Principes généraux

Sous UNIX quasiment tout est accessible grâce à la notion de fichier : des données stockées sur le disque dur jusqu'aux périphériques de l'ordinateur. La notion de fichier est donc centrale dans le fonctionnement du système d'exploitation.

Unix expose différents « *types* » de fichier à l'utilisateur parmi lesquels on retient notamment :

- un **répertoire** [1] (« *directory* » en anglais) contient d'autres fichiers,
- un **lien symbolique** (« *symbolic link* ») est un raccourci vers un autre fichier,
- un **fichier régulier** (« *regular file* ») contient des données dont la nature n'est, a priori, pas spécifiée ou connue par le système.

En imposant que tout fichier soit contenu dans un répertoire, Unix offre une vue arborescente du rangement des données. Le répertoire racine (« *root* » en anglais) de cette hiérarchie est identifiée par le nom `/`. Les autres fichiers sont identifiés, de manière canonique, par la liste la plus courte des répertoires à traverser depuis la racine pour les atteindre, puis par un nom. Cet identifiant est nommé le **chemin absolu** (« *absolute path* ») du fichier.

Unix permet cependant à un même fichier d'avoir plusieurs noms, dans des répertoires quelconques. Un répertoire, par exemple, contient toujours au moins deux autres fichiers : `.` qui est un second nom pour le répertoire lui-même, `..` qui est un autre nom du répertoire qui le contient. Il est ainsi possible d'identifier un fichier par un **chemin relatif**. Pour cela on ne spécifie pas la liste des répertoires à traverser depuis la racine mais depuis un répertoire quelconque.

Dans un chemin :

- les différents répertoires traversés sont séparés les uns des autres par le caractère `/` ;
- on distingue la partie répertoire (« *directory name* »), ce qui est avant le dernier `/`, du nom de base (« *basename* »), ce qui est après le dernier `/` ;
- les noms de bases peuvent généralement comporter entre 1 et 1024 caractères quelconque, sauf le caractère `/`. Unix fait une différence entre les lettres en capitales et les lettres en minuscules, de telle sorte que `toto` et `Toto` représentent deux fichiers différents.

À tout moment il existe un répertoire de travail (« *working directory* ») dans lequel Unix considère être positionné. Il est possible de spécifier un chemin en ne précisant que le nom de base d'un fichier. Dans ce cas Unix considère le chemin relatif à ce répertoire de travail.

La plupart des commandes considèrent que les fichiers dont le nom de base débute par le caractère `.` sont cachés et ne les traitent pas directement.

Un lien symbolique est un fichier qui contient uniquement un chemin vers un autre fichier. Quand elles accèdent à un tel fichier, la plupart des commandes ne traitent pas le fichier lui même mais le fichier identifié par le chemin qu'il contient. Pour être valide un lien symbolique doit donc contenir un chemin absolu ou un chemin relatif au répertoire dans lequel il est contenu.

De nombreuses commandes permettent de manipuler les fichiers via le langage de commandes (« *shell* »). Elles utilisent, et attendent donc, des chemins pour identifier les fichiers qu'elles doivent gérer. Le tableau suivant donne une liste de certaines de ces commandes. Les pages du manuel en ligne vont en donneront des détails sur l'utilisation.

Commandes	Utilité
`pwd`	afficher le nom du répertoire de travail
`cd`	changer le répertoire de travail
`ls`	lister des fichiers
`rm`	supprimer des fichiers
`mkdir`	créer des répertoires
`rmdir`	supprimer des répertoires vides
`od`	afficher les octets d'un fichier sous différents formats
`cat`	afficher le contenu de fichiers
`stat`	afficher les caractéristiques de fichiers
`file`	déterminer la convention de structure du contenu du fichier (son type applicatif)
`cp`	dupliquer des fichiers
`mv`	déplacer (renommer) des fichiers
`ln`	surnommer ou créer un « *lien symbolique* » vers un fichier

1. On dit aussi « *catalogue* » ou « *dossier* »

Sous le capot du système de fichiers

Du point de vue du cœur du système, un fichier est simplement une suite d'informations (ses « *caractéristiques* ») et de données non structurée (son « *contenu* ») stockée dans une table du système.

