Initiation aux systèmes informatiques

Version 0

IUTO

sept. 02, 2018

Contents

Chapter 1 - Prise en main

I. Premier login

Exercice 1 : Je me connecte sur les postes de l'IUT

Allumez un ordinateur et **ouvrir une session**. A quelques exceptions près, sur les postes de l'IUT votre **login** est votre *nom*.

Le mot de passe par défaut est *1a2018-19* pour les 1A et *as2018-19* pour les AS.

Puisque le mot de passe est le même pour tout le monde, la première chose à faire est de le changer.

Exercice 2 : Je change mon mot de passe sur les postes de l'IUT

Ouvrez un terminal

Applications > *Acccessoires* > *Terminal Emulator*

Vous devez voir apparaître un **prompt** qui ressemble à :

```
groot@iutinfo25-14:~$
```

Tapez la commande suivante :

```
passwd
```

Tapez ensuite le mot de passe courant (*Current Kerberos password*)

Choisissez un nouveau mot de passe (*Enter new Kerberos password*)

Comment choisir mon mot de passe ?

Un mot de passe doit être suffisamment compliqué. Il doit comporter :

- au moins 8 caractères
- au moins un caractère alpha-numérique
- majuscules et minuscules

Tapez à nouveau votre nouveau mot de passe (*Retype new Kerberos password*)

Avertissement:

Sous Linux, lorsque vous tapez un mot de passe, rien ne s'affiche à l'écran : C'EST NORMAL !

II. Je me connecte à l'ENT pour activer mon compte

Dans le terminal, tapez la commande

```
firefox
```

puis rendez vous à l'adresse suivante :

```
http://ent.univ-orleans.fr/render.userLayoutRootNode.uP
```

Activez votre compte (vous aurez besoin de votre carte d'étudiant)

Avertissement:
Vous avez donc 2 logins et 2 mots de passe à retenir :

- Un login/mot de passe pour les postes du département informatique de l'IUT
- un login/mot de passe pour accéder à l'ent

III. Quelques sites à mettre en favori

Rappel : pour ouvrir un navigateur, ouvrez un terminal puis tapez la commande :

```
firefox
```

Pour le module M1101 - Introduction à Unix

Les documents utilisés dans le module M1101 :

- http://info.iut45.univ-orleans.fr/docs/m1101/fr/latest/

En particulier, vous trouverez sur ce site cette feuille de TP en format HTML (avec des liens cliquables).

Pour le module M1105 - Documents numériques

Sites de référence des éléments HTML5 :

- https://www.w3.org/community/webed/wiki/HTML/Elements
- https://developer.mozilla.org/en-US/docs/Web/HTML/Element

Sites de référence pour le CSS3 :

- https://www.w3.org/wiki/CSS
- https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

Un validateur de page HTML:

- https://validator.w3.org/#validate_by_input

Chapter 2 - Commandes de base (1)

I. TP : Exerciseur (1)

Objectifs de ce TP

- Vérifier que vous comprenez les termes *arborescence*, *répertoire*, *fichier*
- Commencer à comprendre ce qu'est une *commande* avec éventuellement des *options* et des *arguments*
- Comprendre les notions de *chemin relatif* et *chemin absolu*
- Comprendre ce qu'est un *répertoire courant*, un *répertoire parent*, le *Home* d'un utilisateur
- Comprendre ce que sont les répertoires `/`, `~`, `.` et `..`
- Savoir utiliser de la touche `TAB` pour la complétion automatique, les flèches haut/bas/droite/gauche
- les raccourcis `CTRL+l`, `CTRL+r` et le `bouton milieu de la souris`
- Connaître et savoir utiliser (dans des cas simples) les commandes `cd`, `ls` (avec les options `-alh`), `cat`, `rm` (avec l'option `-r`) `mkdir`, `mv` (pour déplacer et/ou renommer), `nano`, `cp`

Ouvrez un terminal et tapez la commande :

```
ssh -t 192.168.13.201 tpunix 2> /dev/null
```

Vous devez voir apparaître le **prompt** suivant :

```
user@tpunix:~$
```

Vous êtes connectés sur une machine distante sur laquelle tout est permis ! Votre nom de login sur cette machine est *user* et votre mot de passe est *live*. C'est une machine « bac à sable » sur laquelle vous pouvez faire tout ce que vous voulez sans risquer de casser quoi que ce soit. N'hésitez pas à y essayer tout ce qui vous passe par la tête !

Lorsque vous ne savez pas quoi faire (comme par exemple *maintenant*), tapez :

```
tpunix -h
```

Si vous souhaitez repartir du début dans l'exerciseur, au lieu de faire :

```
ssh -t 192.168.13.201 tpunix 2> /dev/null
```

tapez :

```
ssh -t 192.168.13.201 clear tpunix 2> /dev/null
```

Quand vous avez terminé l'exerciseur, relisez les objectifs du TP pour vérifier qu'ils sont atteints.

Si ce n'est pas le cas, demandez des précisions à votre enseignant[1] .

Pour consolider vos nouvelles connaissance, passez ensuite aux exercices (qui seront à faire par écrit)

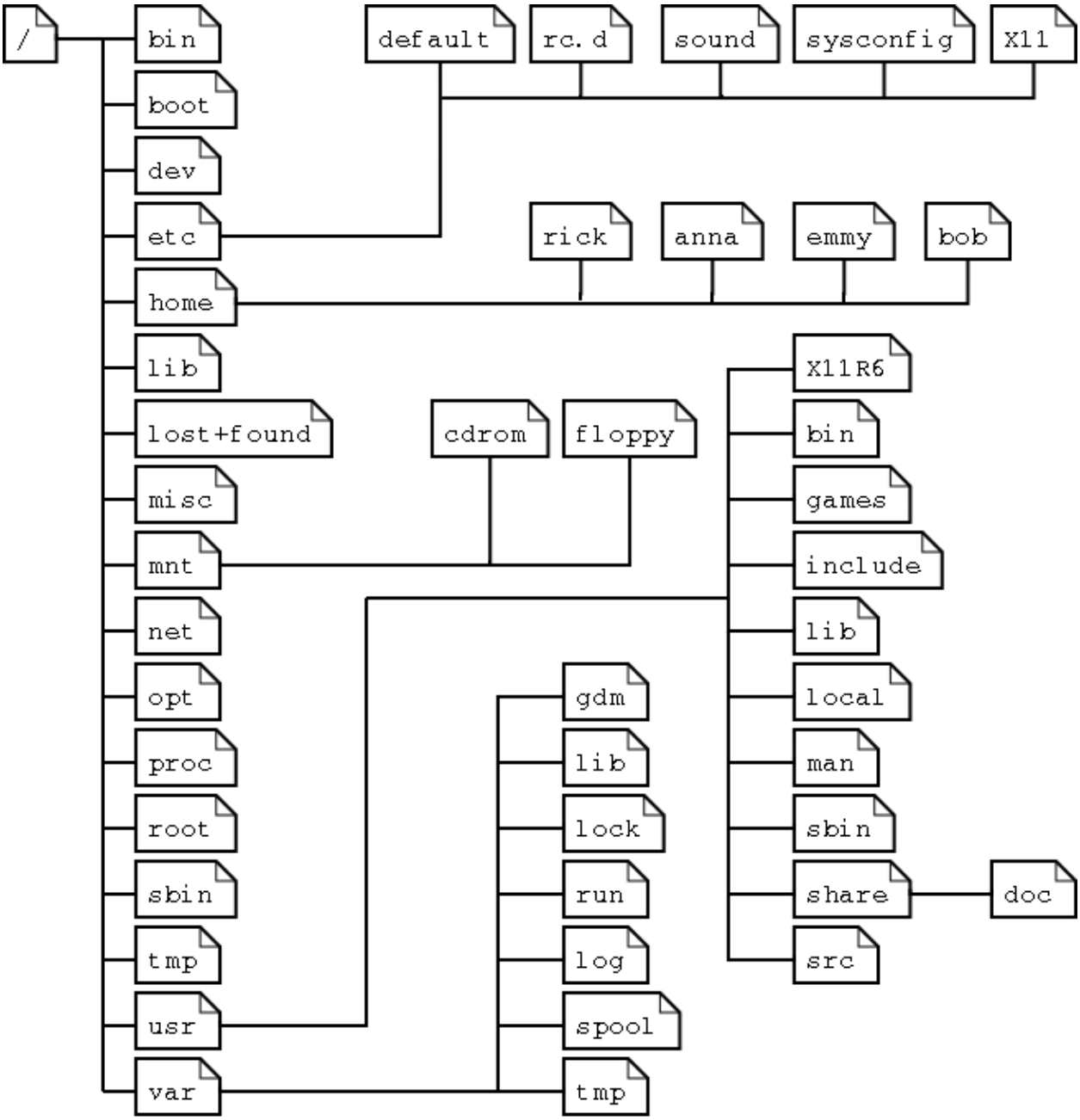
:.1 Notes en bas de page

II. Exercices

Exercice 1 : Explorer l'arborescence

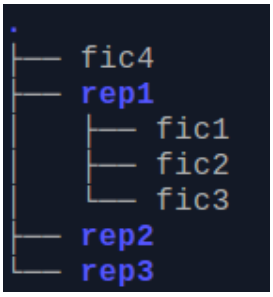
On donne l'arborescence suivante :

[1] Et rappelez vous : il n'y a pas de question idiote, il n'y a que des idiots qui ne posent pas de questions !!



1. Proposez une commande qui permet de se déplacer du répertoire `games` au répertoire `doc`
   (a) en utilisant un chemin relatif
   (b) en utilisant un chemin absolu
2. Vous êtes à la racine. Proposez quatre commandes qui vous permettent de vous déplacer dans votre *Home*.
3. Vous êtes dans votre *Home*
   (a) comment lister le contenu de votre *Home* ?
   (b) comment lister le contenu de votre *Home* (y compris les fichiers et répertoires cachés) ?
   (c) sans modifier le répertoire courant, comment lister le contenu du répertoire `doc`? (plusieurs réponses)

Exercice 2 : Commandes de base



Voici l'arborescence du *Home* de Zig et voici ce qu'il voit à l'écran :

```
zig@iutinfo23-14:~\rep3 $ _
```

1. Quelles commandes peut-il utiliser pour aller dans son *Home* ? (donnez au moins trois réponses différentes)
2. Zig est maintenant dans son *Home* et il n'en bouge plus de tout l'exercice.

1. Quelle commande doit-il taper pour copier le fichier `fic4` dans le répertoire `rep3` ?
2. Quelle commande doit-il taper pour déplacer le fichier `fic1` dans le répertoire `rep3` ?
3. Quelle commande doit-il taper pour supprimer tous les fichiers du répertoire `rep1` ?

Exercice 3 : Gérer les fichiers

Dans chacun des cas suivant, proposez une commande qui permet de résoudre le problème :

1. Je veux créer un répertoire `script` le répertoire courant
2. Je veux effacer le fichier `script1.sh`
3. Je veux déplacer le fichier `script2.sh` dans le répertoire `script`
4. Je veux copier dans `script` tous les fichiers d'extension `.sh` de `/pub/systeme`

Exercice 4 : QCM : Commandes de base

Exercice 5 : Recherches internet

Dans chacun des cas suivant, trouvez le nom de la commande qui permet de :

1. modifier les droits d'accès à un fichier
2. afficher les processus actifs (plusieurs réponses)
3. envoyer un signal d'arrêt à un processus
4. connaitre le type d'un fichier (texte, binaire, image ou autre)

Exercice 6 : Quota et taille des fichiers

Unités utilisées pour la taille des fichiers

- Le bit : c'est la plus petite unité (binary digit). Un bit est un élément pouvant être égal à 0 ou à 1 (deux valeurs possibles donc).
- L'octet (o) : 1 o = 8 bits *
- Le kilo-octet (Ko) : 1 Ko équivaut à 1000 octets.
- Le méga-octet (Mo) : 1 Mo = 1000 Ko.
- Le giga-octet (Go) : 1 Go = 1000 Mo.
- Le téra-octet (To) : 1 To = 1000 Go.

Avertissement:

octet se traduit par *bytes* en anglais

Attention à ne pas confondre *bit* et *byte* !

