Initiation à l'administration d'un poste sous Linux

Équipe Pédagogique Réseau Informatique @
UVCI 2019

Mars 2019
Version 1.0

Table des matières

Objectifs

- *Gérer* le mécanisme de connexion d'un utilisateur
- *Gérer* le système de fichier de Linux
- *Gérer* la protection des fichiers

Introduction

Nous avons appris dans la leçon introductive que GNU/Linux est un système d'exploitation libre, multitâche, multiplate-forme et multi-utilisateur de type UNIX basé sur le noyau Linux.

Ainsi pour permettre à de nombreux utilisateurs d'administrer la même machine, Linux met en œuvre des mécanismes d'identification des utilisateurs, de protection et de confidentialité de l'information, tout en permettant le partage contrôlé nécessaire au travail en groupe. Nous découvrirons les particularités du système de fichier de Linux ainsi que les commandes employée pour assurer la protection des fichiers.

- Toutes les manipulations se feront en mode texte avec la distribution *DEBIAN GNU/Linux* version *9.5*.
- Les commandes utilisées sont également valables pour les distributions dérivées de *DEBIAN*.

Mécanisme de connexion d'un utilisateur

Objectifs

Gérer le mécanisme de connexion d'un utilisateur

1. Connexion

Le système Linux étant un système multi–utilisateur et multitâche, plusieurs personnes sont connectées simultanément et peuvent travailler sans interférer les unes avec les autres.

Au démarrage d'une machine, plusieurs étapes se succèdent :

- Mise sous tension de la machine et de ses périphériques,
- Bootstrap du système (charger le noyau Linux),
- Montage des disques,
- Vérification des systèmes de fichiers (fsck),
- Passage en multi–utilisateur,
- Lancement des services.

On obtient alors, affichée à l'écran, l'invite “ login : ”

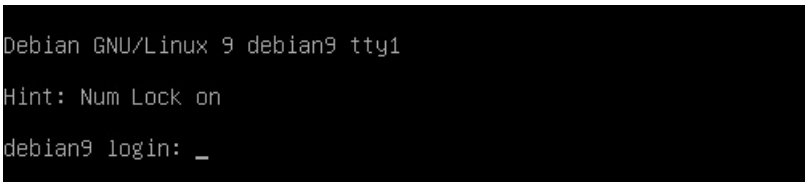


Figure 1 : Invite de connexion (login)

Pour qu'un utilisateur puisse travailler sur le système, il doit s'identifier en indiquant

- tout d'abord son *login* suivi de la touche <*Entrer*> après l'invite *login :*,
- puis son *mot de passe* suivi de la touche <*Entrer*> à la suite de l'invite *passwd :*.

Lorsque l'utilisateur saisit son mot de passe, les caractères saisis ne sont pas affichés à l'écran (on dit qu'il n'y a pas d'écho des caractères sur le terminal). Ce mécanisme permet de garder la confidentialité du mot de passe.

Il apparaît alors à l'écran un certain nombre d'informations (informations générales, date, arrivée de messages, date de dernière connexion). Puis le système lance un programme qui généralement est un interpréteur de commandes (shell). Il indique par une chaîne de caractères, appelée *invite* (ou *prompt*), qu'il est prêt à recevoir une commande. A partir de ce moment, l'utilisateur est connecté, on dit aussi qu'il est entré en session.

Dans l'illustration Figure 2 suivante, le *prompt* est : *uvci@debian9:~$*.

```
debian9 login: uvci
Password:
Last login: Mon Jul 30 03:33:29 GMT 2018 on tty1
Linux debian9 4.9.0-7-amd64 #1 SMP Debian 4.9.110-1 (2018-07-05) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
uvci@debian9:~$
```

Figure 2 : Entrée en session de l'utilisateur uvci sur la machine debian9

Remarque

En cas d'erreur lors de la saisie du nom ou du mot de passe, le système donne à l'utilisateur la possibilité de recommencer en utilisant la combinaison de touches <*ctrl–u*> si la touche <*Entrer*> n'a pas été pressée. Dans le cas contraire, la combinaison <*ctrl–c*> donne à l'utilisateur la possibilité de recommencer.

2. Mot de passe

Lors d'une première connexion, il est fortement conseillé à l'utilisateur de s'attribuer un mot de passe s-il n'en a pas. Ce mot de passe sera *chiffré* (le chiffrage est purement logiciel et non inversible). Il sera impossible de le retrouver à partir du mot chiffré, même pour le super–utilisateur (root).

Si l'utilisateur oublie son mot de passe, l'administrateur ne peut que le détruire pour lui permettre d'en définir un nouveau. Un utilisateur peut à tout moment changer son mot de passe, ou s'en attribuer un par la commande *passwd*. Lors du changement, il faut fournir l'ancien mot de passe.

```
uvci@debian9:~$ passwd
Changement du mot de passe pour uvci.
Mot de passe UNIX (actuel) :
Entrez le nouveau mot de passe UNIX :
Retapez le nouveau mot de passe UNIX :
passwd: password updated successfully
uvci@debian9:~$
```

Saisie de mot de passe sans echo

Figure 3 : Modification du mot de passe de l'utilisateur uvci

Par la suite, lors des diverses connexions de l'utilisateur, la saisie du mot de passe se fera sans écho. De plus l'administrateur peut imposer des contraintes sur le mot de passe (six caractères minimum, un caractère non alphabétique,...). Enfin lorsque le nom et le mot de passe sont corrects, *login* récupère dans le fichier */etc/passwd* toutes les informations utiles pour cet utilisateur.

```
root@debian9:/home/uvci# useradd etudiant1
root@debian9:/home/uvci# passwd etudiant1
Entrez le nouveau mot de passe UNIX :
Retapez le nouveau mot de passe UNIX :
passwd: password updated successfully
root@debian9:/home/uvci#
```

Saisie de mot de passe sans echo

Figure 4 : Création de l'utilisateur etudiant1 et son mot de passe par le super-utilisateur (root)

Attention

- L'utilisateur *root* doit obligatoirement avoir un mot de passe contenant plus de six caractères.
- Il doit être utilisé uniquement pour réaliser des modifications dans votre système.
- L'administrateur doit créer un compte personnel avec la commande *useradd*, puis *passwd* pour lui associer un mot de passe. (voir Figure 4)

3. Gestion des utilisateurs

Lorsque diverses personnes ont accès à un système, il est nécessaire que l'administrateur système gère les utilisateurs. Pour ceci, les commandes usuelles et les fichiers à configurer doivent être connus. Les fichiers importants à connaître sont les fichiers */etc/passwd* et */etc/group*.

