L'objectif de ce TP est de prendre en main l'utilisation d'un terminal, de comprendre les notions de chemins absolus et de chemins relatifs et de découvrir l'existence du manuel. **Toutes ces notions vous seront indispensables tout au long de votre BUT et il est important de les maitriser le plus rapidement possible.**

Pour ce TP, vous devez ouvrir le pdf du sujet sur votre droite, ouvrir votre terminal (Ctrl + Alt + T) sur votre gauche puis réaliser la totalité du sujet depuis votre terminal et sans utiliser votre souris.

Notes :
— Vous pouvez passer de votre terminal à votre sujet en utilisant les touches Alt + ⇆ .
— Vous pouvez vous déplacer dans le pdf en utilisant les flèches directionnelles.

Dans ce TP (et tous les suivants), lorsque vous utilisez un terminal, vous allez devoir utiliser des commandes. L'usage générique d'une commande est :

```
commande [option] [argument]
```

Les éléments entre crochets (`[]`) sont facultatifs. Chaque commande à un usage qui lui est propre et avec des options et des arguments qui lui sont aussi propres. Lorsque le sujet vous demande d'utiliser une commande en particulier, il arrivera qu'il faille ajouter un argument.

1 Premières manipulations

Exercice 1.1 : Ma première commande

L'une des commandes les plus simples d'utilisation est la commande `echo`. Son rôle est de répéter le texte qui lui est passé en argument. Elle s'utilise de la manière suivante :

```
echo <texte>
```

et affiche `<texte>`. (Les éléments entre chevrons (<>) sont à remplacer par le texte approprié.)

Q1. Utilisez la commande `echo` pour afficher le texte `toto` dans votre terminal. Affichez ensuite le texte `truc`.

2 Les dossiers

Exercice 2.1 : Trouver son dossier personnel

Votre système Unix a besoin d'un certain nombre de fichiers pour fonctionner. Il existe un dossier particulier nomé la "racine" et dont le chemin est "`/`". Tous les autres fichiers sont stockés dans différents sous-dossiers de la racine "`/`".

Q1. En utilisant la commande `ls`, observez l'ensemble des sous-dossiers de la racine "`/`". Ici `ls` est la commande et `/` est l'argument.

Dans un premier temps, nous allons nous intéresser uniquement au dossier `/home`. On considèrera que tous les autres sous-dossiers de la racine sont des dossiers propres au système d'exploitation et ne seront étudiés que plus tard.

Q2. En utilisant la commande `ls`, observez le contenu du dossier `/home`.

Le dossier `/home` contient la totalité des documents des utilisateurs de la machine. Dans le système de fichiers de l'IUT, les utilisateurs sont classés par groupes. Vous appartenez au groupe `infoetu`.

Q3. Le dossier `infoetu` est-il présent dans le dossier `/home` ?

Q4. Quel est le chemin absolu du dossier `infoetu` ?
RAPPEL : Un chemin absolu est un chemin qui part de la racine et qui par conséquent commence par "`/`".

Q5. Observez, toujours via le terminal, le contenu du dossier `infoetu`.

Dans le dossier `infoetu`, vous devriez trouver un dossier à votre nom. C'est votre "dossier personnel", aussi appelé "home". Ce dossier vous appartient et c'est dans ce dossier (et surtout ses sous-dossiers) que vous allez travailler et stocker vos données.

Q6. Donnez le chemin absolu de votre dossier personnel.
Aide : Le chemin absolu du dossier personnel de votre enseignant est `/home/infoens/julien.baste`.

Q7. Observer le contenu de votre dossier personnel.

Votre dossier personnel étant le dossier que vous allez utiliser le plus, il existe un alias pour le désigner. Cet alias est "~". Ainsi pour répondre à la question précédente et observer le contenu de votre dossier personnel, vous pouvez aussi simplement exécuter la commande "`ls ~`".