1 octet = 8 bits

Ordre de grandeur de la taille d'un fichier

1. Donnez l'ordre de gandeur de la taille des fichiers suivants :

1. un fichier texte d'une page
2. un document texte d'une page (type LibreOffice)
3. une photo

2. Combien de temps met-ton pour télécharger un film de 700 Mo avec un débit de 2Mbit/s ?

Quota

Avertissement:

Au département informatique, la taille de votre *Home* est limité à 500 Mo. Vous devez régulièrement surveiller que vous ne dépassez pas le maximum autorisé.

En cas de dépassement, votre compte risque d'être gelé et vous ne pourrez plus l'utiliser sans intervention des administrateurs réseau.

Pour déterminer la taille d'un répertoire (votre *Home* par exemple), placez vous dans ce répertoire et tapez la commande `du -sh`

:.1 un extrait du man de la commande *du*

```
NAME
   du - estimate file space usage
OPTIONS
   -s, --summarize
```

(continues on next page)

(continued from previous page)

```
      display only a total for each argument
  -h, --human-readable
      print sizes in human readable format (e.g., 1K 234M 2G)
```

3. Que fait la commande `du`?
4. A quoi servent les options `-s` et `-h`?
5. Quelle est la taille de votre *Home* ?
6. Quelle est la taille du répertoire `/bin/` ? Que trouve-t-on dans ce répertoire ?
7. Quelle est la taille du répertoire `~/.local/share/Trash/files` ? Que trouve-t-on dans ce répertoire ?

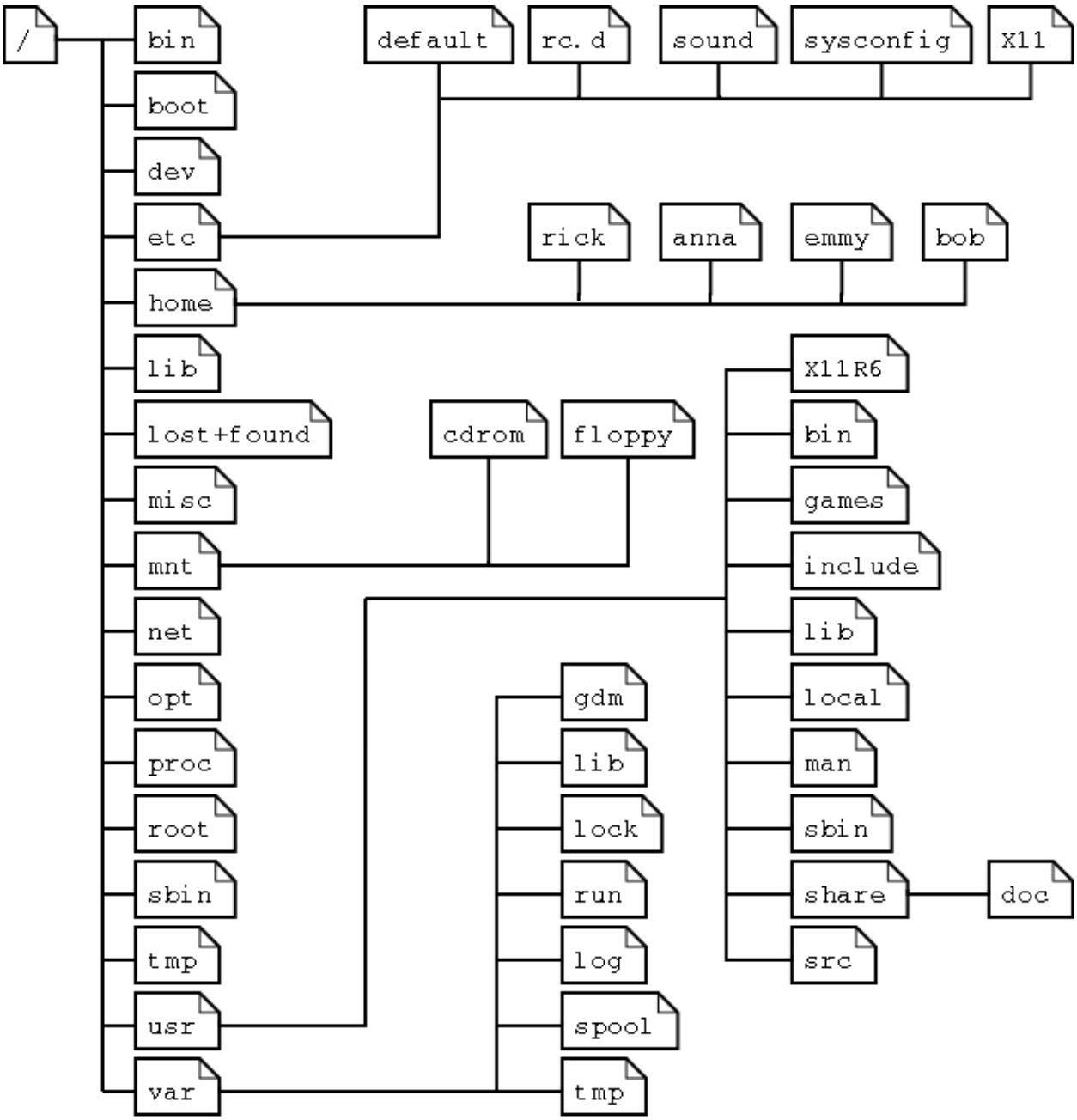
Exercice 7 : Vocabulaire : commandes, options et arguments

Dans chacun des cas suivant, on donne une ligne de commande. Repérez le **nom de la commande**, la ou les **options** et le ou les **arguments**

Ligne de commande	Nom de la commande	Options(s)	Arguments
`ls -l /home/bob/`			
`ls l /home/bob/`			
`ls -al .`			
`ls`			
`mkdir Toto Tata Tutu`			
`cd ..`			
`rm -r ./temp.txt`			
`rm -r ./ temp.txt`			

Exercice 8 : Pour s'entrainer

Avec l'arborescence suivante :



Répondre aux questions :

1. Si on est dans le répertoire `/opt/` comment se déplacer dans le répertoire `/usr/bin` ? Donnez deux solutions différentes.
2. Quelle est la différence entre la commande `cd /home/toto` et `cd home/toto` ? Dans quel(s) cas obtient-on le même résultat ?
3. Comment lister les fichiers du répertoire `/etc` ?
4. À quoi sert l'option `-l` de la commande `ls` ? Illustrez par un exemple que vous commenterez.
5. Lorsqu'on tape `ls -a`, on voit deux répertoires `.` et `..` ; à quoi correspondent-ils ?
6. Quelle est la différence entre un chemin relatif et un chemin absolu ?

7. Quelle commande permet de créer un répertoire ? Donnez un exemple d'utilisation.
8. Quelle commande permet de supprimer un fichier ? Donnez un exemple d'utilisation.
9. Quelle commande permet de supprimer un *répertoire* (et tout son contenu) ? Donnez un exemple d'utilisation.
10. Quelle commande permet d'afficher le contenu d'un fichier texte ?
11. Quelle commande permet de déplacer un fichier ? Donnez un exemple d'utilisation.
12. Donnez une commande qui permette de renommer le fichier `toto.txt` en `tata.txt`
13. À quoi correspond le répertoire ~ ?

Chapter 3 - Commandes de base (2)

I. TP : Exerciseur (2)

Objectifs de ce TP

- Revoir les notions du TP précédent
- Une première approche de la notion d'*utilisateurs*, de *Home* avec les commandes `su` (pour changer d'utilisateur), `exit`, `sudo` et `makeuser`
- Se familiariser avec la notion de *droits* des fichiers (et éventuellement des répertoires)
- **savoir utiliser les commandes**
  - `ls -l` pour lire les droits
  - `chmod` (avec les options `-[ugoa] [+-] [rwx]` et `chown` pour modifier les droits
- Se familiariser avec la notion de *processus*
- Savoir utiliser les commandes `ps aux`, `kill -9 <PID>`, `<cmd> &`, `fg`, `bg` ainsi que les raccourcis `CTRL-C` et `CTRL-Z`

Ouvrez un terminal et tapez la commande :

```
ssh -t 192.168.13.201 tpunix2 2> /dev/null
```

Vous devez voir apparaître le *prompt* suivant :

```
user@tpunix2:~$
```

Comme dans le TP précédent, vous êtes connectés sur une machine distante sur laquelle tout est permis !

De la même façon, lorsque vous ne savez pas quoi faire, tapez :

```
tpunix -h
```

Quand vous serez connectés en tant que *user3* où il faudra taper :

```
tpunix
```

Si vous souhaitez repartir du début dans l'exerciseur, tapez la commande :

```
ssh -t 192.168.13.201 clear tpunix2 2> /dev/null
```

Indication:

Le mot de passe de l'utilisateur « user » est « live »

Le mot de passe de l'utilisateur « user2 » est « password »

Quand vous avez terminé l'exerciseur, relisez les objectifs du TP pour vérifier qu'ils sont atteints.

Si ce n'est pas le cas, demandez des précisions à votre enseignant[1] .

Pour consolider vos nouvelles connaissances, passez ensuite aux exercices (qui seront à faire par écrit)

II. Exercices

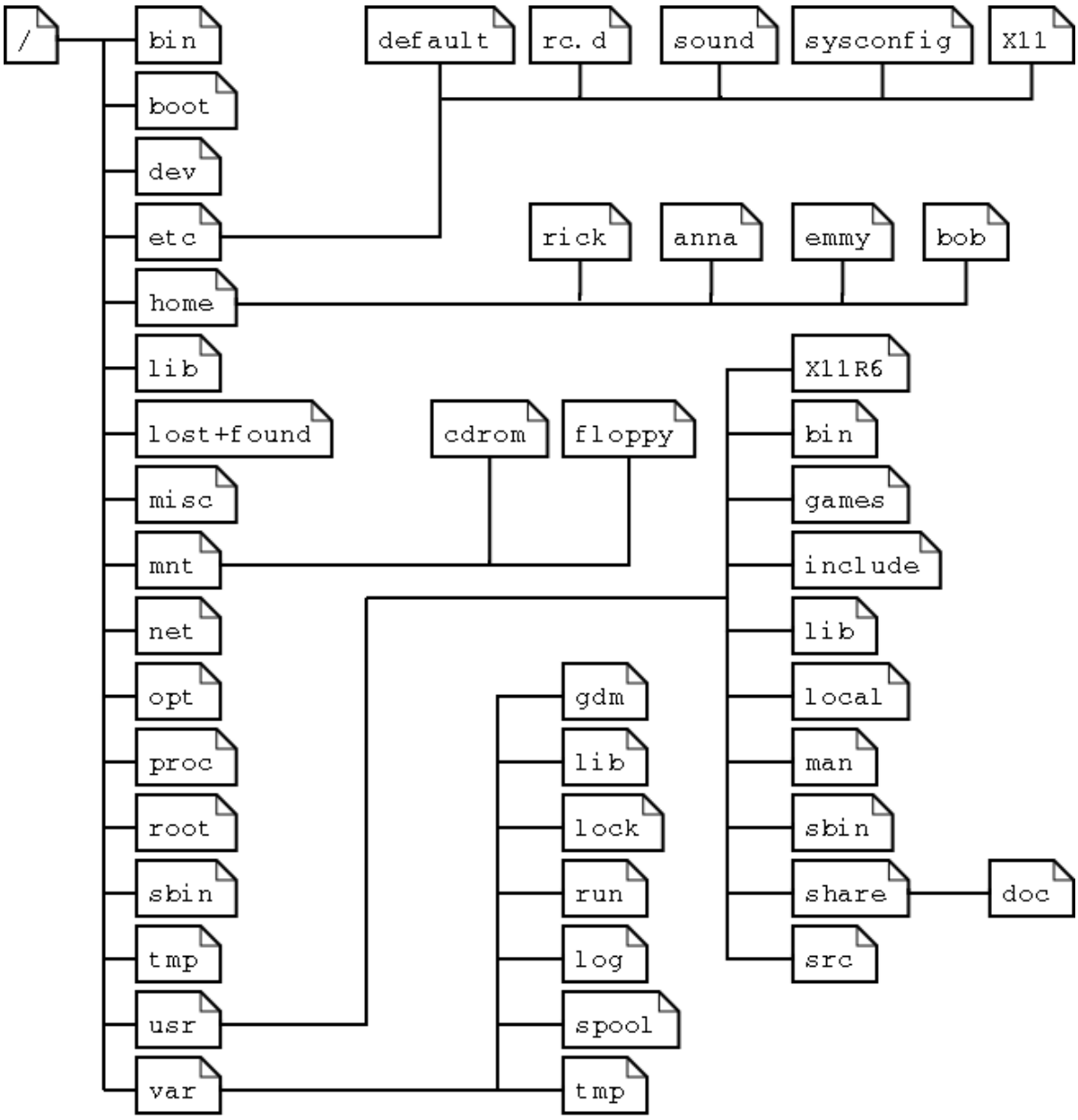
Exercice 1 : Pour revoir les commandes de base

1. Quelles informations nous donne cette copie d'écran ?

```
bob@iutinfo23-14:~/Documents$ ls
Dossier2  Fic1.txt  Fic2.txt  Fic3.txt   Fic4.png    Fic5
```

2. On donne l'arborescence suivante :

[1] Et rappelez vous : Celui qui pose une question risque d'avoir l'air bête cinq minutes, celui qui ne pose pas de question restera bête toute sa vie.



