertoire personnel, et puis de changer la valeurs de **# umask 022** par la valeur désirée.

Changement du propriétaire et du groupe

- ❑ Les commandes de changement du propriétaire et du groupe d'un fichier
- La commande **chown** permet de changer le propriétaire d'un fichier. Pour des raisons de sécurité, seul le **root** peut modifier le propriétaire d'un fichier ou d'un répertoire.
- La commande **chgrp** permet le changement de groupe pour les fichiers ou répertoires cités, à condition que l'utilisateur fasse partie du nouveau groupe et soit propriétaire de ces fichiers ou répertoires.

Changement du propriétaire et du groupe

La commande **chgrp**

chgrp est utilisée pour changer le groupe du fichier ou du répertoire. Le changement de groupe peut être effectué par :

- Le root
- Le propriétaire du fichier si ce dernier est parmi les membres du groupe en question.

Le syntaxe de cette commande :

chgrp [options] nouveau_groupe fichier/répertoire

Les options intéressantes :

- R : Changer l'autorisation sur les fichiers qui sont dans les sous-répertoires du répertoire en question.
- c : Changer l'autorisation pour chaque fichier.
- f : Forcer et ne pas reporter les erreurs .

Changement du propriétaire et du groupe

La commande **chgrp**

Exemple :

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ touch f1
info@ubuntu:~/Documents$ ls -l f1
-rw-rw-r-- 1 info info 0 oct. 12 01:17 f1
info@ubuntu:~/Documents$ id info
uid=1000(info) gid=1000(info) groupes=1000(info),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
info@ubuntu:~/Documents$ chgrp TP f1
chgrp: groupe incorrect : «TP»
info@ubuntu:~/Documents$ sudo groupadd TP
[sudo] Mot de passe de info :
info@ubuntu:~/Documents$ chgrp TP f1
chgrp: modification du groupe de 'f1': Opération non permise
info@ubuntu:~/Documents$ sudo chgrp TP f1
info@ubuntu:~/Documents$ ls -l f1
-rw-rw-r-- 1 info TP 0 oct. 12 01:17 f1
info@ubuntu:~/Documents$
```

Changement du propriétaire et du groupe

La commande **chown**

chown est utilisée pour changer le propriétaire et/ou le groupe propriétaire du fichier ou du répertoire. L'utilisation de cette commande n'est permis que pour le root.

Le syntaxe de cette command :

```
chown [-option] [utilisateur][:groupe] fichier  
[fichier1 fichier2 ..]
```

Elle peut être utilisée pour changer :

- Le propriétaire et le groupe
- Seulement le propriétaire
- Seulement le groupe (devienne alors similaire à **chgrp**)

Les options intéressantes :

-R : Modifie tous ses sous-répertoires et ses sous-fichiers d'une manière récursive.

-h : Modifie uniquement les liens symboliques et pas leur cible.



Changement du propriétaire et du groupe

La commande **chown**

Exemple :

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ ls -l f1
-rw-rw-r-- 1 info TP 0 oct. 12 01:17 f1
info@ubuntu:~/Documents$ sudo groupadd TD
info@ubuntu:~/Documents$ sudo chown Ensamiste:TD f1
info@ubuntu:~/Documents$ ls -l f1
-rw-rw-r-- 1 Ensamiste TD 0 oct. 12 01:17 f1
info@ubuntu:~/Documents$ sudo chown info f1
info@ubuntu:~/Documents$ ls -l f1
-rw-rw-r-- 1 info TD 0 oct. 12 01:17 f1
info@ubuntu:~/Documents$ sudo chown :TP f1
info@ubuntu:~/Documents$ ls -l f1
-rw-rw-r-- 1 info TP 0 oct. 12 01:17 f1
info@ubuntu:~/Documents$
```


ACL (Access Control List)

Limites de la gestion des accès

Nous avons vu que par défaut un système **Unix** attache trois privilèges à un fichier :

1. Les privilèges de l'utilisateur propriétaire (u).
2. Ceux du groupe propriétaire (g).
3. Ceux des autres (o).

Ainsi le partage de fichiers ne peut se faire qu'à travers le principe de groupe. Imaginant alors la situation :

Pour qu'un utilisateur **X** rend un fichier accessible en lecture et écriture à un utilisateur **Y** et uniquement à **Y**, il n'y a pas d'autre solution que de créer un groupe **G** auquel **X** et **Y** appartiennent, puis d'accorder les droits de lecture et d'écriture du fichier au groupe **G**.

