Linux Debian - Administration

m2iformation.fr

Sommaire

Jour 2

- Présentation des différents systèmes de fichiers
- Gestion des partitions traditionnelles (partitions primaires et étendues)
- Gestion des volumes logiques (LVM)
- Formatage
- Montage et démontage des systèmes de fichiers
- Gestion des droits standards (chown, chmod, umask)
- Listes de contrôle d'accès (ACL)

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Jour 2

Présentation des différents systèmes de fichiers

1. ext4 (Fourth Extended Filesystem)

- **Description** : C'est l'un des systèmes de fichiers les plus utilisés sur Linux, en particulier sur les distributions comme Debian. Il est une amélioration des versions précédentes (ext2 et ext3), offrant de meilleures performances, une gestion optimisée de la fragmentation, et une plus grande capacité de stockage.
- **Caractéristiques** :
  - Taille maximale d'un fichier : 16 To (Téraoctets)
  - Taille maximale d'un système de fichiers : 1 Eo (Exaoctet)
  - Journalisation : Gère un journal pour réduire les risques de corruption des données en cas de panne.
  - Récupération rapide des données après un crash.
  - Prise en charge des attributs étendus et des quotas de disque.
- **Cas d'utilisation** : Système de fichiers par défaut dans de nombreuses distributions Linux, idéal pour les serveurs et postes de travail.

Présentation des différents systèmes de fichiers

2. Btrfs (B-tree File System)

- **Description** : Un système de fichiers moderne conçu pour remplacer ext4 à long terme. Il se concentre sur la gestion des grandes quantités de données, la fiabilité, et les fonctionnalités avancées comme la prise en charge de la **gestion des instantanés** (snapshots) et de la **compression** des données.
- **Caractéristiques** :
  - Gestion des snapshots et de la déduplication de données.
  - Taille maximale d'un fichier : 16 Eo.
  - Taille maximale d'un système de fichiers : 16 Eo.
  - Journalisation intégrée.
  - Possibilité d'effectuer des mises à jour atomiques grâce aux snapshots.
- **Cas d'utilisation** : Idéal pour des serveurs où des snapshots fréquents et la gestion de grands volumes de données sont requis. Souvent utilisé sur des systèmes de stockage complexes ou en cloud.

Présentation des différents systèmes de fichiers

3. XFS

- **Description** : Un autre système de fichiers performant, conçu initialement pour les serveurs SGI (Silicon Graphics). XFS est particulièrement adapté aux grandes bases de données et systèmes de fichiers nécessitant une très haute performance.
- **Caractéristiques** :
  - Taille maximale d'un fichier : 8 Eo.
  - Taille maximale d'un système de fichiers : 8 Eo.
  - Excellente gestion de fichiers de grande taille et des I/O (opérations d'entrée/sortie) intensives.
  - Journalisation à haut débit.
  - Optimisé pour la gestion des fichiers fragmentés.
- **Cas d'utilisation** : Utilisé dans les environnements de production de haute performance, comme les serveurs de base de données ou les systèmes où de très grandes quantités de données doivent être stockées et traitées rapidement.

Présentation des différents systèmes de fichiers

4. ZFS (Zettabyte File System)

- **Description** : Un système de fichiers avancé développé par Sun Microsystems, maintenant sous licence CDDL. ZFS combine des fonctionnalités de gestion de volumes avec un système de fichiers, ce qui le rend particulièrement intéressant pour les systèmes nécessitant des fonctionnalités comme la **gestion de volume**, la **correction automatique d'erreurs**, et une **intégrité des données** de haute qualité.
- **Caractéristiques** :
  - Gestion de très grandes quantités de stockage (pouvant atteindre des zettaoctets).
  - Intégrité des données grâce à des contrôles d'intégrité (checksums) automatiques.
  - Compression et déduplication des données.
  - Gestion des snapshots.
  - Résilience face à la corruption des données grâce à la réparation automatique.
- **Cas d'utilisation** : Très utilisé dans les systèmes de stockage massifs ou les infrastructures cloud, ainsi que pour des environnements où la sécurité et l'intégrité des données sont critiques.

