

Administration Système sous Linux (Ubuntu Server)

Grégory Morel

2022-2023

CPE Lyon - 3IRC / 4ETI / 3ICS

Cours 3

Gestion des utilisateurs et des droits

Gestion des utilisateurs et des groupes

Utilisateurs et groupes sous Linux

Rappels sur Linux

- basé sur UNIX, conçu dès l'origine comme un système multi-utilisateurs
- tout est fichier (fichiers, dossiers, configuration matérielle, périphériques...)

⇒ nécessité de définir des groupes (*administration, comptabilité, R&D, etc.*) et de gérer finement les droits sur les fichiers (seule l'équipe R&D a accès aux documents sensibles sur les projets en cours; seul l'admin peut modifier la configuration ou exécuter certains programmes, etc.)

A retenir

Tout utilisateur doit appartenir au moins à un groupe, appelé groupe primaire.
Un utilisateur peut appartenir à plusieurs groupes

Utilisateurs et groupes sous Linux

Rappels sur Linux

- basé sur UNIX, conçu dès l'origine comme un système multi-utilisateurs
- tout est fichier (fichiers, dossiers, configuration matérielle, périphériques...)

⇒ nécessité de définir des groupes (*administration, comptabilité, R&D, etc.*) et de gérer finement les droits sur les fichiers (seule l'équipe R&D a accès aux documents sensibles sur les projets en cours; seul l'admin peut modifier la configuration ou exécuter certains programmes, etc.)

A retenir

Tout utilisateur doit appartenir au moins à un groupe, appelé groupe primaire.
Un utilisateur peut appartenir à plusieurs groupes

Utilisateurs et groupes sous Linux

Rappels sur Linux

- basé sur UNIX, conçu dès l'origine comme un système **multi-utilisateurs**
- tout est fichier (fichiers, dossiers, configuration matérielle, périphériques...)

⇒ nécessité de définir des **groupes** (*administration, comptabilité, R&D, etc.*) et de gérer finement les **droits** sur les fichiers (seule l'équipe R&D a accès aux documents sensibles sur les projets en cours; seul l'admin peut modifier la configuration ou exécuter certains programmes, etc.)

A retenir

Tout utilisateur doit appartenir au moins à un groupe, appelé **groupe primaire**.
Un utilisateur peut appartenir à plusieurs groupes

Deux familles de commandes :

	<code>useradd</code> / <code>userdel</code> <code>groupadd</code> / <code>groupdel</code>	<code>adduser</code> / <code>deluser</code> <code>addgroup</code> / <code>delgroup</code>
Type	Binaire	Script Perl
Interactif	Non	Oui
Usage	Scripts	Ligne de commande

Ajout d'un utilisateur

Dossier personnel

Quand on crée un utilisateur avec **adduser**, un dossier personnel du même nom est automatiquement créé dans **/home**, basé sur le dossier **/etc/skel** (*skeleton*)

Rem. : **useradd** ne le fait pas par défaut, il faut lui préciser l'option **-m**

Important

Tout compte créé sans mot de passe est **inactif**, jusqu'à l'attribution d'un mot de passe (par exemple avec **passwd**)¹

Si aucun groupe n'est spécifié pour le nouvel utilisateur, un **nouveau groupe à son nom** est automatiquement créé²

1. C'est notamment le cas du compte **root** sous Ubuntu

2. Il n'est pas possible de créer un utilisateur portant le même nom qu'un groupe existant

Ajout d'un utilisateur

Dossier personnel

Quand on crée un utilisateur avec **adduser**, un dossier personnel du même nom est automatiquement créé dans **/home**, basé sur le dossier **/etc/skel** (*skeleton*)

Rem. : **useradd** ne le fait pas par défaut, il faut lui préciser l'option **-m**

Important

Tout compte créé sans mot de passe est **inactif**, jusqu'à l'attribution d'un mot de passe (par exemple avec **passwd**)¹

Si aucun groupe n'est spécifié pour le nouvel utilisateur, un **nouveau groupe à son nom** est automatiquement créé²

1. C'est notamment le cas du compte **root** sous Ubuntu

2. Il n'est pas possible de créer un utilisateur portant le même nom qu'un groupe existant

Ajout d'un utilisateur

Dossier personnel

Quand on crée un utilisateur avec **adduser**, un dossier personnel du même nom est automatiquement créé dans **/home**, basé sur le dossier **/etc/skel** (*skeleton*)

Rem. : **useradd** ne le fait pas par défaut, il faut lui préciser l'option **-m**

Important

Tout compte créé sans mot de passe est **inactif**, jusqu'à l'attribution d'un mot de passe (par exemple avec **passwd**)¹

Si aucun groupe n'est spécifié pour le nouvel utilisateur, un **nouveau groupe à son nom** est automatiquement créé²

1. C'est notamment le cas du compte **root** sous Ubuntu

2. Il n'est pas possible de créer un utilisateur portant le même nom qu'un groupe existant

Suppression d'un utilisateur

Dossier personnel

Par défaut, supprimer un utilisateur ne supprime pas son répertoire personnel, sa boîte aux lettres ou tout autre fichier possédé par l'utilisateur sur le système.