1. Si on est dans le répertoire `/opt/` comment se déplacer dans le répertoire `/usr/bin` ? Donnez deux solutions différentes.
2. Quelle est la différence entre la commande `cd /home/toto` et `cd home/toto` ? Dans quel(s) cas obtient-on le même résultat ?
3. Comment lister les fichiers du répertoire `/etc` ?
4. À quoi sert l'option `-l` de la commande `ls` ? Illustrez par un exemple que vous commenterez.
5. Lorsqu'on tape `ls -a`, on voit deux répertoires `.` et `..` ; à quoi correspondent-ils ?
6. Quelle est la différence entre un chemin relatif et un chemin absolu ?
7. Quelle commande permet de créer un répertoire ? Donnez un exemple d'utilisation.
8. Quelle commande permet de supprimer un fichier ? Donnez un exemple d'utilisation.

9. Quelle commande permet de supprimer un *répertoire* (et tout son contenu) ? Donnez un exemple d'utilisation.
10. Quelle commande permet d'afficher le contenu d'un fichier texte ?
11. Quelle commande permet de déplacer un fichier ? Donnez un exemple d'utilisation.
12. Donnez une commande qui permette de renommer le fichier `toto.txt` en `tata.txt`
13. À quoi correspond le répertoire ~ ?

Exercice 2 : Droits et processus

1. Quelle commande permet de lister les processus actifs (on donne ci-dessous un extrait de l'affichage) ?

```
Zig    651  0.6  0.3 757296 27108 ? Sl   08:50   0:22 roquette
Tresh  702  0.0  0.2  14756   376 ? Sl   09:01   0:34 grappin
```

2. Quelle commande permet à Zig de tuer le processus roquette ?
3. Quelle commande permet d'obtenir l'affichage suivant ?

```
drwxr-x--- 1 Zig  Zig  4096  2014-09-07 09:55 .
drwxr-x--- 1 root root 12288 2014-06-21 19:07 ..
drwxr-x--- 1 root root 12288 2014-06-21 19:07 .rengar
drw-r----- 1 Zig  Zig  4096  2014-09-08 11:55 boum
-rwxr--r-- 1 Zig  Zig  4520  2014-09-10 09:55 mine.boum
-rwxr--r-- 1 Zig  Zig  6541  2014-09-10 10:01 bombe.bang
-rwxr--r-- 1 Zig  Zig  7541  2014-09-10 10:02 explosion.bang
-rwxr--r-- 1 Zig  Zig  1541  2014-09-10 10:03 petard.bang
-rw-r--r-- 1 root root 1705  2014-06-21 20:55 drake.gold
```

4. `.rengar` est un fichier ou un répertoire ? Quelles sont ses particularités ?
5. A quoi correspondent les deux répertoires `.` et `..` ?
6. Quelle commande Zig doit-il taper pour donner les droits `rwxr-x---` au répertoire `boum` ?
7. Quelle commande Zig doit-il taper pour copier le fichier `mine.boum` dans le répertoire `boum`
8. Expliquer ce que signifient les droits `r` , `w` et `x` un **fichier**. Donnez un exemple de commande permise dans chaque cas.

III. Bilan

Exercice 1 : Chemins

1. Vous vous trouvez dans le répertoire `/opt/`, quel chemin relatif représenterait le fichier `/etc/resolv.conf` ?
2. Vous vous trouvez dans le répertoire `/etc/`, quel chemin absolu représenterait le fichier `firefox/sysprefs.js` ?
3. Vous vous trouvez dans le répertoire `/etc/`, quel chemin absolu représenterait le fichier `./hostname` ?
4. Vous vous trouvez dans le répertoire `/etc/`, quel chemin absolu représenterait le fichier `../tmp/hostname` ?
5. Vous vous trouvez dans le répertoire `/opt/`, quel chemin relatif représenterait le fichier `/etc/../opt/resolv.conf` ?

6. Vous vous trouvez dans le répertoire `/opt/`, quel chemin relatif représenterait le fichier `/./opt/resolv.conf` ?
7. Vous vous trouvez dans le répertoire `/opt/`, quel chemin relatif représenterait le fichier `/opt/../resolv.conf` ?
8. Vous vous trouvez dans le répertoire `/opt/`, quel chemin relatif représenterait le fichier `/../opt/../resolv.conf` ?

Exercice 2 : Commandes de base

1. Dans `ls -lah /home/`, quelle est la commande employée ? Quelles sont les options ? Quels sont les arguments ?
2. Dans `cp -R /home/toto /home/titi`, quelle est la commande employée ? Quelles sont les options ? Quels sont les arguments ?
3. Dans `mkdir -p /home/toto/tutu /home/toto/tata`, quelle est la commande employée ? Quelles sont les options ? Quels sont les arguments ?
4. Citez les commandes permettant de :
   - se déplacer,
   - créer un dossier,
   - afficher le contenu d'un fichier texte,
   - supprimer un fichier,
   - déplacer un fichier,
   - renommer un fichier,
   - changer les droits d'un fichier,
   - lister les processus,
   - tuer un processus,
   - installer un paquet,
   - mettre à jour la liste des paquets,
   - changer son mot de passe.
5. Comment faites vous pour connaitre les options et arguments possibles d'une commande ?

Exercice 3 : Droits

1. Dans quel cas ne veut on pas qu'un utilisateur puisse lire le contenu d'un fichier ?
2. Le fichier /etc/shadow contient les mots de passe encryptés des utilisateurs. Pourquoi les utilisateurs « lambda » n'ont ils pas les droits de lecture sur ce fichier ? Pourquoi n'ont ils pas les droits d'écriture ? Pourquoi n'ont ils pas les droits d'exécution ?
3. Le fic/etc/sudoers contient la liste des utilisateurs ayant des droits étendus. Pourquoi les utilisateurs « lambda » n'ont ils pas les droits de lecture sur ce fichier ? Pourquoi n'ont ils pas les droits d'écriture ? Pourquoi n'ont ils pas les droits d'exécution ?
4. Comment donner les droits de lecture à tout le monde au fichier `/tmp/exemple` ?
5. Comment supprimer les droits d'écriture au groupe du fichier `/tmp/exemple` ?

6. Quelle commande permet de changer le propriétaire d'un fichier ?

Exercice 4 : Processus

1. Quand un processus est-il créé ?
2. Qu'est-ce que le PID d'un processus ?
3. La quantité de mémoire utilisée par un processus varie-t-elle au cours de l'exécution du processus ?
4. La quantité de processeur utilisée par un processus varie-t-elle au cours de l'exécution du processus ?
5. Quelle commande permet de connaitre les ressources utilisées par les processus ?
6. Quelle commande permet de mettre fin à un processus ?
7. Peut on mettre fin à n'importe quel processus ?
8. Pour un processus, que signifie être en avant plan ?

Exercice 5 : Fichiers

1. Un fichier texte de 100 caractères prendrait environ quel espace sur le disque ?
2. Une image non compressée de taille `1000x1000` prendrait environ quel espace sur le disque ?
3. Un fichier de log qui enregistre chaque commande que vous tapez prendrait combien d'espace sur le disque au bout de 10 jours ?
4. Comment connaitre la taille d'un fichier sur le disque ?

Exercice 6 : Système linux

a : Les fichiers :

La commande `ls -lah /usr/bin/mkpasswd` affiche `-rwxr-xr-x 1 root root 19K janv. 21 2018 /usr/bin/mkpasswd`.

La commande `ls -lah /var/log/syslog` affiche `-rw-r----- 1 syslog adm 2,4M juil. 23 17:24 /var/log/syslog`.

La commande `ls -lah /var/cache/apt/pkgcache.bin` affiche `-rw-r--r-- 1 root root 49M juil. 23 06:02 /var/cache/apt/pkgcache.bin`.

- Pour chacun des fichiers `/usr/bin/mkpasswd`, `/var/log/syslog` et `/var/cache/apt/pkgcache.bin`, indiquez si c'est un exécutable, un fichier de log, un fichier de cache ou encore autre chose.
- Pour chacun de ces trois fichiers, expliquez les droits qui leur sont affectés.

b : Les utilisateurs

1. Rappelez ce qu'est l'UID. Comment connaître votre uid ?
2. Comment créer un utilisateur sous linux ? De quelles informations a-t-on besoin ?
3. Qu'est-ce que le *home* d'un utilisateur ?
4. Comment est-il possible que plusieurs utilisateurs soient connectés simultanément ? Quelle commande permet de connaitre tous les utilisateurs connectés ?

c : Gestion des paquets

- VRAI/FAUX : Un paquet est l'ensemble des programmes d'installation/désinstallation d'un logiciel, d'une bibliothèque, de documentation, etc.
- VRAI/FAUX : Un paquet est un fichier,
- VRAI/FAUX : Un paquet est un ensemble de fichiers,
- VRAI/FAUX : Un paquet doit être trouvé sur internet puis téléchargé avant de pouvoir être installé,
- VRAI/FAUX : Les paquets peuvent dépendre d'autres paquets ou être en conflit avec d'autres paquets
- VRAI/FAUX : Les paquets sont récupérés automatiquement par le système à partir de « dépôts » officiels
- VRAI/FAUX : Les commandes à utiliser sont celles commençant en *apt*.
- VRAI/FAUX : Seul le superutilisateur peut installer/désinstaller des paquets
- Un paquet contient des métas données sur le programme à installer. Parmi les suivantes, lesquelles sont présentes selon vous ?
  - le nom et l'adresse mail d'une personne responsable du paquet,
  - la taille que prendra l'installation sur le disque,
  - la liste des dépendances,
  - la liste des paquets en conflit,
  - un numéro de version,
  - une description,
  - le nombre moyen de téléchargements par semaine,
  - une note sur 5 indiquant la qualité du logiciel,
  - l'emplacement de l'installation.

Chapter 4 - Scripts en bash (1)

I. Intro Scripts 1ère partie

Objectifs de ce TP

- Apprendre à *écrire* et *exécuter* un script bash « simple » (c'est à dire qui n'utilise que des commandes de base).

Exercice 1 : Mon premier script

Les commandes bash, plutôt que d'être tapées dans un terminal, peuvent être écrites dans un *script*. Un script est un fichier (par exemple `monScritp.sh` ou encore `bonjour` que vous avez rencontré dans le TP précédent).

Lors de l'*exécution* du script, les commandes qu'il contient seront exécutées les unes après les autres.

Les premières lignes d'un script forment l'**en-tête**. Il contient de informations comme l'auteur, l'usage du script et toute autre information utile.

- La première ligne de l'en-tête indique quel *interpréteur* doit être utilisé pour lire le script.
- Les autres lignes commençant par des `#` sont des *commentaires* et ne sont pas exécutées.

Par exemple :

```
#!/bin/bash
# Auteur :  A. B.
# Ce script liste le contenu de votre home
ls ~
# Ceci est un commentaire et ne sera pas exécuté
```

1. Quel est l'interpréteur utilisé par ce script ?
2. Recopiez ce script. Quels droits faut il lui donner pour pouvoir l'exécuter ? Exécutez le.
3. Modifiez le script pour qu'il liste le contenu de votre répertoire `Documents`
4. Que se passe-t-il si vous utilisez un autre interpréteur que celui précisé ?