Présentation des différents systèmes de fichiers

5. ReiserFS

- **Description** : Un système de fichiers conçu pour une gestion efficace des petits fichiers et des répertoires contenant de nombreux fichiers. Il est moins utilisé aujourd'hui mais reste une alternative à ext4 dans certains cas spécifiques.
- **Caractéristiques** :
  - Journalisation intégrée.
  - Efficacité accrue pour les petits fichiers.
  - Très bonne gestion de la fragmentation.
- **Cas d'utilisation** : Bien que moins populaire, il a été utilisé dans des systèmes où de nombreux petits fichiers devaient être stockés et manipulés efficacement.

Présentation des différents systèmes de fichiers

6. VFAT

- **Description** : Un système de fichiers basé sur la FAT (File Allocation Table), utilisé principalement pour la compatibilité avec les systèmes Windows ou pour les supports de stockage amovibles (clés USB, cartes SD, etc.).
- **Caractéristiques** :
  - Taille maximale d'un fichier : 4 Go.
  - Absence de journalisation.
  - Compatibilité avec Windows, ce qui le rend très utile pour les systèmes de fichiers partagés entre Linux et Windows.
- **Cas d'utilisation** : Utilisé pour des clés USB ou des cartes mémoire pour assurer la compatibilité entre Linux, Windows et d'autres systèmes.

Présentation des différents systèmes de fichiers

7. NTFS (New Technology File System)

- **Description** : Système de fichiers par défaut des systèmes Windows. Sous Debian, NTFS est principalement utilisé pour le montage de disques ou de partitions Windows en lecture/écriture, grâce à des outils comme `ntfs-3g`.
- **Caractéristiques** :
  - Journalisation.
  - Support des attributs étendus.
  - Support des fichiers volumineux (plus de 4 Go).
- **Cas d'utilisation** : Partage de fichiers entre Linux et Windows, notamment pour accéder aux partitions de disques durs formatées en NTFS.

Présentation des différents systèmes de fichiers

Lorsqu'il s'agit de gérer les systèmes de fichiers sur Debian, plusieurs outils sont disponibles pour formater, vérifier l'intégrité, et monter des partitions.

1. Formater une partition avec `mkfs`

Le formatage d'une partition est la première étape après la création d'une partition. Cela permet de préparer la partition en y installant un système de fichiers, afin qu'elle puisse stocker des données de manière organisée.

Outil : `mkfs` (make filesystem)

L'outil **`mkfs`** est utilisé pour créer un système de fichiers sur une partition. Il prend en charge plusieurs types de systèmes de fichiers, comme **ext4**, **XFS**, **Btrfs**, etc.

Pour formater une partition avec un système de fichiers **ext4**.

```
sudo mkfs.ext4 /dev/sda1
```

options:

- **`-t <type>`** : Spécifie le type de système de fichiers. Nous pouvons choisir ext4, XFS, Btrfs, ou tout autre système pris en charge.
- **`-L <label>`** : Attribue un label à la partition, une étiquette qui permet de l'identifier plus facilement.

Présentation des différents systèmes de fichiers

2. Vérifier l'intégrité du système de fichiers avec `fsck`

L'outil **`fsck`** (file system check) est utilisé pour vérifier et, si nécessaire, réparer l'intégrité d'un système de fichiers. Des problèmes peuvent survenir sur une partition en raison de coupures de courant, de plantages du système ou d'autres problèmes, et **`fsck`** peut détecter et corriger les erreurs dans ces cas.
Pour vérifier l'intégrité du système de fichiers sur la partition **/dev/sda1**.

```
sudo fsck /dev/sda1
```

- Il effectue un contrôle approfondi de la structure du système de fichiers.
- Si des erreurs sont détectées, **`fsck`** tentera de les corriger automatiquement, sauf si nous avons spécifié des options particulières.

