Gestion d’utilisateur sur Linux

Prérequis :

- Avoir une machine virtuelle sous Debian 12

Création et gestion des comptes utilisateurs :

Qu'est-ce qu'un utilisateur sous Linux ?

Un utilisateur est une entité qui peut interagir avec le système d’exploitation

Chaque utilisateur possède également ce qu’on appelle un UID (identifiant unique)

Comment créer un utilisateur ?

La création d'un utilisateur sous Linux se fait généralement via ligne de commande. La commande la plus utilisée pour cela est **useradd**. Voici les étapes basiques :

1. Depuis le terminal ,taper la commande **sudo useradd -m nom_utilisateur** pour créer un nouvel utilisateur avec un répertoire personnel.

```
debian@debian:~$ sudo useradd -m adrien
```

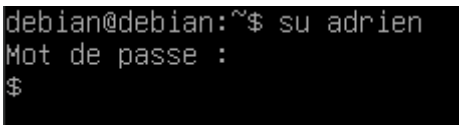
2. Définir un mot de passe pour le nouvel utilisateur avec **sudo passwd nom_utilisateur**.

```
debian@debian:~$ sudo passwd adrien
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
debian@debian:~$
```

Comment changer d'utilisateur sur une console

Pour changer d'utilisateur dans une session terminal, utilisez la commande **su** (substitute user)

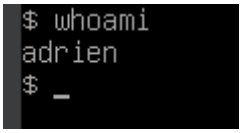
Exemple : **su nom_utilisateur**



vous serez invité à entrer le mot de passe de cet utilisateur, vous permettant de changer de session sans quitter ou fermer la console.

Cette commande est très utile pour tester les droits d'accès ou pour effectuer des opérations administratives sans quitter la session utilisateur actuelle. Pour revenir à votre utilisateur initial, tapez **exit** pour terminer la session de l'utilisateur substitué.

La commande **whoami** permet d'afficher le nom de l'utilisateur actuellement connecté.



Gestion des rôles et des groupes

Rôles des groupes sous Linux

Sous Linux, les groupes servent à organiser les utilisateurs en ensembles pour une gestion simplifiée des permissions. Un groupe peut être attribué à plusieurs utilisateurs, permettant ainsi une administration des droits d'accès collective et efficace. Chaque groupe est doté d'un Identifiant de Groupe Unique (GID) qui lie les utilisateurs du groupe aux ressources et permissions partagées.

Créer et gérer des groupes

Pour créer un groupe sous Linux, la commande utilisée est généralement **groupadd**. Voici les étapes clés pour créer et gérer un groupe :

- **Création** : en ligne de commande, tapez **sudo groupadd [nom_du_groupe]** pour créer un nouveau groupe.

  Exemple : groupadd comptabilite

- **Ajout d'utilisateurs** : pour ajouter un utilisateur à un groupe, utilisez **sudo usermod -a -G [nom_du_groupe] [nom_utilisateur]**.

```
sudo usermod -a -G comptabilite adrien
```

- **Vérification** : vous pouvez lister les membres d'un groupe avec **getent group [nom_du_groupe]**.

```
debian@debian:~$ getent group comptabilite
comptabilite:x:1002:adrien
```

Meilleures pratiques pour attribuer des groupes

Lors de l'attribution de groupes, il faut suivre les bonnes pratiques de sécurité :

- Ajouter des utilisateurs a des groupes seulement si nécessaire
- Revoir périodiquement les appartenances aux groupes de chaque utilisateur.
- Utiliser des groupes système pour les applications spécifiques

Gestion des mots de passe et des clés d'accès

Importance de la sécurité des mots de passe

La sécurité des mots de passe est cruciale pour protéger l'accès aux comptes utilisateurs et aux ressources systèmes.