💡 `userdel` et `deluser` proposent des options pour supprimer le dossier personnel, et des options pour créer une sauvegarde de ce dossier avant suppression¹

1. Ces options peuvent être activées par défaut dans le fichier de configuration `/etc/deluser.conf`

Connaître les utilisateurs et les groupes

Les commandes **users** et **groups** sont des faux-amis!

- **users** : affiche les utilisateurs **actuellement /logués** sur la machine
- **groups** [**user**] : affiche les groupes **auxquels appartient user**¹

Pour connaître tous les utilisateurs / tous les groupes, il faut regarder la première colonne des fichiers **/etc/passwd** / **/etc/group**² :

```
$ cut -d: -f1-4 /etc/passwd
root:x:0:0
daemon:x:1:1
bin:x:2:2
sys:x:3:3
...
```

1. Le groupe primaire est le premier de la liste

2. Il existe une commande **members** non installée par défaut

Connaître les utilisateurs et les groupes

Les commandes **users** et **groups** sont des faux-amis!

- **users** : affiche les utilisateurs **actuellement** / **logués** sur la machine
- **groups** [*user*] : affiche les groupes **auxquels appartient** *user*¹

Pour connaître tous les utilisateurs / tous les groupes, il faut regarder la première colonne des fichiers **/etc/passwd** / **/etc/group**² :

```
$ cut -d: -f1-4 /etc/passwd
root:x:0:0
daemon:x:1:1
bin:x:2:2
sys:x:3:3
...
```


1. Le groupe primaire est le premier de la liste
2. Il existe une commande **members** non installée par défaut

Identifiants d'utilisateur et de groupe

Les nombres qu'on peut voir dans `/etc/passwd` et `/etc/group` sont les **identifiants d'utilisateur** (*uid*) et **de groupe** (*gid*).

La commande **id** permet de les obtenir :

```
$ id batman
uid=1002(batman) gid=1003(dc-comics) groupes=1003(dc-comics)
$ id -g batman
1003
$ id -g -n batman
dc-comics
```

 Le compte utilisateur d'uid **0** est celui du super administrateur (**root**).
Les identifiants < 1000 sont réservés au système.

Modifier les propriétés d'un utilisateur ou d'un groupe

Il faut utiliser les commandes **usermod** / **groupmod** :

- modifier le login d'un utilisateur :

```
sudo usermod -l newUsername oldUsername
```

- déplacer le dossier HOME de l'utilisateur :

```
sudo usermod -m -d newHomedir username
```

- modifier le groupe primaire d'un utilisateur :

```
sudo usermod batman -g avengers
```

- modifier le nom d'un groupe :

```
sudo groupmod -n newGroupName oldGroupName
```

- modifier l'identifiant d'un groupe :

```
sudo groupmod -g newID groupName
```

Pour modifier les informations associées à un utilisateur (nom complet, téléphone, numéro de bureau...), on utilise **chfn** (*CHange Full Name*) :

```
sudo chfn johndoe -f Johnny
```

Modifier les propriétés d'un utilisateur ou d'un groupe

Il faut utiliser les commandes **usermod** / **groupmod** :

- modifier le login d'un utilisateur :

```
sudo usermod -l newUsername oldUsername
```

- déplacer le dossier HOME de l'utilisateur :

```
sudo usermod -m -d newHomedir username
```

- modifier le groupe primaire d'un utilisateur :

```
sudo usermod batman -g avengers
```

- modifier le nom d'un groupe :

```
sudo groupmod -n newGroupName oldGroupName
```

- modifier l'identifiant d'un groupe :

```
sudo groupmod -g newID groupName
```

Pour modifier les informations associées à un utilisateur (nom complet, téléphone, numéro de bureau...), on utilise **chfn** (*CHange Full Name*) :

```
sudo chfn johndoe -f Johnny
```


Modifier les propriétés d'un utilisateur ou d'un groupe

Il faut utiliser les commandes **usermod** / **groupmod** :

- modifier le login d'un utilisateur :

```
sudo usermod -l newUsername oldUsername
```

- déplacer le dossier HOME de l'utilisateur :

```
sudo usermod -m -d newHomedir username
```

- modifier le groupe primaire d'un utilisateur :

```
sudo usermod batman -g avengers
```

- modifier le nom d'un groupe :

```
sudo groupmod -n newGroupName oldGroupName
```

- modifier l'identifiant d'un groupe :

```
sudo groupmod -g newID groupName
```

Pour modifier les informations associées à un utilisateur (nom complet, téléphone, numéro de bureau...), on utilise **chfn** (*CHange Full Name*) :

```
sudo chfn johndoe -f Johnny
```

Modifier les propriétés d'un utilisateur ou d'un groupe

Il faut utiliser les commandes **usermod** / **groupmod** :

- modifier le login d'un utilisateur :

```
sudo usermod -l newUsername oldUsername
```

- déplacer le dossier HOME de l'utilisateur :

```
sudo usermod -m -d newHomedir username
```

- modifier le groupe primaire d'un utilisateur :

```
sudo usermod batman -g avengers
```

