Gestion des Permissions et des Utilisateurs sous Linux

Ilias TOUGUI

L'Ecole Supérieure d'Informatique et du Numérique

INF2132 - Systèmes d'Exploitation

PLAN DU COURS

1. Introduction à la sécurité Linux
2. Gestion des utilisateurs
3. Gestion des groupes
4. Décoder les permissions
5. Modifier les permissions
6. Récapitulatif

OBJECTIFS DU COURS

Ce que vous allez apprendre aujourd'hui

À la fin de ce cours, vous serez capable de :

- **Comprendre** le système de sécurité Linux basé sur utilisateurs et groupes
- **Décoder** les permissions rwx des fichiers et répertoires
- **Manipuler** les comptes utilisateurs et les groupes via la ligne de commande
- **Modifier** les permissions avec chmod, chown et chgrp

Prérequis

Notions de base de la ligne de commande Linux (ls, cd, mkdir, touch).

1. Introduction à la sécurité Linux

POURQUOI LA SÉCURITÉ DES FICHIERS ?

Contexte et problématique

Un système sans sécurité permet à n'importe qui de lire, modifier ou supprimer des fichiers sensibles.

Exemples de risques :

- **Confidentialité :** Documents privés accessibles par tous les utilisateurs
- **Intégrité :** Modification ou suppression accidentelle de fichiers système
- **Disponibilité :** Altération de services critiques par des utilisateurs non autorisés

Solution Linux

Linux utilise un système de permissions héritées d'Unix : chaque fichier/répertoire a un propriétaire, un groupe et des permissions d'accès granulaires.

COMPTES UTILISATEURS LINUX

Identification et authentification

Chaque personne accédant à un système Linux doit avoir un **compte utilisateur unique**.

Composants d'un compte :

- **Nom de connexion (login) :** Chaîne alphanumérique
- **UID (User ID) :** Valeur numérique unique attribuée au compte
- **Mot de passe :** Pour l'authentification (stocké de façon chiffrée)
- **Répertoire personnel (HOME) :** Espace de travail de l'utilisateur
- **Shell par défaut :** Programme interpréteur de commandes (ex: /bin/bash)

À retenir

Les permissions d'accès aux objets système dépendent de l'UID du compte connecté.

LE FICHIER /etc/passwd

Base de données des comptes

Le système Linux utilise un fichier spécial pour associer le nom de connexion à l'UID correspondant. Ce fichier est le fichier /**etc**/**passwd**. Il contient plusieurs informations sur l'utilisateur. Voici à quoi ressemble un fichier /**etc**/**passwd** typique sur un système Linux.

```
$ 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
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
...
rich:x:500:500:Rich Blum:/home/rich:/bin/bash
ilias:x:1000:1000:Ilias Tougui:/home/ilias:/bin/bash
```

LE FICHIER /etc/passwd

Base de données des comptes

Le fichier `/etc/passwd` associe chaque nom de connexion à son UID et contient les informations du compte. Chaque compte est structuré sur une ligne divisée en 7 champs séparés par des caractères ":".

```
login:x:UID:GID:commentaire:home:shell
```

Détail des champs pour l'utilisateur ilias:

- **login :** Nom d'utilisateur (ilias)
- **x :** Indicateur de mot de passe (voir /etc/shadow)
- **UID :** 1000 (identifiant numérique unique)
- **GID :** 1000 (groupe principal)
- **Commentaire :** "Ilias Tougui" (nom complet ou description)
- **Répertoire HOME :** /home/ilias
- **Shell :** /bin/bash

LE COMPTE ROOT ET LES COMPTES SYSTÈME

Administration et sécurité

Le compte root :

- **Administrateur** du système Linux
- **Privilèges illimités** sur tous les fichiers et processus
- **Compte unique** : un seul compte avec UID 0 doit exister

Les comptes système :

- **Définition :** Comptes spéciaux pour les services en arrière-plan (daemon, apache, mysql, sshd, etc.)
- **Rôle :** Permettent aux services d'accéder aux ressources système
- **Pas d'utilisateurs réels :** Aucune personne ne se connecte avec ces comptes

CLASSIFICATION DES UTILISATEURS

Trois catégories distinctes

Linux distingue trois types de comptes selon leur rôle et leur UID :