Politique de mot de passe :

- Changer de mot de passe régulièrement
- Créer un mot de passe fort et complexe

Changer et sécuriser les mots de passe

Pour changer un mot de passe sous Linux, la commande **passwd** est utilisée. Voici comment sécuriser les mots de passe de manière efficace :

- **Changement de mot de passe** : tapez **passwd** pour changer votre mot de passe. Si vous êtes administrateur et souhaitez changer le mot de passe d'un autre utilisateur, utilisez **passwd [nom_utilisateur]**.

```
root@debian:/home/debian# passwd adrien
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
root@debian:/home/debian#
```

- **Politiques de mot de passe** : configurez des politiques de complexité via le fichier **login.defs** pour exiger des mots de passe forts (longueur, diversité de caractères, etc.).
- **Expiration des mots de passe** : définissez une expiration régulière des mots de passe pour encourager les utilisateurs à les renouveler fréquemment. Il suffit de mettre une valeur en jour à la variable **PASS_MAX_DAYS** dans le fichier de configuration

```
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
```

Compréhension du système de permissions Linux

Les bases des permissions (lecture, écriture, exécution)

Sous Linux, le contrôle des accès aux fichiers et répertoires est géré par un système de permissions qui spécifie les droits de lecture, écriture et exécution pour trois catégories d'utilisateurs : le propriétaire du fichier, les membres du groupe auquel le fichier est associé, et tous les autres utilisateurs du système. Ces permissions déterminent respectivement si le fichier peut être lu, modifié ou exécuté.

Les permissions sont habituellement exprimées en notations symboliques ou numériques.

La notation symbolique utilise les lettres **'r' (read, lecture)**, **'w' (write, écriture)**, et **'x' (execute, exécution)** pour indiquer les droits accordés.

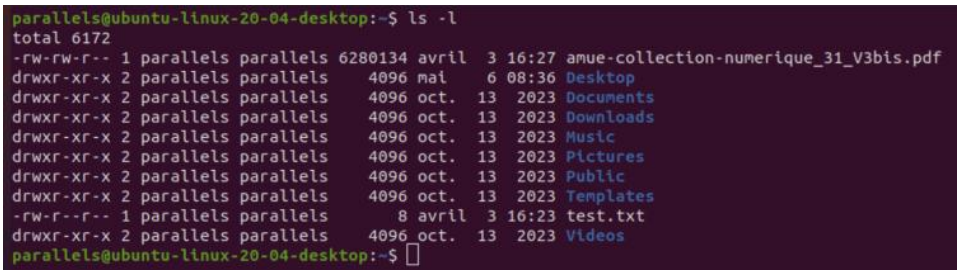
Par exemple :

- 'r' permet de voir le contenu d'un fichier ou de lister les fichiers dans un répertoire.
- 'w' autorise la modification du contenu d'un fichier ou l'ajout et la suppression de fichiers dans un répertoire.
- 'x' permet d'exécuter le fichier comme un programme ou de traverser le répertoire.

Dans la notation symbolique des permissions sous Linux, le caractère **'-' (tiret)** joue un rôle spécifique. Il est utilisé pour indiquer l'absence d'une permission spécifique dans

l'ensemble des droits attribués à une catégorie d'utilisateurs (propriétaire, groupe, et autres).

Ainsi, chaque tiret remplace une permission potentielle (lecture, écriture, ou exécution) et signale qu'elle n'est pas accordée pour cette catégorie spécifique d'utilisateurs. Cette méthode permet une visualisation claire et immédiate des droits attribués, facilitant la gestion des accès de manière sécurisée et contrôlée.



La notation numérique pour les permissions sous Linux est une autre méthode pour exprimer les mêmes droits de lecture, écriture et exécution définis dans la notation symbolique, mais sous forme de chiffres. Chaque type de droit (lecture, écriture, exécution) est associé à une valeur numérique spécifique :

- Lecture (r) est représentée par le **chiffre 4**.
- Écriture (w) est représentée par le **chiffre 2**.
- Exécution (x) est représentée par le **chiffre 1**.

Ces valeurs peuvent être additionnées pour combiner plusieurs permissions.

Par exemple, si vous souhaitez attribuer des permissions de lecture et d'écriture mais pas d'exécution, vous additionnez les valeurs de lecture (4) et d'écriture (2), ce qui donnerait une permission de 6.