Options:

- **`-y`** : Cette option indique à **`fsck`** de réparer automatiquement toutes les erreurs détectées, sans demander confirmation à chaque fois.
- **`-f`** : Forcer la vérification, même si le système de fichiers semble propre.
- **`-n`** : Ne pas faire de modifications, simplement signaler les erreurs trouvées.

Il est important d'utiliser **`fsck`** régulièrement, notamment après un plantage ou une coupure de courant, pour garantir que le système de fichiers reste cohérent et que les données ne sont pas corrompues.

Présentation des différents systèmes de fichiers

3. Monter un système de fichiers avec `mount`

Une fois que nous avons créé et formaté une partition, nous devons la **monter** pour qu'elle devienne accessible dans l'arborescence des fichiers. Le montage d'une partition signifie l'associer à un point de montage, c'est-à-dire un répertoire, afin que le système d'exploitation puisse y accéder et y stocker des fichiers.
L'outil `mount` permet de monter une partition sur un répertoire du système de fichiers.
Pour monter la partition **/dev/sda1** sur le répertoire **/mnt/data**.

```
sudo mount /dev/sda1 /mnt/data
```

Options:

- `-t <type>` : Spécifie le type de système de fichiers à monter, si nous avons besoin de le spécifier explicitement.
- `-o <options>` : Utilise des options spécifiques pour le montage, comme `rw` (lecture/écriture) ou `ro` (lecture seule).

Présentation des différents systèmes de fichiers

4. Monter automatiquement au démarrage via `/etc/fstab`

Pour que le montage soit automatique à chaque démarrage, nous devons ajouter une entrée dans le fichier `/etc/fstab`. Ce fichier contient les informations nécessaires pour monter automatiquement les partitions.

1. Ouvrir le fichier `/etc/fstab` :

```
sudo nano /etc/fstab
```

2. Ajouter une ligne comme celle-ci pour monter **/dev/sda1** sur **/mnt/data** avec un système de fichiers ext4 :

```
/dev/sda1  /mnt/data  ext4  defaults  0  2
```

La gestion des **partitions traditionnelles** sur un disque dur consiste à organiser l'espace de stockage de manière à diviser le disque en plusieurs segments indépendants, appelés **partitions**. Chaque partition peut avoir son propre système de fichiers et être utilisée à différentes fins, comme l'installation d'un système d'exploitation, le stockage de données, ou la gestion de fichiers temporaires.

Gestion des partitions traditionnelles (partitions primaires et étendues)

1. Partitions primaires

Les **partitions primaires** sont les partitions de base créées sur un disque dur. Sur un disque au format **MBR (Master Boot Record)**, nous ne pouvons avoir que **quatre partitions primaires maximum**. Une partition primaire peut contenir des données, un système de fichiers, ou être utilisée pour démarrer un système d'exploitation.

- **Limitation à quatre** : Un disque dur formaté en MBR ne peut pas avoir plus de quatre partitions primaires.
- **Système d'exploitation** : Au moins une des partitions primaires peut être marquée comme **bootable**, ce qui permet au système d'exploitation de démarrer depuis cette partition.
- **Simplicité** : Les partitions primaires sont simples et faciles à gérer. Elles sont idéales pour les systèmes où un petit nombre de partitions est suffisant.

Gestion des partitions traditionnelles (partitions primaires et étendues)

Pour créer une partition primaire, nous utilisons des outils comme `fdisk` ou `parted`.

1. Ouvrir `fdisk` pour gérer le disque dur (par exemple, **/dev/sda**) :

```
sudo fdisk /dev/sda
```

2. Créer une partition primaire :
   - Appuyer sur `n` pour créer une nouvelle partition.
   - Sélectionner `p` pour créer une partition primaire.
   - Suivre les instructions pour spécifier la taille de la partition.
3. Enregistrer les modifications et quitter `fdisk` :

```
w
```

4. Formater la partition nouvellement créée avec un système de fichiers, par exemple ext4 :

```
sudo mkfs.ext4 /dev/sda1
```

Gestion des partitions traditionnelles (partitions primaires et étendues)

2. Partitions étendues

Les **partitions étendues** existent pour dépasser la limitation des quatre partitions primaires. Comme un disque MBR ne peut contenir que quatre partitions primaires, si nous avons besoin de plus de quatre partitions, nous devons créer une partition étendue. Une partition étendue agit comme un conteneur, et à l'intérieur de cette partition, nous pouvons créer plusieurs **partitions logiques**.