1. **Le superutilisateur (root) :**
   - **UID = 0** (toujours et uniquement)
   - Privilèges illimités sur le système entier. Utilisé exclusivement pour l'administration critique
2. **Les utilisateurs système :**
   - **UID : 1-999** (selon les distributions)
   - Comptes pour les services (apache, mysql, sshd, etc.)
3. **Les utilisateurs standards :**
   - **UID $\geq$ 1000**
   - Comptes pour les utilisateurs humains
   - Accès limité à leur espace personnel

ATTENTION : DEUX TYPES DE "SERVICES"

1. **Services du système d'exploitation (OS Services) :**
   - **Définition :** Fonctionnalités fondamentales fournies par le noyau Linux
   - **Exemples :** Gestion des fichiers, exécution de programmes, allocation mémoire...
   - **Rôle :** Composants internes du système d'exploitation
   - **Localisation :** Dans le noyau (kernel) et les bibliothèques système
2. **Programmes serveurs (Daemons) :**
   - **Définition :** Applications qui s'exécutent en arrière-plan
   - **Exemples :** Apache (serveur web), sshd (connexions à distance)
   - **Rôle :** Programmes applicatifs qui *utilisent* les services OS
   - **Exécution :** Via des **comptes système** dédiés (www-data, mysql, sshd)

À retenir

Les comptes système servent à exécuter les programmes serveurs (daemons), pas les services OS !

DISTINCTION : SERVICES OS vs COMPTES SYSTÈME

Scénario : Un serveur web Apache répond aux requêtes HTTP

Étape 1 : Le programme Apache s'exécute

```
$ ps aux | grep apache2
www-data  1234  0.1  2.3  /usr/sbin/apache2
```

- Apache tourne avec le **compte système** www-data (UID < 1000)
- Ce n'est pas un service OS, c'est un **daemon** (programme serveur)

Étape 2 : Apache utilise les services OS

- Service OS **File System** : pour lire les fichiers HTML
- Service OS **I/O operations** : pour envoyer les données réseau
- Service OS **Resource allocation** : pour obtenir mémoire et CPU

Conclusion : Les comptes système permettent d'exécuter des programmes qui utilisent les services OS.

ÉLÉVATION DE PRIVILÈGES : SUDO

Exécuter des commandes en tant que root

Problème : Les utilisateurs standards ne peuvent pas effectuer des tâches d'administration.

Solution : La commande **sudo** (superuser do / substitute user do) permet d'exécuter temporairement une commande avec les privilèges d'un autre utilisateur (généralement root).

Exemple :

```
$ cd /
$ mkdir TPx
mkdir: cannot create directory ‘TPx’: Permission denied

$ sudo mkdir TPx
[sudo] password for ilias:
```

Important : sudo demande *votre* mot de passe, pas celui de root.

SUDO : OPTIONS COURANTES

Utilisation avancée

Commandes sudo courantes :

```
$ sudo <commande>              # Exécuter une commande en tant que root
$ sudo -i                      # Ouvrir un shell root complet
$ sudo -u amine script.sh      # Exécuter en tant qu'amine
$ sudo -l                      # Lister vos privilèges sudo
```

Avantages de sudo :

- **Traçabilité :** Toutes les actions sudo sont enregistrées dans /var/log/auth.log
- **Sécurité :** Évite la connexion directe en tant que root
- **Contrôle granulaire :** Le fichier /etc/sudoers définit qui peut exécuter quoi
- **Audit :** Permet d'identifier précisément qui a fait quoi et quand

BONNES PRATIQUES : PRINCIPE DE MOINDRE PRIVILÈGE

Sécurité et administration

Règles à suivre :

1. **Ne jamais travailler directement en tant que root**
   - Utilisez votre compte personnel et sudo pour les tâches admin
   - Une erreur en tant que root peut détruire le système
2. **Utiliser sudo pour des commandes ponctuelles**
   - Évitez `sudo -i` sauf si nécessaire
   - Préférez `sudo <commande>` pour chaque action
3. **Vérifier avant d'exécuter**
   - Les commandes sudo ont un pouvoir destructeur
   - Exemple dangereux : `sudo rm -rf /*` (ne JAMAIS faire !)

Attention

Avec de grands pouvoirs viennent de grandes responsabilités !

ÉVOLUTION DE LA SÉCURITÉ

Pourquoi un compte par service ?