Exercice 2.2 : Organiser son dossier personnel

Votre dossier personnel étant le dossier dans lequel vous allez stocker toutes vos données, il est de première importance de le structurer avant que le désordre s'installe. (Et je vous assure que ça arrive très vite !) Pour cela, vous allez créer des dossiers. Pour la ressource R1.04, vous allez créer un dossier `R1.04` dans votre dossier personnel.

Jusqu'à maintenant, vous avez uniquement utilisé la commande `ls` suivie du chemin absolu du dossier que vous vouliez observer. Pour créer un dossier, il faut utiliser la commande `mkdir` suivie du chemin absolu du dossier que vous voulez créer. (Si cela peut vous aider, pour l'écriture du chemin, vous pouvez faire comme si le dossier existait déjà.)

Q1. En utilisant la commande `mkdir` suivie du chemin absolu vers le dossier `R1.04` que vous voulez créez, créez le dossier `R1.04` dans votre dossier personnel. En utilisant la commande `ls ~`, vérifiez que le dossier a correctement été créé, et au bon endroit.

Q2. En utilisant la commande `mkdir` suivie d'un chemin absolu, créez un dossier `TP01` dans votre nouveau dossier `R1.04`. En utilisant la commande `ls`, vérifiez que le dossier a correctement été créé, et au bon endroit.

Q3. Sur votre papier libre, dessinez la hiérarchie de fichiers à partir de votre répertoire personnel. (C'est-à-dire votre dossier personnel ainsi que tous les sous-dossiers et les autres fichiers.) Pour cela vous devrez probablement utiliser la commande `ls` plusieurs fois.

Q4. Comparez votre dessin au résultat de la commande "`tree ~`".

Exercice 2.3 : Se déplacer dans ses fichiers

Votre session bash est toujours positionnée quelque part dans votre arborescence de fichier. La commande `pwd` permet d'obtenir le chemin absolu de la position de votre session bash.

Q1. En utilisant la commande `pwd`, déterminez l'emplacement de votre session bash. (La commande `pwd` ne nécessite pas d'arguments.) À quoi correspond cet emplacement ?

Jusqu'à maintenant, nous avons utilisé uniquement des chemins absolus. Ces chemins absolus ont l'avantage de pouvoir être utilisés de la même manière, quelleque soit votre position actuelle dans l'arborescence de fichier. Cet avantage vient avec l'inconvénient que ces chemins absolus sont longs à écrire. Il va souvent être plus simple d'utiliser les chemins dits "relatifs". L'utilisation de chemins relatifs exige de savoir précisément où vous êtes dans l'arborescence de fichiers et où vous voulez aller. En particulier, avant tout usage d'un chemin relatif, l'utilisation des commandes `pwd` et `ls` sont nécessaire pour vérifier votre position et l'existence du fichier cible. Sans arguments, la commande `ls` affiche le contenu du dossier dans lequel vous êtes.

Q2. Utilisez la commande `pwd` puis la commade `ls`. Vérifiez en utilisant le résultat que vous êtes bien dans votre dossier personnel et que ce dossier contient bien le dossier `R1.04`.

La commande `cd` permet de vous déplacer dans le dossier donné en argument. Cet argument peut être soit un chemin absolu soit un chemin relatif.

Q3. En utilisant la commande `cd R1.04`, déplacez-vous dans le dossier `R1.04`.

> Lorsque vous arrivez dans un nouveau dossier, il y a deux vérifications à faire. Vous devez :
> — contrôler où vous êtes en utilisant la commande `pwd` et
> — contrôler ce qu'il y a dans le dossier en utilisant la commande `ls`.
> Ces vérifications doivent être **systématiques**.

Q4. Avez vous bien fait les vérifications d'arriver dans le nouveau dossier ?

Q5. Rendez-vous désormais dans le dossier `TP01`. N'oubliez pas de faire les vérifications à l'arrivée.

