Partie I – Prise en main du terminal texte Linux

1. Le shell Linux

1.1 Qu'est qu'un shell ?

« Shell » (pour *coquille* ou *coque*) est le terme anglais utilisé pour désigner l'interface avec le système d'un système d'exploitation (OS, *Operating System*). Le dessin suivant représente les différentes couches d'un OS.

Les OS basés sur Unix disposent de deux types d'interfaces avec le système :

- une **interface graphique** (GUI, pour *Graphical User Interface*)
- et des **interfaces en ligne de commande** (CLI, pour *Commande Line Interface*).

Les deux types d'interfaces offrent à l'utilisateur à peu près les mêmes fonctionnalités (navigation dans l'arborescence du système, création, suppression, édition de répertoires et de fichiers, lancement de programmes, etc.).

Les interfaces en ligne de commande se présentent sous la forme d'une console (ou « terminal texte »). Les systèmes Linux disposent généralement de 6 terminaux texte représentés par les touches **`F1`** à **`F6`** (**`F7`** est l'interface graphique). Pour passer d'une interface à l'autre, on utilise la combinaison de touches **`Ctrl+Alt+Fn`**, où ***`n`*** est le numéro de la console où l'on veut se rendre.

1.2 Les principaux interpréteurs de commandes Linux

Les terminaux texte Linux sont basés sur un interpréteur de commandes : l'utilisateur saisit des commandes sous la forme de lignes de texte qui sont ensuite exécutées par l'interpréteur de commandes.

Les systèmes Linux disposent de plusieurs interpréteurs de commandes. Les plus courants sont :

- **`sh`** : (*Bourne Shell*, `/usr/bin/sh`) l'ancêtre de tous les interpréteurs de commandes, il est installé sur tous les OS basés sur le système Unix.
- **`bash`** : (*Bourne Again Shell*, `/usr/bin/bash`) une amélioration du Bourne Shell, est l'interpréteur de commandes par défaut de tous les OS basés sur Unix.

`sh` est plus pauvre en fonctionnalités que les autrse interpréteurs de commandes mais reste toujours plus répandu que `bash`.

Lorsqu'on ouvre un terminal texte, une ligne s'affiche qui ressemble à cela :

```
jdupont@etudiants:~ $
```

Cette ligne est une **invite de commandes** (*prompt* en anglais) qui indique que l'interpréteur de commandes est inactif et qu'il attend que l'utilisateur lui donne des commandes à exécuter. L'invite de commande se décompose ainsi :

`jdupont`	`@etudiant`	`:`	`~`	`$`
↑	↑		↑	↑
nom d'utilisateur (le login)	nom de la machine		chemin du répertoire courant	type d'utilisateur connecté

Si l'invite de commandes est :

```
jdupont@etudiants:~/Desktop/travail $
```

cela signifie que l'utilisateur se trouve actuellement dans le répertoire `/home/jdupont/Desktop/travail`.

Le symbole **$** indique que l'utilisateur actuellement connecté est un utilisateur ordinaire. Ce symbole est remplacé par un caractère **#** si l'utilisateur connecté est `root`, l'administrateur de la machine :

```
jdupont@etudiants:~ #
```

1.3 Interprétation des commandes par le shell

Lorsque l'utilisateur saisit une commande dans le terminal texte, par exemple la commande `allo`, l'interpréteur de commandes suit la procédure suivante pour l'exécuter :

(1) Si `allo` est l'une de ses commandes internes, il l'exécute puis redonne la main à l'utilisateur (en réaffichant l'invite de commandes).

Sinon, il passe à l'étape suivante.

(2) Il parcourt le contenu d'une variable d'environnement appelée **`PATH`** (nous reviendrons sur les variables d'environnement plus loin). Le contenu de **`PATH`** est une liste de répertoires dans lesquels il faut chercher les binaires correspondant à la commande à exécuter. Par exemple, si **`PATH`** contient les répertoires suivants (dans l'ordre) :

```
/usr/bin
/bin
/home/jdupont/bin
```

alors l'interpréteur de commandes va chercher les commandes :

```
/usr/bin/allo
/bin/allo
/home/jdupont/bin/allo
```

(3) Si aucun de ces répertoires ne contient une commandes `allo`, l'interpréteur de commandes affiche un message d'erreur pour indiquer à l'utilisateur qu'il ne connaît pas cette commande.

```
jdupont@etudiants:~ $ allo
allo: commande introuvable
```

Exercice 1

1. Vérifiez que votre machine a les 7 interfaces système.

2. Que représente le caractère `$` au début de la ligne de commande ?

3. Quel est le caractère qui indique que l'utilisateur actuellement connecté est l'administrateur `root` ?

4. Quel interpréteur de commandes est installé par défaut sous Linux ?

4. Qu'est-ce qu'un shell ?

- **a.** un interpréteur de commandes
- **b.** un outil de gestion du système de fichiers
- **c.** une interface en ligne de commandes
- **d.** un langage de programmation

5. À quoi sert la commande `tty` ?

6. Affichez la liste des répertoires de la variable `PATH` avec la commande `echo $PATH`. Les répertoires de la liste sont séparés par `':'`. Parcourez les différents répertoires trouvés et observez leur contenu.

7. Affichez la liste des interpréteurs de commandes installés sur votre machine (`/etc/shells`).

8. Dans l'interface grahique, ouvrez un terminal et exécutez la commande `who`.

Rendez-vous ensuite dans la première interface en ligne de commande et connectez-vous avec votre login et votre mot de passe. Exécutez la commande `who`.

Puis rendez-vous dans la troisième interface en ligne de commande et connectez-vous. Exécutez à nouveau la commande `who`.

Enfin, faites de même dans la cinquième interface en ligne de commande.

Que remarquez-vous ? Comment sont désignés les différents terminaux textes sur lesquels vous êtes connectés. Quelle ligne du résultat représente votre connexion dans l'interface graphique ?

9. Utilisez la commande `logout` pour vous déconnecter des interfaces en ligne de commande dans lesquelles vous vous êtres connecté à la question précédente (la première, la troisième et la cinquième). Revenez maintenant à l'interface graphique.

2. Les bases de la programmation shell sous Linux

2.1 Interaction avec l'utilisateur

Affichage à l'écran

`echo` : permet de réaliser des affichages à l'écran.

```
$ echo bonjour le monde !
Bonjour le monde !
$echo a\nb
anb
$echo «a\nb»
a
b
```

Certains caractères ont une signification spéciale pour l'interpréteur de commandes (par exemple, `\n` permet de provoquer un saut de ligne). Pour désactiver l'interprétation de ces caractères, on utilise la commande `echo` avec l'option `-e` et en encadrant le texte à afficher dans des guillemets (simples ou doubles).

```
$ echo «a\nb»
a\nb
$ echo -e a\nb
anb
$ echo -e 'a\nb'
a
b
$ echo -e «a\nb»
a
b
```

Les caractères d'échappement de l'interpréteur de commandes sont les suivants :

Caractère d'échappement	Signification
`\\`	Antislash
`\a`	Sonnerie
`\b`	Effacement du caractère précédent
`\c`	Suppression du saut de ligne par défaut de `echo` en fin de ligne
`\f`	Saut de page
`\n`	Saut de ligne
`\r`	Retour chariot
`\t`	Tabulation horizontale
`\v`	Tabulation verticale

Lecture des entrées du clavier

`read` : permet de lire un texte saisi par l'utilisateur.

```
$ echo -n «Entrez vos nom et prénom > »; read a b;
Entrez votren nom et votre prénom > julien dupont
$ echo «Vous avez entré:»$a $b;
Vous avez entré : julien dupont
```

La commande `echo -n « Entrez vos nom et prénom > »` affiche le texte entre guillements sans afficher de retour à la ligne ensuite. L'utilisateur entre alors les mots `julien` et `dupont` séparés par un espace. La commande `read a b` place alors ces deux mots dans les variables `a` et `b`, respectivement.