Ancienne approche (avant les années 2000) :

- Tous les services s'exécutaient avec le compte **root**
- **Problème majeur :** Si un service est compromis, l'attaquant obtient immédiatement les privilèges root complets
- Accès total au système, tous les fichiers, tous les processus

Approche moderne (isolation des privilèges) :

- **Principe :** Chaque service a son propre compte utilisateur dédié
- **Avantage :** Si un service est compromis, l'attaquant n'a que les privilèges limités de ce compte
- **Confinement :** L'attaquant ne peut pas accéder au reste du système
- **Traçabilité :** Chaque action est attribuée à un compte spécifique

LE CHAMP MOT DE PASSE : ÉVOLUTION

De /etc/passwd à /etc/shadow

Le caractère **"x"** dans le deuxième champ ne signifie PAS que tous les comptes ont le même mot de passe !

Ancienne approche (avant les années 1990) :

- Les mots de passe **chiffrés** étaient stockés directement dans /etc/passwd
- **Problème :** Le fichier /etc/passwd doit être lisible par tous les utilisateurs (beaucoup de programmes en ont besoin pour convertir UID $\rightarrow$ nom)
- **Vulnérabilité :** Avec des logiciels de cassage de mots de passe de plus en plus puissants, les attaquants pouvaient lire le fichier et tenter de décrypter les mots de passe
- Les développeurs Linux ont dû repenser cette architecture

LA SOLUTION : LE FICHIER SHADOW

Séparation des informations sensibles

Architecture moderne (depuis les années 1990) :

- Les mots de passe sont maintenant dans **/etc/shadow**
- Le fichier /etc/passwd contient uniquement "x" pour indiquer que le mot de passe est ailleurs
- **Sécurité renforcée :** Seul root peut lire /etc/shadow
- Seuls les programmes spéciaux (login, passwd, sudo) ont accès à ce fichier

À retenir

/etc/passwd est lisible par tous, /etc/shadow est accessible uniquement à root. Le "x" est un simple indicateur de redirection vers shadow.

LE FICHIER /etc/shadow

Gestion sécurisée des mots de passe

Exemple de ligne :

```
ilias:$6$5eLcH2dfVcyVRpuObKY6fI1:20308:0:99999:7:::
```

Structure (9 champs) :

- **Login :** Nom correspondant à /etc/passwd
- **Mot de passe chiffré :** Algorithme moderne (SHA, bcrypt, etc.)
- **Dernier changement :** Nombre de jours depuis 1970-01-01
- **Jours min :** Délai minimal entre deux changements
- **Jours max :** Expiration du mot de passe
- **Avertissement :** Jours avant expiration pour alerter l'utilisateur
- **Inactivité :** Jours après expiration avant verrouillage du compte
- **Expiration compte :** Date de désactivation du compte
- **Réservé :** Pour usage futur

IDENTIFIER L'UTILISATEUR COURANT

Commandes whoami et id

Qui suis-je ?

```
$ whoami
ilias
```

Affiche simplement le nom de connexion de l'utilisateur actuel.

Informations complètes

```
$ id
uid=1000(ilias) gid=1000(ilias)
groups=1000(ilias),4(adm),24(cdrom),
27(sudo),30(dip),46(plugdev),
100(users),114(lpadmin)
```

Affiche l'UID, le GID et tous les groupes auxquels appartient l'utilisateur.

PAUSE RÉFLEXION

Vérification de compréhension

Questions rapides :

1. Quelle commande permet de visualiser votre UID actuel ?
2. Pourquoi le champ mot de passe de /etc/passwd contient-il "x" ?
3. Quel est l'UID du compte root ?

2. Gestion des utilisateurs

GESTION DES UTILISATEURS

Commandes d'administration

Commandes principales :

- **useradd :** Créer un nouveau compte utilisateur
- **usermod :** Modifier un compte existant
- **userdel :** Supprimer un compte
- **passwd :** Changer le mot de passe
- **chage :** Gérer l'expiration des mots de passe
- **chsh, chfn :** Modifier le shell et les informations utilisateur

Important

Seul le compte root (ou via sudo) peut effectuer ces opérations d'administration.