Il existe une telle table pour chaque périphérique (« *device* » en anglais) de stockage des données (partitions de disque dur, clé USB, etc.). Chaque espace de stockage est identifié par un nombre. Chaque table est indexée par des entiers, c'est-à-dire que chacune de ses lignes est identifiée par un nombre. Du point de vue du noyau du système, un fichier est donc identifié de manière unique par son numéro de périphérique et son index (son « *inode* ») dans la table correspondant à ce périphérique.

Le contenu d'un répertoire est simplement une liste d'inode. Chaque inode est repérée dans cette liste par un nom. Un nom est une chaîne de 1 à 1024 caractères [2] quelconques. Unix fait une différence entre les lettres en capitales (« *uppercase* » en anglais) et les lettres en minuscules (« *lowercase* » en anglais). On dit qu'Unix est sensible à la « *casse* » des caractères, de telle sorte que `toto` et `Toto` représentent deux noms différents. Cette chaîne de caractère est le nom de base d'un fichier dont l'identifiant unique pour le système est l'inode associée dans la liste.

Un répertoire, aussi appelé un « *catalogue* » ou « *dossier* » (« *directory en anglais* »), peut donc contenir plusieurs fichiers. Les noms de base des fichiers dans un répertoire donné sont cependant **uniques** : il ne peut pas y avoir deux fichiers avec le même nom dans un répertoire.

Au final un fichier est vu, par l'utilisateur, comme une entrée dans un catalogue, ou encore un lien vers une inode.

Gestion d'accès

Les droits

Chaque fichier appartient à un utilisateur, son **propriétaire**, et est rattaché à un **groupe**. À chaque fichier est attaché un mode d'accès qui spécifie, en plus du type du fichier, les droits accessibles à chacune des catégories d'utilisateur et donc les manipulations possibles sur le fichier. Trois catégories d'utilisateurs sont considérées

1. le propriétaire,
2. les membres du groupe,
3. les autres utilisateurs.

Pour chacune de ces catégories d'utilisateurs ce mode d'accès spécifie :

1. le droit de « *lecture* » du contenu du fichier (droit `r`)
2. le droit de « *modification* » du contenu du fichier (droit `w`)
3. pour les répertoires, le droit de « *franchissement* » du répertoire, c'est-à-dire le droit d'utiliser le répertoire dans un chemin ; pour les autres fichiers le droit d'« *exécuter* » le contenu du fichier, c'est-à-dire le droit de demander au système de considérer le contenu du fichier comme des instructions à exécuter (droit `x`).

Il est à noter que :

1. Pour modifier les droits d'un fichier, il faut en être le propriétaire ou être l'administrateur du système (utilisateur `root`, d'UID 0).
2. Pour accéder à un fichier, il faut avoir le droit de franchissement de (« *passage dans* ») chacun des répertoires qui constituent son chemin.
3. Pour écrire dans un fichier, il faut avoir l'autorisation de modification sur ce fichier.
4. Pour créer, détruire, déplacer, renommer, ou surnommer un fichier, il faut avoir le droit de modification sur le **répertoire** contenant ce fichier, puisqu'il s'agit d'ajouter ou de supprimer un lien dans un répertoire (une entrée dans le catalogue).