   Essayez avec :

   (a) `/bin/sh`

   (b) `/bin/toto`

   (c) `/usr/bin/python`

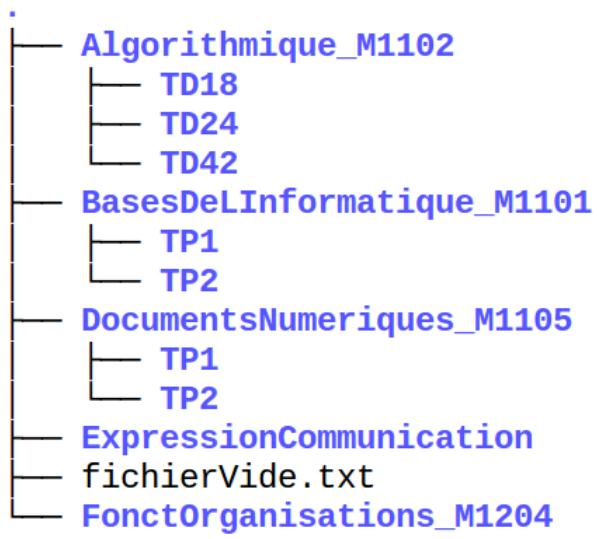
   (d) `/bin/echo`

(e) `/bin/rm`

5. Expliquez les affichages obtenus avec les interpréteurs précédents.
6. Quel « interpréteur » ferait s'afficher le contenu du fichier ?
7. Quel « interpréteur » ferait s'autodétruire le fichier ?
8. Quel interpréteur est utilisé lorsqu'on oublie de le spécifier ?

Exercice 2 : Mise en pratique

1. Ecrire un script `exercice2.sh` qui crée l'arborescence ci contre dans le répertoire courant



2. **Pour revoir les commandes des TP précédents**

   En restant dans votre *Home*, c'est à dire sans utiliser la commande `cd`, quelle commande peut-on utiliser pour :

   (a) déplacer le fichier `fichier_vide.txt` dans le répertoire `TD42`.

   (b) renommer le répertoire `TP1` du répertoire `DocumentsNumeriques_M1105` en `TP1_LibreOfficeWriter`

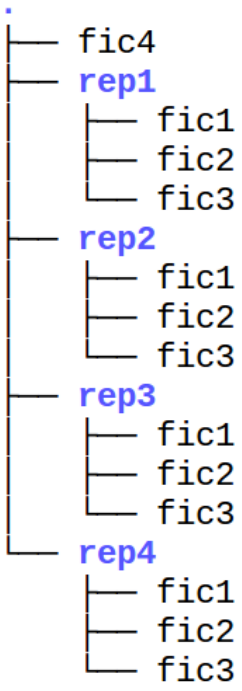
   (c) supprimer tous les répertoires qui sont dans `Algorithmique_M1102`

   (d) copier le fichier `/pub/1A/ASR-M1101/tp1-commandes-de-base.pdf` dans le répertoire `TP1_LibreOfficeWriter` du répertoire `BasesDeLInformatique_M1101`

   (e) copier tous les fichiers de `/pub/1A/DocNum/TP1/` le répertoire `TP1` du répertoire `DocumentsNumeriques_M1105`

II. Exercices

Exercice 1 : Autour des droits



1. Ecrire un script `exercice3.sh` qui crée l'arborescence ci contre dans le répertoire courant avec les droits suivants :

```
-r--------  fic4
dr-x------  rep1
drwx------  rep2
drw-------  rep3
d--x------  rep4
```

   Pour les trois fichiers dans chacun des répertoires `rep1` , `rep2`, `rep3` et `rep4` :

```
---------- fic1
-r-------- fic2
-rw------- fic3
```

2. D'après-vous et **SANS ordinateur** :
   (a) Parmi les 13 fichiers de cette arborescence, lesquels peut-on mettre en argument de la commande `cat` ?
   (b) Même question avec la commande `chmod u+rxw`
   (c) Même question avec la commande `rm`
3. Vérifiez avec le terminal

Exercice 2 : Jocker

1. Comment afficher le nom de tous les fichiers du répertoire courant dont le nom se termine par `.py` ?

2. Comment afficher le nom des fichiers du répertoire courant dont le nom commence par un c ou un d ?
3. Comment afficher le nom des fichiers cachés (et uniquement le fichiers cachés) du répertoire courant ?

4. **Défi !** Comment afficher le nom des fichiers du répertoire courant dont le nom contient exactement 3 caractères ?

Chapter 5 - Coup de projecteur sur le man

I. Introduction au man

La commande `man` permet de consulter les manuels de référence en mode texte. Par exemple, pour consulter la documentation sur une commande `tar`, il suffit de taper :

```
man tar
```

Exercice 1 : Les rubriques du man

Le manuel est structuré en plusieurs rubriques. En voici quelques unes :

- *NAME* La rubrique qui décrit très succinctement la commande (une ligne).
- *SYNOPSYS* La rubrique qui décrit la syntaxe. Les conventions suivantes s'appliquent à la rubrique SYNOPSIS et peuvent être utilisées comme un guide pour les autres rubriques.
  - **texte en gras** : à taper exactement comme indiqué
  - texte souligné ou en *italique* : à remplacer par l'argument approprié décrit dans la description
  - [-abc] : tous les arguments entre [ ] sont facultatifs
  - -a|-b : les options séparées par | ne peuvent pas être utilisées simultanément
  - argument ... : argument peut être répété
  - [expression] ... : toute l'expression située à l'intérieur de [ ] peut être répétée.
- *DESCRIPTION* La description de la commande détaillée dans laquelle les éléments soulignés du synopsys sont souvent repris. Les options sont souvent décrites dans cette rubrique.
- *OPTIONS* Les options (parfois intégrés dans la description)
- *EXEMPLES* La rubrique qui donne des exemples d'utilisation (n'existe pas toujours)
- Allez voir le `man` de la commande de votre choix.

Dans chaque cas, répondre aux questions suivantes SANS UTILISER L'ORDINATEUR !

Dans quelle rubrique (NAME, SYNOPSYS, DESCRIPTION, OPTIONS ou EXEMPLES) peut-on retrouver les extraits suivants ? Toujours sans ordinateur, trouvez (si possible) à quelle commande chaque extrait correspond.

1. ```
   -a, --all

   write counts for all files, not just directories
   ```
   ```

2. ```
   whoami - print effective userid
   ```

3. ```
   Display the current time in the given FORMAT, or set the system date
   ```

4. ```
   -r, -R, --recursive

   remove directories and their contents recursively

   -d, --dir

   remove empty directories
   ```

5. sort - sort lines of text files

6. ```
   Print the full filename of the current working directory.
   ```

7. ```
   To run the command ``wc /etc/hosts`` and show the default information:

   time wc /etc/hosts
   ```

8. ```
   --follow-name

   Normally, if the input file is renamed while an F command is
   executing, less will continue to display the contents of the
   original file despite its name change.
   ```

9. ```
   sort [OPTION]... [FILE]...
   sort [OPTION]... --files0-from=F
   ```

10. ```
    Create archive.tar from files foo and bar.

    tar -cf archive.tar foo bar
    ```

Exercice 2 : touch et cp

Répondez aux questions **sans ordinateur** .

Voici un extrait du `man` de la commande `touch` :

```
NAME
   touch - change file timestamps!

SYNOPSIS
   touch [OPTION]... FILE...

DESCRIPTION
   Update the access and modification times of each FILE to the current␣
↪time.
   A FILE argument that does not exist is created empty
```

1. À quoi sert la commande `touch` ?
2. Dans le SYNOPSIS, pourquoi `OPTION` est-il entre crochets ? Pourquoi `FILE` n'est pas entre crochet ? Que signifient les trois petits points qui sont après `FILE` ?

Voici un extrait du `man` de la commande `cp` :

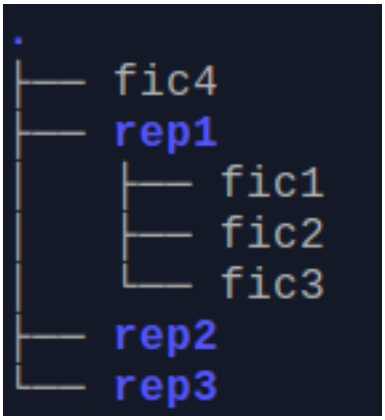
```
NAME
  cp - copy files and directories

SYNOPSIS
   cp [OPTION]... [-T] SOURCE DEST
   cp [OPTION]... SOURCE... DIRECTORY
   cp [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
   Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

   -R, -r, --recursive
   copy directories recursively
```

3. A quoi sert la commande `cp` ?
4. Que font les commandes suivantes ?

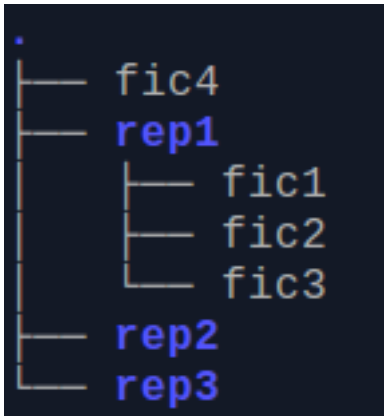


- `cp fic4 toto4`
- `cp rep1 fic4`
- `cp -r rep1 toto5`
- `cp fic4 rep2 rep3`

II. Exercices

Exercice 1 : Petites recherches dans le man

- Dans *votre home*, créez le répertoire `ASR-P1/TP4/`. Dans ce répertoire, créez l'arborescence suivante :



- Répondre aux questions suivants en utilisant uniquement le `man` (c'est-à-dire sans Google et sans essayer de taper les commandes dans le terminal). Ensuite (et seulement ensuite), vérifiez éventuellement vos réponses à l'aide du terminal.

  1. Pourquoi la commande suivante est incorrecte ?

     ```
     rm -a fic4
     ```

  2. Que font les commandes suivantes ?

     ```
     mv fic4 rep1 rep2
     ```

     ```
     mv rep1 rep2 fic4
     ```

     ```
     mv rep1 rep2 rep4
     ```

     ```
     rmdir rep1
     ```

     ```
     rmdir fic4
     ```

     ```
     rmdir rep2
     ```

Exercice 2 : Encore des recherches dans le man

1. Que font les commandes suivantes ? (Répondre par une seule phrase, et il n'est pas nécessaire de tout apprendre par coeur !!!)

   - `tar`
   - `adduser`
   - `true`
   - `cut`
   - `rmdir`
   - `which`
   - `kill`
   - `pwd`
   - `whoami`

- `python`
- `python3`

2. Lister le contenu du répertoire `/bin/`. Vérifiez que les fichiers que vous y voyez sont des commandes existantes.
3. **Défi** !!! Que fait la commande suivante ?

```
tar -cfv archive.tar
```

Chapter 6 - Retour sur les droits (fichiers et répertoires)

I. Droits des fichiers

Lorsque vous tapez `ls -l` vous obtenez quelque chose comme ça :

```
-rwxr-xr-x    1 john etu 43B 14 jui 11:55 toto.sh
```

La partie « john etu » signifie que le fichier appartient à l'utilisateur **john** et est dans le groupe **etu** .

La partie `-rwxr-xr-x` se lit en oubliant le tiret du début, puis en décomposant en trois :

- `rwx` (utilisateur)
- `r-x` (groupe)
- `r-x` (autres)

Chaque partie est composée de trois lettres :

- Droits de lecture (r) : on peut par exemple lire le fichier avec un logiciel.
- Droits d'écriture (w) : on peut modifier le fichier et le vider de son contenu.
- Droits d'exécution (x) : on peut exécuter le fichier s'il est prévu pour, c'est-à-dire si c'est un fichier exécutable.

Les trois parties correspondent à différents utilisateurs :

- La première partie correspond aux droits du propriétaire du fichier.
- La seconde partie correspond aux droits des utilisateurs appartenant au groupe auquel le fichier appartient (relisez cette phrase plusieurs fois...) .
- La dernière partie correspond aux droits des gens qui ne sont ni le propriétaire du fichier, et qui n'appartienent pas au groupe du fichier.

Seul le propriétaire d'un fichier (ou *root*) peut changer ses permissions d'accès.

Exercice 1 : Lecture des droits

Sur mon système, il y a trois utilisateurs :

alice, qui appartient aux groupes etu et admin bob, qui appartient aux groupes etu et bob candy, qui appartient aux groupes etu et admin .

Pour chacun des fichiers suivants, dites, pour chacun des utilisateurs, les droits qu'il a sur le fichier.

- `-rwxr-x--- 1 alice etu 43B 14 jui 11:55 fichier1`

- `-rwxr-x--- 1 candy bob 54K 14 jui 11:56 fichier2`
- `-rwxr-x--- 1 alice admin 3M 14 jui 11:57 fichier3`
- `-rwxr-x--- 1 bob bob 1B 14 jui 11:58 fichier4`

Exercice 2 : Modification des droits

La commande permettant de changer les droits est `chmod`.

- Pour donner les droits de lecture pour le groupe on fera :