3.1. Le fichier /etc/passwd

Toutes les informations des utilisateurs du système (login, mots de passe, ...) sont dans le fichier */etc/passwd*. Ce fichier est accessible en lecture à tous les utilisateurs et chacune de ses lignes possède le format suivants :

```
1 login : password : UID : GID : [commentaire] : répertoire : [prog_de_démarrage]
```

Les sept (7) champs séparés par le caractère ":" sont détaillés comme suit :

- login : Nom du compte de l'utilisateur
- password : Mot de passe de l'utilisateur (codé bien sûr et remplacé par un caractère x)
- UID : User ID, identifiant utilisateur. C'est un entier qui identifie l'utilisateur sous Linux.
- GID : Group ID, identifiant de groupe. C'est un entier qui identifie le groupe de l'utilisateur.
- [commentaire] : Il contient des informations sur l'utilisateur ou son nom réel.
- répertoire : C'est le répertoire courant de l'utilisateur après sa connexion au système.
- [prog_de_démarrage] : C'est la commande exécutée après connexion au système. Il s'agit généralement d'un interpréteur de commandes (shell).

```
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
_apt:x:104:65534::/nonexistent:/bin/false
avahi-autoipd:x:105:109:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
messagebus:x:106:110::/var/run/dbus:/bin/false
uvci:x:1000:1000:uvci,,,:/home/uvci:/bin/bash
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
etudiant1:x:1001:1001::/home/etudiant1:
```

Figure 5 : Extrait d'un fichier /etc/passwd

Remarque

- Les champs cités entre crochets ne sont pas obligatoires.
- Les notions de *UID* et *GID* sont importantes pour la protection des fichiers. Pour cette raison, il ne faut pas les changer inconsidérément.

3.2. Le fichier /etc/group

Un *groupe d'utilisateurs* rassemble un certain nombre d'utilisateurs pouvant facilement partager des fichiers. Ce groupe est répertorié dans le fichier */etc/group* dont chaque ligne se compose de différents champs séparés par ":" et dont le format est le suivant :

```
1 nom_de_groupe : champ_special : GID : [membre1, membre2, ...]
```

Les quatre (4) champs séparés par le caractère ":" sont détaillés comme suit :

- nom_de_groupe : le nom du groupe.
- champ_special : C'est soit un champ vide ou contenant un caractère x ou *.
- GID : Group ID, identifiant de groupe. C'est un entier qui identifie le groupe de l'utilisateur.
- [membre1, membre2, ...] : la liste des utilisateurs membres du groupe. Elle est facultative.

Remarque

- L'identifiant de groupe GID permet de faire le lien entre les fichiers */etc/group* et */etc/passwd*
- Un utilisateur a un groupe principal de rattachement et peut appartenir à d'autres groupes.

Figure 6 : Extrait d'un fichier /etc/group

Dans l'illustration Figure 6 ci-dessus, on remarque que l'utilisateur *uvci* appartient aux groupes *uvci*, *cdrom*, *floppy*, *audio*, *dip*, *video*, *plugdev*, *netdev* et *bluetooth* qu'on obtient avec la commande *groups*.

4. Déconnexion

Pour sortir de session, en mode texte, on utilise la commande *exit* suivie de la touche <*Entrer*>. La déconnexion est effective lorsque l'invite *login :* apparaît.

Figure 7 : Fermeture d'une session

5. Exercice

[Solution n°1 p 24]

Exercice

Afin de travailler sur un système GNU/Linux, un utilisateur doit s' en indiquant

- premièrement son suivi de la touche <Entrer> après l'invite login :,
- ensuite son suivi de la touche <Entrer> à la suite de l'invite :.

Exercice

Lorsque l'utilisateur saisit son mot de passe,

- ☐ il n'y a pas d'écho des caractères sur le terminal.
- ☐ il y a écho des caractères sur le terminal.
- ☐ les caractères saisis ne sont pas affichés à l'écran
- ☐ les caractères saisis sont affichés à l'écran.

Exercice

La commande utilisée pour créer l'utilisateur examen est

- ◯ groups examen
- ◯ useradd examen
- ◯ passwd examen
- ◯ examen <ctrl–u>

Exercice

La commande utilisée pour créer ou modifier le mode passe de l' utilisateur examen est

- ◯ groups examen
- ◯ useradd examen
- ◯ passwd examen
- ◯ examen <ctrl–u>

Exercice

Remplir le tableau ci-dessous avec les informations des utilisateurs de la station de travail debian9 dont un extrait du fichier /etc/passwd est le suivant :

```
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
_apt:x:104:65534::/nonexistent:/bin/false
avahi-autoipd:x:105:109:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
messagebus:x:106:110::/var/run/dbus:/bin/false
uvci:x:1000:1000:uvci,,,:/home/uvci:/bin/bash
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
etudiant1:x:1001:1001::/home/etudiant1:
```

Utilisateur	UID	GID	Repertoire
root			/root
uvci			
etudiant1			

II

Organisation et manipulation du système de fichiers

Objectifs

Gérer le système de fichier de Linux

1. Arborescence sous GNU/Linux

Contrairement au système de fichiers Windows, il n'existe pas de lecteurs A:, C:, etc...