`$a` est le contenu de la variable dont le nom est `a`.

L'option **`-t`** de la commande `read` permet de limiter le temps laissé à l'utilisateur pour saisir le texte demandé, en nombre de secondes.

```
$ echo -n «Entrez votre login > »; read -t 3 a b;
Entrez votre login > jdupont
$ echo «Vous avez entré :»$a $b ;
Vous avez entré : jdupont
```

Avec la commande `read -t 3 a`, l'utilisateur a 3 secondes pour entrer son login. Après ces 3 secondes écoulées, le terminal reprend la main.

L'option **`-s`** de `read` permet de masquer le texte saisit par l'utilisateur, ce qui peut être utile pour entrer un mot de passe par exemple.

```
$ echo -n «Entrez votre mot de passe >»; read -s pwd;
Entrez votre mot de passe >
$ echo «Vous avez entré:» $pwd
Vous avez entré : jkd23dt
```

À la deuxième ligne, l'utilisateur a entré son mot de passe mais sa saisie est restée invisible.

2.2 Syntaxe des commandes

Composition d'une ligne de commandes

Une commande est composée du nom d'une commande, éventuellement suivie d'une ou plusieurs options et d'un ou plusieurs arguments.

Une ligne de commande simple : `commande [-options] [arguments]`

Une ligne de commandes séquentielles séparées par `;` : `commande1 ; commande2 ; …`

```
$ ls -lisa ~/Desktop
$ cd ~/Desktop ; pwd ;
/home/jdupont/Desktop
```

La première ligne de commande utilise la commande `ls` avec ses options `-l`, `-i`, `-s` et `-a` et avec pour argument le répertoire `~/Desktop`. La deuxième ligne de commande déplace l'utilisateur dans le répertoire `~/Desktop` avec la commande `cd` et affiche le chemin absolu de sa position avec la commande `pwd`.

Certaines commandes ont un argument par défaut.

```
$ ls
$ ls .      (ces deux premières lignes sont équivalentes)
$ cd
$ cd ~
$ cd /home/jdupont
            (ces trois lignes sont équivalentes, l'utilisateur courant est 'jdupont')
```

L'argument par défaut de la commande `ls` est `.`, le répertoire courant. L'argument par défaut de la commande `cd` est `~`, le répertoire personnel de l'utilisateur.

Options : courtes, longues

Chaque commande a une liste d'options qui lui sont spécifiques. L'ordre des options utilisées n'est pas important.

Les options courtes sont précédées d'un signe 'moins' **`-`**. Plusieurs options courtes peuvent être spécifiées après un seul **`-`**.

Chaque option longue est précédée de deux signes 'moins' **`--`**.

```
$ ls -l -R -a
$ ls -lRa
$ ls -Ral    (ces trois lignes sont équivalentes)
$ rm -f -i toto.txt
$ rm -fi toto.txt
$ rm --force --interactive toto.txt    (ces trois lignes sont équivalentes)
```

Arguments

Si on utilise plusieurs options, l'argument qui est demandé par chaque option est fourni juste après le nom de cette option : `commande -optionX argument_de_X – optionY argument_de_Y`

```
$ read -s -t 3 pwd
```

Cette commande masque la saisie de l'utilisateur (option `-s`), limite son temps de saisie à 3 secondes (option `-t` et son argument `3`) et place la saisie de l'utilisateur dans la variable `pwd`.

En cas d'ambiguïté (par exemple un argument qui commence par un caractère – comme celui qu'on place devant les options), on sépare les options des arguments par `--` (après ces deux caractères, le shell interprète toutes les chaînes de caractères comme des arguments et non des options).

```
$ rm -p -repertoire/sous-repertoire    (cette première ligne est ambigüe)
$ rm -p -- -repertoire/sous-repertoire
```

2.3 Premières commandes

Utilisateurs du système

`who` : liste les utilisateurs actuellement connectés sur le système

`who -q` : liste uniquement les noms de connexion et fait le total du nombre d'utilisateurs actuellement connectés

`who am i` : affiche uniquement la ligne concernant l'utilisateur qui est actuellement connecté

`whoami` : indique l'identité de l'utilisateur actuellement connecté

`finger` : affiche une description plus précise des utilisateurs actuellement connectés (nom, login, terminal de connexion, temps de connextion, date de connexion, etc.).

`finger <utilisateur>` : affiche encore plus d'informations sur l'utilisateur `utilisateur`.

```
$ who
jdupont    pts/0        2013-01-25 21:02 (:0)
kdurand   pts/1         2013-01-24 02:13 (:0)
```

```
$ who -q
julien root
# utilisateurs=2
$ who am i
jdupont    pts/0        2013-01-25 21:02 (:0)
$ whoami
jdupont
$ finger
Login     Name      Tty      Idle  Login Time   Office     Office Phone
marie     marie      pts/0     -     Jan 25 21:02 (:0)
$ finger jdupont
Login: jdupont                         Name: Julien Dupont
Directory: /home/jdupont                    Shell: /bin/bash
On since Fri Jan 25 21:02 (CET) on pts/0 from :0
No mail.
No Plan.
```

Déconnexion

La déconnexion d'un terminal texte se fait de trois manières différentes :

- avec la commande `exit`
- avec la commande `logout`
- en tapant la combinaison de touches `C+d`

Temps

`date` : indique la date et l'heure du système,

`date +«nous sommes le %x»` : pour formater l'affichage de la commande `date`

`cal` : affiche le calendrier

```
$ date +«nous sommes le %x»
nous sommes le 05/01/2013
$ cal 6 2013
     Juin 2013
di lu ma me je ve sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
```

2.4 Documentation

man

Le système Linux dispose d'un manuel électronique installé par défaut. Ce manuel est divisé en 9 sections principales :

Section 1 : commandes utilisateurs

Section 2 : appels système

Section 3 : bibliothèques de programmation (Perl, libc, etc.)

Section 4 : fichiers spéciaux et périphériques

Section 5 : fichiers de configuration

Section 6 : jeux

Section 7 : divers

Section 8 : commandes d'administration

Section 9 : routines noyau

`man <arg>` : cette commande permet de consulter la page du manuel électronique décrivant un élément `arg` qui est une commande, un fichier,... Cette commande recherche la page de manuel décrivant l'élément recherché `arg` en parcourant les sections du manuel dans cet ordre : 1,8,2,3,4,5,6,7,9. Ainsi, le terme recherché est d'abord comparé aux commandes existantes avant d'être comparé aux appels système et autres noms de fichiers de configuration. L'ordre de parcours de `man` dans le manuel peut être modifié dans le fichier `/etc/man.config`.