```
chmod g+r fichier
```

- Pour donner les droits d'écriture pour le propriétaire on fera :

```
chmod u+w fichier
```

- Pour donner les droits d'exécution pour les autres (*others*) on fera :

```
chmod o+x fichier
```

- Pour donner les droits d'exécution pour tous (*all*) on fera :

```
chmod a+x fichier
```

- Pour enlever des droits, on fera la même chose en remplaçant + par - .

1. On a un fichier `texte.txt` avec les droits `--wx------` . Quelle commande doit on taper pour qu'il ait les droits :
   (a) `-rwxrw-r--`
   (b) `-rwx----w-`
   (c) `-rw-------`
   (d) `-rw-rw-rw-`
   (e) `-rw-r--r--`
2. Pour le fichier précédent qui est un fichier texte, quels sont les droits « raisonnables » à lui donner ?

II. Permissions des répertoires

Pour les répertoires comme pour les fichiers, lorsqu'on fait ls -l on voit apparaitre quelque chose comme :

```
drwxrwxr-x   5 john etu  4096 mars  16 13:32 Bureau
```

Remarquez le `d` du début qui pour un fichier était un `-`. Pour le reste on retrouve les même lettres mais avec des significations différentes :

- Droits de lecture (r) : l'affichage du contenu du répertoire est autorisé (`ls`)
- Droits d'écriture (w) : on peut créer, supprimer ou modifier le nom des fichiers qu'il contient, quels que soient les droits d'accès des fichiers de ce répertoire (même s'ils ne possèdent pas eux-mêmes le droit en écriture). (`touch, rm, mv`)
- Droits d'exécution (x) : l'accès (ou ouverture) du répertoire est autorisé (`cd`).

Globalement, en considérant qu'un répertoire est un fichier contenant la liste des noms des fichiers, on comprend bien le comportement de r et w. Pour x, il faut l'apprendre !

1. Le répertoire /opt/ a les droits suivants : `drwxr-xr-x root root` . Dans ce réprtoire, il y a un fichier `truc` qui a les droits `-rw-rw-rw- root root`. L'utilisateur john qui n'appartient pas au groupe root peut il :
   - lister les fichiers du répertoire ?
   - Modifier le contenu du fichier truc ?
   - Supprimer le fichier truc ?
   - Renommer le fichier truc?
   - Créer un fichier dans le répertoire /opt ?
2. Pour que personne à part vous puissent lister les fichiers que vous avez dans votre home, quels droits devez-vous donner, et à quoi ?
3. Pour que d'autres étudiants de votre groupe puissent lire un fichier README.txt dans votre home, quels droits devez vous donner à votre home et à votre fichier README.txt ? Cela vous expose-t-il à ce qu'on puisse accéder à vos données ?

Exercice 1 : Chmod et code hexa

1. Dans chaque cas, préciser les droits de `fichier` à l'issue des commandes suivantes :

- `chmod 734 fichier`
- `chmod 022 fichier`
- `chmod 601 fichier`
- `chmod 754 fichier`
- `chmod 776 fichier`
- `chmod 555 fichier`
- `chmod 777 fichier`
- `chmod 500 fichier`
- `chmod 550 fichier`

2. Dans chaque cas, préciser la commande à utiliser pour attribuer à `fichier` les droits suivants :
   - `-rwxrw-r--`
   - `-rwx----w-`
   - `-rwx------`

III. Système de fichiers

Vous avez vu précédemment qu'un fichier est un peu plus que des données : il a des droits associés, un propriétaire, une date de dernière modification, etc.

Pour mieux comprendre comment fonctionne le stockage des fichiers sur un disque dur nous regardons ici la manière dont c'est fait sous linux.

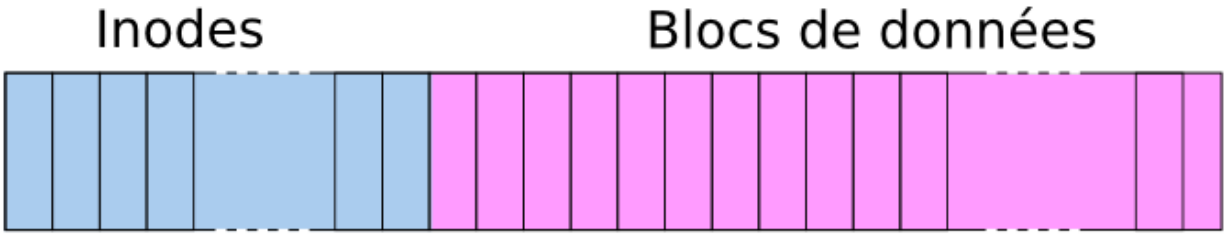
L'espace disque est à la disposition de linux : pour chaque partition, il peut accéder au premier octet, au deuxième, etc. Pour stocker les fichiers, linux va utiliser un protocole pour définir « où » mettre les données. Ce protocole s'appèle « système de fichiers ».

Il existe de nombreux systèmes de fichiers (FAT, NTFS, ext2, ext3, ext4, xfs, …), nous n'allons pas rentrer dans les détails d'un système, mais voir comment cela fonctionne dans les grandes lignes.

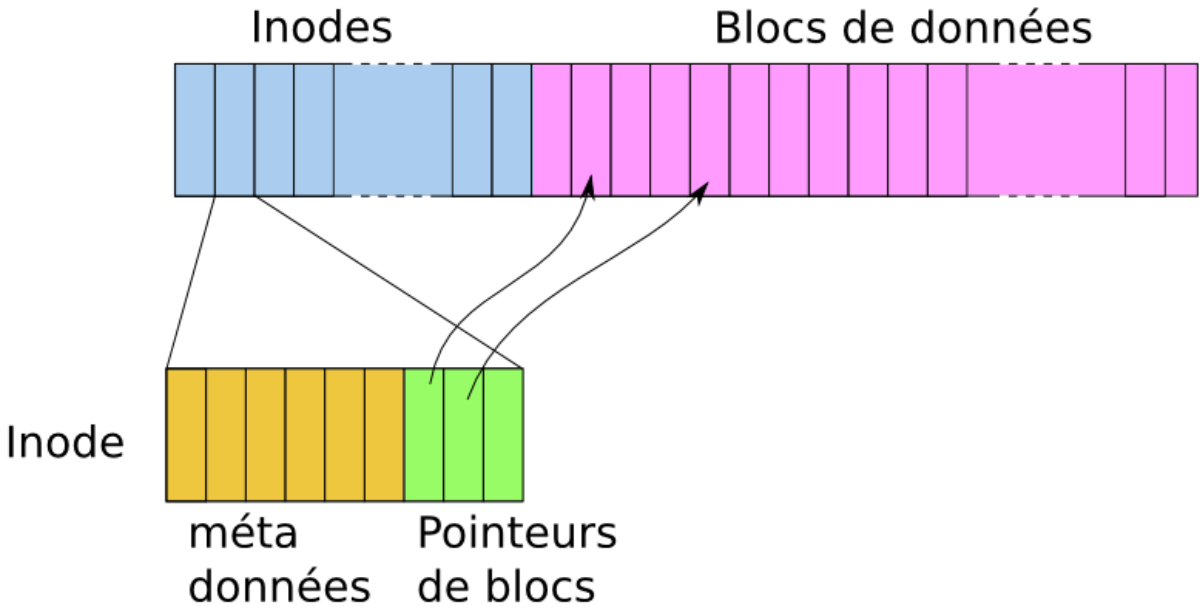
Exercice 1 : Inodes et données

Sur le schéma suivant, on représente les octets de la partition, chaque rectangle représente un groupe de 256 octets :

Cet espace est découpé en deux parties : une liste d'inodes (en bleu) et les blocs de données (en violet) :



Les blocs de données contiennent les données des fichiers (ce qu'on va voir en regardant le contenu hexadécimal d'un fichier), et les inodes contiennent les méta-données : permissions, taille du fichier, date d'accès/de création, etc. ainsi que l'indication de l'endroit où se trouvent les données associées :



Chaque fichier est un inode. Les données du fichier sont pointées par les pointeurs de blocs. Le numéro d'inode est la position de l'inode dans la liste des inodes. Pour connaitre le numéro d'inode des fichiers, vous pouvez faire `ls -i`. *Attention:* le nom d'un fichier ne fait pas partie des méta-données. Il n'est pas non plus dans les données. Mais où est il ?

Exercice 2 : Répertoires

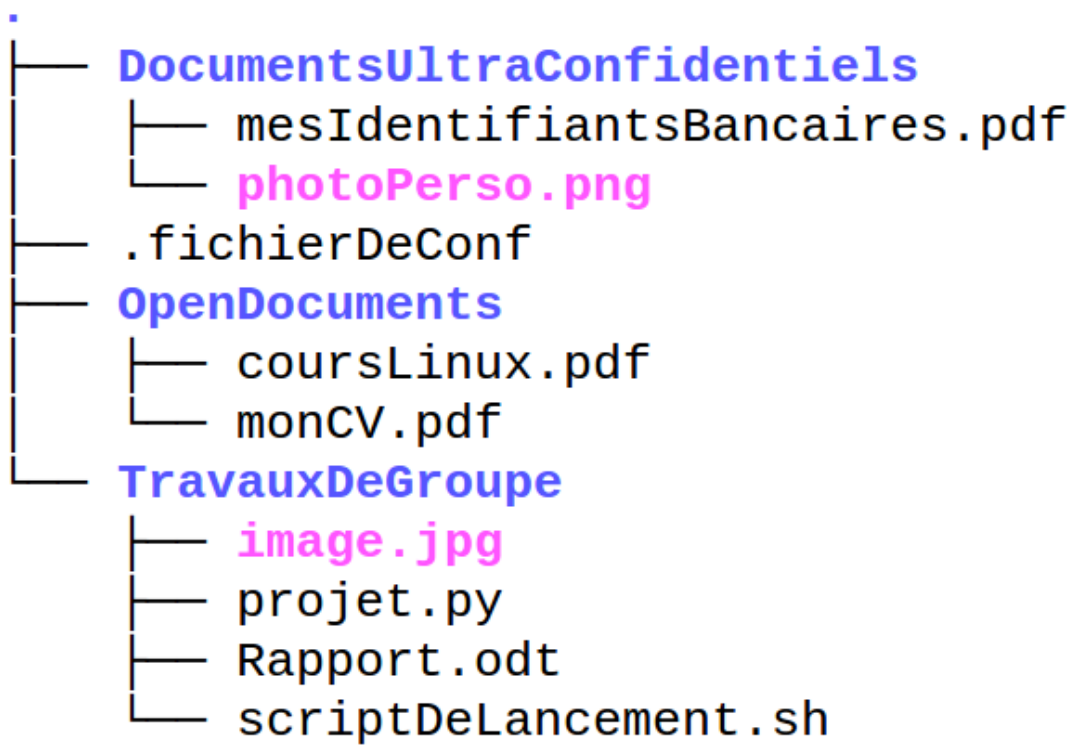
Un répertoire est stocké comme un fichier, par un inode. Les données pointées par les pointeurs de blocs sont une liste de fichiers : une association « nom de fichier -> numéro d'inode ». *A retenir:* un répertoire est un fichier associant à chaque nom de fichier un numéro d'inode. Les répertoires contiennent *seulement* l'association « nom de fichier » -> « numéro d'inode »

1. Quel est le numéro d'inode du dossier / ? **nota :** c'est toujours le cas, c'est le point de départ pour parcourir les fichiers.
2. Remarquez que /dev/ et / ont le même numéro d'inode.. La raison de cela est que ce ne sont pas les mêmes systèmes de fichiers ! (vous pouvez l'observer en faisant `df`)
3. On peut donc imaginer que deux noms de fichiers soient associés au même inode. Dans ce cas, que se passe-t-il si on modifie le contenu du premier fichier ? Pour essayer, créez un fichier texte nommé « fichier1 » contenant ce que vous voulez. Lancez la commande `ln fichier1 fichier2`. Regardez les numéros d'inode des deux fichiers. Modifiez le contenu de fichier1, que se passe-t-il ?

IV. Exercices

Exercice 1 : Droits raisonnables

Voici ce que j'ai dans mon *Home*. Donnez les droits *raisonnables* de chacun des 3 répertoires et des 9 fichiers.



Exercice 2 : Droits et processus

1. Quelle commande permet de lister les processus actifs (on donne ci-dessous un extrait de l'affichage) ?