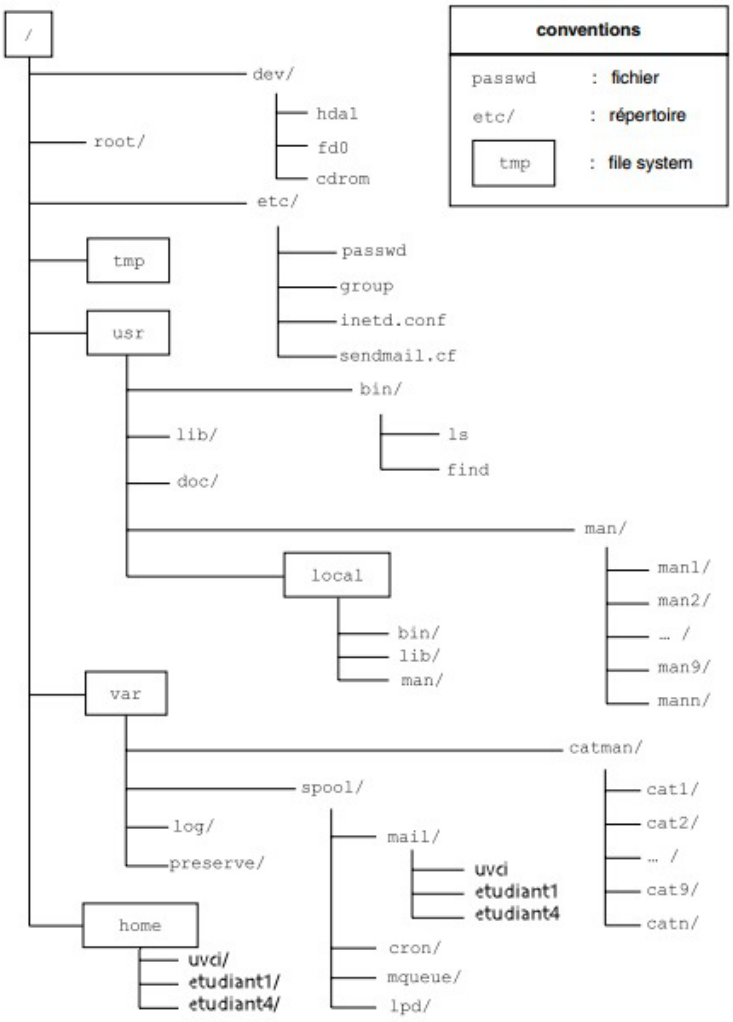


Figure 8 : Arborescence des fichiers GNU/Linux

L'entrée du système de fichiers GNU/Linux se situe à la racine, notée / . Il est organisé en une structure arborescente dont les nœuds sont des *répertoires* et les feuilles sont des *fichiers ordinaires*.

Fondamental

Une règle importante dans les systèmes Gnu/Linux est : " *TOUT EST FICHIER* ".

Le système GNU/Linux ne connaît que trois types de fichiers :

- *les fichiers ordinaires* (regular files) : Ils servent à mémoriser les programmes et les données des utilisateurs et du système.
- *les fichiers répertoires ou répertoires* (directories) : Chaque répertoire contient la liste et la référence des fichiers placés sous son contrôle et la référence du répertoire père dont il dépend.
- *les fichiers spéciaux* : Ils désignent les périphériques, les tubes ou autres supports de communication inter-processus. Les fichiers spéciaux associés aux périphériques peuvent être caractères (terminaux) ou blocs (disque). Cela signifie que les entrées/sorties (E/S) se font soit caractère par caractère, soit bloc par bloc, un bloc contenant n caractères (512, 1024 ou 2048).

2. Classification des fichiers sous GNU/Linux

La syntaxe d'un nom de fichier n'est pas très stricte.

- Il est recommandé de limiter le nom d'un fichier à *14 caractères* au plus et de n'utiliser que les *lettres majuscules* ou *minuscules*, les *chiffres* et certains caractères : le point (.), le tiret (-), le souligné (_).
- Linux autorise au minimum *un caractère* et jusqu'à *255 caractères* pour le nom du fichier.

Le point (.) joue un rôle particulier dans le nom d'un fichier.

- Les fichiers dont le nom commence par un point (.) sont des *fichiers cachés*. Ex : *.profile*
  Ils n'apparaissent pas dans la liste des fichiers en tapant la commande *ls* (sans argument).
- Le point sert également à suffixer les noms des fichiers et facilite leur gestion. Il n'y a pas de syntaxe précise mais il existe toutefois un certain nombre de conventions :

```
agri_hack.cpp  :  fichier source C++
compile.o      :  fichier binaire objet
page-test.php  :  fichier source php
fich.ext.bak | fich.ext.old | fich.ext.ori : Version initiale de fich.ext
```

Attention

- GNU/Linux différencie les majuscules des minuscules contrairement à Windows.
  Ainsi *examen.php* et *Examen.php* sont deux fichiers différents.
- Les utilisateurs ayant des claviers français (AZERTY) doivent éviter les caractères accentués.
- Les caractères spéciaux suivants sont à proscrire absolument dans le nom des fichiers:
  \ > < | $? & [ ] * ! " ‘ ( ) ` @ ~ <espace>

Certains caractères spéciaux ont une signification particulière. En voici quelques un :

```
  *    désigne toute chaîne de 0 à n caractères,
  ?    désigne un caractère quelconque,
[...]  désigne un caractère quelconque appartenant à l'ensemble entre crochets.
```

Exemple

- *fich.** : désigne tous les fichiers de nom *fich* et ayant un suffixe quelconque.
- *uvci?* : désigne tous les fichiers ayant un nom de 5 caractères dont les 4 premiers sont *uvci*.
- *[a f]* : désigne n'importe quelle lettre comprise entre a et f.
- *[a z]** : désigne tous les noms commençant par une lettre minuscule.

3. Désignation des fichiers

Un fichier est repéré par son nom et sa position dans l'arborescence : son chemin d'accès (*pathname*). La syntaxe de ce chemin d'accès est très précise et peut être décrite des deux manières suivantes.

3.1. Chemin d'accès absolu

Il permet d'accéder à un fichier quelconque dans l'arborescence du système de fichiers. Il est composé d'une suite de noms de répertoires séparés par le caractère /. Il commence toujours par le caractère / (répertoire racine) et se termine par le nom du fichier que l'on veut atteindre. La longueur du chemin d'accès absolu d'un fichier est limitée à 1024 caractères.

Exemple

```
1  /var/spool/mail/uvci
2  /home/uvci/examen.html
```

3.2. Chemin d'accès relatif

La désignation d'un fichier par son chemin d'accès absolu devient rapidement lourde avec le nombre de répertoires intermédiaires à désigner. Tout utilisateur peut se positionner sur n'importe quel répertoire de l'arborescence qui devient son répertoire courant (*current working directory*).

A partir de ce répertoire courant, l'utilisateur construit son propre sous–arbre de répertoires et de fichiers.