- modifier le nom d'un groupe :

```
sudo groupmod -n newGroupName oldGroupName
```

- modifier l'identifiant d'un groupe :

```
sudo groupmod -g newID groupName
```

Pour modifier les informations associées à un utilisateur (nom complet, téléphone, numéro de bureau...), on utilise **chfn** (*CHange Full Name*) :

```
sudo chfn johndoe -f Johnny
```

Modifier les propriétés d'un utilisateur ou d'un groupe

Il faut utiliser les commandes **usermod** / **groupmod** :

- modifier le login d'un utilisateur :

```
sudo usermod -l newUsername oldUsername
```

- déplacer le dossier HOME de l'utilisateur :

```
sudo usermod -m -d newHomedir username
```

- modifier le groupe primaire d'un utilisateur :

```
sudo usermod batman -g avengers
```

- modifier le nom d'un groupe :

```
sudo groupmod -n newGroupName oldGroupName
```

- modifier l'identifiant d'un groupe :

```
sudo groupmod -g newID groupName
```

Pour modifier les informations associées à un utilisateur (nom complet, téléphone, numéro de bureau...), on utilise **chfn** (*CHange Full Name*) :

```
sudo chfn johndoe -f Johnny
```

Modifier les propriétés d'un utilisateur ou d'un groupe

Il faut utiliser les commandes **usermod** / **groupmod** :

- modifier le login d'un utilisateur :

```
sudo usermod -l newUsername oldUsername
```

- déplacer le dossier HOME de l'utilisateur :

```
sudo usermod -m -d newHomedir username
```

- modifier le groupe primaire d'un utilisateur :

```
sudo usermod batman -g avengers
```

- modifier le nom d'un groupe :

```
sudo groupmod -n newGroupName oldGroupName
```

- modifier l'identifiant d'un groupe :

```
sudo groupmod -g newID groupName
```

Pour modifier les informations associées à un utilisateur (nom complet, téléphone, numéro de bureau...), on utilise **chfn** (*CHange Full Name*) :

```
sudo chfn johndoe -f Johnny
```

Ajouter un utilisateur à un groupe

- Ajouter un utilisateur existant à un groupe (secondaire) existant :
`usermod -a -G nom_groupe nom_utilisateur`
- Modifier le groupe primaire d'un utilisateur :
`usermod -g nom_groupe nom_utilisateur`
- Ajouter un *nouvel* utilisateur et spécifier son groupe primaire (existant) :
`useradd -g nom_groupe nom_utilisateur`
- Ajouter un *nouvel* utilisateur et spécifier un groupe secondaire (existant) :
`useradd -G nom_groupe nom_utilisateur`

Administrer les groupes

L'administration des groupes peut aussi se faire via la commande **gpsswd** :

-a utilisateur groupe	ajoute l' <i>utilisateur</i> au <i>groupe</i>
-d utilisateur groupe	enlève l' <i>utilisateur</i> du <i>groupe</i>
-r groupe	enlève le mot de passe du <i>groupe</i>
-A user,...	configure la liste des administrateurs
-M user,...	configure la liste des membres

💡 Pour les traitements par lots, on se tournera vers la commande **newusers**

Fichier `/etc/passwd`

Contient toutes les informations relatives aux utilisateurs (login, mots de passe,...)

Seul le superutilisateur (**root**) doit pouvoir le modifier!

Format : 7 champs séparés par ' : '

- nom du compte
- mot de passe¹
- identifiant utilisateur (**UID**, ≥ 1000 pour les utilisateurs "non système")
- identifiant de groupe (**GID**)
- commentaire / nom réel
- répertoire de connexion (**\$HOME**)
- interpréteur de commandes de l'utilisateur (**\$SHELL**)

1. Un 'x' dans cette colonne signifie que le mot de passe est chiffré dans le fichier `/etc/shadow`.
Une chaîne vide peut être refusée par certaines applications

Fichier /etc/passwd

Exemple :

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
spiderman:x:1000:1000:Peter Parker:/home/spiderman:/bin/bash
...
```

Il ne faut jamais éditer directement ce fichier, notamment pour essayer de modifier un mot de passe : toute chaîne de caractères dans ce champ est interprétée comme un mot de passe chiffré. A la place, il faut utiliser la commande **passwd**.

Pour en savoir plus : **man 5 passwd**

Fichier contenant les mots de passe chiffrés; 9 champs séparés par ' : '

- login
- mot de passe chiffré
- date du dernier changement de mot de passe (en jours depuis le 1^{er} janvier 1970)
- âge minimum (nombre de jours avant de pouvoir changer le mot de passe)
- âge maximum (doit être > âge minimum)
- période d'avertissement de l'expiration du mot de passe
- durée d'autorisation du mot de passe après expiration
- date de fin de validité du compte (depuis le 1^{er} janvier 1970)
- champ réservé pour une utilisation future

Fichier /etc/shadow

Exemple¹ :