Dans chaque dossier, il existe deux sous-dossiers spéciaux : "`.`" et "`..`". Le sous-dossier "`.`" représente le dossier lui-même. Nous verrons plus tard à quoi cela peut servir d'avoir un tel sous-dossier dans chaque dossier. Le sous-dossier "`..`" représente le dossier parent. Dans notre cas, le sous-dossier "`..`" du dossier `/home/infoetu/user/R1.04/TP01` correspond au dossier `/home/infoetu/user/R1.04`. Les sous-dossiers "`..`" ont donc une importance très claire : ils permettent de remonter dans la hiérarchie de fichiers en utilisant les chemins relatifs.

Q6. En utilisant la commande "`cd ..`", retournez dans le dossier `R1.04`. N'oubliez pas de faire les vérifications à l'arrivée.

Q7. Dans le dossier `R1.04` dans lequel normalement vous êtes, créez un dossier `cours` et rendez-vous dans ce dossier. (Vous pourrez par exemple enregistrer les slides du cours dans ce dossier.) N'oubliez pas de faire les vérifications à l'arrivée.

Q8. Les chemins relatifs peuvent être légèrement plus complexes que ceux utilisés jusqu'à maintenant. Vous pouvez par exemple revenir dans le dossier parent puis aller dans le dossier `TP01` en une seule étape via la commande `cd ../TP01`. Exécutez cette commande. En utilisant une seule commande, revenez dans le dossier cours depuis le dossier `TP01`.

Q9. En utilisant la commande `cd` et un chemin absolu, déplacez-vous dans le dossier racine. N'oubliez pas de faire les vérifications à l'arrivée.

Q10. Déplacez-vous dans le dossier parent de la racine. Les vérifications d'usages vous montrent que vous êtes revenu à la racine. Pourquoi ne vous êtes vous pas déplacé ?

Q11. Lorsque vous êtes perdu, il est important de toujours savoir revenir dans son dossier personnel. En utilisant la commande `cd` et un chemin absolu, retournez dans votre dossier personnel. N'oubliez pas de faire les vérifications à l'arrivée.

3 Manipulation de fichiers réguliers

Pour manipuler des fichiers, il existe plusieurs commandes importantes :

— `touch <chemin/vers/le/nouveau/fichier>`
↳ crée un fichier vide
— `nano <chemin/vers/le/fichier>`
↳ permet d'éditer un fichier texte
— `cat <chemin/vers/le/fichier>`
↳ affiche le contenu d'un fichier
— `cp <chemin/vers/le/fichier/à/copier> <chemin/vers/le/nouveau/fichier>`
↳ copie un fichier
— `mv <chemin/vers/le/fichier/à/déplacer> <chemin/vers/le/nouvel/emplacement>`
↳ déplace un fichier dans l'arborescence de fichier (et peut aussi le renommer)
— `rm <chemin/vers/le/fichier>`
↳ supprime un fichier
— `file <chemin/vers/le/fichier>`
↳ donne la convention de structure du fichier
— `stat <chemin/vers/le/fichier>`
↳ afficher les caractéristiques de base d'un fichier
— `ln <chemin/vers/le/fichier/à/surnommer> <chemin/vers/emplacement/du/lien>`
↳ Crée une nouvelle entrée dans l'arborescence de fichier pour un fichier donné (surnomage)
— `diff <chemin/vers/le/fichier1> <chemin/vers/le/fichier2>`
↳ donne la différence entre le contenu de deux fichiers

Rappel : Le texte entre chevrons <> doit être remplacé par le texte approprié (ici des chemins vers le fichier voulu. Le but de cet exercice est de prendre en main ces commandes.

Exercice 3.1 : Créer, éditer, afficher

Q1. Rendez-vous dans le dossier `TP01` et créez un fichier vide du nom de `notes`. Avez-vous choisi un chemin absolu ou relatif pour créer votre fichier ? Pourquoi ce choix ?

Il arrivera que vous ayez besoin de modifier des fichiers textes directement depuis votre terminal. Il existe une multitude d'éditeurs de texte qui peuvent remplir cette tâche. Nous utiliserons dans ce cours `nano` qui a l'avantage principal d'être très simple d'utilisation.