ACL (Access Control List)

Limites de la gestion des privilèges

Le problème ici est que la création du groupe n'est possible que pour l'administrateur du système qui seul peut ajouter un nouveau groupe d'utilisateurs.

En outre si un autre utilisateur **Z** souhaitait rejoindre le projet, il faudrait à nouveau lui faire une demande à l'administrateur pour ajouter le nouveau membre **Z** au groupe **G**.

On voit donc clairement que la gestion des privilèges proposée par défaut dans Unix a l'inconvénient d'être trop **statique** pour l'exemple présenté, et peut devenir très vite un casse-tête qui peut conduire à des problèmes de sécurité.

ACL (Access Control List)

Introduction aux ACL

Nous vous proposons donc ici de découvrir quelques commandes (ACL) permettant de régler les privilèges plus finement et de manière plus autonome.

Mais tout d'abord, pour travailler avec les ACL, il y a deux prérequis :

- ❑ Le noyau doit supporter les ACL.
- ❑ Le système de fichier est monté avec l'option **acl**

Sachez que les ACL ne peuvent être utilisées que si le noyau les supportes (Sinon, il faut recompiler le noyau...).

Sur **Ubuntu**, le noyau prend en charge les ACL, mais elles ne sont pas nativement activées.

ACL - configuration

Configuration ACL du noyau

Vérifiant la configuration du noyau à travers cette commande lancée en mode super-user :

```
# grep ACL /boot/config-*
```

La ligne suivante indique que le support général des ACL est présent :

```
CONFIG_****_FS_POSIX_ACL=y
```

les étoiles sont remplacées par les systèmes de fichiers prises en charge par les ACL (exemples: EXT2,EXT3,EXT4,JFS,...)

NB: comme nous pouvons voir, les systèmes de fichiers habituels à Windows (Fat16,Fat32)

En conséquences : Une clé USB formatée sous Windows ne sera pas prise en charge par les ACL.

```
info@ubuntu:~/Documents$ sudo su
root@ubuntu:/home/info/Documents# grep ACL /boot/config-*
/boot/config-5.4.0-42-generic:CONFIG_EXT4_FS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_REISERFS_FS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_JFS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_XFS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_BTRFS_FS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_F2FS_FS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_FS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_SHIFT_FS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_TMPFS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_JFFS2_FS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_EROFS_FS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_NFS_V3_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_NFSD_V2_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_NFSD_V3_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_CIFS_SUPPORT=m
/boot/config-5.4.0-42-generic:CONFIG_CIFS_POSIX_ACL=y
/boot/config-5.4.0-42-generic:CONFIG_CIFS_SMB_ACL=y
```



ACL - installation

Configuration ACL du noyau

Votre système de fichiers est de type : EXT3, donc normalement les ACL sont supportées et activées, mais nous ne pouvons toujours pas les modifier, pour cela nous devons installer le paquet **acl** qui comporte les deux commandes nécessaires pour cette modification, en tapant la commande suivante :

```
# apt-get install acl
```

NB :

- 1- Cela peut nécessiter une connexion internet
- 2- Parfois il est nécessaire de re-monter les partitions avec l'option acl"

```
exit
info@ubuntu: ~/Documents$ sudo apt-get install acl
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
acl est déjà la version la plus récente (2.2.53-6).
acl passé en « installé manuellement ».
Le paquet suivant a été installé automatiquement et n'est plus nécessaire :
  libfprint-2-tod1
Veuillez utiliser « sudo apt autoremove » pour le supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 17 non mis à jour.
info@ubuntu:~/Documents$ █
```



ACL – Types

La mise en place des ACL implique trois notions principales:

1. ACL « minimale »: Composée exclusivement d'éléments de type propriétaire, groupe et « reste du monde », l'ACL minimale est une traduction « en ACL » des droits d'accès traditionnels Unix.
2. ACL étendue : L'ACL étendue prolonge les droits de l'ACL minimale. Elle contient au moins un élément de type mask et peut contenir des éléments de type utilisateur et/ou groupe.
3. ACL par défaut : Les ACL par défaut ne peuvent être appliquées qu'aux répertoires et définissent de quels droits un objet du système de fichiers devra hériter (de son répertoire parent) lors de sa création.

ACL – les commandes

Le paquetage **acl** installé, permet de disposer de deux nouvelles commandes pour la gestion des ACL sous les trois types d'ACL. C'est deux commandes sont :

- ❑ **setfacl** (**Eng** - *set file's ACL* « régler l'ACL du fichier ») : mise en place de ces autorisations.
- ❑ **getfacl** (**Eng** - *get file's ACL* « récupérer l'ACL du fichier ») examen des autorisations d'un fichier.