```
Zig    651  0.6  0.3 757296 27108 ? Sl   08:50   0:22 roquette
Tresh  702  0.0  0.2  14756   376 ? Sl   09:01   0:34 grappin
```

2. Quelle commande permet à Zig de tuer le processus roquette ?
3. Quelle commande permet d'obtenir l'affichage suivant ?

```
drwxr-x--- 1 Zig  Zig  4096  2014-09-07 09:55 .
drwxr-x--- 1 root root 12288 2014-06-21 19:07 ..
drwxr-x--- 1 root root 12288 2014-06-21 19:07 .rengar
drw-r----- 1 Zig  Zig  4096  2014-09-08 11:55 boum
-rwxr--r-- 1 Zig  Zig  4520  2014-09-10 09:55 mine.boum
-rwxr--r-- 1 Zig  Zig  6541  2014-09-10 10:01 bombe.bang
-rwxr--r-- 1 Zig  Zig  7541  2014-09-10 10:02 explosion.bang
-rwxr--r-- 1 Zig  Zig  1541  2014-09-10 10:03 petard.bang
-rw-r--r-- 1 root root 1705  2014-06-21 20:55 drake.gold
```

4. A quoi correspondent les deux répertoires `.` et `..`
5. Quelle commande Zig doit-il taper pour donner les droits `rwxr-x---` au répertoire `boum` ?
6. Quelle commande Zig doit-il taper pour copier le fichier `mine.boum` dans le répertoire `boum`
7. Expliquer ce que signifient les droits `r` , `w` et `x` un **répertoire**. Donnez un exemple de commande permise dans chaque cas.
8. Zig veut supprimer le fichier `drake.gold`. Comment doit-il faire ? Expliquez et justifiez soigneusement

Exercice 3 : J'ai le droit ? droits des fichiers et répertoires

Voici le contenu du répertoire `/home/cersei/ASR/Eval3/`

```
-r------- 1 cersei  cersei 4242 2014-09-07 09:55 fic4
dr-x----- 1 cersei  cersei 4096 2014-09-07 09:55 rep1
drwx----- 1 cersei  cersei 4096 2014-09-07 09:55 rep2
drw------ 1 cersei  cersei 4096 2014-09-07 09:55 rep3
d--x----- 1 cersei  cersei 4096 2014-09-07 09:55 rep4
```

Il y a trois fichiers dans chacun des répertoires `rep1` , `rep2`, `rep3` et `rep4` :

```
--------- 1 cersei  cersei 1784 2014-09-07 09:55 fic1
-r------- 1 cersei  cersei 1692 2014-09-07 09:55 fic2
-rw------ 1 cersei  cersei 1515 2014-09-07 09:55 fic3
```

Parmi les commandes suivantes, précisez lesquelles

- s"**exécutent** sans problème (c'est à dire sans message d'erreur)
- renvoient message d"**erreur**

Expliquez chacune de vos réponses

1. Avec la commande `ls`

`ls rep1`	
`ls rep2`	
`ls rep3`	
`ls rep4`	

2. Avec la commande `touch`

`touch rep1/ fic5`	
`touch rep2/ fic5`	
`touch rep3/ fic5`	
`touch rep4/ fic5`	

3. Avec la commande `cd`

`cd rep1`	
`cd rep2`	
`cd rep3`	
`cd rep4`	

4. Avec la commande `rm`

`rm rep1/fic3`	
`rm rep2/fic3`	
`rm rep3/fic3`	
`rm rep4/fic3`	

5. Avec la commande `cat`

`cat fic4`	
`cat rep1/ fic2`	
`cat rep2/ fic2`	
`cat rep3/ fic2`	

Chapter 7 - Scripts en bash (2)

I. Intro Scripts 2eme partie

Objectifs de ce TP

- Consolider les compétences *comprendre* et *écrire* un script bash
- Comprendre et utiliser les notions de *variables* et *arguments*

Exercice 1 : Arguments d'un script

Créez le script `exArg.sh` suivant :

```
#!/bin/bash
# Auteur :  A. B.
echo le premier argument est $1
```

1. Qu'affiche ce script si vous l'exécutez sans argument ?
2. Qu'affiche ce script si vous tapez : `./exArgs.sh toto` ?
3. Qu'affiche ce script si vous tapez : `./exArgs.sh toto titi tutu` ?
4. Qu'affiche ce script si vous tapez : `./exArgs.sh toto titi tutu` ?
5. Qu'affiche ce script si vous tapez : `./exArgs.sh "toto titi tutu"` ?
6. Qu'affiche ce script si vous tapez : `./exArgs.sh 'toto titi tutu'` ?
7. Qu'affiche ce script si vous tapez : `./exArgs.sh toto\ titi tutu` ?
8. Quel est l'intérêt des `"` (guillemets), de `'` (l'apostrophe), et de \ (antislash appelé *caractère d'échappement*) ?
9. Comment afficher le deuxième argument du script ?

1. Écrivez un script qui affiche `j'ai 18 ans` . Proposez plusieurs solutions pour afficher l'apostrophe.
2. Écrivez un script qui affiche `toto a dit:"c'est chouette l'ASR"!.`
3. Écrivez un script qui affiche `$1 est le premier argument`. Quelle autre manière de faire avez-vous trouvé que d'utiliser le caractère d'échappement ?
4. Créez un fichier dont le nom est `$toto`.
5. Créez un fichier dont le nom est `Bonjour tout le monde`
6. Écrivez un script qui prend en argument deux mots et crée le fichier dont le nom est obtenu en écrivant le premier mot suivi d'un espace puis du second mot.

Exercice 2 : Jockers

Créez le script suivant `jockers.sh` :

```
#!/bin/bash
# Ce script affiche le nombre d'arguments et les premiers.
echo Le nombre d\'arguments est $#
echo Premier argument : $1, deuxieme : $2, et ensuite : $3 $4 $5 $6 $7 $8 $9
```

1. Exécutez ce script de manière à voir s'afficher : `Premier argument : Bonjour, deuxieme : tout, et ensuite : le monde !!!`
2. Exécutez ce script de manière à voir s'afficher : `Premier argument : \*, deuxieme : , et ensuite :`
3. Que s'est il passé lorsque vous avez essayé `./jocker.sh *` ?
4. Sans essayer, que pensez vous qu'il va s'afficher si on tape `echo *` ? Vérifiez le.
5. Sans essayer, que pensez vous qu'il va s'afficher si on tape `echo *.sh` ? Vérifiez le.
6. Sans essayer, que pensez vous qu'il va s'afficher si on tape `echo *trololololo` ? Vérifiez le.
7. Déplacez vous dans le répertoire `/bin`. Essayez : `echo b*`, `echo ??`, `echo l?` . À quoi correspond le caractère `?` ?
8. Combien de fichiers dont le nom est composé de trois caractères y a-t-il dans le dossier `/bin` ?
9. combien de fichiers dont le nom commence par un c et se termine par « conf » y a-t-il dans le dossier `/etc` ?
10. Proposez des noms de fichiers qui correspondraient à :
    - *toto*
    - .??
    - c*d*.conf
11. Proposez des noms de fichier correspondant à `c*` mais pas à `c*?` .
12. Déplacez vous dans le répertoire `/bin`. Essayez : `echo [br]*`, `echo {ba,re}*`. À quoi correspondent `[]` et `{}` ?
13. Comment afficher tous les fichiers dont le nom commence par un nombre ?
14. Comment afficher les fichiers dont le nom se termine par `.sh` ou par `.conf` ?
15. Comment supprimer les fichiers dont le nom se termine par `.aux` ou par `.log` ?
16. Comment afficher les fichiers dont le nom contient un b, un c ou un d ?
17. Comment afficher les fichiers dont le nom contient plus de deux lettres et se termine par un b, un c ou un d ?

II. Exercices

Exercice 1 : Ecrire un script

Version 1

Ecrire un script `rendExecutable.sh` qui prend en *argument* un nom de fichier et qui le rend exécutable pour l'utilisateur puis affiche un message de compte-rendu.

Version 2

Ecrire un script `rendExecutable.sh` qui prend en *argument* un nom de fichier et qui

1. Regarde si ce fichier est exécutable, puis le rend exécutable pour l'utilisateur si ce n'est pas le cas.
2. Affiche un message de compte-rendu dans chaque cas

Version 3

Ecrire un script `rendExecutable.sh` qui prend en *argument* un nom de fichier et qui

1. Vérifie que l'*argument* désigne bien un fichier
2. Regarde si ce fichier est exécutable, puis le rend exécutable pour l'utilisateur si ce n'est pas le cas.
3. Affiche un message de compte-rendu dans chaque cas

Indication:

L'instruction if permet d'effectuer des opérations si une condition est réalisée.

```
if condition
then
   instruction(s)
else
   instructions
fi
```

Les opérateurs de tests sur les objets du système de fichiers

- `[ -e $FILE ]` vrai si l'objet désigné par $FILE existe dans le répertoire courant,
- `[ -s $FILE ]` vrai si l'objet désigné par $FILE existe dans le répertoire courant et si sa taille est supérieure à zéro,
- `[ -f $FILE ]` vrai si l'objet désigné par $FILE est un fichier dans le répertoire courant,
- `[ -r $FILE ]` vrai si l'objet désigné par $FILE est un fichier lisible dans le répertoire courant,
- `[ -w $FILE ]` vrai si l'objet désigné par $FILE est un fichier inscriptible dans le répertoire courant,
- `[ -x $FILE ]` vrai si l'objet désigné par $FILE est un fichier exécutable dans le répertoire courant,
- `[ -d $FILE ]` vrai si l'objet désigné par $FILE est un répertoire dans le répertoire courant.

Exercice 2 : Pour s'entrainer

1. Écrivez un script qui prend un *argument* en paramètre et qui affiche : `L'argument est` suivi de l'argument. Proposez deux solutions différentes.
2. Écrivez un script qui prend en *argument* un nom de fichier et supprime ce fichier.

Avertissement:

Attention quand vous manipulez la commande `rm`

III. Pour aller plus loin

Exercice 1 : Lire un script avec des conditionnelles

Voici le script `mystere.sh`

```
#!/bin/bash
if [ $# -eq 0 ]
then
    echo "Eh Eh"
else
    if [ $1 = 'banzai' ]
    then
        echo "Ah Ah $*"
    else
        echo 'Oh Oh $1'
    fi
fi
```

1. Qu'affichera ce script si on tape la commande `./mystere.sh` ?
2. Qu'affichera ce script si on tape la commande `./mystere.sh yoda` ?
3. Qu'affichera ce script si on tape la commande `./mystere.sh banzai` ?
4. Qu'affichera ce script si on tape la commande `./mystere.sh banzai yoda` ?

Indication:

L'instruction if permet d'effectuer des opérations si une condition est réalisée Les opérateurs de tests disponibles sont :

pour les **chaînes de caractères** :

- `c1 = c2` vrai si c1 et c2 sont égaux ;
- `c1 != c2` vrai si c1 et c2 sont différents ;
- `-z c` vrai si c est la chaîne vide ;
- `-n c` vrai si c n'est pas la chaîne vide.

Pour les **nombres** :

- `n1 -eq n2` vrai si n1 et n2 sont égaux (equal) ;
- `n1 -ne n2` vrai si n1 et n2 sont différents (non equal);
- `n1 -lt n2` vrai si n1 est strictement inférieur à n2 (lower than);
- `n1 -le n2` vrai si n1 est inférieur ou égal à n2 (lower or equal);
- `n1 -gt n2` vrai si n1 est strictement supérieur à n2 (greater than) ;
- `n1 -ge n2` vrai si n1 est supérieur ou égal à n2 (greater or equal).

Chapter 8 - Processus

I. Processus

Selon wikipedia : « Un processus est un programme en **cours d'exécution** par un ordinateur. »

Sur nos machines, différents processus sont executés au même moment. En réalité, chaque programme utilise le processeur à tour de rôle, mais tellement rapidement que ça nous donne l'impression que chaque programme tourne en simultané avec les autres.

En système, nous allons voir le processus comme un peu plus que simplement un programme en cours d'exécution. Il a aussi des méta-données :

- un id
- un propriétaire,
- une date de début d'exécution,
- un état,
- un parent,
- un terminal auquel le processus est rataché,
- etc.

1. Pour voir quel sont les processus dont vous êtes le propriétaire et qui sont rattachés à un terminal, tapez :