- **Une seule partition étendue** : Il ne peut y avoir qu'une seule partition étendue par disque.
- **Conteneur pour partitions logiques** : La partition étendue ne contient pas directement de données, mais peut héberger un nombre illimité de partitions logiques (pratiquement limité par la taille du disque).
- **Évite la limitation des quatre partitions** : Une fois la partition étendue créée, nous pouvons y ajouter des partitions logiques supplémentaires.

Sur un disque où trois partitions primaires sont déjà créées, nous pouvons créer une partition étendue pour ajouter des partitions logiques supplémentaires. Par exemple, nous pourrions utiliser ces partitions logiques pour séparer des données, des fichiers de swap, ou des fichiers temporaires.

Gestion des partitions traditionnelles (partitions primaires et étendues)

1. Utiliser `fdisk` pour créer une partition étendue :
    - Ouvrir `fdisk` sur le disque (par exemple, **/dev/sda**) :

```
sudo fdisk /dev/sda
```

- Appuyer sur `n` pour créer une nouvelle partition.
- Sélectionner `e` pour créer une partition étendue.
- Suivre les instructions pour spécifier la taille.

2. Créer des partitions logiques à l'intérieur de la partition étendue :
    - Après avoir créé la partition étendue, nous pouvons créer des partitions logiques de la même manière que des partitions primaires.
    - Lors de la création de nouvelles partitions, elles seront numérotées **/dev/sda5**, **/dev/sda6**, etc. (les partitions logiques commencent à partir de `5`).
3. Formater les partitions logiques créées avec un système de fichiers :

Gestion des partitions traditionnelles (partitions primaires et étendues)

3. MBR vs GPT

L'utilisation de **MBR** pour gérer les partitions est une approche traditionnelle, mais elle est de plus en plus remplacée par le format **GPT (GUID Partition Table)**, qui est plus moderne et offre davantage de flexibilité. Contrairement à MBR, GPT permet de créer un nombre illimité de partitions (pratiquement 128 partitions), tout en supportant des disques de très grande taille (au-delà de 2 To).

- **Pas de limitation à 4 partitions** : GPT permet de créer de nombreuses partitions sans avoir besoin de partitions étendues et logiques.
- **Support des disques de plus de 2 To** : Contrairement à MBR, qui est limité à 2 To par disque.
- **Meilleure protection des données** : GPT inclut plusieurs copies de la table des partitions, ce qui améliore la sécurité en cas de corruption.

Gestion des volumes logiques (LVM)

La **Gestion des Volumes Logiques (LVM)** est un système utilisé dans Linux pour fournir une gestion flexible et dynamique de l'espace disque. Contrairement aux partitions traditionnelles, LVM permet de regrouper plusieurs disques ou partitions en un seul espace de stockage (appelé **volume logique**), que nous pouvons redimensionner facilement sans interrompre les services.

1. **Volumes physiques (Physical Volumes, PV)** :
   - Un **volume physique** correspond à une partition ou un disque brut. Nous convertissons une partition ou un disque en volume physique pour qu'il puisse être utilisé par LVM. Par exemple, un disque ou une partition comme **/dev/sda1** ou **/dev/sdb** peut être un volume physique.
   - Ces volumes physiques sont ensuite regroupés en un groupe de volumes.
2. **Groupes de volumes (Volume Groups, VG)** :
   - Un **groupe de volumes** est un pool d'espace de stockage qui contient plusieurs volumes physiques. LVM regroupe ces volumes physiques pour créer un espace de stockage plus important et plus flexible.
   - Nous pouvons créer, agrandir ou réduire des volumes logiques à partir de ce groupe de volumes.