Par exemple, pour afficher le manuel d'utilisation de la commande `finger` :

```
$ man finger
```

Pour forcer la recherche dans une section précise du manuel (par exemple la section 7) :

```
$ man 7 signal
```

`apropos <arg>` : cette commande affiche les pages du manuel qui concernent l'élément `arg`.

```
$ apropos who
at.allow (5)         - determine who can submit jobs via at or batch
at.deny (5)          - determine who can submit jobs via at or batch
bsd-from (1)         - print names of those who have sent mail
from (1)             - print names of those who have sent mail
w (1)                - Show who is logged on and what they are doing.
w.procps (1)         - Show who is logged on and what they are doing.
who (1)              - show who is logged on
whoami (1)           - print effective userid
whois (1)            - client for the whois directory service
```

`--help`

Les commandes Linux disposent de l'option `--help` qui permet d'afficher la syntaxe générale de la commande et ses options les plus utilisées.

`/usr/share/doc`

Les sous-répertoire de `/usr/share/doc` contiennent une documentation détaillée des applications installées sur le compte utilisateur.

which

Cette commande cherche dans les répertoires de la variable d'environnement `PATH` le fichier exécutable correspondant au nom de son argument.

Par exemple :

```
$ which ls
```

2.5 Raccourcis clavier

Affichage

`C+l` ou `clear` : efface le contenu du terminal et repositionne l'invite de commandes sur la première ligne du terminal (sans effacer l'historique des commandes utilisées)

`M-Page suiv` : descend d'une demi-page dans l'affichage du terminal

`M-Page préc` : remonte d'une demi-page dans l'affichage du terminal

Édition

`Deb` : déplace le curseur en début de ligne

`Fin` : déplace le curseur en fin de ligne

`C+w` : efface le dernier mot

`C+u` : efface le début de la ligne de commande à partir de la position du curseur

`C+k` : efface la fin de la ligne de commande à partir de la position du curseur

Historique

`C+r` : recherche une chaîne de caractères dans l'historique de commandes. Si la chaîne est contenue dans plusieurs commandes de l'historique, une pression supplémentaire sur `C+r` permet de remonter encore dans la liste des commandes de l'historique qui contiennent cette chaîne de caractères.

`C+j` ou `Echap` : termine une recherche initiée avec `C+r` et permet à l'utilisateur de modifier la ligne de commande trouvée avant de l'exécuter

`C+g` : annule une recherche initiée avec `C+r`

Divers

`C+c` : interrompt la commande en cours sans attendre la fin de son exécution normale

`C+d` : termine la saisie au clavier attendue par une commande en lui envoyant le caractère de fin de fichier `EOF` (*End of File*)

Exercice 2

1. Quelles sont les commandes dont la syntaxe est valide parmi les suivantes :
   - **a.** `$ commande -p -o`
   - **b.** `$ commande -o-p`
   - **c.** `$ commande -po`
   - **d.** `$ commande -po arg_option_p arg_option_o`
   - **e.** `$ commande -p arg_options_o -o arg_option_p`
   - **f.** `$ commande -p --arg_option_p`
   - **g.** `$ commande arg1arg2`
   - **h.** `$ commande`
   - **i.** `$ commande arg -p -o`
2. Affichez le nombre d'utilisateurs actuellement connectés sur la machine.
3. Affichez la liste des utilisateurs connectés avec leur temps de connexion.
4. Écrivez une suite de commandes qui demande à l'utilisateur de saisir le login de l'un des utilisateurs de la machine puis affiche les informations les plus détaillées possibles sur l'utilisateur correspondant.
5. Un utilisateur peut-il utiliser la séquence `'dfhg'` comme mot de passe de connexion ?
6. Effacez rapidement les dernières commandes de votre terminal texte de manière à revenir en première ligne. Existe-t-il une autre manière de le faire ?
7. Comment faire pour rappeler la dernière commande exécutée ?
8. Affichez la chaîne de caractères `''a e i o ''` sans les guillemets, avec un espace entre `'a'` et `'e'`, deux espaces entre `'e'` et `'i'`, cinq espaces entre `'i'` et `'o'`, et 3 espaces après `'o'`. Que se passe-t-il ?
9. Affichez à présent votre nom et votre prénom séparés par une tabulation et suivis d'un retour à la ligne.
10. Affichez la date et l'heure actuelles dans ce format : `Nous sommes le <nom_jour> <num_jour> / <mois>', il est <heure> heures, <minute> minutes et <seconde> secondes`

    Par exemple : `Nous sommes le samedi 03 / 01, il est 20 heures, 15 minutes et 51 secondes`
11. Affichez le calendrier du mois de juin de l'année 2007.
12. Que fait la commande `time` ? Donnez-en trois exemples d'utilisation de complexité variable.
13. Affichez le calendrier de l'année 1759 en comptant son temps d'exécution.

    Parcourez les mois de l'année 1759. Que remarquez-vous ? Cherchez l'explication dans le `man` de la commande.
14. Recherchez dans les dernières lignes commandes celles où vous avez utilisé une commande contenant la chaîne `who` et choisissez celle que vous avez utilisée pour répondre à la question 2.
15. Quelle documentation faut-il consulter pour obtenir des informations sur l'utilisation d'une commande du système (`date`, `who`, …).
16. Quelle section du manuel électronique décrit les bibliothèques de programmation ?
17. Utilisez la commande `uname` pour afficher le nom de la machine sur laquelle vous êtes connectés ainsi que

les informations relatives à votre OS et la version du noyau du système.

18. À quoi servent les commandes `head` et `tail` ?

19. Utilisez les commandes `head` et `tail` pour extraire du fichier les lignes 10 à 20 `/etc/passwd`.

20. Quels sont les exécutables des commandes suivantes : `ls`, `cp`, `mv`, `man`, `info`, `cd`, `cron`, `chroot` ?

21. Quel fichier faut-il éditer pour modifier l'ordre de parcours de `man` dans le manuel ?

22. Affichez l'aide du shell avec la commande `man bash` et parcourez le fichier.

3. Le système de fichiers Linux

3.1 Les principaux répertoires

Le systèmes de fichiers Linux est organisé sous la forme d'une hiérarchie de répertoires (un arbre) donc la racine est **/** (*slash*). Voici les principaux répertoires que contient la racine :