Le système utilise un nombre octal [3] sur 3 chiffres pour stocker les droits. Du poids fort au poids faible on utilise un chiffre (donc 3 bits) pour les droits accessibles :

1. au propriétaire du fichier
2. aux membres du groupe du fichier
3. aux autres utilisateurs

Les chiffres octaux correspondent aux droits à attribuer pour chacune de ces trois catégories d'attribution :

2. Cette chaîne ne peut pas contenir de caractères `/`
3. C'est-à-dire en base 8, n'utilisant que les chiffres de 0 à 7

Nombre octal	Nombre binaire	Droits équivalents
0	000	`---`
1	001	`--x`
2	010	`-w-`
3	011	`-wx`
4	100	`r--`
5	101	`r-x`
6	110	`rw-`
7	111	`rwx`

Modification

La commande `chmod` permet de modifier les droits d'accès d'un fichier :

`chmod` ⟨*mode*⟩ ⟨*chemins*⟩

Le ⟨*mode*⟩ peut être spécifié par le nombre octal codant l'intégralité des droits ou dans une version symbolique permettant une modification plus souple. En version symbolique le ⟨*mode*⟩ est composé de trois groupes de caractères. Le second groupe ne peut contenir qu'un seul caractère :

⟨*personne*⟩⟨*action*⟩⟨*accès*⟩

Personne		Action		Accès	
`u`	propriétaire	`+`	ajouter	`r`	lecture
`g`	groupe	`-`	enlever	`w`	écriture
`o`	autres	`=`	initialiser	`x`	éxécution/franchissement
`a`	tous				

Lorsqu'une commande tente de crée un fichier, elle spécifie les droits d'accès qu'elle aimerait avoir pour ce fichier. Généralement pour un répertoire elle demande l'intégralité des droits (`0777`) et pour les autres types de fichiers elle demande tous les droits sauf le droit d'exécution (`0666`). Le système n'accorde pas automatiquement tous les droits demandés. Certains seront accordés d'autres seront refusés en fonction d'un « *masque de protection* » (`umask`).

Les droits obtenus seront le résultat des droits demandés masqués par la valeur de ce masque. La commande `umask` permet de connaître ou modifier la valeur du masque courant.

Les tubes

Contrairement aux processus lancés simultanément qui s'exécutent sans relation entre eux, des processus concurrents sont synchronisés via la production d'information par l'un et la consommation d'information par l'autre. Il est nécessaire pour cela, que le processus producteur soit apte à produire des caractères sur la sortie standard et que le processus consommateur soit apte à lire des caractères sur l'entrée standard.

C'est ce que réalise la fonctionnalité des tubes (« *pipe* ») dont la syntaxe est :

```
commande1 | commande2
```

De ce fait il existe de nombreuses commandes UNIX qui profitent de ce genre de communication, notamment les « *filtres* ». Les filtres sont des programmes qui lisent des données (généralement ligne par ligne) sur leur entrée standard, y appliquent un certain nombre de manipulations avant de les retourner sur leur sortie standard.

Le tableau suivant résume un certain nombre de ces filtres. Vous obtiendrez plus d'informations en consultant les pages du manuel concernées.

`cat`	retourne les lignes lues sans modification.
`cut`	ne retourne que certaines parties de chaque lignes lues.
`grep`	retourne uniquement les lignes lues qui correspondent à un modèle particulier ou qui contiennent un mot précis.
`head`	retourne les premières lignes lues.
`more`	retourne les lignes lues par bloc (dont la taille dépend du nombre de lignes affichables par le terminal) en demandant une confirmation à l'utilisateur entre chaque bloc.
`sort`	trie les lignes lues.
`tail`	retourne les dernières lignes lues.
`tee`	envoie les données lues sur la sortie standard **ET** dans un fichier passé en paramètre.
`tr`	remplace des caractères lus par d'autres.
`uniq`	supprime les lignes consécutives identiques.
`wc`	retourne le nombre de caractères, mots et lignes lus.
`sed`	version filtre de la commande `ed` (les requêtes `ed` utilisées s'appliquent à chaque ligne lue)

Exercices

Exercice 1 : Manipulation du système de fichiers

Q 1. Créez une hiérarchie `tpfs-retour`, semblable à celle ci-dessous, dans votre répertoire de travail principal sous le répertoire `systeme`.