```
$ cat /etc/shadow
...
spiderman:$6$TIUqp8Ec$qa0wMI...:17892:0:99999:7:::
systemd-coredump:!!:17892:::::::::
toto:!:17893:0:99999:7:::
...
```

Il ne faut jamais éditer directement ce fichier, il est géré par **passwd**.

Pour en savoir plus : **man passwd** et **man 5 shadow**

-
1. Les 3 premiers caractères du champ 'mot de passe' indiquent l'algorithme de hachage utilisé (par exemple **\$6\$** pour SHA-512). Un **'!'** indique que le compte est verrouillé ou que le mot de passe n'a jamais été défini

Fichiers `/etc/group` et `/etc/gshadow`

Contiennent la liste des utilisateurs appartenant aux différents groupes, et les mots de passe de groupes.

Format :

- nom du groupe
- champ special
- numero de groupe
- liste des membres

Un utilisateur **peut appartenir à plusieurs groupes** ; lorsqu'il se connecte, il appartient au **groupe primaire spécifié dans `/etc/passwd`**.

L'administrateur d'un compte peut gérer :

- la date d'**expiration** du compte : date à laquelle il sera désactivé
- la date d'**expiration** du mot de passe : date à laquelle l'utilisateur devra changer son mot de passer
- une durée d'**inactivité** du mot de passe : nombre de jours pendant lesquels l'utilisateur peut changer son mot de passe avant blocage de son compte

Règle de sécurité

Définir une durée de vie maximale pour les mots de passe des utilisateurs

Gestion des comptes et des mots de passe

La commande **chage** permet de lister et modifier les informations de validité d'un mot de passe :

```
$ sudo chage -l batman
Dernier changement de mot de passe      : déc. 27, 2018
Fin de validité du mot de passe         : jamais
Mot de passe désactivé                  : jamais
Fin de validité du compte                : jamais
Nombre minimum de jours entre les changements de mot de
passe                                   : 0
Nombre maximum de jours entre les changements de mot de
passe                                   : 99999
Nombre de jours d'avertissement avant la fin de validité du
mot de passe                            : 7
```

Gestion des comptes et des mots de passe

Forcer un utilisateur à changer son mot de passe

```
passwd -e nom_utilisateur
```

Verrouiller un mot de passe¹


```
passwd -l nom_utilisateur
```

Verrouiller un compte : mettre une date d'expiration passée

- `usermod --expiredate 0 nom_utilisateur`
- `chage --expiredate 0 nom_utilisateur`

Réactiver un compte : mettre "" ou -1

- `usermod --expiredate "" nom_utilisateur`
- `chage --expiredate -1 nom_utilisateur`

1.  N'empêche pas une connexion par un autre moyen, par ex. SSH!

Le superutilisateur : root

L'utilisateur **root**

Quelques caractéristiques :

- c'est un **super-utilisateur** : il possède **toutes** les permissions d'administration du système, en particulier :
 - il est le seul utilisateur à pouvoir modifier le dossier racine **/**, d'où son nom
 - il peut accéder à n'importe quel fichier ou dossier
 - il peut prendre l'identité de n'importe quel autre utilisateur **sans connaître leur mot de passe!**
- son identifiant (**uid**) est toujours **0**
- il est symbolisé par le symbole **#** sur l'invite de commande ¹
- contrairement aux autres utilisateurs, son répertoire personnel est dans **/root** et non dans **/home**

1. Comportement par défaut; mais on peut le changer en modifiant la variable d'environnement **PS1**

L'utilisateur **root**

Quelques caractéristiques :

- c'est un **super-utilisateur** : il possède **toutes** les permissions d'administration du système, en particulier :
 - il est le seul utilisateur à pouvoir modifier le dossier racine **/**, d'où son nom
 - il peut accéder à n'importe quel fichier ou dossier
 - il peut prendre l'identité de n'importe quel autre utilisateur **sans connaître leur mot de passe!**
- son identifiant (**uid**) est toujours **0**
- il est symbolisé par le symbole **#** sur l'invite de commande ¹
- contrairement aux autres utilisateurs, son répertoire personnel est dans **/root** et non dans **/home**

1. Comportement par défaut; mais on peut le changer en modifiant la variable d'environnement **PS1**

L'utilisateur **root**

Quelques caractéristiques :

- c'est un **super-utilisateur** : il possède **toutes** les permissions d'administration du système, en particulier :
 - il est le seul utilisateur à pouvoir modifier le dossier racine **/**, d'où son nom
 - il peut accéder à n'importe quel fichier ou dossier
 - il peut prendre l'identité de n'importe quel autre utilisateur **sans connaître leur mot de passe!**
- son identifiant (**uid**) est toujours **0**
- il est symbolisé par le symbole **#** sur l'invite de commande¹
- contrairement aux autres utilisateurs, son répertoire personnel est dans **/root** et non dans **/home**

1. Comportement par défaut; mais on peut le changer en modifiant la variable d'environnement **PS1**

L'utilisateur **root**

Quelques caractéristiques :

- c'est un **super-utilisateur** : il possède **toutes** les permissions d'administration du système, en particulier :
 - il est le seul utilisateur à pouvoir modifier le dossier racine **/**, d'où son nom
 - il peut accéder à n'importe quel fichier ou dossier
 - il peut prendre l'identité de n'importe quel autre utilisateur **sans connaître leur mot de passe**!
- son identifiant (**uid**) est toujours **0**
- il est symbolisé par le symbole **#** sur l'invite de commande¹
- contrairement aux autres utilisateurs, son répertoire personnel est dans **/root** et non dans **/home**

1. Comportement par défaut; mais on peut le changer en modifiant la variable d'environnement **PS1**

L'utilisateur **root**

Sous Ubuntu, le compte **root** est **désactivé par défaut** (pas de mot de passe)

Avantages :

- évite de devoir changer et communiquer aux administrateurs le nouveau mot de passe root quand quelqu'un quitte l'entreprise
- impossible de lancer une attaque par force brute sur le compte **root** pour trouver le mot de passe ; à la place, le hacker doit connaître le nom d'un compte local doté des droits d'administration
- un seul mot de passe à retenir pour les administrateurs (le leur)
- on peut même choisir pour chaque utilisateur les commandes qu'il a le droit de lancer avec **sudo**

Inconvénients :

- on perd la sécurité de la double authentification (deux mots de passe)

Pas de solution parfaite ! L'administrateur doit veiller à la sécurité du système

Les commandes **su** et **sudo**

La commande **su**

su *utilisateur* permet de prendre l'identité d'un autre utilisateur (**su** = substitute / switch user¹):