/				La racine unique du système
	/boot			Le noyau Linux et d'autres fichiers lancés à l'amorçage du système
	/home			Les répertoires personnels des utilisateurs de la machie. Par exemple, les comptes utilisateurs de `jean`, `victor` et `kara` existeront dans les répertoires `/home/jean`, `/home/victor` et `/home/kara`, respectivement. Pour séparer physiquement les données utilisateurs des données système, ce répertoire peut être installé sur une partition à part.
	/root			Ce répertoire a le même rôle que le précédent mais il est réservé à l'utilisateur `root`, l'administrateur de la machine.
	/bin			(*binaires*) les exécutables de base nécessaires au fonctionnement du système (les commandes `ls` et `mkdir` par exemple)
	/sbin			(*super binaires*) toutes les commandes d'administration système essentielles (les commandes de partitionnement et de gestion des périphériques réseau, par exemple)
	/lib			(*librairies*) les binaires compilés pour Linux font appel à des bibliothèques de fonctions, ce qui permet notamment d'alléger la taille des fichiers puisque plusieurs exécutables peuvent ainsi utiliser la même portion de code contenue dans l'une de ces bibliothèques. Le répertoire `/lib` regroupe les bibliothèques utilisées par les binaires contenus dans `/bin` et dans `/sbin`.
	/opt			Paquetages d'applications supplémentaires
	/usr			Hiérarchie secondaire
		/bin, /sbin, /lib		Tous les programmes qui ne sont pas dans `/bin`, `/sbin` et `/lib`.
		/src		(*sources*) les sourcees des logiciels développés sous licence libre (GPL), qu'on peut donc modifier et recompiler. Par exemple, les sources du noyau Linux sont dans `/usr/src/linux`.
		/include		Les fichiers d'internationalisation sont dans `/usr/share/locale`, la documentation est dans `/usr/share/man`, `/usr/share/info` et `/usr/share/doc`.
		/local		Applications et documents propres à la machine locale
			/bin	Binaires des programmes locaux
			/sbin	Binaires système locaux
			/src	Fichies sources locaux
			/lib	Bibliothèques partagées locales
			/include	Fichiers d'en-tête C et C++ locaux
		/games		Tous les jeux installés sur le système
	/etc			(*et caetera*) fichiers de configuration et scripts de démarrage du système ne trouvant pas leur place dans les autres répertoires
		/rc.d		Scripts de démarrage et de contrôle des services
		/sysconfig		La configuration des périphériques
	/dev			Les périphériques connectés au système (`/dev/sda`, `/dev/cdrom`,etc.)
		/null		Répertoire poubelle vers lequel on redirige toutes les données dont on veut se débarasser (fichiers et affichages, comme les messages d'erreur produits par une commande, par exemple)

3.2 Chemins absolus, personnels et relatifs

Un chemin est une suite de répertoires et sous-répertoires séparés par le caractère **/** (*slash*).

```
repertoire/sous-repertoire/sous-sous-repertoire/...
```

Il existe trois différentes manières de spécifier le chemin d'un fichier ou d'un répertoire qui peuvent être utilisées indifféremment.

Le chemin absolu

Tout chemin absolu commence par **/** (la racine) et indique tous les répertoires qu'il faut traverser à partir de la racine pour arriver jusqu'au fichier ou répertoire en question. Le répertoire absolu d'un fichier/répertoire est donc toujours le même quelle que soit la position où l'on se trouve dans l'arborescence.

Exemple : Le chemin absolu du sous-répertoire `/include` qui est dans `/usr/local` est donc `/usr/local/include`.

Le chemin personnel

Tout chemin personnel commence par **`~utilisateur`** (qui référence le répertoire personnel `/home/utilisateur` de l'utilisateur `utilisateur`) et indique tous les répertoires qu'il faut traverser à partir de `/home/utilisateur` pour arriver jusqu'au fichier/répertoire en question. Le chemin personnel est donc différent en fonction du compte utilisateur où l'on se trouve. Attention : dans un chemin, le caractère **~** ne peut être précédé de rien d'autre.

Exemple : Le chemin personnel du sous-répertoire `/include` qui est dans `/usr/local` peut être :

Chemin personnel:	**Si je suis dans :**
`~/documents`	`/home/jean`
`~jean/documents`	`/home/kara`

Un chemin relatif

C'est un chemin qui indique la position du fichier/répertoire en question par rapport à la position où l'on se trouve dans l'arborescence. Il existe donc au moins autant de chemins relatifs d'un fichier/répertoire que de positions où l'on peut se trouver dans l'arborescence.

Chaque répertoire du système contient deux fichiers spéciaux :

- **.** : qui référence le répertoire courant
- **..** : qui référence le répertoire parent du répertoire courant

Exemple : Le chemin relatif du sous-répertoire `/include` qui est dans `/usr/local` peut être :

Chemin relatif :	**Si je suis dans :**
`usr/local/include`	`/`
`local/include`	`/usr`
`include` ou `./include`	`/usr/local`
`../include`	`/usr/local/bin`

3.3 Organisation physique des fichiers

Dans le système de fichiers Linux, chaque fichier/répertoire est représenté par :

- un **inode** (*index node*) unique : une structure qui contient toutes les informations relatives à ce fichier à l'exception de son nom
- et des **blocs de données** : qui contiennent les données stockées dans le fichier par l'utilisateur

Le système de fichiers Linux est séparé en deux parties distinctes :

- la table des inodes, dans laquelle chaque inode est identifié par un numéro
- et les blocs de données de tous les fichiers et répertoires de la table des inodes

Un fichier peut avoir plusieurs liens, donc plusieurs noms. Il suffit pour cela que tous ses noms pointent sur le même inode.

L'inode d'un fichier/répertoire contient les informations suivantes :

- type : le type de fichier, ordinaire (**-**), répertoire (**d**), lien symbolique (**l**), bloc (**b**), caractère (**c**), tube (**p**) ou socket (**s**).
- droits : les droits qu'ont les utilisateurs de la machine sur le fichier (on reviendra plus tard sur les droits d'utilisation d'un fichier)
- liens : le nombre de noms différents (liens physiques) qui pointent vers les blocs de données du fichier
- UID : (*User ID*) l'identifiant de l''utilisateur propriétaire du fichier
- GID : (*Group ID*) l'identifiant du groupe auquel appartient l'utilisateur propriétaire du fichier

  (une machine peut avoir plusieurs groupes d'utilisateurs)
- taille : la taille du fichier en nombre d'octets
- atime : la date de dernière lecture
- mtime : la date de dernière modification
- ctime : la date de dernière connexion

Les blocs de données d'un fichier contiennent les données que l'utilisateur y a stockées.

Les blocs de données d'un répertoire contiennent une table de correspondance entre les noms des fichiers/répertoires qu'il contient et leur inode.

```
$ ls -ld /etc/motd /lib /dev/sda /dev/null /etc/rc.local /dev/initctl /dev/log
drwxr-xr-x  4 root root  420 2013-01-23 21:34 /dev/input         (répertoire)
srw-rw-rw-  1 root root    0 2013-01-23 21:34 /dev/log           (socket)
crw-rw-rw-  1 root root 1, 3 2013-01-23 21:34 /dev/null          (caractère)
brw-rw----  1 root disk 8, 0 2013-01-23 21:34 /dev/sda           (bloc)
lrwxrwxrwx  1 root root   13 2011-10-28 14:27 /etc/motd -> /var/run/motd
                                                                 (lien symbolique)
-rwxr-xr-x  1 root root  306 2011-04-26 00:52 /etc/rc.local      (fichier ordinaire)
```

Dans le système de fichiers Linux, le nom d'un fichier peut être dupliqué par des pseudonymes qu'on appelle « liens ». Cela peut se faire de deux manières :

- soit en créant un nouveau lien vers les blocs de données du fichier ; c'est ce qu'on appelle **lien physique**
- soit en créant un nouveau lien vers le nom du fichier ; c'est ce qu'on appelle **lien symbolique**

3.4 Commandes d'exploration du systèmes de fichiers

Navigation

`cd` : (*change directory*) permet de changer de répertoire courant. Prend en argument le chemin du répertoire où l'on veut aller.

L'argument par défaut de cette commande est `~`, le répertoire personnel de l'utilisateur connecté dans le terminal texte en cours.

`-` est le répertoire précédent (dans l'historique)

`pwd` : (*print working directory*) affiche le chemin absolu du répertoire courant. Ne nécessite aucun argument.