Les contraintes que vous devrez respecter :

- vous ne devez pas utiliser la commande `cd`,
- vous devez créer les fichiers réguliers (ceux dont la majuscule du nom est en capitale) en leur mettant comme contenu le nom du fichier en caractères minuscules.

```
systeme
‘-- tpfs
    |-- a-faire
    |   |-- Algo2
    |   |-- Anglais
    |   ‘-- Math
    ‘-- fait
        |-- Gestion
        ‘-- Algo
```

Q 2. Mettez dans un fichier nommé `reponses` du répertoire `tpfs` les réponses (une réponse par ligne) aux questions suivantes :

1. placez-vous dans le répertoire `a-faire` et donnez la commande que vous avez exécutée pour cela, `cd a-faire`
2. donnez le chemin absolu du fichier `Gestion`, `/home/infoetu/login/systeme/tpfs/fait/Gestion`
3. donnez le chemin relatif le plus court vers `Algo`, `Algo`
4. donnez le chemin relatif le plus court vers `Anglais`. `Anglais`

Q 3. Vérifiez que les chemins que vous avez spécifiés sont corrects en utilisant la commande `cat` pour visualiser leurs contenus.

```
cat /home/infoetu/login/systeme/tpfs/fait/Gestion
cat ../fait/Algo
cat Anglais
```

Le dernier exemple c’est pour insister sur le fait que le nom du fichier c’est un chemin réduit à sa plus simple expression.

Q 4. Utilisez la commande `ls` pour trouver les **inodes** des différents fichiers créés.

Il faut trouver l’option `-i` de `ls` dans `ls(1)`.

La commande `tree` peut aussi être utile.

Q 5. Sur une feuille, représentez la hiérarchie avec les différents inodes.

Q 6. Trouvez le numéro de périphérique de chacun des fichiers.

Un appel à la commande `stat` suffit largement. Comme ils sont tous sur le même périphérique (disque NFS) pas la peine de le faire sur tous.

Q 7. Sur une feuille, représentez sous forme de tableaux le contenu des répertoires `tpfs`, `a-faire` et `fait`.

Pour `tpfs`, sur ma machine ça donne, via `ls -R1i ..` :

```
tpfs:              tpfs/afaire:        tpfs/fait:
6791490 .          6799750 .           6791561 .
6791437 ..         6791490 ..          6791490 ..
6799750 afaire     6753359 Algo2       6753198 Algo
6791561 fait       6753402 Anglais     6753337 Gestion
                   6753360 Math
```

Sur les machines de TP ce sera forcément différent. Il faut surtout se souvenir que dans tout répertoire il y a `.` et `..`

Vous mettrez à jour ces tableaux en fonction des opérations demandées dans la suite.

Q 8. Réalisez la suite d’opérations demandées en vous assurant :

- que votre répertoire de travail reste `a-faire` (**vous ne devez pas utiliser la commande** `cd`),

– de mettre à jour chacun des tableaux correspondant au contenu des répertoires,
– d'ajouter une (**et une seule**) ligne dans le fichier `reponses` correspondant à la ligne de commandes que vous avez utilisée.

1. recopiez le fichier `Algo2` dans le fichier `Algorithmique`,
   `cp Algo2 Algorithmique`
   Création d'un nouvel inode/fichier au contenu identique à celui d'algo2.

```
tpfs:              tpfs/afaire:              tpfs/fait:
6791490 .          6799750 .                 6791561 .
6791437 ..         6791490 ..                6791490 ..
6799750 afaire     6753359 Algo2             6753198 Algo
6791561 fait       6753795 Algorithmique     6753337 Gestion
                   6753402 Anglais
                   6753360 Math
```

2. recopiez le fichier `Algo2` dans le fichier `Algo` du répertoire `fait`,
   `cp Algo2 ../fait/Algo`
   On écrase le contenu du fichier algo. Aucune autre modification du système de fichiers.
3. renommez le fichier `Anglais` en `English`,
   `mv Anglais English`
   Uniquement un changement du nom dans le répertoire.

```
tpfs:              tpfs/afaire:              tpfs/fait:
6791490 .          6799750 .                 6791561 .
6791437 ..         6791490 ..                6791490 ..
6799750 afaire     6753359 Algo2             6753198 Algo
6791561 fait       6753795 Algorithmique     6753337 Gestion
                   6753402 English
                   6753360 Math
```