```
ps
```

2. Essayez ensuite la commande

```
ps -o pid
```

3. Remplacez pid par `args` puis par `pid,args`

L'option `-o` de ps permet d'afficher certaines métadonnées des processus :

- PID : l'identifiant du processus (Process ID)
- ARGS : la commande qui a été utilisée pour lancer le processus
- PPID : l'identifiant du processus parent (Parent Process ID)
- UID : l'identifiant de l'utilisateur qui a lancé le processus
- TTY : le terminal correspondant au processus

4. En cherchant dans le manuel, trouvez quelle(s) option(s) de ps permettent d'afficher les informations de tous les processus

II. Signaux

Rappelez vous qu'un processus est un peu plus qu'un programme en cours d'exécution. Il a ses méta-données et est aussi capable de *communiquer* avec le reste du système. Cela se fait à l'aide de signaux.

Les principaux signaux pour nous sont :

- SIGKILL (9) : demande de fin **immédiate**.
- SIGTERM (15) : demande de fin (mais on laisse le temps au programme de s'arrêter tranquilement) .
- SIGINT (2) : à peu près pareil que SIGTERM. C'est le message envoyé lorsqu'on appuie sur ctrl-c .
- SIGSTOP (17) : demande au processus de se mettre en pause (stop).
- SIGCONT (19) : demande au processus de redémarrer.

Les numéros écrits à coté des signaux sont leur code. Pour envoyer un signal à un processus, il faut taper :

```
kill -CODE PID
```

où CODE est le code du signal et PID l'identifiant du processus auquel on veut envoyer le signal.

1. Lancez firefox si ce n'est pas déjà fait. Demandez l'arrêt de firefox à l'aide du signal 9. Pour cela, commencez par trouver le PID du processus correspondant à firefox.
2. Relancez firefox. Demandez au processus de se mettre en pause (SIGSTOP). Observez. Demandez lui de reprendre son exécution (SIGCONT).
3. Depuis un terminal, on peut envoyer le signal SIGINT à un processus à l'aide de ctrl-c . Lancez par exemple `top` depuis un terminal et tapez ctrl-c . La commande se termine, est elle encore présente dans la liste des processus ?
4. Depuis un terminal, on peut envoyer le signal SIGSTOP à un processus à l'aide de ctrl-z . Lancez par exemple `top` depuis un terminal et tapez ctrl-z . La commande se termine, est elle encore présente dans la liste des processus ?
5. Essay la commande `bg`. Que se passe-t-il ?
6. Lancez firefox depuis un teminal. Stoppez le à l'aide de ctrl-z. Tapez la commande `fg`. Que se passe-t-il ?

III. SSH

La commande `ssh` vous permet d'ouvrir un terminal sur une machine distante. Par exemple :

```
ssh info22-12
```

1. Essayez !
2. Selon vous, quels peuvent être les intérêts de cette commande ?

Vous pouvez aussi utiliser ssh pour exécuter une commande à distance :

```
ssh info22-12 ls
```

3. Essayez avec les commandes `ls`, `date`, `hostname`, `who`
4. Écrivez un script qui affiche toutes les personnes connectées dans votre salle.
5. Pouvez vous lancer firefox à distance ? Pour pouvoir lancer une commande graphique, utilisez l'option -X de ssh.

Chapter 9 - Redirections

I. Redirection

Exercice 1 : L'entrée standard

- Dans le terminal, tapez `read maVariable2`. Le terminal attend que vous tapiez du texte, on dit que read lit son **entrée standard** ; mettez par exemple « Bonjour ». Que contient maintenant la variable `$maVariable2` ?

À retenir

- **L'entrée standard** est le flux par lequel du texte peut être entré dans un programme. Les données sont habituellement lues depuis le clavier à moins d'une **redirection**

- Écrivez un script qui affiche « Quel est votre nom ? », puis lit l'entrée standard et la stocke dans une variable `$nom`, et enfin affiche Bonjour suivi de la variable `$nom`.
- Écrivez un script qui lit l'age de l'utilisateur sur l'entrée standard, et affiche « vous avez XX ans ».
- Écrivez un script qui prend en argument un age et affiche « vous avez XX ans ».
- Écrivez un script qui prend en argument un nom de fichier et affiche le contenu du fichier.
- Écrivez un script qui lit un nom de fichier sur son entrée standard et affiche le contenu du fichier.
- Écrivez un script qui prend en argument un dossier puis lit sur son entrée standard un nom de fichier, et affiche le contenu du fichier situé dans le dossier passé en argument.

Exercice 2 : Calculs

- Que fait la commande `expr 1 + 1` ?
- Quelle commande bash permet d'obtenir le résultat de la division de 714 par 17 ?
- Comment multiplier 6 par 7 ?
- Écrivez un script qui prend comme argument le type d'opération qu'on veut faire (`+` , `-`, `x`, `/`), puis attend deux nombres sur son entrée standard, et affiche le résultat du calcul.

Exercice 3 : La sortie standard

- Créez un nouveau répertoire vide dans le répertoire de ce TP, et déplacez vous-y.

- Tapez la commande **`echo Bonjour tout le monde`**. Bravo, la phrase « Bonjour tout le monde » est affichée sur la **sortie standard**.

À retenir

- (Selon Wikipedia) La **sortie standard** est le flux de sortie dans lequel les données sont écrites par le programme. Les données sont habituellement écrites à l'écran, à moins d'une **redirection**.
- On redirige la sortie standard vers un fichier avec soit **>** soit **>>** .

- Faites maintenant suivre la même commande de : **`> salutations.txt`** . La phrase « Bonjour tout le monde » n'est pas affichée sur la sortie standard. En effet, tout ce qui aurait du s'afficher sur la sortie standard a été **redirigée** vers le fichier `salutations.txt`. Comment pouvez vous le vérifier ?
- Créez un fichier *FichierETC.txt* contenant la liste de tous les fichiers du dossier `/etc/`.
- Écrivez un script `monNom.sh` qui lit votre nom sur son entrée standard, et l'écrit dans le fichier `monNom.txt` .
- Au lieu d'utiliser **>** pour la redirection, essayez **>>**. Essayer de faire successivement : **`echo Bonjour >> salutations.txt`** et **`date >> salutations.txt`**` . Quelle est la différence entre **>** et **>>** ?
- Écrivez un script qui lit votre nom sur son entrée standard, et écrit dans le fichier `monNom.txt` la phrase : Je m'appelle XXX et il est HH:MM heures . Aidez vous du man de date et du man de echo pour la mise en forme.

Exercice 4 : L'entrée standard

- Écrivez un script `troisLignes.sh` qui lit 3 lignes sur son entrée standard et les affiche.
- Exécutez la commande **`troisLignes.sh`** suivie de **`< monNom.sh`**. Cettre fois, au lieu de lire son entrée standard, c'est le contenu du fichier qui a été lu. On a **redirigé l'entrée standard**.

À retenir

- On redirige l'entrée standard à partir d'un fichier avec le caractère **<**

- Essayez la commande tr avec les arguments abcdf et ABCDEF: **`tr abcdef ABCDEF`**. Cette commande attend du texte sur son entrée standard. Pour signifier qu'on n'a plus rien à envoyer sur l'entrée standard (fermer l'entrée standard), on tapera `ctrl-d`. Que fait cette commande ?
- Écrivez un script qui prend en argument un nom de fichier, et écrit ce qui est tapé sur l'entrée standard dans le fichier passé en argument, en remplaçant tous les a par des @ et les e par des 3.
- Écrivez un script qui prend en argument un nom de fichier (par exemple truc.txt), lit ce fichier et crée un fichier MAJ avec le même contenu mais où toutes les lettres sont en majuscule.

II. Exercices

Exercice 1 : Ecrire un script

Version 1

Ecrire un script `rendExecutable.sh` qui demande un nom de fichier sur son entrée standard et qui le rend exécutable pour l'utilisateur puis affiche un message de compte-rendu.

Version 2

Ecrire un script `rendExecutable.sh` qui demande un nom de fichier sur son entrée standard et qui

1. Regarde si ce fichier est exécutable, puis le rend exécutable pour l'utilisateur si ce n'est pas le cas.
2. Affiche un message de compte-rendu dans chaque cas

Version 3

Ecrire un script `rendExecutable.sh` qui demande un nom de fichier sur son entrée standard et qui

1. Vérifie que le paramètre désigne bien un fichier
2. Regarde si ce fichier est exécutable, puis le rend exécutable pour l'utilisateur si ce n'est pas le cas.
3. Affiche un message de compte-rendu dans chaque cas

Indication:

L'instruction if permet d'effectuer des opérations si une condition est réalisée.

Tests

Les opérateurs de tests sur les objets du système de fichiers

- `[ -e $FILE ]` vrai si l'objet désigné par $FILE existe dans le répertoire courant,
- `[ -s $FILE ]` vrai si l'objet désigné par $FILE existe dans le répertoire courant et si sa taille est supérieure à zéro,
- `[ -f $FILE ]` vrai si l'objet désigné par $FILE est un fichier dans le répertoire courant,
- `[ -r $FILE ]` vrai si l'objet désigné par $FILE est un fichier lisible dans le répertoire courant,
- `[ -w $FILE ]` vrai si l'objet désigné par $FILE est un fichier inscriptible dans le répertoire courant,
- `[ -x $FILE ]` vrai si l'objet désigné par $FILE est un fichier exécutable dans le répertoire courant,
- `[ -d $FILE ]` vrai si l'objet désigné par $FILE est un répertoire dans le répertoire courant.

Exercice 2 : Ecrire un script

Ecrire un script `compte.sh` qui prend en *paramètre* une extension et un répertoire *rep* et qui compte le nombre de fichiers dont le nom se termine par l'extension (précédée d'un point) dans le répertoire *rep*. Par exemple :

```
bob@info42_13:~/Documents/InitSysteme$ ./compte.sh py /home/bob
Le nombre de fichiers dont le nom se termine par .py
dans le répertoire /home/bob est :
13
```

Exercice 3 : Pour vérifier ses connaissances

La commande `ssh info25-03 who` affiche le nom des personnes loggées sur la machine *info25-03* et affiche par exemple :

```
chabin    :0     2015-10-09 07:57 (:0)
robert    pts/1  2015-10-09 08:34 (:0.0)
```

1. Comment écrire ces informations dans un fichier `user.log` ?
2. Proposez un script qui prend en *argument* un nom de machine (par exemple `info25-03`) et qui
   - crée un fichier `info25-03.log` dans lequel il écrit le nom des personnes logguées sur cette machine
   - affiche les informations dans le terminal
3. **Défi** Proposez un script qui prend en *argument* un nom de machine (par exemple *info25-03*) et qui :
   - crée un fichier dont le nom est au format **MACHINE.JOUR-MOIS-ANNEE.log** (par esemple `info25-03.19-09-2017.log` dans lequel il écrit le nom des personnes logguées sur cette machine
   - affiche dans le teminal le nom (et uniquement le nom) des personnes logguées sur cette machine

Exercice 4 : Entrée standard

- Écrivez un fichier `essai.txt` qui, lorsque vous taperez la commande : **`tr Mi To < essai.txt`** affiche **`Toto`** .
- Écrivez un fichier `revelations.txt` qui, lorsque vous taperez la commande : **`tr foqrp hsmab < revelations.txt`** affiche **`J'aime le bash`** .
- Écrivez un fichier `lignes.txt` qui, lorsqu'on tape la commande **`wc -l < lignes.txt`** affiche **`10`**.

Exercice 5 : Sortie standard

- À l'aide de la commande **`seq`** et de la redirection de la sortie standard, écrivez un fichier contenant les nombres de 1 à 1000.
- Créez un fichier contenant tous les nombres de 1 à 1000, mais où les 0 sont remplacés par des @ . (pensez à la commande **`tr`**)
- Créez un fichier contenant tous les nombres de 1 à 1000, mais où on a enlevé tous les chiffres autres que 0 (en utilisant l'option **`-d`** de tr par exemple ..). Essayez aussi d'enlever les retours à la ligne (symbole **`n`**) .
- Combien y a-t-il de 0 si on écrit tous les nombres de 1 à 10000 ?

Exercice 6 : Grep

La commande **`grep`** lit son entrée standard et affiche les lignes qui contiennent l'argument passé à grep. Par exemple : **`grep toto < truc.txt`** affichera toutes les lignes de `truc.txt` qui contiennent le mot *toto*.

- Écrivez dans un fichier la liste des fichiers de votre home.
- À l'aide de *grep*, listez les fichiers de votre home qui contiennent « txt » .
- Créez un fichier qui contient la liste des fichiers de /usr/bin qui contiennent « rm » .
- Combien ce fichier contient-il de lignes ?
- Combien de fichiers de /usr/bin contiennent la lettre « o » ?

Avec grep, on peut aussi spécifier des contraintes sur le texte recherché. Par exemple : **`grep ^toto`** affichera toutes les lignes commençant par toto. **`grep toto$`** affichera toutes les lignes terminant par toto.

- Dans un fichier, écrivez la liste des fichiers de /usr/bin terminant par n.
- Si on écrit les nombres de 1 à 10000, combien commencent par 1 ?

Chapter 10 - Système et Python

I. Bash et Python

Exercice 1 : Script python

Créez un répertoire pour ce TP et déplacez vous y. Écrivez le script python suivant dans un fichier script.py :