Gestion des volumes logiques (LVM)

3. **Volumes logiques (Logical Volumes, LV)** :
   - Un **volume logique** est l'espace alloué dans un groupe de volumes. Il fonctionne comme une partition classique, mais avec la flexibilité d'être agrandi ou réduit dynamiquement.
   - Un volume logique peut héberger des systèmes de fichiers (comme ext4, XFS, etc.) et peut être utilisé de la même manière que des partitions traditionnelles.

Gestion des volumes logiques (LVM)

LVM présente plusieurs avantages majeurs par rapport aux partitions traditionnelles :

- **Redimensionnement facile** : Nous pouvons augmenter ou réduire la taille des volumes logiques sans réinstaller le système ni affecter les données existantes.
- **Gestion dynamique** : Si nous manquons d'espace sur un volume logique, nous pouvons ajouter de nouveaux disques ou partitions au groupe de volumes et agrandir les volumes logiques.
- **Snapshots** : LVM permet la création de **snapshots**, des copies instantanées d’un volume logique à un moment donné, ce qui est utile pour les sauvegardes ou les tests.
- **Gestion de l'espace disque étendu** : LVM permet de regrouper plusieurs disques ou partitions physiques en un seul volume, ce qui facilite la gestion d'un grand nombre de disques et l'extension du stockage au fil du temps.

Gestion des volumes logiques (LVM)

Pour configurer LVM, nous devons suivre quelques étapes pour convertir des disques ou partitions en volumes physiques, créer un groupe de volumes, puis créer et gérer des volumes logiques.

1. Convertir un disque ou une partition en volume physique (PV)

Avant de créer des groupes de volumes et des volumes logiques, nous devons préparer les partitions ou disques pour qu'ils puissent être utilisés par LVM.

1. **Afficher les disques disponibles** :

```
lsblk
```

2. **Créer un volume physique** avec `pvcreate` :
   Par exemple, si nous voulons utiliser **/dev/sdb1** comme volume physique pour LVM :

```
sudo pvcreate /dev/sdb1
```

3. **Vérifier la création du volume physique** :
   Utilisons la commande `pvdisplay` pour vérifier :

```
sudo pvdisplay
```

Gestion des volumes logiques (LVM)

2. Créer un groupe de volumes (VG)

Une fois les volumes physiques créés, nous devons les regrouper dans un **groupe de volumes**. C'est dans ce groupe que nous allons créer les volumes logiques.

1. **Créer un groupe de volumes** avec `vgcreate` :
   Nous pouvons regrouper un ou plusieurs volumes physiques dans un groupe de volumes. Par exemple, pour créer un groupe de volumes appelé **vg_data** à partir de **/dev/sdb1** :

```
sudo vgcreate vg_data /dev/sdb1
```

2. **Vérifier le groupe de volumes** :
   Utilisons `vgdisplay` pour afficher les informations sur le groupe de volumes :

```
sudo vgdisplay
```

Gestion des volumes logiques (LVM)

3. Créer des volumes logiques (LV)

Une fois le groupe de volumes créé, nous pouvons créer un **volume logique** qui sera utilisé pour stocker les données, exactement comme une partition traditionnelle.

1. **Créer un volume logique** avec `lvcreate` :
   Par exemple, pour créer un volume logique de 10 Go appelé **lv_data** dans le groupe de volumes **vg_data** :

```
sudo lvcreate -L 10G -n lv_data vg_data
```

2. **Vérifier le volume logique** :
   Utilisons `lvdisplay` pour afficher les informations sur les volumes logiques :

```
sudo lvdisplay
```

Gestion des volumes logiques (LVM)

3. **Formater le volume logique** avec un système de fichiers :
   Une fois le volume logique créé, nous devons le formater pour pouvoir y stocker des fichiers. Par exemple, pour formater le volume logique **lv_data** en **ext4** :

```
sudo mkfs.ext4 /dev/vg_data/lv_data
```

4. **Monter le volume logique** :
   Enfin, nous devons monter le volume logique pour qu'il soit accessible depuis le système de fichiers :

```
sudo mount /dev/vg_data/lv_data /mnt/data
```

Gestion des volumes logiques (LVM)

4. Redimensionner un volume logique

Un des avantages majeurs de LVM est la capacité à redimensionner les volumes logiques à la volée. Nous pouvons augmenter la taille d'un volume logique si nous manquons d'espace.