4. déplacez le fichier `English` dans le répertoire `fait`,
   `mv English ../fait`
   Opération uniquement sur les contenus de répertoire.

```
tpfs:              tpfs/afaire:              tpfs/fait:
6791490 .          6799750 .                 6791561 .
6791437 ..         6791490 ..                6791490 ..
6799750 afaire     6753359 Algo2             6753198 Algo
6791561 fait       6753795 Algorithmique     6753402 English
                   6753360 Math              6753337 Gestion
```

5. faites en sorte que le fichier `Math` s'appelle également `Abandon` dans le répertoire `fait` (on le surnomme),
   `ln Math ../fait/Abandon`
   Création d'un lien physique. On ajoute une entrée dans le répertoire `fait` avec le même inode.

```
tpfs:              tpfs/afaire:              tpfs/fait:
6791490 .          6799750 .                 6791561 .
6791437 ..         6791490 ..                6791490 ..
6799750 afaire     6753359 Algo2             6753360 Abandon
6791561 fait       6753795 Algorithmique     6753198 Algo
                   6753360 Math              6753402 English
                                             6753337 Gestion
```

6. éditez le fichier `Abandon` et y mettre le mot `regret`,
   `vi Abandon`
   Utilisation de vi ou autre pour modifier le contenu. Aucune autre modification du système de fichiers.
7. visualisez le contenu du fichier `Math`,
   `cat Math`
   On constate le changement car on a édité un lien physique. Aucune autre modification du système de fichiers.
8. créez un lien **symbolique** nommé `Persevere` pointant sur le fichier `Abandon`,
   `ln -s ../fait/Abandon Persevere`
   Création d'un nouvel inode pour le lien symbolique.

```
tpfs:              tpfs/afaire:              tpfs/fait:
6791490 .          6799750 .                 6791561 .
6791437 ..         6791490 ..                6791490 ..
6799750 afaire     6753359 Algo2             6753360 Abandon
6791561 fait       6753795 Algorithmique     6753198 Algo
                   6753360 Math              6753402 English
                   6735517 Persevere         6753337 Gestion
```

9. visualisez le contenu du fichier **Persevere**,
   `cat Persevere`
   On constate qu'on visualise le contenu de **Abandon**. Aucune autre modification du système de fichiers.
10. supprimez le fichier **Abandon**,
    `rm ../fait/Abandon`
    L'entrée du répertoire est supprimée mais pas l'inode car il reste un lien physique.

```
tpfs:              tpfs/afaire:              tpfs/fait:
6791490 .          6799750 .                 6791561 .
6791437 ..         6791490 ..                6791490 ..
6799750 afaire     6753359 Algo2             6753198 Algo
6791561 fait       6753795 Algorithmique     6753402 English
                   6753360 Math              6753337 Gestion
                   6735517 Persevere
```

11. visualisez le contenu du fichier **Persevere**,
    `cat Persevere`
    Erreur car le lien symbolique n'est plus valide. Aucune autre modification du système de fichiers.
12. visualisez le contenu du fichier **Math**,
    Pas de changement. Aucune modification du système de fichiers.
13. créez le répertoire **plustard**.
    `mkdir plustard`

```
tpfs:              tpfs/afaire:              tpfs/fait:         tpfs/afaire/plustard:
6791490 .          6799750 .                 6791561 .          6807985 .
6791437 ..         6791490 ..                6791490 ..         6799750 ..
6799750 afaire     6753359 Algo2             6753198 Algo
6791561 fait       6753795 Algorithmique     6753402 English
                   6753360 Math              6753337 Gestion
                   6735517 Persevere
                   6807985 plustard
```