La notation numérique utilise trois chiffres pour représenter les permissions pour trois catégories d'utilisateurs : le propriétaire du fichier, le groupe auquel appartient le fichier, et tous les autres utilisateurs. Chaque chiffre peut prendre une valeur de 0 à 7, calculée par l'addition des droits individuels :

- 0 : aucun droit (0)
- 1 : exécution seule (1)
- 2 : écriture seule (2)
- 3 : écriture et exécution (2+1)
- 4 : lecture seule (4)
- 5 : lecture et exécution (4+1)

- 6 : lecture et écriture (4+2)
- 7 : lecture, écriture et exécution (4+2+1)

Permissions pour les utilisateurs, les groupes et les autres

Supposons un fichier **exemple.txt** avec les permissions suivantes :

- Propriétaire : lecture et écriture
- Groupe : lecture seulement
- Autres : aucune permission

En notation symbolique, cela serait représenté par **rw-r-----,**

- Pour le propriétaire **(rw-)** :
    - r signifie que le propriétaire a le droit de lire le fichier.
    - w signifie que le propriétaire a également le droit d'écrire ou de modifier le fichier.
    - Le - à la place du x indique que le propriétaire n'a pas le droit d'exécuter ce fichier comme un programme.
- Pour le groupe **(r--)** :
    - r indique que les membres du groupe ont le droit de lire le fichier.
    - Les deux tirets suivants (--) remplacent les droits w (écriture) et x (exécution), signifiant que les membres du groupe ne peuvent ni modifier ni exécuter le fichier.
- Pour les autres utilisateurs **(---)** :

Les trois tirets signifient que les autres utilisateurs n'ont aucun droit sur le fichier. Ils ne peuvent ni lire, ni écrire, ni exécuter le fichier.

Et en notation numérique par **640** :

- Le propriétaire a des droits de lecture et d'écriture **(6 = 4+2)**,
- Le groupe a uniquement le droit de lecture **(4)**,
- Les autres utilisateurs n'ont aucun droit **(0)**.

Modification des permissions et propriétés

Utiliser chmod, chown, et chgrp

Pour gérer les permissions de fichiers et de répertoires, Linux offre plusieurs commandes :

- **chmod (change mode)** : modifie les permissions d'un fichier ou répertoire. Exemple : **chmod 755 fichier** donne au propriétaire toutes les permissions contre lecture et exécution aux autres.
- **chown (change owner)** : change le propriétaire d'un fichier. Exemple : **chown utilisateur fichier** change le propriétaire du fichier.
- **chgrp(change group)** : change le groupe associé à un fichier. Exemple : **chgrp groupe fichier** change le groupe du fichier.

Erreurs courantes à éviter

Ne donnez pas de permissions d'exécution à des fichiers **non-exécutables**, cela peut être une faille de sécurité.

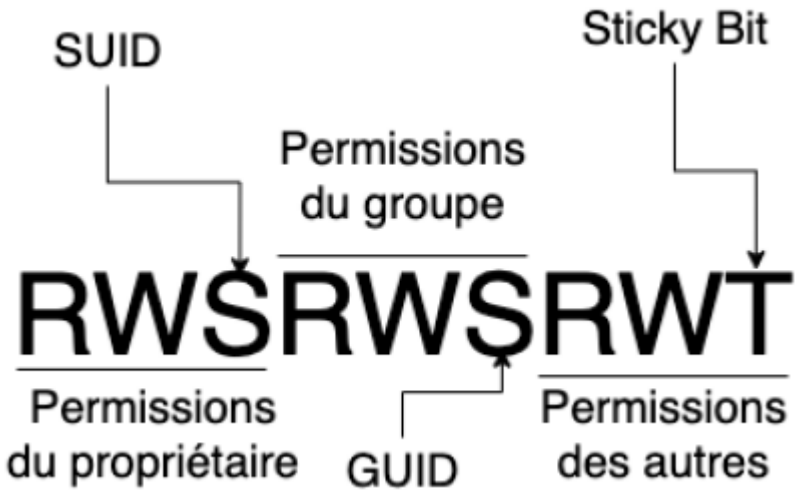
Évitez d'utiliser **chmod 777**, qui donne des permissions complètes à tout le monde, ce qui peut exposer des données sensibles.