Exemple

```
1  Chemin absolu       :  /home/uvci/incubateur/projet/agrihack2018
2  Répertoire courant  :  /home/uvci
3  Chemin relatif      :  incubateur/projet/agrihack2018
```

Remarque

Tout répertoire contient au moins deux entrées :

```
1   .    # représente le répertoire lui-même.
2   ..   # représente le répertoire père.
```

Exemple

```
1  Répertoire courant       :  /home/uvci
2  Chemin relatif           :  ../uvci/examen/exercice_1
3  Chemin absolu equivalent :  /home/uvci/examen/exercice_1
```

4. Manipulation des répertoires

Pour bien organiser son espace de travail, il est souvent utile de grouper ses fichiers par centre d'intérêt en créant des sous-répertoires. Nous allons voir quelques commandes qui permettent de gérer les répertoires.

4.1. pwd (Print Working Directory)

La commande *pwd* affiche le chemin d'accès complet du répertoire courant.

Exemple

```
1  uvci@debian9:~$ pwd          # Résultat : /home/uvci
```

4.2. cd (Change Directory)

La commande *cd* permet de changer de répertoire de travail. Si aucun argument n'est indiqué, le contenu de la variable *HOME* est utilisé comme nouveau répertoire de travail. Sa syntaxe est :

```
cd répertoire
```

Exemple

```
uvci@debian9:~$ pwd          # Résultat : /home/uvci
uvci@debian9:~$ cd exam      # Changement du répertoire courant
uvci@debian9:~/exam$ pwd     # Résultat : /home/uvci/exam
uvci@debian9:~/exam$ cd      # Utilisation de cd sans argument
uvci@debian9:~$ pwd          # Résultat : /home/uvci
```

4.3. mkdir (MaKe DIRectory)

La commande *mkdir* crée un répertoire pour chaque nom passé en argument. La protection de ce(s) répertoire(s) est déterminée par la valeur de *umask* ou à l'aide de l'option *m*. Sa syntaxe est :

```
mkdir [option] répertoire
```

- *-m* : mode permet d'affecter la protection définie par mode au répertoire créé.
- *-p* : permet de créer *répertoire* ainsi que tous ses répertoires pères s'ils n'existaient pas.

Exemple

```
# Créer le répertoire semestre1 puis le répertoire examen dans semestre1
uvci@debian9:~$ mkdir semestre1 semestre1/examen
# Créer du répertoire semestre2 et son fils examen dans le répertoire courant
uvci@debian9:~$ mkdir -p /home/uvci/semestre2/examen
```

4.4. rmdir (ReMove DIRectory)

La commande *rmdir* permet de supprimer un répertoire. Cela n'est possible que si le répertoire est vide et si l'utilisateur n'est pas positionné dans le répertoire qu'il veut supprimer. Sa syntaxe est :

```
rmdir [option] répertoire
```

- *-p* : permet de supprimer *répertoire* ainsi que tous ses répertoires pères.

Exemple

```
# Supprimer le sous-dossier examen vide avant son répertoire père semestre1
uvci@debian9:~$ rmdir semestre1/examen semestre1
# Supprimer semestre2 contenant un sous-dossier examen vide avec l'option -p
uvci@debian9:~$ mkdir -p semestre2/examen
```

5. Manipulation des fichiers

Quel que soit le travail à faire sur un poste de travail GNU/Linux, certaines tâches élémentaires telles que lister le contenu d'un répertoire, copier, effacer, ou afficher des fichiers sont récurrentes. Nous allons dans la suite voir brièvement les commandes permettant de les réaliser.

5.1. ls (LiSt files)

La commande *ls* permet d'obtenir la liste et les caractéristiques des fichiers contenus dans un répertoire. Sans argument, la commande *ls* affiche la liste des noms des fichiers du répertoire courant par ordre alphabétique.

```
ls [option] [fichrep]
```

- *-a* : liste des fichiers cachés (par exemple : .bashrc, .profile, ...).
- *-l* : liste détaillée des caractéristiques de chaque fichier du répertoire.
- *-R* : liste récursive des répertoires et sous-répertoires à partir du répertoire courant.

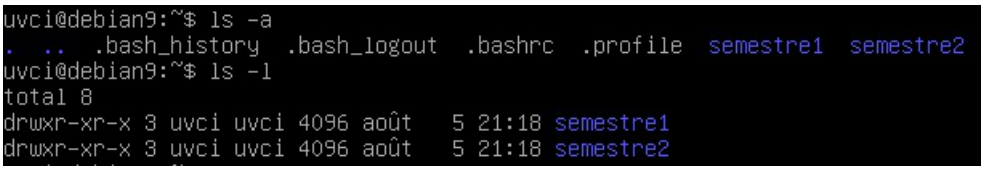


Figure 9 : Résultats des commandes ls -a et ls -l sur le répertoire /home/uvci

5.2. cat (conCATenate)

La commande *cat* est une commande multiusage qui permet d'afficher, de créer, de copier et de concaténer des fichiers. Elle lit le ou les fichiers spécifiés et les affiche sur l'écran (sortie standard). Si aucun fichier n'est spécifié, la commande *cat* lit l'entrée standard. Sa syntaxe est :

```
cat [option] [fichier]
```

- *-b* : numéroter les lignes non vides.
- *-n* : numéroter toute les lignes.
- *-s* : remplacer plusieurs lignes vides consécutives par une seule.

Exemple

```
uvci@debian9:~$ cat /etc/passwd      # Affichage du contenu de /etc/passwd
uvci@debian9:~$ cat > slogan_uvci    # Création d'un fichier
Mon université avec moi, partout et à tout moment. <Entrer> <ctrl-d>
uvci@debian9:~$ cat f1 > f2          # Copie du fichier f1 dans le fichier f2
uvci@debian9:~$ cat f1 f2 f3 > f123  # Concaténation de f1, f2 et f3 dans f123.
```

- Le caractère " $>$ " permet de mettre en œuvre le mécanisme de redirection.
- $<$*Entrer*$>$ marque la fin de la ligne et $<$*ctrl–d*$>$ est le caractère de fin de fichier.

5.3. cp (CoPy)

La commande *cp* permet de réaliser la copie de fichiers. Elle s'utilise sous trois formes :

```
cp [option] fsource  fdestination
cp [option] fsource ... répertoire
cp [option] repsource  repdestination
```

- *-i* : demande de confirmation, en cas de copie sur un fichier existant.
- *-r* : copie récursive c'est à dire copie des fichiers et sous-répertoires de *repsource* .

Attention

Le répertoire *repdestination* ne doit pas être un sous-répertoire du répertoire *repsource*.