14. faites en sorte que le fichier **Math** s'appelle également **Sauvegarde** dans le répertoire **/tmp**. En vous aidant des tableaux que vous avez mis à jour, expliquez le problème et proposez une autre solution.
    `ln Math /tmp/Sauvegarde`
    Il y aura une erreur car un inode est lié à un système de fichier (un périphérique de stockage reconnu par le système, « *i.e.* » disque ou partition). Donc on ne peut pas faire de lien physique entre 2 partitions. Il faut alors faire un lien symbolique.
    `ln -s $HOME/systeme/tpfs/afaire/Math /tmp/Sauvegarde`

Q 9. Déterminez le nombre de noms du répertoire **a-faire** et à quoi ils correspondent.

a-faire dans tpfs + . dans **a-faire** + .. dans **plustard** = 3

Exercice 2 : Gestion des droits

Q 1. Placez vous dans le répertoire **tpfs**. A l'aide de la commande **id** déterminez votre groupe principal (**gid**). Vérifiez ensuite (sans vous déplacer) que le répertoire **tpfs** est accessible aux membres de votre groupe. Si ce n'est pas le cas, rendez le accessible.

Le groupe, c'est **infoetu**. Vérifier les droits x (et r) pour le groupe **infoetu** depuis le répertoire de connection. Si nécessaire modifier avec la commande **chmod**

N'utilisez plus la commande cd jusqu'à la fin du TP

Q 2. Créez un répertoire **prive** dans lequel vous créerez un fichier nommé **prive** contenant votre nom de login.

Q 3. En utilisant la forme **symbolique**, interdire l'accès au répertoire **prive** pour les membres du groupe et les autres. La commande est-elle unique ?

`chmod g-x,o-x prive`
On peut utiliser différentes combinaisons pour arriver au même résultat.

Q 4. Créez un répertoire **partage** :

- Dans ce répertoire créez un fichier `lecture` dans lequel vous mettrez votre nom de login. Ce fichier devra être consultable mais non modifiable par les membres de votre groupe principal et non lisible/modifiable par les autres. Modifiez les droits en utilisant la forme **numérique**.
  `chmod 640 lecture`
- Dans le répertoire `partage` créez un fichier `ecriture` dans lequel vous mettrez votre nom de login. Ce fichier devra être consultable et modifiable par les membres de votre groupe principal mais pas par les autres. Modifiez les droits en utilisant la forme **numérique**.
  `chmod 660 ecriture`

Q 5. Demandez à votre voisin de tester vos droits en :

- essayant de lire le contenu du fichier `prive`,
- essayant de lire puis de modifier le contenu du fichier `lecture`,
- ajoutant son nom de login à votre fichier `ecriture`,

Q 6. Modifiez les droits, en utilisant la forme numérique, du répertoire `partage` de façon à ce que les membres de votre groupe puisse y ajouter des fichiers.

`chmod 770 partage`

Q 7. Demandez à votre voisin d'ajouter un fichier portant son login comme nom dans votre répertoire `partage`.

Q 8. Essayez de modifier ce fichier en y ajoutant votre login sur la dernière ligne. Que se passe-t-il? Pourquoi?

Il manque par défaut les droits d'écriture sur les fichiers crées par d'autre.

Q 9. Essayez de supprimer ce fichier. Que se passe-t-il? Pourquoi?

Le fichier est supprimé puisque vous avez le droit de modifier le répertoire `partage` qui vous appartient.

Q 10. Demandez à votre voisin de supprimer votre fichier `lecture`. Que se passe-t-il? Pourquoi?

Aucun problème puisque les membres du groupe ont le droit de modifier le répertoire `partage`.

Q 11. Dans le répertoire `tpfs`, éditer le fichier `salut` avec le contenu suivant :

salut

```
1 echo Hello World
```

Q 12. Tentez d'exécuter ce fichier en tapant `./salut`. Modifiez ensuite les droits en utilisant la forme **symbolique** afin qu'il puisse s'exécuter. Testez à nouveau.

`chmod u+x salut`

Q 13. Déterminez votre masque courant de création de fichier.

`umask`

Par défaut ça doit être 0022.