```
#!/XXX
# Auteur :  A. B.
# Ce script dit bonjour
print("Bonjour")
```

- Par quoi devez vous remplacer les XXX du début du script pour qu'il fonctionne ?
- Exécutez le script et redirigez la sortie pour que Bonjour soit écrit dans le fichier sortie.txt.

Exercice 2 : Arguments en python

Créez et exécutez le script python suivant:

```
#!/XXX
# Auteur :  A. B.
# Ce script affiche ses arguments

import sys
print(sys.argv)
```

- Quel est le type de sys.argv ? Que contient sys.argv ?
- Modifiez le script pour qu'il affiche : « Le premier argument est : » suivi du premier argument.
- Modifiez votre script pour qu'il affiche le nombre d'arguments.
- Modifiez votre script pour qu'il affiche les arguments séparés par des espaces, comme le fait la commande echo. Pour cela, créez une chaine de caractères initialement vide, à laquelle vous ajoutez les mots de sys.argv en utilisant une boucle for.
- Modifiez votre script pour que si le premier argument tapé est « -s », les arguments soient affichés non plus séparés par des espaces, mais par des virgules.

Exercice 3 : Entrée standard en python

Créez et exécutez le script python suivant :

```
#!/XXX
# Auteur :  A. B.
# Ce script vous dit bonjour
print("Quel est votre nom ?")
nom=input()
print("Bonjour "+nom)
```

- Modifiez le script pour qu'il demande aussi votre age et affiche « Bonjour XX vous avez XX ans »
- Modifiez le script pour qu'il affiche « Bonjour XX, vous êtes mineur » ou « Bonjour XX, vous êtes majeur » selon l'age entré. Pour convertir une chaine `chaine` en entier, on fait : `int(chaine)`.
- Modifiez le script pour qu'il prenne en argument un age. Si l'argument est passé, l'age n'est pas demandé et celui donné en argument est utilisé. Si l'argument n'est pas passé, le comportement est celui de la question précédente.

Exercice 4 : Recopie.py

- Créez un script qui lit son entrée standard et (re)affiche ce qui a été tapé.
- Modifiez le script pour qu'une fois que ce qui a été tapé a été affiché, il recommence, i.e. lit à nouveau l'entrée standard, et la recopie.
- Modifiez le script pour que ceci se répète indéfiniment (avec une boucle while) .
- Modifiez le script pour qu'il ajoute au début de chaque ligne le numéro de la ligne (i.e. 1 pour la première ligne, 2 pour la seconde, etc.) .
- Utilisez ce script pour créer un fichier `script_numerote.txt` contenant les lignes du fichier `script.py` numérotées.

II. Exercices

Exercice 1 : Problèmes

Pour chacune des questions suivantes, vous devrez imaginer un script python vous aidant à résoudre le problème. On se base ici sur le `texte du Comte de Monte-Cristo`.

- Créez un fichier comte-numerote.txt où toutes les lignes du texte ont été numérotées.
- Créez un fichier comte-un-sur-deux.txt contenant une ligne sur deux du texte original.
- Créez un fichier comte-M.txt contenant toutes les lignes commençant par M.
- Créez un fichier comte-nbcarac.txt où chaque ligne est précédée de son nombre de caractères.
- Créez un fichier comte-alenvers.txt où les lignes du texte sont dans l'ordre inverse : la dernière ligne du texte est la première et la première est la dernière. (Nota : pour cette question il faut savoir manipuler les listes en python)
- Créez un fichier comte-séparé.txt où toutes les lignes commençant par un M sont au début et toutes les autres sont à la fin.

Exercice 2 : Pour s'entrainer

- Écrivez un script python qui prend en argument un mot et recopie sur sa sortie standard toutes les lignes lues sur son entrée standard et qui contiennent ce mot.
- Écrivez un script python qui permet de compter le nombre de fois qu'un mot passé en argument apparaît dans l'entrée standard. Utilisez le pour compter le nombre de fois que le mot « et » apparaît dans le comte de Monté-Cristo .
- J'ai un fichier python dans lequel je me suis trompé, j'ai utilisé des tabulations au lieu de 4 espaces (!!) pour l'indentation. Proposez un script qui me permette de régler le problème facilement.

Chapter 11 - Système et Python (suite)

I. Scripts Python et variables d'environnement

Exercice 1 : Rappels

En bash, vous pouvez définir des variables d'environnement soit « locales », soit transmises aux programmes appelés :

```
maVariableLocale=5     # Local
echo $maVariableLocale
export maVariable=5    # Transmise
```

Vous pouvez afficher les variables d'environnement à l'aide de la commande **`env`**

Exercice 2 : En python

Les variables d'environnement sont aussi accessibles en python :

```
import os
print os.environ
```

- Écrivez un script python qui affiche le nom de login de l'utilisateur en utilisant la variable d'environnement « USER ».
- Ajoutez une ligne à votre script pour qu'il affiche le répertoire courant (PWD).
- Ajoutez une ligne à votre script pour qu'il affiche le chemin vers votre home.
- Ajoutez une ligne pour qu'il affiche le contenu de la variable d'environnement « TOTO ». Essayez de définir cette variable avant d'appeler votre script.

II. Encodage des charactères

III. Execution de commandes bash

Exercice 1 : subprocess.run

Depuis python vous pouvez exécuter des commandes bash de la manière suivante :

```
import subprocess
entree_standard = "banane"
retours=subprocess.run(["tr","a","b"],stdout=subprocess.PIPE,input=entree_standard.
↪encode("utf-8"))
print(retours.stdout.decode("utf-8"))
```

- Écivez un script python qui affiche tous les fichiers du dossier courant.

Remarque : en python, vous pouvez transformer une chaine de caractères en une liste de chaines en utilisant la méthode split des chaines qui prend en argument le caractère permettant de découper la chaine :

```
chaine = "Bonjour tout le monde. Ceci est un exemple."
print(chaine.split(" "))
#Affiche :  ['Bonjour', 'tout', 'le', 'monde.', 'Ceci', 'est', 'un', 'exemple.']
print(chaine.split("."))
#Affiche : ['Bonjour tout le monde', ' Ceci est un exemple', '']
```

- Écrivez un script python qui affiche tous les fichiers du dossier courant dont le nom commence par « t » .
- Écrivez une script python qui affiche les noms des fichiers du répertoire courant suivi d'une * si c'est un répertoire et de rien si c'est un fichier.

Exercice 2 : Application

- A l'aide d'un script python, céez un dossier contenant 10 fichiers vides : « fichier-0 », « fichier-1 », . . . , « fichier-9 ». (pour rappel, on peut créer un fichier vide avec la commande **`touch`**). Assurez vous que si vous vouliez créer 1000 fichiers vides, il n'y aurait pas beaucoup de choses à changer dans votre script.
- Modifiez votre script pour que le préfixe (« fichier ») et le nombre de fichiers soient des arguments du script. Par exemple, en faisant **`./monScript.py toto 12`**, il génère les fichiers `toto-0`, . . . , `toto-11` .
- Écrivez un script bash qui prend deux arguments: un nom de fichier et une chaine de caractères, et crée le fichier avec comme contenu la chaine de caractères. Par exemple **`creeFichier.sh toto12 bonjour`** crée le fichier `toto12` et y écrit bonjour. Pour vous mettre sur la voie, utilisez echo et la redirection de sortie standard.
- Créez un script qui prend en argument un prefixe et un nombre de fichiers, et crée les fichiers prefixe-0, . . . , comme précédemment, mais dont le contenu est le numéro du fichier. Ainsi, prefixe-0 contiendra « 0 », prefixe-1 contiendra « 1 » , etc. (utilisez le script de la question précédente)

IV. Boucles en bash

V. Exercices

Exercice 1 : Autour du web

La commande `curl http://www.meteofrance.com/mf3-rpc-portlet/rest/pluie/452340` récupère les prévisions météo dans l'heure pour Orléans à condition que la variable d'environnement http_proxy soit définie et vaille : `http://wwwcache.univ-orleans:3128/`

- Essayez cette commande en redirigeant la sortie vers un fichier « meteo.txt ».

Dans la suite on travaillera avec ce fichier pour ne pas faire des requêtes inutiles sur le site de meteofrance (et prendre le risque qu'il se mette à nous refuser de répondre ..). On remplacera donc la commande précédente par **`cat meteo.txt`**

- Observez les données récupérées.
- Écrivez une script python qui affiche la météo pour les dix prochaines minutes.
- Modifiez le script pour qu'il affiche "Aucun risque de pluie dans l'heure" s'il n'y a aucun risque de pluie, et "risque de pluie dans l'heure" sinon.

Exercice 2 : Autour des données système

Le code python suivant permet d'afficher « Bonjour » toutes les secondes.

```
#!/usr/bin/python3
import time

while(True):
  print("Bonjour")
  time.sleep(1)
```

- Écrivez une script python qui affiche l'heure courante en permanence. (Vous pouvez utiliser la commande « clear » de bash pour effacer l'écran).
- Écrivez un code qui permet d'afficher en permanence le processus prenant le plus de ressources (pensez à regarder le man de ps pour trier les processus par utilisation de CPU).
- Écrivez un script python qui affiche la liste des utilisateurs loggués. Chaque utilisateur ne doit apparaître qu'une seule fois !

- **Défi** Écrivez une script qui affiche la liste des utilisateurs présents dans la salle. (vous pouvez utiliser ssh et who)

Chapter 12 - Les commandes à connaître

I. Explorer l'arborescence

- ls
- cd

II. Gérer les fichiers

- cp
- mv
- rm
- mkdir

Chapter 13 - La commande grep

Téléchargez le fichier `insee.csv` qui contient des informations sur les villes de France : chaque ligne contient le nom de la commune, le code postal et le département (ces éléments étant séparés par des points-virgules).

1. commande permet de savoir le nombre de communes de code postal 45400 ?
2. Quelle commande permet de savoir le nombre de communes dont le nom contient le mot Saint mais pas Laurent ?
3. Quelle commande permet de lister le nom des communes du département OISE ? Attention à ne lister *que ces communes* : par exemple, on ne doit pas lister les communes du Val d'Oise.
4. On remarque que certains noms de communes sont utilisés plusieurs fois. Par exemple Olivet est une ville du Loiret, mais aussi de la Mayenne. On aimerait savoir quel nom de ville est le plus utilisé. Quelle commande permet de répondre à cette question ?

Chapter 14 - Les commandes grep et cut

Téléchargez le fichier `championsLOL.csv`. Ce fichier contient des informations sur les champions de LOL : chaque ligne contient le nom d'un champion, son genre (m pour masculin, f pour féminin) du champion, ainsi que ses rôles dans le jeu (ces éléments étant séparés par des « deux-points »).

1. Quelle commande permet d'afficher la première ligne du fichier ?
2. Quelle commande permet d'afficher le nombre de champions *Combattants* ?
3. Quelle commande permet d'afficher le nombre *Combattants* qui ne sont pas des *Tanks* ?
4. Quelle commande permet d'afficher le nom des champions féminins ?
5. Comment afficher la liste des différents rôles principaux possibles (en évitant les doublon) ?
6. **Défi** On aimerait savoir quel est le rôle de plus représenté. Quelle commande permet de répondre à cette question ?

Chapter 15 - Pipe : lire des commandes

Que font les commandes suivantes ?

1. `ls .. | grep -i "^a"`
2. `ls *.txt | sort >> toto.txt`
3. `cat < toto.txt`
4. `echo toto.txt $PATH`

Chapter 16 - Indices and tables

- genindex
- modindex
- search