1. **Étendre un volume logique** avec `lvextend` :
   Si nous voulons ajouter 5 Go supplémentaires au volume logique **lv_data** :

```
sudo lvextend -L +5G /dev/vg_data/lv_data
```

2. **Redimensionner le système de fichiers** pour qu'il utilise l'espace supplémentaire :
   Pour **ext4**, nous utilisons `resize2fs` :

```
sudo resize2fs /dev/vg_data/lv_data
```

Pour **XFS**, nous utilisons `xfs_growfs` :

```
sudo xfs_growfs /mnt/data
```

Gestion des volumes logiques (LVM)

5. Créer un snapshot avec LVM

LVM permet de créer des **snapshots**, des copies instantanées d'un volume logique. Ces snapshots sont utiles pour des sauvegardes ou des tests avant d'effectuer des changements importants.

1. **Créer un snapshot** avec `lvcreate` :
   Par exemple, pour créer un snapshot du volume logique **lv_data** dans **vg_data** avec une taille de 5 Go :

```
sudo lvcreate --size 5G --snapshot --name lv_data_snap /dev/vg_data/lv_data
```

2. **Monter le snapshot** pour l'utiliser :

```
sudo mount /dev/vg_data/lv_data_snap /mnt/snapshot
```

Gestion des volumes logiques (LVM)

6. Supprimer un volume logique ou un groupe de volumes

Si nous voulons supprimer un volume logique ou un groupe de volumes, nous devons démonter le volume logique, puis utiliser les commandes de suppression.

1. **Démonter le volume logique** :

```
sudo umount /mnt/data
```

2. **Supprimer le volume logique** avec `lvremove` :

```
sudo lvremove /dev/vg_data/lv_data
```

3. **Supprimer le groupe de volumes** avec `vgremove` (si plus aucun volume logique n'est présent dans le groupe) :

```
sudo vgremove vg_data
```

4. **Supprimer le volume physique** avec `pvremove` :

```
sudo pvremove /dev/sdb1
```

TP 3 Gestion des volumes logiques (LVM)

Objectif :

Dans cet exercice, les stagiaires utiliseront **LVM (Logical Volume Manager)** pour gérer dynamiquement l'espace de stockage dans un environnement **VirtualBox**. Ils vont créer un volume logique, le formater avec un système de fichiers, puis configurer son montage permanent via `fstab` et **systemd**.

Prérequis :

- Environnement VirtualBox fonctionnel avec une machine virtuelle Debian ou une autre distribution Linux.
- Compréhension de base des partitions, systèmes de fichiers, et du gestionnaire de volumes logiques (LVM).
- Une machine virtuelle avec une partition ou un disque supplémentaire non alloué.

1. **Ajouter un disque supplémentaire à la machine virtuelle**
2. **Créer un Volume Physique (PV)**
3. **Créer un Groupe de Volumes (VG)**
4. **Créer un Volume Logique (LV)**
5. **Formater le Volume Logique**
6. **Monter le Volume Logique**

TP 3 Gestion des volumes logiques (LVM)

7. **Configurer le Montage Permanent avec `fstab`**
8. **Configurer le Montage Permanent avec `systemd`**
9. **Vérifications**

Gestion des droits standards (chown, chmod, umask)

1. `chown` : Changer le propriétaire et le groupe des fichiers

Le propriétaire d'un fichier ou d'un dossier détermine qui peut le modifier ou y accéder. Dans Debian, chaque fichier a un propriétaire et un groupe. Le propriétaire est l'utilisateur qui a créé le fichier, et le groupe est souvent celui auquel appartient cet utilisateur.