Affichage

`ls` : (*list*) liste le contenu d'un répertoire. Prend en argument les chemins d'un ou plusieurs répertoires dont on veut afficher le contenu.

L'argument par défaut de cette commande est `.`, le répertoire courant.

Ses options les plus courantes sont les suivantes :

`-l` : affiche aussi le contenu de l'inode de chaque entrée du répertoire (voir la section 3.3 plus haut)

-u : affiche aussi la date de dernier accès (atime)

`-a` : affiche aussi les fichiers cachés (qui commencent par le signe `.`)

`-R` : (*recursive*) affiche aussi le contenu des sous-répertoires

`-t` : la sortie est triée par date de modification du plus ancien au plus récent

```
$ pwd
/home/jdupont/
(les deux commandes suivantes sont équivalentes)
$ ls
Bureau   Documents   Images   Musique   Public   Téléchargements   Vidéos
Workspace  tp1.txt
$ ls .
Bureau   Documents   Images   Musique   Public   Téléchargements   Vidéos
Workspace  tp1.txt
$ cd Documents
$ pwd
/home/jdupont/Documents
$ ls -l ..      (affiche le contenu du répertoire parent: /home/jdupont)
drwxr-xr-x  7 jdupont jdupont    4096 2013-01-23 23:57 Bureau
drwxr-xr-x  3 jdupont jdupont    4096 2012-12-25 11:10 Documents
drwxr-xr-x  2 jdupont jdupont    4096 2011-10-28 14:50 Images
drwxr-xr-x  3 jdupont jdupont    4096 2011-11-06 10:42 Musique
drwxr-xr-x  2 jdupont jdupont    4096 2011-10-28 14:50 Public
drwxr-xr-x  2 jdupont jdupont    4096 2012-11-21 06:24 Téléchargements
drwxr-xr-x  2 jdupont jdupont    4096 2011-10-28 14:50 Vidéos
drwxr-xr-x  3 jdupont jdupont    4096 2011-11-21 18:23 workspace
-rwxr-xr-x  3 jdupont jdupont    4096 2011-11-21 18:23 tp1.txt
```

Dans la sortie de la commande `ls -l`, chaque ligne se décompose ainsi :

d	rwxr-xr-x	2	jdupont	jdupont	4096	2011-10-28 14:50	Documents
↑	↑	↑	↑	↑	↑	↑	↑
type	droits	liens	UID	GID	taille	mtime	nom

3.5 Commandes d'édition du système de fichiers

Création/Suppresion

`mkdir` : (*make directory*) crée un répertoire ou une arborescence. Prend en argument le nom du ou des répertoires à créer

`-p` : crée aussi les répertoires parents qui n'existent pas

```
$ ls
tp1   tp2    tp3    tp4    tp5
$ mkdir -p tp6/exo1 tp6/exo2
$ ls tp6
exo1  exo2
```

`touch` : crée un nouveau fichier. Prend en argument le nom du fichier à créer

`rm` : (*remove*) supprime un fichier ou un répertoire. Prend en argument une liste d'un ou plusieurs chemins de répertoires et/ou fichiers à supprimer

`-i` : `--interactive`. Demande confirmation avant de supprimer chaque fichier/répertoire

`-R` : `--recursive`. Suppression récursive pour les répertoires (supprime aussi leurs sous-répertoires)

`-f` : `--force`. Force la suppression

`rmdir` : (*remove directory*) supprime un répertoire ou une arborescence. Prend en argument une liste d'une ou plusieurs arborescences

```
$ touch exo1.txt exo2.txt exo3.txt
$ mkdir -p exo1 exo2/q1.txt exo3/q2.txt
$ ls
exo1.txt    exo2.txt     exo3.txt     /exo1  /exo2  /exo3
$ rm -i exo1.txt exo2.txt exo3.txt
rm : supprimer fichier vide «exo1.txt» ? y
rm : supprimer fichier vide «exo2.txt» ? y
rm : supprimer fichier vide «exo2.txt» ? n
$ ls
exo3.txt    /exo1  /exo2  /exo3
```

Déplacement

`cp` : (*copy*) copier un fichier ou un répertoire. Prendre en arguments : (i) une liste d'un ou plusieurs fichiers/répertoires à copier et (ii) le chemin du répertoire de destination

`-r` : `--recursive`. Pour un répertoire, copie aussi le contenu de ses sous-répertoires

```
$ mkdir -p tp1/exo1 tp1/exo2 tp3/exo3
$ ls tp1
exo1  exo2   exo3
$ cp -r tp1 ..     (copie le contenu du répertoire 'tp1' et de ses sous-répertoires
                   dans le répertoire parent '..' du répertoire courant)
```

Lorsqu'on copie un fichier, on peut donner un nom différent à sa copie en spécifiant son nouveau nom :

```
$ cp exo1.txt ../tp1-exo1.txt          (copie le fichier 'exo1.txt' dans le
                                  répertoire parent en le renommant 'tp1-exo1.txt')
$ ls ..
tp1-exo1.txt
```

`mv` : (*move*) déplace un fichier. Prend en arguments : (i) une liste d'un ou plusieurs chemins de fichiers à déplacer et (ii) la destination du déplacement

```
$ touch exo1.txt exo2.txt
$ ls
exo1.txt    exo2.txt
$ mv exo1.txt ..          (déplace 'exo1.txt' dans le répertoire parent '..')
$ ls
exo2.txt
$ ls ..
exo1.txt
```

Exercice 3

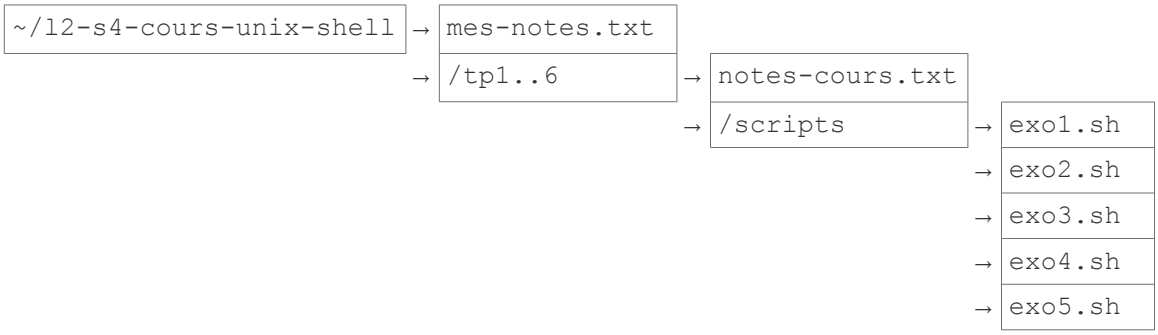
1. Quels sont les fichiers dont le nom commence par le caractère "`.`".
2. Recherchez dans votre système de fichiers au moins un fichier de chacun des types suivants :
   - **a.** un fichier ordinaire :
   - **b.** un répertoire :
   - **c.** un lien symbolique :
   - **d.** un bloc de données :
   - **e.** un caractère :
   - **f.** un tube :
   - **g.** un socket :
3. Quel est le type des chemins suivants : absolu (A), personnel (P) ou relatif (R) ?
   - **a.** `/usr/local/bin`:
   - **b.** `~/alire.txt` :
   - **c.** `/etc` :
   - **d.** `./services` :
   - **e.** `../home` :
   - **f.** `~/Desktop` :
4. Citez deux commandes qui ont un argument par défaut.
5. Que fait la commande `cd -` ?
6. Déplacez-vous dans le répertoire `/Desktop` de votre compte utilisateur. Retournez ensuite dans votre répertoire personnel (`/home/utilisateur`) sans taper son chemin. Puis redescendez dans le répertoire `/Desktop` (le répertoire où vous étiez juste avant) également sans taper son chemin.
7. Dans votre répertoire personnel, listez la totalité du contenu du répertoire courant en affichant le contenu de l'inode de chaque élément.
8. Depuis votre répertoire personnel, affichez le contenu des répertoires `/usr/local` et `/usr/share` sans vous y déplacer et en utilisant une seule commande.
9. Toujours depuis votre répertoire personnel, affichez l'arborescence du répertoire `/usr/local`.
10. Affichez les informations relatives à votre répertoire personnel sans lister son contenu.
11. Où sont stockés les noms des fichiers sur un système de fichiers Linux ?
    - **a.** dans les blocs de données réservés aux fichiers
    - **b.** dans l'inode des fichiers
    - **c.** dans les blocs de données réservés aux répertoires
    - **d.** dans l'inode des répertoires
12. À quoi servent les commandes `find` et `locate` ?
13. Donnez deux couples d'exemples d'utilisation de `find` et de `locate` qui donnent le même résultat.