```
batman$ users ; whoami
batman
batman
batman$ su spiderman
Mot de passe :                #(de spiderman)
spiderman$ users ; whoami
batman
spiderman
```

Si *utilisateur* n'est pas spécifié, c'est l'utilisateur **root** qui est implicite
💡 ⇒ impossible sous Ubuntu par défaut puisque **root n'a pas de mot de passe**

1. Et pas "super user" !

La commande **su**

Pour mettre fin à une session shell, on utilise la commande **exit** (ou CTRL+D).

⇒ permet de reprendre l'identité qu'on avait juste avant la commande **su** :

```
batman$ whoami
batman
batman$ su spiderman
Mot de passe :                #(de spiderman)
spiderman$ whoami
spiderman
spiderman$ exit
batman$ whoami
batman
```

La commande **su**

Par défaut, **su** conserve l'environnement de l'appelant. Pour prendre l'identité d'un utilisateur et son environnement, on utilise **su -l** (ou **su -**):

```
batman$ export VAR=maVariable
batman$ su spiderman
spiderman$ echo $VAR
maVariable      #VAR existe car l'environnement initial est conservé
spiderman$ exit
batman$ su -l spiderman
spiderman$ echo $VAR
                #VAR n'existe pas dans l'environnement de spiderman
spiderman$
```


La commande **sudo**

sudo -u utilisateur commande : exécuter *commande* en tant que *utilisateur*

Différences entre **su** et **sudo** :

- avec **su** on prend l'identité de quelqu'un d'autre, pas avec **sudo**
- les commandes sont enregistrées dans l'historique de celui qui les tape réellement
- toutes les commandes exécutées avec **sudo** sont journalisées, dans `/var/log/auth.log`
- Si un utilisateur non autorisé tente une action **sudo**, l'administrateur en est automatiquement informé
- **su** demande le mot de passe de l'utilisateur *cible*, alors que **sudo** demande le mot de passe de l'utilisateur *courant*
- **sudo** évite de laisser indéfiniment une session ouverte en tant qu'administrateur

La commande **sudo**

sudo -u utilisateur commande : exécuter *commande* en tant que *utilisateur*

Différences entre **su** et **sudo** :

- avec **su** on prend l'identité de quelqu'un d'autre, pas avec **sudo**
- les commandes sont enregistrées dans l'historique de celui qui les tape réellement
- toutes les commandes exécutées avec **sudo** sont journalisées, dans `/var/log/auth.log`
- Si un utilisateur non autorisé tente une action **sudo**, l'administrateur en est automatiquement informé
- **su** demande le mot de passe de l'utilisateur *cible*, alors que **sudo** demande le mot de passe de l'utilisateur *courant*
- **sudo** évite de laisser indéfiniment une session ouverte en tant qu'administrateur

La commande **sudo**

sudo -u utilisateur commande : exécuter *commande* en tant que *utilisateur*

Différences entre **su** et **sudo** :

- avec **su** on prend l'identité de quelqu'un d'autre, pas avec **sudo**
- les commandes sont enregistrées dans l'historique de celui qui les tape réellement
- toutes les commandes exécutées avec **sudo** sont journalisées, dans **/var/log/auth.log**
- Si un utilisateur non autorisé tente une action **sudo**, l'administrateur en est automatiquement informé
- **su** demande le mot de passe de l'utilisateur *cible*, alors que **sudo** demande le mot de passe de l'utilisateur *courant*
- **sudo** évite de laisser indéfiniment une session ouverte en tant qu'administrateur

La commande **sudo**

sudo -u utilisateur commande : exécuter *commande* en tant que *utilisateur*

Différences entre **su** et **sudo** :

- avec **su** on prend l'identité de quelqu'un d'autre, pas avec **sudo**
- les commandes sont enregistrées dans l'historique de celui qui les tape réellement
- toutes les commandes exécutées avec **sudo** sont journalisées, dans **/var/log/auth.log**
- Si un utilisateur non autorisé tente une action **sudo**, l'administrateur en est automatiquement informé
- **su** demande le mot de passe de l'utilisateur *cible*, alors que **sudo** demande le mot de passe de l'utilisateur *courant*
- **sudo** évite de laisser indéfiniment une session ouverte en tant qu'administrateur

La commande **sudo**

sudo -u utilisateur commande : exécuter *commande* en tant que *utilisateur*

Différences entre **su** et **sudo** :

- avec **su** on prend l'identité de quelqu'un d'autre, pas avec **sudo**
- les commandes sont enregistrées dans l'historique de celui qui les tape réellement
- toutes les commandes exécutées avec **sudo** sont journalisées, dans **/var/log/auth.log**
- Si un utilisateur non autorisé tente une action **sudo**, l'administrateur en est automatiquement informé
- **su** demande le mot de passe de l'utilisateur *cible*, alors que **sudo** demande le mot de passe de l'utilisateur *courant*
- **sudo** évite de laisser indéfiniment une session ouverte en tant qu'administrateur

La commande **sudo**

sudo -u utilisateur commande : exécuter *commande* en tant que *utilisateur*

Différences entre **su** et **sudo** :

- avec **su** on prend l'identité de quelqu'un d'autre, pas avec **sudo**
- les commandes sont enregistrées dans l'historique de celui qui les tape réellement
- toutes les commandes exécutées avec **sudo** sont journalisées, dans **/var/log/auth.log**
- Si un utilisateur non autorisé tente une action **sudo**, l'administrateur en est automatiquement informé
- **su** demande le mot de passe de l'utilisateur *cible*, alors que **sudo** demande le mot de passe de l'utilisateur *courant*
- **sudo** évite de laisser indéfiniment une session ouverte en tant qu'administrateur

Question

Si `sudo` demande le mot de passe de l'utilisateur `courant`, n'importe qui peut exécuter des commandes en tant que superadministrateur?

💡 Heureusement, non! `sudo` repose sur un fichier de configuration, (`/etc/sudoers`) qui indique qui a les "droits sudo" :