Syntaxe :

```
chown [nouveau_propriétaire]:[nouveau_groupe] fichier
```

- `nouveau_propriétaire` : Le nom de l'utilisateur à qui vous voulez assigner le fichier.
- `nouveau_groupe` : Le nom du groupe auquel vous voulez assigner le fichier. Vous pouvez ignorer cette partie si vous ne souhaitez pas changer le groupe.
- `fichier` : Le fichier ou le répertoire cible.

Gestion des droits standards (chown, chmod, umask)

2. `chmod` : Modifier les droits d'accès aux fichiers

`chmod` permet de définir les permissions d'accès pour les fichiers et dossiers. Il y a trois types de permissions :

- **r** : Lecture (Read)
- **w** : Écriture (Write)
- **x** : Exécution (Execute)

Ces permissions sont attribuées à trois catégories :

- **u** : Utilisateur propriétaire (user)
- **g** : Groupe propriétaire (group)
- **o** : Autres (others)

Syntaxe :

```
chmod [options] mode fichier
```

Il existe deux façons principales d'utiliser `chmod` :

- **Symbolique** : Utiliser des lettres pour ajouter, enlever, ou définir les permissions (comme `u+r` pour ajouter la permission de lecture à l'utilisateur).
- **Numérique** : Utiliser des chiffres pour représenter les permissions (comme `755`).

Gestion des droits standards (chown, chmod, umask)

3. `umask` : Définir les permissions par défaut

Lorsque vous créez un nouveau fichier ou répertoire, les permissions par défaut sont définies par le masque d'utilisateur, ou `umask`. Ce masque soustrait certaines permissions à celles qui sont normalement attribuées à un fichier ou un répertoire nouvellement créé.

Syntaxe :

```
umask [valeur]
```

La valeur de `umask` est soustraite des permissions maximales possibles pour déterminer les permissions par défaut. Les permissions maximales sont :

- **666** pour les fichiers (lecture et écriture pour tous),
- **777** pour les répertoires (lecture, écriture, exécution pour tous).

Listes de contrôle d'accès (ACL)

Les **Listes de Contrôle d'Accès (ACL)** permettent de définir des permissions plus granulaires que celles offertes par les mécanismes traditionnels de gestion des permissions avec `chown` et `chmod`. Grâce aux ACL, vous pouvez accorder des permissions spécifiques à des utilisateurs ou des groupes particuliers pour des fichiers ou répertoires, sans être limité aux trois catégories classiques (utilisateur propriétaire, groupe propriétaire, et autres).

1. Qu'est-ce qu'une ACL ?

Une ACL permet de définir des permissions spécifiques pour plusieurs utilisateurs ou groupes, indépendamment des permissions définies via `chmod`. Cela offre plus de flexibilité dans la gestion des droits d'accès.

2. Activer les ACL sur un système de fichiers

Avant de pouvoir utiliser les ACL, il faut s'assurer que le système de fichiers prend en charge cette fonctionnalité. Généralement, les systèmes de fichiers comme **ext4** supportent les ACL, mais vous devez parfois les activer manuellement.

3. Utiliser les ACL

Une fois ACL activé, vous pouvez commencer à gérer les permissions supplémentaires sur vos fichiers et répertoires.

Listes de contrôle d'accès (ACL)

1. Affichage des ACL

Pour voir les ACL définies sur un fichier ou un répertoire, utilisez la commande `getfacl` :

```
getfacl nom_du_fichier
```

2. Définir des ACL avec `setfacl`

La commande `setfacl` permet d'ajouter, modifier ou supprimer des permissions ACL pour un fichier ou un répertoire.

```
setfacl -m u:[utilisateur]:[permissions] nom_du_fichier
```

- `u:[utilisateur]` : Définit l'utilisateur à qui vous donnez des permissions spécifiques.
- `[permissions]` : Peut inclure `r` (lecture), `w` (écriture), `x` (exécution).