CRÉER UN UTILISATEUR : useradd

Valeurs par défaut

Paramètres par défaut:

```
$ useradd -D
```

- **GROUP=100 :** Le nouvel utilisateur est ajouté à un groupe commun avec GID 100
- **HOME=/home :** Le répertoire personnel sera créé dans /home/nom_utilisateur
- **INACTIVE=-1 :** Le compte ne sera pas désactivé automatiquement après expiration du mot de passe
- **EXPIRE= :** Aucune date d'expiration du compte n'est définie par défaut
- **SHELL=/bin/bash :** Le shell bash est utilisé par défaut
- **SKEL=/etc/skel :** Le système copie le contenu de /etc/skel dans le répertoire HOME de l'utilisateur
- **CREATE_MAIL_SPOOL=yes :** Un fichier de messagerie est créé pour recevoir les mails

CRÉER UN UTILISATEUR : useradd

Options

Commande de base :

```
$ useradd -m test
```

Options utiles :

- **-m :** Créer le répertoire HOME (recommandé)
- **-c "commentaire" :** Ajouter une description
- **-s /bin/zsh :** Définir un shell
- **-g groupe :** Spécifier le groupe principal
- **-G groupes :** Ajouter à des groupes secondaires
- **-e YYYY-MM-DD :** Date d'expiration du compte

EXEMPLE : CRÉATION D'UN UTILISATEUR

Étapes détaillées

Créer un compte pour "amine", développeur, avec shell zsh et expiration le 31/12/2025.

Commande complète :

```
$ sudo useradd -m -c "Amine, Dev" -s /bin/zsh \
        -e 2025-12-31 amine
```

Vérification :

```
& grep amine /etc/passwd
amine:x:1001:1001:Ahmed, Dev:/home/amine:/bin/bash
```

Définir le mot de passe :

```
$ passwd amine
```

MODIFIER UN UTILISATEUR : usermod

Changements post-création

Syntaxe :

```
usermod [options] login
```

Options principales :

- **-c "nouveau commentaire" :** Modifier la description
- **-d /nouveau/home :** Changer le répertoire HOME
- **-s /bin/tcsh :** Changer le shell
- **-l nouveau_login :** Renommer le compte
- **-L :** Verrouiller le compte (lock)
- **-U :** Déverrouiller le compte (unlock)
- **-g groupe :** Changer le groupe principal
- **-G groupe1,groupe2 :** Modifier les groupes secondaires
- **-a -G groupe :** Ajouter un groupe sans écraser les autres

SUPPRIMER UN UTILISATEUR : userdel

Nettoyage et précautions

Syntaxe de base :

```
$ userdel username        # Supprime le compte seulement
$ userdel -f username     # Supprime par force
```

Comportement par défaut (sans -f) :

- Supprime l'entrée dans /etc/passwd et /etc/shadow
- **Conserve** le répertoire HOME et les fichiers
- Les fichiers appartiennent maintenant à l'UID orphelin

Précaution importante

Avant de supprimer un compte avec -f, vérifiez qu'aucun fichier important n'est dans le HOME. Dans un environnement partagé, archivez les données avant suppression.

CHANGER LES MOTS DE PASSE : passwd

Gestion par utilisateur et administrateur

Utilisation par l'utilisateur :

```
$ passwd                # Change son propre mot de passe
```

Le système demande l'ancien mot de passe puis le nouveau (deux fois).

Utilisation par root :

```
$ passwd amine          # Change le mot de passe d'amine
```

Root n'a pas besoin de connaître l'ancien mot de passe.

Options utiles :

- **-l username :** Verrouiller le compte (lock)
- **-u username :** Déverrouiller le compte (unlock)
- **-e username :** Forcer le changement au prochain login
- **-d username :** Supprimer le mot de passe (connexion sans mot de passe !)

AUTRES UTILITAIRES UTILISATEURS

chsh, chfn, chage

chsh : Changer le shell

```
$ sudo chsh -s /bin/zsh amine
```

Le shell doit être listé dans /etc/shells.

chfn : Modifier les informations

```
$ sudo chfn ilias
```

Ces informations sont stockées dans le champ commentaire de /etc/passwd.

chage : Gérer l'expiration du mot de passe

```
$ chage -E 2025-12-31 amine   # Expiration du compte
$ chage -M 90 amine       # Mot de passe valide 90 jours
$ chage -W 7 amine        # Avertir 7 jours avant expiration
```

3. Gestion des groupes

INTRODUCTION AUX GROUPES LINUX

Partage de ressources entre utilisateurs

Limitation des comptes utilisateurs individuels :

- Les comptes utilisateurs sont excellents pour contrôler la sécurité **individuelle**
- **Problème :** Ils ne permettent pas facilement de partager des ressources entre plusieurs utilisateurs
- **Exemple :** Comment permettre à 5 développeurs de modifier les mêmes fichiers de projet ?