```
batman$ sudo cat /etc/sudoers
...
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL ...
```

💡 \Rightarrow pour donner / retirer les privilèges sudo à un utilisateur il suffit de l'ajouter / l'enlever du groupe `sudo`¹

1. On peut aussi éditer le fichier `/etc/sudoers` à l'aide de `visudo`

Question

Si `sudo` demande le mot de passe de l'utilisateur `courant`, n'importe qui peut exécuter des commandes en tant que superadministrateur ?

💡 Heureusement, non ! `sudo` repose sur un fichier de configuration, (`/etc/sudoers`) qui indique qui a les "droits sudo" :

```
batman$ sudo cat /etc/sudoers
...
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL ...
```

💡 ⇒ pour donner / retirer les privilèges sudo à un utilisateur il suffit de l'ajouter / l'enlever du groupe `sudo`¹

1. On peut aussi éditer le fichier `/etc/sudoers` à l'aide de `visudo`

Mais si j'ai *vraiment* besoin d'être **root** ?

Plusieurs possibilités :

1. **sudo -s** : lance un nouveau shell avec l'environnement de l'utilisateur original
2. **sudo -i** : lance un nouveau shell avec l'environnement de l'utilisateur cible (ici, root)

	sudo -s	sudo -i
HOME=	/home/foo	/root
PWD=	/home/foo	/root
PATH=	/usr/local/sbin:/usr/local/bin: /usr/sbin:/usr/bin:/sbin:/bin:/usr/X11R6/bin	/usr/local/sbin:/usr/local/bin: /usr/sbin:/usr/bin:/sbin:/bin:/usr/games
fichier(s) exécuté(s)	/home/foo/.bashrc	/etc/environment, /root/.login, /root/.profile, /root/.bashrc

3. **sudo su** : contourne l'absence de mot de passe de **root** en passant par **sudo** ⇒ **pas recommandé**

Changer de groupe temporairement

La commande **newgrp** permet de **changer de groupe** (GID) de l'utilisateur **le temps d'une nouvelle session**.

💡 C'est donc l'équivalent de la commande **su** pour les groupes

1

C'est possible dans les cas suivants :

- l'utilisateur est root,
- l'utilisateur est membre du groupe,
- l'utilisateur n'est pas membre du groupe, mais le groupe est protégé par un mot de passe et l'utilisateur connaît ce mot de passe

Important

La commande **newgrp** ne fonctionne donc pas si l'utilisateur n'est pas membre du groupe et que le groupe n'a pas de mot de passe

1. L'équivalent de la commande **sudo** pour les groupes est **sg**

Changer de groupe temporairement

Exemple :