Q2. Utilisez la commande `nano` pour ouvrir le fichier `notes`.

L'éditeur `nano` est désormais ouvert et est prêt à éditer le fichier nommé `notes` dans votre dossier `TP01`. L'ensemble des commandes utiles à l'utilisation de `nano` est indiqué en bas de votre terminal. Il faut comprendre `^X` comme la combinaison Ctrl + X et comprendre `M-U` comme la combinaison Alt + U . (En réalité le `M` correspond à la touche Meta qui est une touche virtuelle associée par défaut à Alt .)

Q3. Dans le fichier `notes`, écrivez "Notes sur le TP01" à l'aide de `nano`. Découvrez comment sauvegarder le texte nouvellement écrit puis quittez `nano`.

Q4. Ouvrez à nouveau le fichier `notes` en utilisant `nano` et vérifiez que votre texte a bien été enregistré.

Q5. Au fur et à mesure du TP, toujours en utilisant `nano`, prenez vos notes concernant le TP dans le fichier `notes`.

Q6. Dans votre dossier `TP01`, créez un dossier `Bidule` et un dossier `Chouette`.

Q7. Déplacez-vous dans votre dossier `Bidule`, créez-y un fichier nommé `toto` qui contient le texte `truc`.

Q8. La commande `cat` vous permet d'afficher le contenu d'un fichier. En utilisant deux lignes de commandes, affichez deux fois le contenu du fichier `toto`. La première fois en utilisant un chemin relatif et la deuxième fois en utilisant un chemin absolu. Dans les deux cas, vous devez afficher `truc`, si ce n'est pas le cas, c'est que vous vous êtes trompé dans la ligne de commandes.

Q9. Depuis le dossier `TP01`, en utilisant deux lignes de commandes, affichez deux fois le contenu du fichier `toto`. La première fois en utilisant un chemin relatif et la deuxième fois en utilisant un chemin absolu. Quelles différences observez-vous dans les chemins avec la question précédente ?

Q10. Depuis le dossier `Chouette`, en utilisant deux lignes de commandes, affichez deux fois le contenu du fichier `toto`. La première fois en utilisant un chemin relatif et la deuxième fois en utilisant un chemin absolu.

Exercice 3.2 : Renommer, copier, surnommer

Q1. Retournez dans votre dossier `Bidule`. (N'oubliez pas les vérifications standards lorsque vous arrivez dans votre dossier.)

Q2. En utilisant `stat`, notez le numéro de l'inode du fichier `toto`.

Q3. Renommez votre fichier `toto` en `tata`. Le numéro d'inode du fichier a-t-il changé ? Pourquoi ?

Q4. Déplacez le fichier `tata` dans votre dossier `Chouette`. Le numéro d'inode du fichier a-t-il changé ? Pourquoi ?

Q5. Déplacez-vous dans le dossier `Chouette`.

Q6. Copiez le fichier `tata`. La copie doit s'appeler `titi` et être placée dans le dossier `Chouette`. Le numéro d'inode de `titi` est-il le même que celui de `tata` ? Pourquoi ?

Q7. Affichez le contenu du fichier `tata` puis le contenu du fichier `titi`.

Q8. Éditez le fichier `titi` pour qu'il contienne (uniquement) le texte `machin`.

Q9. Affichez le contenu du fichier `tata` puis le contenu du fichier `titi`. Le contenu du fichier `tata` a-t-il changé ? Pourquoi ?

Q10. Surnommez le fichier `tata`. Le nouveau fichier doit s'appeler `tutu` et être placé dans le dossier `Chouette`. Le numéro d'inode de `tutu` est-il le même que celui de `tata` ? Pourquoi ?

Q11. Affichez le contenu du fichier `tata` puis le contenu du fichier `tutu`.