Exemple

```
uvci@debian9:~$ cp f1 f2             # Copie du fichier f1 dans le fichier f2
uvci@debian9:~$ cp f1 semestre1      # Copie de f1 dans le répertoire semestre1
uvci@debian9:~$ mkdir sauvegarde     # Création du répertoire sauvegarde
# Copie uniquement tous les fichiers du répertoire semestre1 dans sauvegarde
uvci@debian9:~$ cp semestre1/* sauvegarde
# Copie toute l'arborescence du répertoire semestre2 dans sauvegarde
```

```
7  uvci@debian9:~$ cp -r semestre2 sauvegarde
```

5.4. mv (MoVe)

La commande *mv* permet de changer le nom d'un fichier ou de transférer un ou plusieurs fichiers dans un autre répertoire. Cette dernière opération est à utiliser avec précaution car les fichiers ne seront plus dans leur répertoire d'origine. Il est possible de transférer des répertoires vers d'autres répertoires. Pour cela, les répertoires source et destination doivent appartenir au même “file system”. Elle s'utilise sous trois formes :

```
1  mv [option] fsource  fdestination
2  mv [option] fsource ... répertoire
3  mv [option] repsource  repdestination
```

- *-f* : ne pas demander de confirmation avant d'écraser.
- *-i* : demande de confirmation. Option conseillée pour éviter d'écraser *fdestination*.

Exemple

```
1  uvci@debian9:~$ mv slogan_uvci slogan.txt  # Renomme slogan_uvci en slogan.txt
2  uvci@debian9:~$ mv sauvegarde bkp  # Renomme le repertoire sauvegarde en bkp
3  # Deplace le fichier slogan.txt dans bkp et le renomme en slogan_uvci.txt
4  uvci@debian9:~$ mv slogan.txt bkp/slogan_uvci.txt
```

5.5. rm (ReMove)

La commande *rm* permet de supprimer un ou plusieurs fichiers. Il s'agit de l'effacement du lien. Si celui-ci était unique, il y a effacement des données. Si l'accès en écriture au fichier cité n'est pas autorisé, la commande affiche les protections du fichier en mode octal et attend la confirmation de la suppression du fichier.

```
1  rm [option] fichrep ...
```

- *-f* : ne demande pas de confirmation de suppression si le droit d'écriture *w* est absent.
- *-i* : demande de confirmation de chaque suppression pour éviter un effacement par inadvertance . Option fortement conseillée.
- *-r* : suppression récursive. Si *fichrep* est un répertoire alors toute sa sous-arborescence sera supprimée.

Attention

Le droit de supprimer un fichier n'est pas lié au droit d'écriture dans le fichier mais à celui d'écriture dans le répertoire qui le contient.

Exemple

```
1  uvci@debian9:~$ rm bkp/slogan_uvci.txt  # Suppression de bkp/slogan_uvci.txt
2  uvci@debian9:~$ rm bkp/*  # Suppression de tous les fichiers du repertoire bkp
3  # Suppression du répertoire bkp et de toute sa sous-arborescence.
4  uvci@debian9:~$ rm -r bkp
```

6. Exercice

[Solution n°2 p 24]

Exercice

1. La commande pour renommer un fichier ou un répertoire est *[cat / cp / mkdir / mv]* .
2. Quelles sont les commandes permettant d'afficher le contenu d'un fichier.

Commandes	cp	cat	more	du	less	mv
Réponses	*[oui / non]*	*[oui / non]*	*[oui / non]*	*[oui / non]*	*[oui / non]*	*[oui / non]*

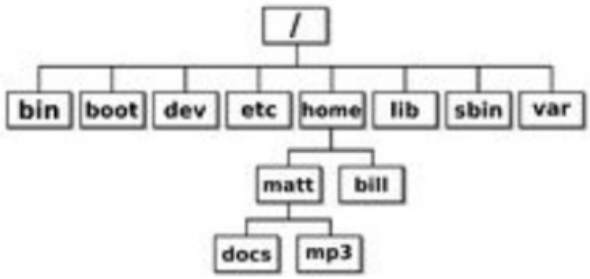
Exercice

Un est une façon de stocker les informations et de les organiser dans des sur des mémoires secondaires ou (disque dur, SSD, CD-ROM, clé USB, disquette, etc.). Dans les systèmes Gnu/Linux, tout est et on distingue :

- les qui stockent les programmes, les données des utilisateurs et du système.
- les ou qui contiennent la liste et la référence des fichiers placés sous leur contrôle et la référence de leur répertoire père.
- les qui désignent les périphériques, les tubes ou autres supports de communication inter-processus.

Exercice

On considère l'arborescence suivante d'une station de travail sur laquelle vous êtes connecté avec le login matt.



Donner la commande qui permet de :

1. Lister simplement votre répertoire courant :
2. Créer les dossiers cours et exos comme sous-répertoires de docs en utilisant des chemins relatifs:
3. Se déplacer dans le répertoire mp3 est :
4. Afficher le nouveau répertoire courant : . Le résultat de la commande sera :
5. Supprimer le répertoire exos supposé vide avec confirmation en utilisant son chemin absolu :
6. Déplacer le répertoire cours dans le dossier mp3 (répertoire courant) sans confirmation et le renommer en uvci :
7. Créer dans le répertoire courant un ficher nommé uvci.doc avec comme texte "J'aime mon université l'UVCI." :

   uvci.doc <Enter>

   <>

   <>

Droits d'accès ou permissions

Objectifs

Gérer la protection des fichiers.

1. Droits d'accès aux fichiers

À chaque fichier est associé un ensemble d'indicateurs précisant les droits d'accès à celui-ci. Pour chaque fichier il existe *trois types d'utilisateurs* :

- le *propriétaire* du fichier (*u*),
- les membres du *groupe* propriétaire du fichier (*g*)
- les *autres utilisateurs* (*others* en anglais) du système (*o*).

Pour chaque fichier et par type d'utilisateur, il existe *trois droits* principaux : autorisation d'écriture (*w*), autorisation de lecture (*r*) et autorisation d'exécution (*x*).

Sous GNU/Linux il existe différents types de fichiers dont les symboles sont entre parenthèses. Ce sont les fichiers ordinaires (-), les fichiers répertoires (*d*), les fichiers spéciaux ou périphériques accédés en mode caractère (*c*), les fichiers spéciaux ou périphériques accédés en mode bloc (*b*), les tubes nommés ou named pipes (*p*), les liens symboliques (*l*) et les sockets (*s*).