14. Quelle commande permet d'afficher les informations contenuees dans l'inode d'un fichier ou répertoire ?

15. À quoi correspond l'inode de `..` ?

16. Placez-vous dans votre répertoire personnel et créez-y l'arborescence suivante en utilisant le moins de commandes possible :

Indice : Vous pourrez, entre autres, utiliser les commandes `mkdir` et `cp` avec leurs options. Les fichiers `.sh` sont vides pour le moment et seront complétés au fur et à mesure des TPs.



17. Que fait la commande `ln` ?

18. Placez-vous dans le répertoire `/l2-s4-cours-unix-shell`. Utilisez la commande `ln` pour créer un lien physique vers le fichier `mes-notes.txt` qui s'appelle `mes-notes.txt.link`.

Affichez l'inode de ces deux fichiers (avec la commande `ls`). Que remarquez-vous ?

À présent, modifiez le contenu du fichier `mes-notes.txt` puis affichez le contenu du fichier `mes-notes.txt.link`. Le fichier `mes-notes.txt.link` est-il modifié ? Qu'en concluez-vous ?

Supprimez maintenant le fichier `mes-notes.txt` et affichez le contenu de `mes-notes.txt.link`. Que se passe-t-il ?

19. Refaites la procédure de la question 13 mais en créant cette fois un lien symbolique. Quelle sont les différences ?

20. Utilisez l'option `-type` de la commande `find` pour afficher la liste des fichiers de type bloc qui se trouvent dans le système de fichiers complet de votre machine.

21. Utilisez les options `-type` et `-name` de la commande `find` pour afficher la liste des fichiers qui sont des liens symbolique et dont le nom se termine par « `.so` » dans le système de fichiers complet de votre machine.

22. Utilisez l'option `-user` de la commande `find` pour afficher la liste des fichiers ordinaires dont le nom se termine par « `.c` » dans le compte personnel de votre voisin.

23. Utilisez l'option `-links` de la commande `find` pour afficher la liste des fichiers ordinaires qui ont moins de 2 liens symboliques. Affichez ensuite la liste des fichiers ordinaires qui ont plus de 2 liens symboliques.

24. Affichez le numéro d'inode des entrées de votre répertoire personnel.

25. Que fait la commande suivante ?

```
$ ls -lSr /dev
```

26. Choisissez un fichier parmi ceux qui ont été affichés lorsque vous avez lancé la deuxième commande de la question précédente. Affichez son numéro d'inode. Utilisez ensuite l'option `-inum` de la commande `find` pour afficher la liste des liens qui pointent sur son inode.

27. Déplacez-vous dans l'arborescence `~/l2-s4-cours-unix-shell` créée à la question 16. Après chaque déplacement :

- affichez le contenu de la variable `!$` (attention : il faut saisir la commande d'affichage à chaque fois au lieu de la relancer depuis l'historique)

- utilisez cette variable pour lister le contenu du répertoire où vous êtes arrivé

28. Déplacez-vous dans l'arborescence `~/l2-s4-cours-unix-shell` créée à la question 16. Après chaque déplacement :

- listez le contenu du répertoire où vous êtes arrivé en utilisant la combinaison de touches `Echap+.` (point).

29. À quoi servent les raccourcis `Ctrl+t` et `Ctrl+w` ?

4. Droits d'accès aux fichiers et répertoires

4.1 Utilisateurs et groupes

Le système Linux est multiutilisateur. Chaque personne utilisant une machine possède un « compte utilisateur » sur le système et peut partager des fichiers avec d'autres utilisateurs de la machine. Le système Linux propose en effet la notion de « groupe d'utilisateurs ». Un utilisateur doit obligatoirement être membre d'un groupe au moins.

Les différents utilisateurs d'un système Linux sont identifiés par : (i) un numéro unique : l'UID (*User's Identifier*), (ii) un nom d'utilisateur unique (*login*) et (iii) un mot de passe unique (*password*).

Les groupes d'utilisateurs sont représentés par : (i) un nom unique et (ii) un numéro unique : le GID (*Group's Identifier*).

Lorsqu'un nouveau fichier est créé, c'est l'UID de l'utilisateur qui l'a créé et le GID de son groupe principal qui sont utilisés comme utilisateur et groupe propriétaire, respectivement.

Les droits d'utilisation d'un fichier ou répertoire sont définis pour :

utilisateurs	**notation symbolique**	**signification**
Un utilisateur	`u` (*user*)	Le propriétaire du fichier ; en principe, celui qui l'a créé
Un groupe	`g` (*group*)	Le groupe principal du propriétaire du fichier, peut être modifié par le propriétaire du fichier
Les autres	`o` (*other*)	Tout utilisateur autre que le propriétaire et qui n'est pas membre du groupe propriétaire
Tous les utilisateurs	`a` (*all*)	Toutes les catégories d'utilisateurs

On distingue trois types de comptes d'utilisateurs dans un système Linux :

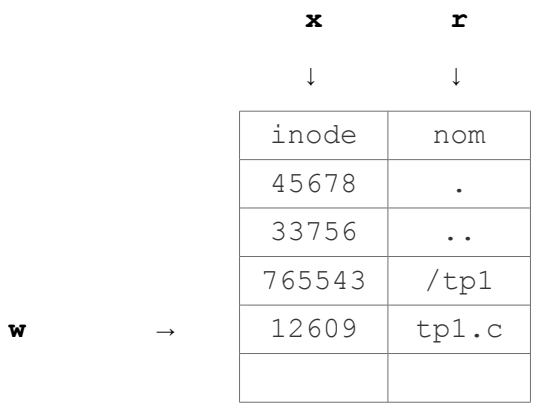
comptes	**utilisation**	**UID**
root	(« super-utilisateur) l'administrateur de la machine, il peut accéder à tout le système	0
apache, bin, daemon,...	Une série de compte qui servent à faciliter la gestion des droits d'accès de certaines applications et démons.	Entre 1 et 499
jean, sébastien, kara, ...	Les comptes utilisateurs associés à des personnes réelles	Supérieur à 499

4.2 Droits d'utilisation d'un fichier

Les droits standard

Droit	**Sur un fichier**	**Sur un répertoire**
lecture	Droit de lire le contenu du fichier	Droit de lecture sur la totalité des entrées du répertoire. Sans ce droit, on peut accéder à une entrée individuelle. On peut ainsi lire un fichier dans un répertoire même sans le droit de lecture.
écriture	Droit de modifier le contenu du fichier	Droit de modifier les entrées du répertoire, c'est-à-dire créer ou supprimer les entrées du répertoire.
exécution	Droit d'exécuter le fichier	Droit d'accéder aux entrées du répertoire. Sans ce droit, aucun accès au répertoire et à sa sous-arborescence n'est autorisée.