Solution : Les groupes

- Un **groupe** permet à plusieurs utilisateurs de partager un ensemble commun de permissions
- S'applique aux objets système : fichiers, répertoires, périphériques
- Simplifie la gestion des permissions pour des équipes ou projets

GROUPES PRIMAIRE ET SECONDAIRE

Comprendre la différence

Définitions :

- **Groupe primaire :** Groupe principal obligatoire assigné à chaque utilisateur
- **Groupe secondaire :** Groupes supplémentaires optionnels pour le partage de ressources

Caractéristiques principales :

- Un utilisateur possède **un seul** groupe primaire
- Un utilisateur peut appartenir à **jusqu'à 15** groupes secondaires
- Les nouveaux fichiers créés héritent du groupe primaire
- Groupe primaire : stocké dans `/etc/passwd`
- Groupes secondaires : stockés dans `/etc/group`

GROUPES PRIMAIRE ET SECONDAIRE

Comprendre la différence

Exemple : Afficher les groupes d'un utilisateur

```
$ id ilias
uid=1001(ilias) gid=1001(ilias)
groups=1001(ilias),27(sudo),1002(dev)
```

Ici, **ilias** (gid=1001) est le groupe primaire. Les groupes **sudo** et **dev** sont des groupes secondaires.

STRUCTURE DES GROUPES

GID, nom de groupe et stratégies de distribution

Identifiants de groupe :

- Chaque groupe possède un **GID** (Group ID) unique, similaire aux UID pour les utilisateurs
- Chaque groupe a également un **nom de groupe** unique et lisible
- Le système utilise le GID en interne, le nom sert à l'administration

Gestion des groupes :
Des utilitaires permettent de créer et gérer vos propres groupes :

- **groupadd**: Créer un nouveau groupe
- **groupmod**: Modifier un groupe
- **groupdel**: Supprimer un groupe

LE FICHIER /etc/group

Base de données des groupes

Structure du fichier :

Chaque ligne définit un groupe avec 4 champs séparés par " :".

```
nom_groupe:x:GID:liste_membres
```

Exemples:

```
$ cat \etc\group
root:x:0:root
adm:x:4:syslog,ilias
sudo:x:27:ilias
...
ilias:x:1000:
amine:x:1001:
```

LE FICHIER /etc/group

Base de données des groupes

Exemples:

```
$ cat \etc\group
root:x:0:root
adm:x:4:syslog,ilias
sudo:x:27:ilias
...
ilias:x:1000:
amine:x:1001:
```

Explications :

- **nom_groupe :** Nom textuel du groupe
- **x :** Mot de passe de groupe (rarement utilisé)
- **GID :** Identifiant numérique unique
- **liste_membres :** Utilisateurs appartenant au groupe (séparés par des virgules)

CRÉER UN GROUPE : groupadd

Syntaxe et exemple

Commande de base :

```
$ groupadd nom_groupe
```

Le système attribue automatiquement le prochain GID disponible ($\geq$ 500 ou 1000).

Options utiles :

- **-g GID :** Spécifier un GID explicite
- **-r :** Créer un groupe système (GID < 500)

Exemple :

```
$ sudo groupadd shared
$ tail -1 /etc/group
shared:x:1002:
```

CRÉER UN GROUPE : groupadd

Syntaxe et exemple

Exemple :

```
$ sudo groupadd shared
$ tail -1 /etc/group
  shared:x:1002:
```

Le groupe "shared" est créé avec GID 1002, mais **aucun membre** pour l'instant.

Ajouter des membres ensuite :

```
$ sudo usermod -a -G shared ilias
$ sudo usermod -a -G shared amine
$ tail -1 /etc/group
  shared:x:1002:ilias,amine
```

MODIFIER UN GROUPE : groupmod

Renommer ou changer le GID

Syntaxe :

```
groupmod [options] nom_groupe
```

Options :

- **-n nouveau_nom :** Renommer le groupe
- **-g nouveau_GID :** Changer le GID

Exemple : Renommer un groupe

```
$ sudo groupmod -n dev shared
$ grep dev /etc/group
dev:x:1002:ilias,amine
```

Le groupe "shared" est maintenant nommé "dev". Les membres et le GID restent inchangés.