Ainsi, à chaque fichier GNU/Linux sont associés 10 attributs :

- *1* pour désigner le type et
- *9* pour la protection (*3* pour le propriétaire, *3* pour le groupe et *3* pour les autres utilisateurs)

L'ensemble des renseignements sur un fichier est obtenu en utilisant la commande `ls – l fichier`.

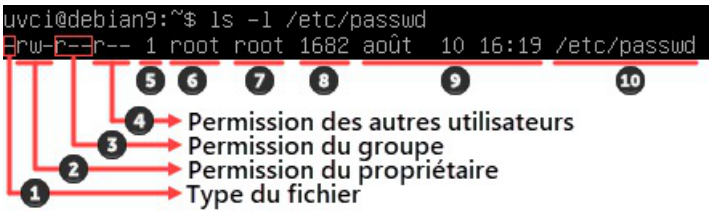


Figure 10 : Affichage des informations du fichier /etc/passwd

- La chaîne `-rw-r--r--` représente les protections du fichier */etc/passwd*.
- (*5*) représente le nombre de liens sur ce fichier (un dans l'exemple).
- (*6*) et (*7*) indiquent l'utilisateur (`root`) et le groupe (`root`) propriétaires du fichier.
- (*8*), (*9*) et (*10*) sont respectivement la taille du fichier en octets au champ , la date et de l'heure de la dernière modification du fichier et enfin le nom du fichier.

Exemple

- `c rw- rw- r–` : Fichier spécial caractère avec lecture et écriture pour le propriétaire et pour le groupe, et lecture pour les autres (par exemple les terminaux).
- `- rwx r-x r–` : Fichier ordinaire avec lecture, écriture et exécution permises pour le propriétaire, lecture et exécution pour le groupe et seulement lecture pour les autres. Il est

donc impossible aux membres du groupe et aux autres utilisateurs d'écrire dans ce fichier.

2. Droits d'accès aux répertoires

Les droits d'accès aux répertoires ont une interprétation légèrement différente de celle concernant les fichiers. Les informations d'un répertoire sont obtenues par la commande *ls – dl* repertoire.

```
uvci@debian9:~$ ls -dl /home/uvci
drwxr-xr-x 9 uvci uvci 4096 août  12 12:34 /home/uvci
```

Figure 11 : Affichage des informations du répertoire /home/uvci

L'interprétation des protections pour les répertoires est la suivante :

- *r* : autorise la lecture du contenu du répertoire comme dans le cas des fichiers; permet donc de voir la liste des fichiers qui sont dans le répertoire.
- *w* : autorise la création, la suppression et le changement du nom d'un élément du répertoire. Cette permission est indépendante de l'accès aux fichiers dans le répertoire.
- *x* : autorise l'accès au répertoire (à l'aide de la commande cd).

Exemple

Considérons les répertoires *Unix* et *Unix1* de l'utilisateur *uvci* dont les droits sont illustrés ci-dessous. Chacun d'eux contient un fichier *code.sh* avec les protections *rwx*.

```
drwxr-xr-x 4 uvci uvci 4096 août  11 06:31 Unix
dr-xr-xr-x 4 uvci uvci 4096 août  12 15:48 Unix1
```

- *Unix* a pour protection `drwxr` , seul *uvci* pourra modifier et supprimer son fichier *code.sh*
- *Unix1* a pour protection `dr-xr` , seul *uvci* pourra modifier son fichier *code.sh* mais il ne pourra pas le supprimer. En effet l'utilisateur *uvci* n'a pas l'autorisation *w* (autorisation de création, suppression, modification du nom d'un élément du répertoire).

3. Droits étendus ou bits de permission spéciale

3.1. Droit SUID

Ce droit s'applique aux fichiers exécutables. Il permet d'allouer temporairement à un utilisateur les droits du propriétaire du fichier durant son exécution. Il est utilisé lorsqu'une tâche, d'un utilisateur classique, nécessite des droits supplémentaires (ceux de root). C'est un dispositif de sécurité essentiel à utiliser avec précaution.

Il est symbolisé par *s* ou *S* qui remplace temporairement le droit *x* du propriétaire. C'est un *s* si le droit d'exécution *x* du propriétaire est présent, ou un *S* sinon. Sa valeur octale est *4000*.

Exemple

Le droit *SUID* est utilisé pour le programme */bin/mount*. `-rwsr-xr-x` correspond en octal à *4755*.

```
uvci@debian9:~$ ls -l /bin/mount
-rwsr-xr-x 1 root root 44304 mars   7 18:29 /bin/mount
```

3.2. Droit SGID

Ce droit fonctionne comme le droit *SUID*, mais appliqué aux groupes. Il donne à un utilisateur les droits du groupe auquel appartient le propriétaire de l'exécutable et non plus les droits du propriétaire. Il a une tout autre utilisation s'il est appliqué à un répertoire. Lorsqu'un fichier est créé dans un répertoire portant le droit *SGID*, alors ce fichier se verra attribuer par défaut le groupe du répertoire. De plus, si c'est un autre répertoire qui est créé dans le répertoire portant le droit *SGID* alors il portera également ce droit.

Il est symbolisé par *s* ou *S* qui remplace temporairement le droit *x* du groupe. C'est un *s* si le droit d'exécution *x* du groupe est présent, ou un *S* sinon. Sa valeur octale est *2000*.

Exemple

L'utilisateur *uvci* créé le dossier *repSGID* auquel il attribue le droit *SGID*. Ainsi ses droits `drwxr-sr-x` correspondent en octal à *2755*.

```
uvci@debian9:~$ mkdir repSGID
uvci@debian9:~$ chmod g+s repSGID
uvci@debian9:~$ ls -dl repSGID
drwxr-sr-x 2 uvci uvci 4096 août  13 05:42 repSGID
```

3.3. Sticky Bit (bit collant)

Ce droit est utilisé pour manier de façon plus subtile les droits d'écriture d'un répertoire. En effet, le droit d'écriture signifie que l'on peut créer et supprimer les fichiers de ce répertoire. Le *sticky bit* permet de faire la différence entre les deux droits. Lorsqu'il est positionné sur un répertoire, il interdit la suppression d'un fichier qu'il contient à tout utilisateur autre que le propriétaire du fichier. Pour les fichiers, il indique que ce fichier doit encore rester en mémoire vive après son exécution.