Q12. Éditez le fichier `tutu` pour qu'il contienne (uniquement) le texte `machin`.

Q13. Affichez le contenu du fichier `tata` puis le contenu du fichier `tutu`. Le contenu du fichier `tata` a-t-il changé ? Pourquoi ?

Q14. Copiez le fichier `tata`. La copie doit aussi s'appeler `tata` et être placée dans le dossier `Bidule`. Le numéro d'inode des deux fichiers `tata` sont-ils les mêmes ? Pourquoi ?

Q15. Affichez le contenu des deux fichiers `tata`.

Q16. Éditez le fichier `tata` présent dans le dossier `Bidule` pour qu'il contienne (uniquement) le texte `bidule`.

Q17. Affichez le contenu des deux fichiers `tata`. Le contenu du fichier `tata` du dossier `Chouette` a-t-il changé ? Pourquoi ?

Q18. Dans le dossier `Chouette`, utilisez `stat` sur les fichiers `tata` et `tutu`. Comparez leur numéro d'inode ainsi que le nombre de liens. Supprimez le fichier `tutu` puis effectuez à nouveau un `stat` sur le fichier `tata`. Que constatez-vous ?

Exercice 3.3 : Deux commandes d'affichage : `echo` et `cat`

Lorsqu'il s'agira d'afficher du texte dans votre terminal, il y a deux commandes qui reviennent très fréquemment : `echo` et `cat`. Si ces deux commandes sont des commandes d'affichage, elles ont des rôles profondément différents.

- `echo` permet de répéter le texte donné en argument.
- `cat` permet d'afficher le contenu d'un fichier (texte principalement, mais pas que).

Q1. Depuis votre dossier `Chouette`, exécutez les lignes suivantes :
- `echo tata`
- `echo truc`

— `cat tata`
— `cat truc`

Q2. Comment expliquer les résultats que vous avez obtenus ?

Q3. J'appuie sur l'importance de comprendre la différence entre `echo` et `cat`. Toute confusion pendant un devoir vous ferra échouer à la question. Vous êtes prévenus.

4 Gestion des droits

Chaque fichier est possédé par un propriétaire et un groupe. Du plus chaque personne peut avoir ou non les droits de lecture, d'écriture et d'exécution/franchissement. Les droits peuvent se résumer en un tableau. On donnera ici un exemple classique de droits sur un fichier régulier.

	Lecture (r)	Écriture (w)	Exécution franchissement (x)
Propriétaire	r	w	-
Membre du groupe	r	-	-
Autre	r	-	-

Pour des raisons techniques, on préfèrera généralement donner les droits en ligne. On donne en premier les droits du propriétaire puis les droits du groupe puis ceux des autres. L'exemple précédent donnera donc en ligne les droits `rw-r--r--`.

La commande `stat` et la commande `ls` avec l'option `-l` permettent d'obtenir ces informations sur les droits des fichiers. Il est à noter que lors de l'utilisation de ces commandes, la ligne des droits est précédée par un caractère correspondant au type de fichier. On rencontrera principalement `-` pour les fichiers réguliers, `d` pour les dossiers et `l` pour les liens. Pour un fichier ordinaire, notre exemple donne donc : `-rw-r--r--`.

Si l'on rentre dans les détails,
— le droit de lecture donne la possibilité de lire le fichier,
— le droit d'écriture permet de le modifier et
— le droit d'exécution/franchissement permet :
  — d'executer le fichier si s'est un fichier régulier ou
  — d'accéder au contenu du dossier si c'est un dossier.

Il est à noter que si vous n'avez pas les droits de franchissement sur un dossier, il vous est alors impossible de consulter et d'agir sur les fichiers contenus dans ce dossier.