Pour bien comprendre les droits sur les répertoires, on peut considérer ces derniers comme une table associant les inodes aux noms des entrées du répertoire :



Pour accéder à `tp1.c` et à `/tp1`, il faut avoir les droits `r` et `x` :

- `r` permet de connaître leur nom
- `x` permet de connaître leur numéro d'inode, qui est nécessaire pour les manipuler puisque c'est l'inode qui permet d'accéder :
  - pour un fichier : à ses blocs de données
  - pour un répertoire : à ses entrées

Le droit `w` sur un répertoire permet en plus de modifier la liste de ses entrées, c'est-à-dire de créer, supprimer ou modifier ses entrées.

Les droits spéciaux

Le système Linux propose des droits supplémentaires dits « d'endossement » : **`SUID`** (pour *SetUID*) et **`SGID`** (pour *SetGID*). Ces droits donnent accès non pas à un fichier mais à une commande. Ils concernent uniquement les fichiers binaires (du code compilé) et non les scripts (à l'exception des scripts Perl).

droit	**fichier**	**répertoire**
`SUID`	Pour exécuter un fichier en endassant l'identité de son propriétaire	
`SGID`	Pour exécuter un fichier en endassant l'identité du groupe principal de son propriétaire	Le groupe des nouvelles entrées créées dans le répertoire est celui du répertoire (qui a pu être changé) au lieu du groupe principal de son propriétaire
`Sticky Bit`		Interdiction de supprimer ou de modifier le nom d'une entrée du répertoire par un autre utilisateur que son propriétaire (très utile pour le travail collaboratif)

Par exemple, la commande `passwd` permet aux utilisateurs de changer leur mot de passe que les droits d'accès au fichier `/etc/shadow` (qui contient les mots de passes des utilisateurs) sont ainsi positionnés :

```
$ ls -l /etc/shadow
-r-------- 1 root root 1130 2013-01-05 16:32 /etc/shadow
(seul root a le droit de lecture sur ce fichier)
```

Les utilisateurs ordinaires peuvent modifier leur mot de passe car les droits du fichier `/usr/bin/passwd`, le fichier binaire qui est exécuté lorsqu'on lance cette commande, a les droits suivants :

```
$ ls -l /usr/bin/passwd
-rws--x--x 1 root root 42824 2011-06-24 11:28 /usr/bin/passwd
```

Lorsqu'on lance cette commande, elle est exécutée sous l'identité de `root`, son propriétaire, grâce au droit SUID.

4.3 Gestion des droits d'utilisation d'un fichier

`umask` : cette commande permet d'afficher et de déterminer les droits d'accès par défaut (ou « masque) des fichiers créésss.

`chmod` : cette commande permet à l'utilisateur et à root de modifier les droits d'utilisation d'un fichier/répertoire. Elle peut être utilisée de deux manières, selon qu'on note les droits d'utilisateurs en notation symbolique ou en notation numérique.

Notation symbolique

La commande `chmod` (*change mode*) permet de modifier les droits d'utilisation sur un fichier/répertoire. En notation symbolique, sa syntaxe est la suivante :

```
$ chmod utilisateur[operateur]droit(s) <fichier>
```

Avec la notation symbolique, les différents droits d'utilisation sont notés ainsi :

droit	Notation symbolique	Signification
lecture	**r**	*read*
écriture	**w**	*write*
exécution	**x**	*execute*
SUID	**s** (à la place de x)	*SetUID*
	S (si x est positionné en même temps)	
SGID	**s** (à la place de x)	*SetGID*
	S (si x est positionné en même temps)	
Sticky Bit	**t**	

Les opérateurs d'affectation des droits sont les suivants :

Ajouter le(s) droit(s) spécifié(s)	**+**
Supprime le(s) droit(s) spéficié(s)	**-**
Fixe tous les droits	**=**

Exemples :

```
$ ls -l tp1.txt
-rw---x-wx  3 jdupont jdupont    4096 2011-11-21 18:23 tp1.txt
$ chmod u+x tp1.txt
(ajoute à l'utilisateur 'u' le droit d'exécution 'x' sur le fichier 'tp1.txt')
$ ls -l tp1.txt
-rwx--x-wx  3 jdupont jdupont    4096 2011-11-21 18:23 tp1.txt

$ chmod g-w toto.txt
(supprime au groupe 'g' le droit d'écriture 'w' sur le fichier)
$ ls -l tp1.txt
-rwx--x--x  3 jdupont jdupont    4096 2011-11-21 18:23 tp1.txt
```

```
$ chmod go=r toto.txt
(assigne au groupe 'g' et aux autres utilisateurs 'o' le droit de lecture 'r' uniquement, les droits d'écriture et d'exécution sont donc supprimés)
$ ls -l tp1.txt
-rwxr-xr-x  3 jdupont jdupont    4096 2011-11-21 18:23 tp1.txt

$ chmod u+rwx,g+rw,o+r toto.txt
(assigne à l'utilisateur les droits r, w et x, au groupe les droits r et w, et aux autres utilisateurs le droit r)
```

Notation numérique

En notation numérique, la syntaxe de `chmod` est la suivante :

```
$ chmod droits <fichier>
```

En notation octale, les droits de chaque utilisateur sont notés ainsi : lecture : r = 0 ou 4

écriture : w = 0 ou 2

exécution : x = 0 ou 1

(0 correspond à −, qui signifie qe l'utilisateur n'a pas le droit en question)

Avec la notation numérique, les différents droits d'utilisation sont notés ainsi :

Droits (symbolique)	Notation binaire	Notation octale
---	000	0
--x	**001**	**1**
-w-	**010**	**2**
-wx	011	3
r--	**100**	**4**
r-x	101	5
rw-	110	6
rwx	111	7
SUID		4000
SGID		2000
Sticky Bit		1000

Exemple :

les droits de l'utilisateur propriétaire : rwx = r + w + x = 4 + 2 + 1 = **7**

les droits de son groupe principal : -wx = r + w + x = 0 + 2 + 1 = **3**

les droits des autres utilisateurs : r-x = r + w + x = 4 + 0 + 1 = **5**

la commande d'assignation des droits est donc la suivante :

```
$ ls -l tp1.txt
-rw---x-wx  3 jdupont famille    4096 2011-11-21 18:23 tp1.txt
$ chmod 735 tp1.txt
$ ls -l tp1.txt
-rwx-wxr-x  3 jdupont famille    4096 2011-11-21 18:23 tp1.txt
```

Exemple :