SUPPRIMER UN GROUPE : groupdel

Supprimer un groupe secondaire

Syntaxe :

```
groupdel [options] nom_groupe
```

Options principales :

- **-f, –force :** Force la suppression même si le groupe est encore utilisé
- **-h, –help :** Affiche l'aide et quitte

Exemple : Supprimer un groupe secondaire

```
$ sudo groupdel dev
$ groups amine
```

Le groupe secondaire "dev" est supprimé. Les utilisateurs amine et ilias conservent leurs groupes primaires respectifs.

4. PERMISSIONS DES FICHIERS

AFFICHAGE DES PERMISSIONS : ls -l

Comprendre la première colonne

Commande :

```
$ ls -l
-rw-rw-r-- 1 ilias dev  1024 Oct 21 14:30 document.txt
drwxr-xr-x 2 ilias dev  4096 Oct 21 14:35 projet/
```

Décomposition de "-rw-rw-r--" :

- **1er caractère :** Type d'objet
  - **-** = fichier régulier
  - **d** = répertoire (directory)
  - **l** = lien symbolique (link)
- 3 caractères suivants (rw-) : Permissions du propriétaire (owner)
- 3 suivants (rw-) : Permissions du groupe (group)
- 3 derniers (r--) : Permissions pour les autres (others)

SIGNIFICATION DES SYMBOLES rwx

Lecture, écriture, exécution

Pour un fichier régulier :

- **r (read) :** Lecture du contenu du fichier
- **w (write) :** Modification ou suppression du fichier
- **x (execute) :** Exécution du fichier comme programme/script

Pour un répertoire :

- **r :** Lister le contenu du répertoire (ls)
- **w :** Créer, supprimer, renommer des fichiers dans le répertoire
- **x :** Traverser le répertoire (cd) et accéder aux fichiers

Symbole "-" :
Un tiret indique que la permission est **refusée**.

PAUSE RÉFLEXION

Exercice rapide

Décoder les permissions suivantes :

1. -rwxr-xr-x (fichier)
2. drwxrwx-- (répertoire)
3. -rw------- (fichier)
4. drwxr-xr-x (répertoire)

PAUSE RÉFLEXION

Exercice rapide

Décoder les permissions suivantes :

1. -rwxr-xr-x (fichier)
2. drwxrwx-- (répertoire)
3. -rw------- (fichier)
4. drwxr-xr-x (répertoire)

Réponses :

1. Propriétaire : lecture, écriture, exécution. Groupe et autres : lecture et exécution.
2. Propriétaire et groupe : accès complet. Autres : aucun accès.
3. Propriétaire : lecture et écriture. Groupe et autres : aucun accès.
4. Propriétaire : accès complet. Groupe et autres : lecture et traversée uniquement.

LES TROIS NIVEAUX DE SÉCURITÉ

Propriétaire, groupe, autres

Hiérarchie des permissions :

1. **Propriétaire (owner/user) :** La personne qui a créé le fichier (ou à qui il a été transféré)
2. **Groupe (group) :** Tous les membres du groupe propriétaire du fichier
3. **Autres (others/world) :** Tous les autres utilisateurs du système

Le système vérifie d'abord si vous êtes le propriétaire, puis si vous êtes dans le groupe, sinon vous êtes "autres".

Exemple :

- amine possède le fichier "rapport.txt" avec permissions `-rw-r------`
- amine peut lire et écrire (rw-)
- Les membres du groupe "dev" peuvent lire (r--)
- Les autres utilisateurs ne peuvent rien faire (---)

MODE OCTAL DES PERMISSIONS

Représentation numérique

Principe : Chaque triplet rwx est représenté par un chiffre octal (0-7).

Conversion binaire → octal :

Symbole	Binaire	Octal
---	000	0
--x	001	1
-w-	010	2
-wx	011	3
r--	100	4
r-x	101	5
rw-	110	6
rwx	111	7

MODE OCTAL DES PERMISSIONS

Représentation numérique

Principe : Chaque triplet rwx est représenté par un chiffre octal (0-7).