Q 14. Dans le répertoire `tpfs` créez un répertoire `tmp`.

`mkdir tmp`

Q 15. En comparant les droits demandés (`0777`), votre masque et les droits accordés par le système, déterminez quelle expression a été utilisée pour déterminer les droits qui ont été fixé à `tmp`.

⟨*demandes*⟩ `AND NOT` ⟨*umask*⟩

Q 16. Modifiez votre masque pour que la création des fichiers permettent par défaut la modification par les membres du groupe.

`umask 0002`

Q 17. Vérifiez que votre masque est correct en créant un fichier `verif` dans le répertoire `tmp`.

Un simple `touch tmp/verif` suivi d'un `ls -l tmp/verif` devrait suffire.

Exercice 3 : Recherche de données dans les bases système

Q 1. La commande `getent passwd` permet d'afficher la liste des informations concernant les utilisateurs du système. Pour mémoire le service technique du département gère entre autres des utilisateurs étudiants du département informatique, du département génie biologique, etc. En utilisant les tubes et des filtres écrivez les commandes permettant d'effectuer les actions suivantes :

1. afficher le nombre d'utilisateurs du système.

   `getent passwd | wc -l`

2. afficher le nombre de prénoms différents parmi tous les utilisateurs.

```
getent passwd | cut -d: -f5 | cut -d. -f1 | tr '[:upper:]' '[:lower:]' | sort | uniq | wc -l
```

3. afficher le prénom le plus utilisé parmi les utilisateurs ainsi que le nombre d'utilisateurs portant ce prénom.

```
getent passwd | cut -d: -f5 | cut -d. -f1 | tr '[:upper:]' '[:lower:]' | sort | uniq -c | sort | tail -1
```

4. afficher le nombre d'étudiants inscrits au département informatique, sachant que les étudiants du département informatique ont leur répertoire principal dans `/home/infoetu`.

```
getent passwd | grep :/home/infoetu | wc -l
```

5. afficher le nombre d'étudiants inscrits au département informatique sur le terminal courant et la liste des prénoms et noms de ces étudiants sur un second terminal. Chaque ligne affichée sur ce second terminal devra comprendre le prénom d'un étudiant puis un caractère tabulation (`\t`) puis le nom de cet étudiant.

```
getent passwd | grep infoetu | cut -d: -f5 | tr . \\t | tee /dev/pts/... | wc -l
```

6. afficher la lettre la plus souvent utilisée comme première lettre d'un prénom avec sa fréquence d'utilisation.

```
getent passwd | cut -d: -f5 | cut -c1 | tr '[:upper:]' '[:lower:]' | sort | uniq -c | sort |tail -1
```

7. afficher le nombre de groupes unix principaux différents. Au département tous les utilisateurs d'un même groupe ont leur répertoire principal dans un répertoire portant le nom de ce groupe. Par exemple tous les enseignants ont leur répertoire dans `/home/infoens`.

```
getent passwd | cut -d/ -f3 | sort | uniq | wc -l
```

8. afficher le nombre d'utilisateurs homonymes (noms identiques pour 2 utilisateurs).

```
getent passwd | cut -d: -f5 | cut -d. -f 2 | tr '[:upper:]' '[:lower:]' | sort | uniq -d | wc -l
```

Q 2. La commande `last` permet d'obtenir la liste des derniers utilisateurs s'étant connectés sur la machine sur laquelle elle est exécutée. En utilisant les tubes et des filtres écrivez les commandes permettant d'effectuer les actions suivantes :

– afficher la liste des utilisateurs s'étant connecté sur la machine. Cette liste devra être triée dans l'ordre décroissant du nombre de connexion.

```
last | cut -c 1-8 | sort | uniq -dc | sort -rn | cut -c 9-
```

– afficher la liste des utilisateurs s'étant connecté juste avant un redémarrage (« *utilisateur* » `reboot`) de la machine.

```
last | grep -A 1 reboot | grep -v -e -- | grep -v reboot | cut -c 1-8 | sort | uniq
```