Gestion des droits d'accès spécifiques

setuid, setgid et sticky bit

Ces permissions spéciales modifient le comportement des fichiers et répertoires :

- **setuid** : le fichier s'exécute avec les permissions de son propriétaire, et non de l'utilisateur qui l'exécute.
- **setgid** : les fichiers créés dans un répertoire héritent le groupe du répertoire plutôt que le groupe de l'utilisateur qui les crée.
- **sticky bit** : utilisé principalement sur des répertoires pour permettre aux utilisateurs de supprimer uniquement les fichiers qu'ils possèdent.



setuid (Set User ID)

Le bit **setuid** modifie le comportement d'un exécutable en faisant en sorte que le programme s'exécute avec les permissions de son propriétaire, et non avec les permissions de l'utilisateur qui l'exécute. Cela est particulièrement utile pour les programmes qui nécessitent des privilèges d'accès élevés pour fonctionner correctement, tout en étant accessibles à des utilisateurs ayant des droits limités.

Supposons qu'un administrateur système souhaite permettre aux utilisateurs de changer leurs mots de passe sans donner à ces utilisateurs l'accès administratif complet. Le fichier exécutable qui gère le changement de mot de passe peut être possédé par le super utilisateur (root) et configuré avec le bit setuid :

-rwsr-xr-x 1 root root 12345 Jan 10 10:00 /usr/bin/passwd

Ici, le **s** dans les permissions (**rws**) indique que le bit **setuid** est activé. Lorsqu'un utilisateur exécute ce programme, il s'exécute avec les privilèges du propriétaire (**root**), permettant ainsi le changement de mot de passe.

setgid (Set Group ID)

Le bit **setgid**, lorsqu'il est appliqué à un répertoire, fait en sorte que tous les fichiers créés dans ce répertoire héritent du groupe du répertoire, plutôt que du groupe de l'utilisateur qui les crée. Cela est utile pour maintenir des groupes de travail cohérents au sein de répertoires partagés, où l'accès doit être géré collectivement.

Dans un environnement de développement logiciel, un répertoire **/projects** peut être configuré pour que tout nouveau code ou documentation créé dans ce répertoire soit automatiquement attribué au groupe developers :

drwxr-sr-x 2 root developers 4096 Jan 10 10:00 /projects

Le **s** dans les permissions du groupe (**r-s**) indique que le bit **setgid** est actif. Les fichiers créés par n'importe quel membre du groupe **developers** conserveront ce groupe, facilitant la collaboration et le partage des fichiers dans le projet.

Sticky bit

Le **sticky bit** est utilisé principalement sur des répertoires partagés où plusieurs utilisateurs peuvent créer des fichiers. Lorsqu'il est activé, il permet à l'utilisateur de supprimer ou de modifier ses propres fichiers uniquement, et non ceux des autres utilisateurs, même si les permissions du répertoire permettraient autrement ces actions.

Sur un serveur de fichiers partagé, le répertoire **/shared/docs** peut être utilisé par plusieurs utilisateurs pour stocker des documents :

drwxrwxrwt 3 admin staff 4096 Jan 10 10:00 /shared/docs

Le **t** à la fin des permissions (**rwt**) indique que le **sticky bit** est activé. Ainsi, chaque utilisateur peut gérer ses propres fichiers sans risquer d'affecter les fichiers des autres, ce qui est essentiel pour éviter les suppressions accidentelles ou malveillantes.

Quand et comment utiliser ces permissions spéciales

Utilisez **setuid** et **setgid** avec prudence, surtout sur des scripts ou des binaires pour éviter des failles de sécurité.

Appliquez le **sticky bit** sur des répertoires partagés où les utilisateurs ont besoin de contrôler uniquement leurs propres fichiers, comme /tmp.

L'utilisation de ces bits spéciaux doit être faite avec prudence, car une mauvaise configuration peut entraîner des risques de sécurité. Ils sont néanmoins des outils puissants pour gérer des environnements multi-utilisateurs complexes, en offrant des mécanismes pour maintenir les privilèges nécessaires et protéger les données partagées.