```
batman$ groups
batman
batman$ id
uid=1002(batman) gid=1003(batman) groupes=1003(batman)
batman$ newgrp avengers
Mot de passe :                # mot de passe du groupe avengers
batman$ groups
avengers batman
batman$ id
uid=1002(batman) gid=1004(avengers) groupes=1004(avengers),
1003(batman)
```

Gestion des droits

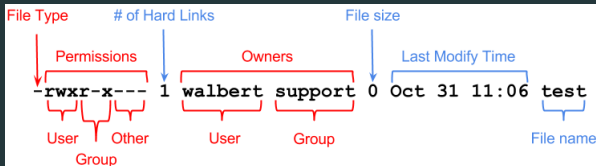
Droits

La commande `ls -l` montre que tout fichier ou dossier dispose de **trois types de droits** :

	Fichier	Dossier
Lecture (r ead)	Afficher le contenu	
Ecriture (w rite)	Modifier	Créer / Supprimer des fichiers
Exécution (e xecute)	Exécuter un binaire / script	Rentrer dans le dossier

Ces droits sont présents en 3 exemplaires :

- pour l'utilisateur propriétaire du fichier / dossier
- pour le groupe propriétaire du fichier / dossier
- pour tous les autres utilisateurs



Types de fichiers (premier caractère) :

- - : fichiers "normaux"
- d : dossiers
- l : liens symboliques
- c : périphériques en mode caractère (ex. : clavier, terminal)
- b : périphériques en mode bloc (ex. : disques durs)
- s : sockets locales (communications entre tâches, par ex. avec le serveur X)
- p : tubes nommés (*named pipes*) (communications entre tâches, par ex. le service d'impression)

Modification des droits : **chmod**

La commande **chmod** (*change mode*) permet de modifier les droits existant sur les fichiers¹. Elle s'utilise de deux manières :

- soit en spécifiant explicitement les droits :

```
chmod [u g o a] [+ - =] [r w x] nom_du_fichier
```

Ex. :

- **chmod u+x** : donne au propriétaire le droit d'exécuter le fichier
- **chmod go-rwx** : enlève tous les droits aux utilisateurs autres que le propriétaire
- **chmod a+r** : donne le droit en lecture à tout le monde

1. Et bien sûr les dossiers (qui sous Linux sont aussi des fichiers). On peut utiliser l'option **-R** pour appliquer des modifications récursivement au contenu d'un dossier.

Modification des droits : **chmod**

- soit en fournissant une **valeur numérique en octal** correspondant aux droits de chacun :



Ex. :

- **chmod 541 fichier** \Leftrightarrow **chmod u=rx,g=r,o=x fichier**
- **chmod 644 fichier** : droit traditionnellement donné aux fichiers
- **chmod 755 dossier** : droit traditionnellement donné aux dossiers

La commande **stat** permet de récupérer les droits sur un fichier :

```
$ stat -c %A fichier
-r-xr---x
$ stat -c %a fichier
541
```


Contient les valeurs par défaut des programmes `adduser`, `addgroup`, `deluser`, `delgroup`.


Par exemple, la variable `DIR_MODE` indique quelles permissions sont attribuées à un dossier lors de sa création.

⇒ `0755` par défaut : tout le monde peut lister le contenu des dossiers personnels des utilisateurs

⇒ on remplace cette valeur par exemple par `0700`

Droits par défaut : **umask**

Un **umask** (*user file creation mode mask*) permet de définir les **droits par défaut** d'un nouveau fichier / dossier.

 Ce masque est une valeur (en octal) qui indique les droits que **n'aura pas** le fichier / dossier.

```
$ umask
0022
$ umask -S      # sous forme symbolique
u=rwx,g=rx,o=rx
$ touch nouveau_fichier
$ ll nouveau_fichier
-rw-r--r-- 1 user user 0 janv. 15 11:53 nouveau_fichier
```

En interne, le masque est appliqué par une fonction **AND NOT** sur la valeur **666** pour un fichier, et **777** pour un dossier.

Droits par défaut : **umask**

La commande **umask** permet aussi de modifier la valeur du masque pour la durée de la session :

```
$ umask 007
$ umask -S
u=rwx,g=rwx,o=
$ touch nouveau_fichier
$ ll nouveau_fichier
-rw-rw---- 1 user user 0 janv. 15 11:53 nouveau_fichier
$ umask g-rw
$ umask -S
u=rwx,g=x,o=
$ touch nouveau_fichier2
$ ll nouveau_fichier2
-rw----- 1 user user 0 janv. 15 11:53 nouveau_fichier2
```

Sticky bit

Tous les utilisateurs ont la permission **w** sur le dossier **/tmp** : en théorie, tout le monde peut donc écrire et supprimer n'importe quel fichier à l'intérieur! Cependant, quand on affiche les droits sur ce dossier, on remarque une permission inhabituelle :

```
$ stat -c %A /tmp  
drwxrwxrwt
```

Ce **t** est un bit particulier appelé **sticky bit**; il indique que **dans ce dossier¹, seuls le propriétaire d'un fichier, le propriétaire du dossier ou **root** ont le droit de renommer ou supprimer ce fichier**

1. Le sticky bit peut aussi être appliqué aux fichiers "ordinaires". Cependant cette utilisation est désuète et ne sera pas abordée ici

Comme les autres droits, le sticky bit se modifie avec **chmod** :