Pour pouvoir modifier le mode d'utilisation d'un fichier, on utilise la commande `chmod`. Cette commande n'est utilisable que par le propriétaire du fichier. La syntaxe de `chmod` est la suivante :

`chmod` ⟨`mode`⟩ ⟨`fichiers`⟩

Les modifications à effectuer sur le mode courant sont spécifiées par un code dont la syntaxe est :

⟨personne⟩⟨action⟩⟨accès⟩

⟨personne⟩	
`u`	propriétaire
`g`	groupe
`o`	autres
`a`	tous

⟨action⟩	
`+`	ajouter
`-`	enlever
`=`	initialiser

⟨accès⟩	
`r`	lecture
`w`	écriture
`x`	exécution/franchissement

Ainsi pour retirer à tout le monde les droits de lecture d'un fichier, on utilisera `chmod a-r <fichier>`.

Exercice 4.1 : Les droits du propriétaire

Q1. Rendez-vous dans le dossier `Chouette`. N'oubliez pas les vérifications standard.

Q2. Dans ce dossier, observez les droits que vous avez sur le fichier `tata`.

Q3. Observez le contenu du fichier `tata` en utilisant `cat`, retirez-vous le droit de lecture sur ce fichier puis observez à nouveau son contenu.

Q4. Observez les droits qu'il vous reste sur le fichier `tata`. Normalement il vous reste les droits d'écriture. Tentez d'éditer le fichier avec nano. Pourquoi ne réussissez-vous pas à faire l'édition ?

Nous verrons plus tard qu'il est possible d'éditer un fichier sans avoir les droits de lecture.

Q5. Restaurez vos droits de lecture sur le fichier `tata` puis retirez les droits d'écriture.

Q6. Regardez le contenu de `tata` en utilisant `cat`. Pouvez-vous consulter le fichier ?

Q7. Ouvrez le fichier `tata` avec `nano`. Pouvez-vous ouvrir le fichier ? Que constatez-vous ?

Q8. Remettez les droits `-rw-r--r--` sur le fichier `tata`.

Q9. Testez que vous pouvez à nouveau lire le fichier et l'éditer.

Q10. Rendez-vous dans le dossier `TP01` puis observez que vous avez les droits d'exécution sur le dossier `Chouette`. Retirez ces droits d'exécution.

Q11. Utilisez la commande `ls` pour observer le contenu de `Chouette`. Que constatez-vous ?

Q12. Pouvez-vous lire le contenu du fichier `Chouette/tata` ?

Q13. Redonnez-vous les droits d'exécution sur le dossier `Chouette`.

Exercice 4.2 : Droits du groupe

Cet exercice est à faire si votre voisin en est au même stade que vous.

Q1. Rendez-vous dans le dossier personnel de votre voisin. Y avez-vous accès ?

Q2. Reprenez les questions de l'exercice précédent sauf que l'un d'entre vous effectue les modifications de droits cette fois-ci en modifiant les droits du groupe et l'autre fait les observations d'accès.

5 Le manuel

Exercice 5.1 : L'utilisation du manuel

Il existe une multitude de commandes et chaque commande à un usage qui lui est propre. Afin de savoir comment utiliser chaque commande, votre système est fourni avec un manuel. Pour pouvoir consulter le manuel d'une commande, on utilise `man` :

```
man <commande>
```

Lorsque vous êtes dans le manuel, il y a plusieurs touches à connaitre :
— flèches haut et bas pour se déplacer dans le texte.
— "`q`" pour quitter le manuel.
— "`/`" suivi d'un mot à rechercher pour chercher un mot, puis `n` pour passer à l'occurrence suivante.

NOTE : Lorsque vous lisez une page du manuel, les sections capitales à lire en entier sont **NAME**, **SYNOPSIS**, et **DESCRIPTION**. Ces sections permettent de comprendre le fonctionnement général de la commande. Les autres parties permettent de vous éclairer sur des usages particuliers de la commande et sont généralement survolées à la cherche d'une information précise.

Q1. Lisez les section name, synopsis et description de la page du manuel de `man` afin de comprendre comment le manuel est structuré.

Q2. La commande `mkdir` envoie une erreur si le dossier que vous voulez créer existe déjà. En allant chercher dans le manuel (`man mkdir`), trouvez l'option qui permet d'ignorer cette erreur.