Exemple : -rw-r–r--

- Fichier: -
- Propriétaire : rw- = 6
- Groupe : r-- = 4
- Autres : r-- = 4
- **Mode octal : 644**

Exemple : d----w-r--

- Repertoire: d
- Propriétaire : --- = 0
- Groupe : -w- = 2
- Autres : r-- = 4
- **Mode octal : 024**

MODES OCTAUX COURANTS

Usages typiques

Fichiers :

- **644** (-rw-r--r--) : Fichier texte lisible par tous, modifiable par le propriétaire
- **600** (-rw-------) : Fichier privé (ex: clés SSH, mots de passe)
- **755** (-rwxr-xr-x) : Script exécutable par tous
- **700** (-rwx------) : Script/programme privé

Répertoires :

- **755** (drwxr-xr-x) : Répertoire accessible par tous, modifiable par propriétaire
- **750** (drwxr-x---) : Répertoire accessible au groupe, invisible aux autres
- **700** (drwx------) : Répertoire entièrement privé
- **775** (drwxrwxr-x) : Répertoire partagé en écriture avec le groupe

À éviter

777 (rwxrwxrwx) : Accès complet pour tout le monde ! Risque de sécurité majeur.

UMASK : LE PROBLÈME

Permissions par défaut des nouveaux fichiers

Lorsqu'un utilisateur crée un fichier, le système lui attribue des permissions par défaut.

Exemple :

```
$ touch newfile
$ ls -al newfile
-rw-rw-r-- 1 ilias ilias 0 Oct 21 22:56 newfile
```

Observation : Le fichier est lisible par tous les utilisateurs du système !

Question : Comment contrôler ces permissions par défaut pour tous les fichiers futurs ?

UMASK : LA SOLUTION

Masquer les permissions non désirées

Le **umask** définit un masque pour supprimer certaines permissions par défaut.

Afficher le umask actuel :

```
$ umask
0002
```

Signification :

- Premier "0" : bits spéciaux (SUID/SGID/sticky)
- "002" : masque octal (user, group, other)

Utilité : En environnement multi-utilisateurs, umask assure que les fichiers créés ont des permissions sécurisées par défaut.

UMASK : CALCUL DES PERMISSIONS

Soustraire le masque des permissions maximales

Formule : Permissions finales = Permissions maximales - umask

Permissions maximales par défaut :

- Fichiers : 666 (rw-rw-rw-)
- Répertoires : 777 (rwxrwxrwx)

Exemple avec umask = 0002 :

```
Fichier : 666 - 002 = 664 (rw-rw-r--)
Répertoire : 777 - 002 = 775 (rwxrwxr-x)
```

Résultat : Les autres (other) ne peuvent que lire, pas écrire.

AFFICHER LES PERMISSIONS EN MODE OCTAL

La commande stat

La commande **stat** affiche des informations détaillées sur un fichier, y compris les permissions en mode octal.

Afficher toutes les informations :

```
$ stat newfile
File: newfile
Size: 0         Blocks: 0          IO Block: 4096
Access: (0664/-rw-rw-r--)  Uid: (1000/  ilias)
Access: 2025-10-21 22:56:12.123456789 +0100
Modify: 2025-10-21 22:56:12.123456789 +0100
```

Afficher uniquement le mode octal :

```
$ stat -c '%a' newfile
664
```

Utile pour vérifier rapidement les permissions numériques d'un fichier.

MODIFIER LE UMASK

Personnalisation

Modifier temporairement (session en cours):

```
$ umask 027
$ touch test_file
$ ls -l test_file
-rw-r----- 1 ilias ilias 0 Oct 21 15:00 test_file
```

Permissions : 666 - 027 = 640 (rw-r-----)

Modifier définitivement : Ajouter la ligne `umask 027` dans :

- `~/.bashrc` ou `~/.profile` (pour l'utilisateur)
- `/etc/profile` ou `/etc/bash.bashrc` (pour tous les utilisateurs)

Valeurs umask courantes :

- **022 :** Fichiers rw-r--r--, répertoires rwxr-xr-x (par défaut)
- **027 :** Fichiers rw-r-----, répertoires rwxr-x--- (groupe seul peut lire)
- **077 :** Fichiers rw-------, répertoires rwx------ (totalement privé)

5. Modifier les permissions

CHMOD : MODE OCTAL

Syntaxe et exemples

Syntaxe :

```
chmod mode fichier
```

Exemples :

```
$ chmod 644 document.txt    # rw-r--r--
$ chmod 755 script.sh       # rwxr-xr-x
$ chmod 700 secret.key      # rwx------
```

L'option **-R** applique les permissions à tous les fichiers et sous-répertoires.

```
$ chmod -R 755 mon_repertoire/
```

CHMOD : MODE SYMBOLIQUE

Syntaxe [ugoa][+-=][rwx]

Format :

```
chmod [qui][opération][permissions] fichier
```

Qui :