Il est symbolisé par t ou T qui remplace temporairement le droit x des autres utilisateurs. C'est un t si le droit d'exécution x des autres utilisateurs est présent, ou un T sinon. Sa valeur octale est *1000*.

Exemple

Le *sticky bit* est utilisé sur le répertoire temporaire */tmp* dont les droits `drwxrwxrwt` correspondent en octal à *1777*.

```
uvci@debian9:~$ ls -dl /tmp
drwxrwxrwt 7 root root 4096 août  13 06:25 /tmp
```

4. Opérations sur les droits d'accès

4.1. Modification des droits d'accès

La protection d'un fichier ou d'un répertoire ne peut être modifiée que par le propriétaire ou le super-utilisateur *root* à l'aide de la commande *chmod* (CHange MODe). Elle présente deux modes d'utilisation : *octal* et *symbolique*.

4.1.1. Mode octal

Ce mode utilise la description des protections par un nombre *octal*. Sa syntaxe est :

chmod droit_octal fichier

Pour l'utiliser, on associe à chacun des types de *droits* (r, w et x) un bit prenant la valeur *0* si l'utilisateur n'a pas ce droit ou *1* si l'utilisateur possède ce droit. On obtient des triplets (b b b) pour chacun des types d'utilisateurs qu'on convertit ensuite en octal.

Droits	Binaire	Octal	Commentaires
---	000	0	Aucun droit
--x	001	1	Exécutable
-w-	010	2	Écriture
-wx	011	3	Écrire et Exécuter
r--	100	4	Lire
r-x	101	5	Lire et Exécuter
rw-	110	6	Lire et Écrire
rwx	111	7	Lire, Écrire et Exécuter

Figure 12 : Valeur binaire et octale des droits d'accès

Exemple

L'utilisateur *uvci* créé un fichier *script.sh* qu'il veut être le seul à pouvoir modifier, mais que les personnes de son groupe pourront lire et que tous pourront exécuter. Que doit-il faire ?

Les droits de chaque type d'utilisateur sont résumés dans le tableau suivant :

Utilisateur	U (uvci)	G (Groupe)	O (Autres)
Droits d'accès	r w x	r – x	- - x
Valeur binaire	1 1 1	1 0 1	0 0 1
Valeur octale	7	5	1

La commande à exécuter est donc : `chmod 751 script.sh`

4.1.2. Mode symbolique

Ce mode permet une description *absolue* ou *relative* des droits d'accès. Sa syntaxe est :

chmod [*who*] *op* [*permission*] fichier

- *who* est la combinaison de `u` (user=propriétaire), `g` (groupe), `o` (other=autre) ou `a` (all=tous) pour `ugo`,
- *op* peut être l'opérateur `+` pour ajouter un droit d'accès, `–` pour supprimer un droit d'accès et `=` pour affecter un droit de manière absolue (tous les autres bits sont remis à 0)
- *permission* est la combinaison de `r` (read=lire), `w` (write=écrire) ou `x` (exécuter). Il peut s'agir aussi des permissions spéciales : `s` (pour les droits *SUID* ou *SGID*) ou `t` (pour le *Sticky Bit*).

Exemple

- *chmod u-w fichier* : supprime le droit d'écriture au propriétaire.
- *chmod g+r fichier* : ajoute le droit de lecture pour le groupe.
- *chmod ug=x fichier* : accès uniquement en exécution pour le propriétaire et le groupe.
- *chmod u+rw,g+r,o+r,a-x fichier* : ajoute les droits de lecture et écriture au propriétaire, ajoute le droit de lecture au groupe et aux autres puis supprime le droit d'exécution à tous.

4.2. Droits d'accès à la création d'un fichier

La protection d'un fichier, ainsi que le nom du propriétaire et le nom du groupe auquel il appartient, sont établis à sa création et ne peuvent être modifiés que par son propriétaire.

La commande *umask* permet de définir un masque de protection des fichiers (et répertoires) lors de leur création. Cette commande se trouve en général dans le fichier *.bashrc*, mais elle peut être exécutée à tout moment. Le masque est exprimé en octal ou sous forme symbolique.

umask [*masque*] [fichier]

Exemple : Que fait la commande umask 022 ?

La valeur *022* est soustraite de la permission permanente (*111 111 111* ou *777* en octal) :

umask 022 permet de créer des fichiers répertoires avec la protection par défaut `drwxr-xr-x`.

⚠ *Attention*

Pour les fichiers ordinaires, *umask 022* donnera une protection de type `-rw-r--r--` car la possibilité d'exécution n'est pas autorisée sur les fichiers ordinaires.

5. Exercice

[Solution n°3 p 25]

Exercice

Sous GNU/Linux, à chaque fichier sont associés attributs :

- pour désigner le et
- pour la (pour le , pour le groupe et pour les utilisateurs)

Exercice

Identifier les assertions qui sont correctes.

☐ Le SUID s'applique aux fichiers répertoires

☐ Le SGID s'applique au droit x du groupe des utilisateurs.

☐ La valeur octale du sticky Bit est 2000.

☐ Le Sticky Bit interdit la suppression du fichier d'un répertoire auquel il est appliqué à tout utilisateur autre que le propriétaire du fichier.

☐ La valeur octale du SUID est 4000

Exercice

On considère un fichier dont les droits sont : `drwxr-xr-t`.

1. *Quel est son type ?*
2. *Quel bit de permission spéciale possède t'il ?*
3. *Remplir le tableau du détail de ses permissions*
4. *Quelle est la valeur de ses droits en octal ?*

Vos réponses :

1. C'est un fichier *[ordinaire / répertoire / spécial caractère / spécial bloc / socket]* .
2. Il possède le bit de permission *[SUID / SGID / Sticky Bit / Bit collant]* de symbole et de valeur octale .
3. Tableau du détail de ses permissions

Utilisateur	U	G	O
Droits d'accès			
Valeur binaire			
Valeur octale			

4. La valeur des droits de ce fichier en octal est .

Exercice

Un utilisateur créé un fichier lance_app.sh qu'il veut être le seul à pouvoir modifier mais que les personnes de son groupe pourront lire. Il veut aussi que seul les autres utilisateurs ne pourront pas l'exécuter. De plus il souhaite lui attribuer le droit SUID.

Aider le à trouver les droits et la bonne commande à utiliser pour les attribuer.