ACL – la commande setfacl

Attribution des ACL étendues

La syntaxe:

```
setfacl -m u:utilisateur:permissions fichier
```

Permet de modifier (c'est le sens de l'option **m**) les autorisations de l'**utilisateur** (argument **u:**). Les autorisations **permissions** peuvent être définies en utilisant la notation symbolique sur le **fichier**.

-m, --modify - modifier les ACL d'un fichier ou répertoire

-x, --remove - supprime des entrées ACLs

-b, --remove-all - supprime toutes les entrées ACLs

-L, --logical - suivi des liens symboliques

-R, --recursive - application des ACLs de façon récursive

Regardez le manuel `man setfacl` pour plus de détails.

ACL – la commande getfacl

Examiner les autorisations associées à un fichier

La syntaxe:

```
getfacl [option] fichier
```

La commande **getfacl** nous permet d'afficher les ACL du fichier.

Les quelques options :

- R** permet de voir les ACLs de façon récursive.
- L** pour le suivi des liens symboliques.

Regardez le manuel man getfacl pour plus de détails.

ACL – exemple : donner droits à un utilisateur

Lors de sa création, un fichier possède des autorisations minimales

À travers les droits par défaut du masque utilisé.

La commande `ls -l` et `getfacl` affichent des résultats identiques. Mais dès que nous attribuons les droits à un nouveau utilisateur par `setfacl`, nous remarquons l'apparition le signe (+) dans `ls -l` qu'il y a des ACL étendues qui sont définies sur ce fichier.

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ ls -l
total 0
-rw-rw-r-- 1 info TP 0 oct. 12 01:17 f1
info@ubuntu:~/Documents$ getfacl f1
# file: f1
# owner: info
# group: TP
user::rw-
group::rw-
other::r--
```

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ setfacl -m u:Ensamiste:rwX f1
info@ubuntu:~/Documents$ ls -l
total 0
-rw-rwxr--+ 1 info TP 0 oct. 12 01:17 f1
info@ubuntu:~/Documents$ getfacl f1
# file: f1
# owner: info
# group: TP
user::rw-
user:Ensamiste:rwX
group::rw-
mask::rwX
other::r--
```

ACL — exemple : donner droits à un groupe

De la même manière, les droits ACL peuvent être attribuée à un groupe, en remplaçant le **u** par **g**

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ setfacl -m g:TD:rwx f1
info@ubuntu:~/Documents$ getfacl f1
# file: f1
# owner: info
# group: TP
user::rw-
user:Ensamiste:rwx
group::rw-
group:TD:rwx
mask::rwx
other::r--
```


ACL – exemple : appliquer un masque

Voyons maintenant l'intérêt du masque ACL. Supposant qu'on désire :

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ setfacl -m m::r-x f1
info@ubuntu:~/Documents$ getfacl f1
# file: f1
# owner: info
# group: TP
user::rw-
user:Ensamiste:rw-
group::rw-
group:TD:rw-
mask::r-x
other::r--
#effective:r-x
#effective:r--
#effective:r-x
```

Supprimer pour tous les utilisateurs (en dehors de moi, le owner), le droit en écriture. Je supprime le droit en écriture dans le masque :

setfacl -m m::r-x fichier

Les droits attribués aux utilisateurs ne change pas, mais les droits **effectives** changent en fonction du masque.

Par exemple ici, les entités qui vont avoir une restriction de droit à cause du masque : smi1, est les membres du groupe.

ACL – exemple : appliquer un masque

Si nous désirons enlever à **Ensamiste** tout droit ACL sur ce fichier :

```
info@ubuntu: ~/Documents
info@ubuntu:~/Documents$ setfacl -x u:Ensamiste f1
info@ubuntu:~/Documents$ getfacl f1
# file: f1
# owner: info
# group: TP
user::rw-
group::rw-
group:TD:rwx
mask::rwx
other::r--

info@ubuntu:~/Documents$ setfacl -b f1
info@ubuntu:~/Documents$ getfacl f1
# file: f1
# owner: info
# group: TP
user::rw-
group::rw-
other::r--
```

À travers cette commande :
setfacl -x u:smi1 fichier
Et il redevient un utilisateur normal soumis aux règles classiques.

Vous pouvez aussi supprimer l'ensemble des droits ACL d'un fichier :
setfacl -b fichier

Prochain chapitre

Nous allons entamer la gestion des processus : le principe
et les commandes associées