- **u** = user (propriétaire)
- **g** = group
- **o** = others
- **a** = all (u+g+o)

Opération :

- **+** : Ajouter une permission
- **-** : Retirer une permission
- **=** : Définir exactement ces permissions (remplace les autres)

CHMOD SYMBOLIQUE : EXEMPLES

Utilisation pratique

Ajouter l'exécution pour le propriétaire :

```
$ chmod u+x script.sh
$ ls -l script.sh
-rwxr--r-- 1 ilias dev 256 Oct 21 15:15 script.sh
```

Retirer l'écriture pour le groupe et les autres :

```
$ chmod go-w fichier.txt
-rw-r--r-- 1 ilias dev 1024 Oct 21 15:16 fichier.txt
```

Définir exactement rw pour tous :

```
$ chmod a=rw partage.txt
-rw-rw-rw- 1 ilias dev 512 Oct 21 15:17 partage.txt
```

Ajouter lecture pour tout le monde :

```
$ chmod a+r public.txt
-rw-r--r-- 1 ilias dev 2048 Oct 21 15:18 public.txt
```

CHOWN : CHANGER LE PROPRIÉTAIRE

Syntaxe et exemples

Syntaxe :

```
$ chown [propriétaire][:groupe] fichier
```

Changer uniquement le propriétaire :

```
$ sudo chown amine newfile
$ ls -l newfile
-rw-r--r-- 1 amine ilias 0 Oct 21 15:20 newfile
```

Changer propriétaire ET groupe :

```
$ sudo chown amine:dev newfile
-rw-r--r-- 1 amine dev 0 Oct 21 15:20 newfile
```

Changer uniquement le groupe :

```
$ sudo chown :dev newfile
-rw-r--r-- 1 amine dev 0 Oct 21 15:20 newfile
```

CHGRP : CHANGER LE GROUPE

Alternative à chown pour le groupe

Syntaxe :

```
$ chgrp groupe fichier
```

Exemple :

```
$ chgrp dev rapport.txt
$ ls -l rapport.txt
-rw-r--r-- 1 ilias dev 2048 Oct 21 15:25 rapport.txt
```

Conditions :

- Le propriétaire du fichier peut changer le groupe
- L'utilisateur doit être membre du groupe initial et le groupe cible
- Root peut changer le groupe vers n'importe quel groupe

PAUSE RÉFLEXION

Mini-exercice pratique

Scénario :
Vous avez un fichier `script.sh` avec permissions `-rw-r--r--`.

Objectifs :

1. Rendre le script exécutable pour le propriétaire
2. Retirer tous les droits pour "others"
3. Changer le groupe vers "dev"

Solution attendue :

1. `chmod u+x script.sh`
2. `chmod o-r script.sh`
3. `chgrp dev script.sh`

RÉCAPITULATIF : SÉCURITÉ LINUX

Points clés à retenir

Comptes utilisateurs et groupes :

- Chaque utilisateur a un UID unique et appartient à un ou plusieurs groupes
- Les fichiers /etc/passwd, /etc/shadow, /etc/group gèrent les comptes
- Commandes : useradd, usermod, userdel, groupadd, groupmod

Permissions :

- Trois niveaux : propriétaire, groupe, autres
- Trois types : lecture (r), écriture (w), exécution (x)
- Mode octal (644, 755) et symbolique (u+x, go-w)
- umask définit les permissions par défaut

Modification :

- chmod : changer les permissions
- chown : changer le propriétaire (et le groupe)
- chgrp : changer le groupe uniquement

COMMANDES DE RÉFÉRENCE

Aide-mémoire

Gestion utilisateurs :

- `useradd -m login` : Créer utilisateur avec HOME
- `usermod -a -G groupe login` : Ajouter à un groupe
- `userdel -r login` : Supprimer utilisateur et HOME
- `passwd login` : Changer mot de passe
- `id login` : Afficher UID, GID, groupes

Gestion groupes :

- `groupadd nom` : Créer groupe
- `groupmod -n nouveau ancien` : Renommer
- `groupdel nom` : Supprimer groupe

COMMANDES DE RÉFÉRENCE

Aide-mémoire

Permissions :

- `chmod 755 fichier` : Mode octal
- `chmod u+x,go-w fichier` : Mode symbolique
- `chown user:group fichier` : Changer propriétaire et groupe
- `umask 002` : Définir permissions par défaut