- ☐ Les droits du fichier sont : - rws r-x ---
- ☐ Les droits du fichier sont : - rwS r-x -w-
- ☐ La valeur octale des droits du fichier est 4750
- ☐ La valeur octale des droits du fichier est 851
- ☐ La commande à saisir est chmod u+rwx, u+s, g+rx, o-rwx *lance_app.sh*
- ☐ La commande à saisir est chmod 4750 *lance_app.sh*
- ☐ La commande à saisir est chmod 2752 *lance_app.sh*

Exercice

On considère un fichier mon_script.sh dont les droits sont : - rwx r-x r-x .

Lesquelles de ces commandes permettent de supprimer le droit d'exécution à tous les types d'utilisateurs ?

- ☐ chmod 644 *mon_script.sh*
- ☐ chmod u-x,g-x,o-x *mon_script.sh*
- ☐ chmod a-x *mon_script.sh*
- ☐ chmod ogu-x *mon_script.sh*
- ☐ chmod u=rw,go=r *mon_script.sh*

Solutions des exercices

> Solution n°1

Exercice p. 9

Exercice

Afin de travailler sur un système GNU/Linux, un utilisateur doit s' identifier en indiquant

- premièrement son login suivi de la touche <Entrer> après l'invite login :,
- ensuite son mot de passe suivi de la touche <Entrer> à la suite de l'invite passwd :.

Exercice

- [x] il n'y a pas d'écho des caractères sur le terminal.
- [ ] il y a écho des caractères sur le terminal.
- [x] les caractères saisis ne sont pas affichés à l'écran
- [ ] les caractères saisis sont affichés à l'écran.

Exercice

- ( ) groups examen
- (•) useradd examen
- ( ) passwd examen
- ( ) examen <ctrl–u>

Exercice

- ( ) groups examen
- ( ) useradd examen
- (•) passwd examen
- ( ) examen <ctrl–u>

Exercice

Utilisateur	UID	GID	Repertoire
root	0	0	/root
uvci	1000	1000	/home/uvci
etudiant1	1001	1001	/home/etudiant1

> Solution n°2

Exercice p. 16

Exercice

1. La commande pour renommer un fichier ou un répertoire est mv .
2. Quelles sont les commandes permettant d'afficher le contenu d'un fichier.

Commandes	cp	cat	more	du	less	mv
Réponses	non	oui	oui	non	oui	non

Exercice

Un système de fichiers est une façon de stocker les informations et de les organiser dans des fichiers sur des mémoires secondaires ou mémoires de masse (disque dur, SSD, CD-ROM, clé USB, disquette, etc.). Dans les systèmes Gnu/Linux, tout est fichier et on distingue :

- les fichiers ordinaires qui stockent les programmes, les données des utilisateurs et du système.
- les fichiers répertoires ou répertoires qui contiennent la liste et la référence des fichiers placés sous leur contrôle et la référence de leur répertoire père.
- les fichiers spéciaux qui désignent les périphériques, les tubes ou autres supports de communication inter-processus.

Exercice

1. Lister simplement votre répertoire courant : ls
2. Créer les dossiers cours et exos comme sous-répertoires de docs en utilisant des chemins relatifs: mkdir docs/cours docs/exos
3. Se déplacer dans le répertoire mp3 est : cd mp3
4. Afficher le nouveau répertoire courant : pwd . Le résultat de la commande sera : /home/matt/mp3
5. Supprimer le répertoire exos supposé vide avec confirmation en utilisant son chemin absolu : rmdir -i /home/matt/docs/exos
6. Déplacer le répertoire cours dans le dossier mp3 (répertoire courant) sans confirmation et le renommer en uvci : mv -f /home/matt/docs/cours uvci
7. Créer dans le répertoire courant un ficher nommé uvci.doc avec comme texte "J'aime mon université l'UVCI." :

   cat > uvci.doc <Enter>

   J'aime mon université l'UVCI. < Enter >

   < ctrl-d >

> Solution n°3

Exercice p. 22

Exercice

Sous GNU/Linux, à chaque fichier sont associés 10 attributs :

- 1 pour désigner le type et
- 9 pour la protection (3 pour le propriétaire , 3 pour le groupe et 3 pour les autres utilisateurs)

Exercice

☐ Le SUID s'applique aux fichiers répertoires

☑ Le SGID s'applique au droit x du groupe des utilisateurs.

☐ La valeur octale du sticky Bit est 2000.

☑ Le Sticky Bit interdit la suppression du fichier d'un répertoire auquel il est appliqué à tout utilisateur autre que le propriétaire du fichier.

☑ La valeur octale du SUID est 4000

Exercice

Vos réponses :

1. C'est un fichier `répertoire`.
2. Il possède le bit de permission `Sticky Bit` de symbole `t` et de valeur octale `1000`.
3. Tableau du détail de ses permissions

Utilisateur	U	G	O
Droits d'accès	`r w x`	`r - x`	`r - x`
Valeur binaire	`1 1 1`	`1 0 1`	`1 0 1`
Valeur octale	`7`	`5`	`5`

4. La valeur des droits de ce fichier en octal est `1755`.

Explications :

1. C'est un *répertoire* car les permissions commencent par la lettre *d* qui symbolise les *fichiers répertoires* sous GNU/Linux.
2. Les permissions contiennent la lettre *t* à la place du droit *x* (exécution) des autres utilisateurs. Il s'agit bien du *Sticky Bit*.
3. Les droits des autres utilisateurs sont r - x car le Sticky Bit a pour valeur *t*. Si par contre le Sticky Bit avait pour valeur *T* alors ces droits seraient r - - .

Exercice

☑ Les droits du fichier sont : - rws r-x ---

☐ Les droits du fichier sont : - rwS r-x -w-

☑ La valeur octale des droits du fichier est 4750

☐ La valeur octale des droits du fichier est 851

☑ La commande à saisir est chmod u+rwx, u+s, g+rx, o-rwx *lance_app.sh*

☑ La commande à saisir est chmod 4750 *lance_app.sh*

☐ La commande à saisir est chmod 2752 *lance_app.sh*

Exercice

☑ chmod 644 *mon_script.sh*

☑ chmod u-x,g-x,o-x *mon_script.sh*

☑ chmod a-x *mon_script.sh*

☑ chmod ogu-x *mon_script.sh*

☑ chmod u=rw,go=r *mon_script.sh*