3. Supprimer une ACL

Pour supprimer une ACL attribuée à un utilisateur, vous pouvez utiliser la commande suivante :

```
setfacl -x u:bob projet.txt
```

4. **Définir des ACL par défaut**
   Les ACL par défaut s'appliquent automatiquement à tous les nouveaux fichiers et répertoires créés à l'intérieur d'un répertoire particulier. Pour définir une ACL par défaut, utilisez l'option `-d` :

Listes de contrôle d'accès (ACL)

4. Visualisation des permissions avec `ls`

Lorsque vous utilisez des ACL, le résultat de la commande `ls -l` change légèrement. Vous verrez un `+` à la fin des permissions si des ACL sont en place :

```
ls -l projet.txt
```

Résultat :

```
-rw-rw-r--+ 1 alice developpeurs 0 Oct 20 10:10 projet.txt
```

Le `+` indique que des ACL supplémentaires sont définies pour ce fichier.

TP 4 : Droits Classiques et Création d'un Répertoire Collaboratif

Objectif :

Dans ce TP, vous allez manipuler les droits d'accès standards des fichiers et des répertoires sous Debian. Vous allez créer un répertoire collaboratif où plusieurs utilisateurs pourront travailler ensemble, tout en contrôlant les permissions d'accès de manière précise.

1. **Manipulation des droits classiques sur des fichiers et répertoires :**
    1. Créez un fichier appelé `document.txt` dans votre répertoire personnel.
    2. Modifiez les permissions de ce fichier pour que :
        - Le propriétaire ait les droits de lecture et d'écriture.
        - Le groupe ait les droits de lecture uniquement.
        - Les autres utilisateurs n'aient aucun droit d'accès.
    3. Vérifiez que les permissions ont bien été appliquées avec la commande `ls -l`.
    4. Changez le propriétaire du fichier `document.txt` à un autre utilisateur présent sur le système.
    5. Modifiez à nouveau les permissions pour que le propriétaire ait tous les droits (lecture, écriture, et exécution), et que le groupe et les autres utilisateurs n'aient que des droits de lecture.

TP 4 : Droits Classiques et Création d'un Répertoire Collaboratif

2. **Création d'un répertoire collaboratif :**
    1. Créez un répertoire appelé `projet_collaboratif` dans votre répertoire personnel.
    2. Assurez-vous que ce répertoire est accessible pour :
        - Le propriétaire avec des droits complets (lecture, écriture, exécution).
        - Le groupe avec des droits complets (lecture, écriture, exécution).
        - Les autres utilisateurs avec des droits de lecture uniquement.
    3. Ajoutez plusieurs utilisateurs à un groupe spécifique qui sera en charge de collaborer dans ce répertoire.
    4. Modifiez les permissions pour que tous les membres de ce groupe puissent :
        - Lire, écrire, et exécuter dans le répertoire.
        - Les fichiers créés dans ce répertoire doivent automatiquement appartenir à ce groupe et suivre les mêmes permissions.
    5. Testez les permissions en créant des fichiers à l'intérieur du répertoire `projet_collaboratif` avec différents utilisateurs appartenant au groupe collaboratif, puis vérifiez les permissions de chaque fichier.
    6. Modifiez les permissions pour qu'un fichier spécifique dans ce répertoire ne soit modifiable que par le propriétaire, tout en restant accessible en lecture pour les autres membres du groupe.

TP 4 : Droits Classiques et Création d'un Répertoire Collaboratif

3. **Gestion des erreurs et des permissions :**
    1. Vérifiez les permissions des fichiers et répertoires à chaque étape avec la commande `ls -l`.
    2. Testez ce qui se passe si un utilisateur non membre du groupe essaie d'ajouter un fichier dans le répertoire collaboratif. Quelles sont les erreurs rencontrées ?
    3. Corrigez ces erreurs en modifiant les permissions du répertoire ou des fichiers selon les besoins du projet collaboratif.