```
chmod +t dossier # active le sticky bit
```

On peut aussi utiliser la notation octale. Dans ce cas, on a besoin d'un **quatrième chiffre octal** pour indiquer cette permission supplémentaire¹ :

- **chmod 1777 dossier # active le sticky bit**
- **chmod 0777 dossier # désactive le sticky bit**

1. On verra plus loin qu'on peut donner d'autres valeurs à ce premier chiffre

Changement de propriétaire ou de groupe : **chown** et **chgrp**

La commande **chown** (*change owner*) permet de modifier le propriétaire d'un fichier / dossier :

```
chown [-R] utilisateur cible
```

La commande **chgrp** (*change group*) permet de modifier le groupe d'un fichier / dossier :

```
chgrp [-R] groupe cible
```

💡 On peut aussi utiliser le raccourci :

```
chown [-R] utilisateur:groupe cible
```

💡 Seul **root** a le droit d'utiliser **chown**, mais tout utilisateur peut utiliser **chgrp** sur un fichier qu'il possède et s'il appartient au nouveau groupe

Compléments sur les droits

Le programme `nano` appartient à `root`, mais n'importe qui a le droit de l'exécuter :

```
$ ll /usr/bin/nano  
-rwxr-xr-x root root /usr/bin/nano
```

Si je lance ce programme, avec quels droits s'exécute-t-il ?

💡 Un programme s'exécute avec les droits de l'utilisateur **qui l'a lancé**.

⇒ Par conséquent, un utilisateur `lambda` ne peut pas utiliser `nano` pour lire / modifier des fichiers sur lesquels il n'a pas les droits.

Droits d'endossement

Sauf que... parfois, on aimerait que ce soit le cas !

Exemple

Tout utilisateur doit pouvoir utiliser la commande **passwd** pour modifier son mot de passe. Mais ce programme a besoin d'écrire dans les fichiers **/etc/passwd** et **/etc/shadow**, qui appartiennent à **root** !

"Solution 1" : utiliser **sudo** ? \Rightarrow on ne peut pas donner les droits **sudo** à tous les utilisateurs !

Solution 2 : permettre aux utilisateurs d'exécuter ce programme comme s'ils étaient **root** : c'est le rôle du droit **setuid** (*Set User ID*) :

```
$ stat -c %A /usr/bin/passwd  
-rwsr-xr-x
```

Ce **s** indique que ce programme est exécuté avec les droits de son propriétaire.

Droits d'endossement

Il existe un droit équivalent pour les groupes, appelé **setgid** (*Set Group ID*)

💡 Comme les autres droits, les droits d'endossement se modifient avec **chmod** :

- **chmod u+s fichier** # pour activer le setuid
- **chmod g+s fichier** # pour activer le setgid

💡 La valeur du **setuid** est 4 et la valeur du **setgid** est 2, on peut aussi utiliser la notation octale sur le 4^è **chiffre** (le même que le sticky bit) :

- **chmod 4755 fichier** # pour activer le setuid
- **chmod 6755 fichier** # pour activer les setuid et setgid

Droits d'endossement

💡 Si un fichier / dossier est setuid ou setgid, il y a un **s** à la place du droit d'exécution¹ :

```
$ stat -c %A /usr/bin/passwd  
-rwsr-xr-x
```

💡 "setuid-er" un dossier n'a aucun effet; en revanche, un dossier peut être "setgid-é" : tous les fichiers ou sous-dossiers créés à l'intérieur appartiennent alors automatiquement au même groupe que le dossier (pratique pour les dossiers partagés)

💡 Note : pour des raisons de sécurité, il est impossible de "setuid-er" un script Bash²

1. s minuscule si le fichier est exécutable, majuscule sinon

2. Voir par exemple <http://www.faqs.org/faqs/unix-faq/faq/part4/section-7.html>

Attributs supplémentaires

On peut définir un certain nombre d'attributs supplémentaires sur un fichier :

- il est automatiquement compressé après chaque écriture
- il ne doit pas être exporté en cas de sauvegarde du système
- il est immuable (= ne peut être modifié)
- il peut être récupéré après suppression ¹

Ces attributs se modifient avec **chattr**. Par exemple, la commande suivante empêche quiconque, **même root**, de supprimer un fichier :

```
$ sudo chattr +i toto ; sudo rm toto  
rm: cannot remove 'toto': Operation not permitted
```

On peut lister les attributs avec la commande **lsattr** :

```
$ lsattr toto  
----i-----e---- toto
```

1. Certaines de ces options ne sont pas disponibles avec ext4

La gestion des droits sous Linux peut s'avérer limitée, notamment en entreprise.

Exemple

Plusieurs utilisateurs doivent accéder à une ressource partagée alors qu'ils n'ont pas de groupe commun.

Ces situations nécessitent diverses astuces lourdes à mettre en œuvre et à entretenir, comme la création de groupes intermédiaires.

Les **ACL** permettent de pallier ce manque grâce à une gestion avancée des droits.

Pour en savoir plus : <https://doc.ubuntu-fr.org/acl>