L'option ainsi trouvée a une deuxième fonctionnalité très pratique : il permet de créer des dossiers dans des dossiers n'existant pas encore (en les créant au passage). Testez cette fonctionnalité.

Q3. Sous Unix, un fichier est caché s'il commence par un "`.`". Dans votre dossier `TP01`, en utilisant la commande `mkdir`, créez le dossier "`.toto`". Utilisez la commande `ls` pour afficher le contenu du dossier `TP01`. Le dossier `.toto` n'est pas visible. Pour pouvoir l'afficher, vous allez devoir trouver dans le manuel (`man ls`) l'option nécessaire pour que `ls` affiche les fichiers cachés (c'est-à-dire les fichiers commençant par un "`.`").

Q4. En lisant le manuel de la commande `touch`, créez un fichier dont le dernier accès est dans le futur (à la date de votre prochain anniversaire).

Q5. En lisant le manuel de la commande `echo`, trouvez comment ne pas terminer par un retour à la ligne (le prompt enchainera donc directement après le texte retourné).

Q6. En lisant le manuel de la commande `cp`, trouvez comment copier un dossier et son contenu.

Q7. En lisant le manuel de la commande `mv`, trouvez comment afficher tous les déplacements effectués par la commande lors de son exécution.

Q8. En lisant le manuel de la commande `rm`, trouvez comment supprimer un dossier. Trouvez aussi comment faire en sorte que vous n'ayez pas à valider chaque suppression.

Q9. En lisant le manuel de la commande `ln`, trouvez comment créer un lien symbolique.

Q10. En lisant le manuel de la commande `stat`, trouvez comment :
— afficher uniquement les droits d'acces au fichier.
— afficher uniquement le type de fichier.

— afficher uniquement le propriétaire du fichier.
— afficher uniquement le groupe associé au fichier.

Q11. À l'aide du manuel et en observant le résultat des lignes de commande suivantes, trouvez ce que font les commandes suivantes :

Commandes	Exemples d'utilisation
`ls`	`ls`
`cd`	`cd ~`
`pwd`	`pwd`
`exit`	`exit`
`echo`	`echo "toto"`
`cat`	`cat /etc/passwd`
`date`	`date`
`wc`	`wc /etc/passwd`
`cal`	`cal`
`which`	`which cal`
`touch`	`touch essai`
`grep`	`grep root /etc/passwd`
`cut`	`cut -f 3`
`tr`	`tr abcdefghijklmnopqrstuvwxyz cdefghijklmnopqrstuvwxyzab`
`quota`	`quota -v`

6 Toujours plus loin

Exercice 6.1 : Un petit jeu

Q1. Terminez Terminux, un petit jeu pour prendre en main tous ces concepts :
— En version courte en français : http://luffah.xyz/bidules/Terminus/
— En version longue en anglais : http://www.mprat.org/Terminus/

Exercice 6.2 : L'édition de texte

Sous Unix, il existe deux éditeurs de texte très puissant qui sont `emacs` et `vim`. Ces deux éditeurs sont relativement compliqués à prendre en main, mais leur maitrise permet de gagner énormément de temps lors de l'écriture de textes. (Sachez que ce TP est écrit en utilisant `emacs` par exemple.) La maitrise de ses éditeurs de textes vous fera gagner du temps même en dehors de l'utilisation d'`emacs` et `vim` puisqu'une partie des raccourcis d'`emacs` peuvent être utilisées directement dans un terminal et les commandes utilisées dans `vim` se retrouve lors de l'utilisation d'autres puissantes commandes tel que `sed` (que nous verrons plus tard).

Q1. Ouvrez `emacs` puis saisissez la combinaison de touche ctrl + h puis la touche t . Cela ouvre le tutoriel emacs, il ne vous reste plus qu'à le suivre.

Q2. Découvrez l'utilisation de `vim`.