- À partir de l'exemple précédent, on veut ajouter le droit SUID à l'utilisateur : `rws-wxr-x`

  → On ajoute 4000 aux droits précédemment calculés : `4000 + 735 = 4735`

- À partir de l'exemple précédent, on veut ajouter le droit SGID à l'utilisateur : `rwx-wsr-x`

  → On ajoute 2000 aux droits précédemment calculés : `2000 + 735 = 2735`

```
$ ls -l tp1.txt
-rw---x-wx  3 jdupont famille    4096 2011-11-21 18:23 tp1.txt
$ chmod 4735 tp1.txt
$ ls -l tp1.txt
-rws-wxr-x  3 jdupont famille    4096 2011-11-21 18:23 tp1.txt
```

4.4 Gestion des utilisateurs d'un fichier

`chown` : (*change owner*) cette commande permet de changer le propriétaire et le groupe d'un fichier

Pour changer de propriétaire : `chown <nouv_proprietaire> fichier`

```
$ ls -l tp1.txt
-rwx-wxr-x  3 jdupont famille    4096 2011-11-21 18:23 tp1.txt
$ chown root tp1.txt
$ ls -l tp1.txt
-rwx-wxr-x  3 root famille    4096 2011-11-21 18:23 tp1.txt
```

Pour changer de groupe : `chown :<nouv_groupe> fichier`

```
$ ls -l tp1.txt
-rwx-wxr-x  3 jdupont famille    4096 2011-11-21 18:23 tp1.txt
$ chown :root tp1.txt
$ ls -l tp1.txt
-rwx-wxr-x  3 jdupont root    4096 2011-11-21 18:23 tp1.txt
```

Pour changer le propriétaire et le groupe : `chown <nouv_proprietaire>:<nouv_groupe> fichier`

```
$ ls -l tp1.txt
-rwx-wxr-x  3 jdupont famille    4096 2011-11-21 18:23 tp1.txt
$ chown root:root tp1.txt
$ ls -l tp1.txt
-rwx-wxr-x  3 root   root    4096 2011-11-21 18:23 tp1.txt
```

Pour changer de propriétaire seulement si l'actuel propriétaire est un certain utilisateur, par exemple `jdupont` :

`chown --from=<actuel_proprietaire> <nouv_proprietaire> fichier`

```
$ ls -l tp1.txt
-rwx-wxr-x  3 jdupont famille    4096 2011-11-21 18:23 tp1.txt
$ chown --from=jdupont root tp1.txt
$ ls -l tp1.txt
-rwx-wxr-x  3 root famille    4096 2011-11-21 18:23 tp1.txt
```

Pour changer de groupe seulement si l'actuel groupe est un certain groupe donné, par exemple `famille` :

```
$ ls -l tp1.txt
-rwx-wxr-x  3 jdupont famille    4096 2011-11-21 18:23 tp1.txt
$ chown -from=:famille amis tp1.txt
$ ls -l tp1.txt
-rwx-wxr-x  3 jdupont amis     4096 2011-11-21 18:23 tp1.txt
```

Exercice 4

1. Utilisez les commandes `id` et `groups` pour afficher les informations en rapport avec tous les utilisateurs et groupes d'utilisateurs de votre machine. Utilisez la documentation de ces commandes pour connaître les différentes manières de les utiliser.

   Affichez la liste des groupes auxquels vous appartenez.

2. À quoi sert le compte `root` ?

3. Utilisez la commande `umask` pour afficher le masque de votre compte utilisateur.

4. Quel masque correspond aux droits suivants :

   a. `rw-rw-r--` :

   b. `rwxrwxr-x` :

   c. `-------w-` :

   d. `rwxr—r--` :

   e. `r—r-----` :

5. Un utilisateur ordinaire est identifié par :

   a. un UID égal à 0

   b. un UID supérieur ou égal à 500

   c. un masque à 022

   d. un groupe principal à `root`

6. Convertissez les droits suivants en notation octale :

   a. `rw-rw-r--` :

   b. `rwxrwxr-x` :

   c. `-------w-` :

   d. `rwxr—r--` :

   e. `r—r-----` :

7. Changez le masque de votre compte à `0002`. Créez un fichier vide `f1.txt` et un répertoire vide `r1.txt`. Quels sont les droits de chacun ?

8. Changez maintenant votre masque à `27`. Créez un fichier vide `f2.txt` et un répertoire vide `r2.txt`. Que signifie le masque `27` ?

9. Affichez les droits de `f1.txt` et de `r1.txt`. Que remarquez-vous ?

10. Donnez aux autres utilisateur les droits `r`, `w` et `x` sur `r1`. Assignez au répertoire `r2` les mêmes droits que `r1`. Créez un répertoire `r3` et donnez-lui les mêmes droits qu'à `r1` et `r2`.

11. Créez un fichier `secret` dans `r3` et modifiez ses droits de manière à ne laisser que le droit de lecture à l'utilisateur propriétaire (tous les autres droits sont supprimés).

12. Connectez-vous au premier terminal texte `tty1` avec le login et le mot de passe d'un camarade.

    - Peut-il lire le fichier `secret` ? Pourquoi ?

- Peut-il le supprimer ? Pourquoi ?

13. Que fait la commande `whereis` ? Quelles sont ses options qui permettent d'afficher uniquement les fichiers binaires ?

14. Exécutez la commande `which ls` pour afficher le fichier binaire qui est exécuté lorsqu'on saisit la commande `ls`. Créez un répertoire `/tmp` dans votre répertoire personnel et copiez-y le fichier binaire trouvé en le renommant `monls`.

Affichez les droits du fichier binaire original et de sa copie `~/tmp/monls`. Quelle est la différence ?

Ajoutez le droit UID à `~/tmp/monls`. Vous utiliserez pour cela la commande `chmod` avec la notation symbolique des droits.

Positionnez les droits d'accès au fichier `~/tmp/monls` de manière à ce que votre voisin puisse exécuter la copie du binaire. Demandez-lui d'afficher le contenu de son répertoire personnel en utilisant la copie du binaire.

15. Utilisez l'option `-perm` de la commande `find` pour afficher la liste des fichiers de votre compte personnel qui ont les droits suivants : `rwxr--r--`. Les droits doivent être exprimés en notation octale.

16. Utilisez l'option `-perm` de la commande `find` pour afficher la liste des fichiers de votre compte personnel qui ont au moins le droit SUID. Les droits doivent être exprimés en notation octale.

17. Les répertoires `/tmp` et `/var/tmp` contiennent des données temporaires. Le répertoire `/tmp` peut être purgé à n'importe quel moment. Dans la plupart des distributions, il est nettoyé à chaque démarrage du système. Les données contenues dans `/var/tmp` sont quand à elles conservées d'un démarrage à l'autre.

Même si elles sont temporaires, les entrées du répertoire `/tmp` ne doivent pas être supprimées sans raison. En effet, elles peuvent représenter des données qui sont en cours d'utilisation par une ou plusieurs applications. Leur suppression peut donc entraîner un plantage du système.

Que fait la commande suivante ?

```
find /tmp -type f -atime +14
```

Affichez les droits du répertoire `/tmp`. Que remarquez-vous ?

Créez un fichier et un répertoire quelconques dans le répertoire `/tmp`. Demandez ensuite à vos camarades de le modifier. Que se passe-t-il ? Pourquoi ?