NUMP1 : TP Unix - 1

Rémi Giraud - remi.giraud@enseirb-matmeca.fr

2026-2027

1 Rappel

1.1 Création et suppression de répertoires

La commande `mkdir` (make directory) permet de créer un répertoire. Lors de la création de ce répertoire, deux fichiers spéciaux sont créés : `.` et `..` Ces deux fichiers spéciaux sont des liens vers le répertoire courant et le répertoire "parent" utile pour l'exécution de certaines commandes prenant un répertoire en argument.

Question 1 Créez un répertoire `tmp` et un répertoire `Projets` dans votre répertoire d'accueil. Plusieurs possibilités, par exemple :

```
$ mkdir ~/tmp
$ cd
$ mkdir Projets
$ cd Projets
$ ls -als
$ pwd
$ cd ..
$ pwd
```

Si le chemin est celui d'un fichier (ou répertoire) déjà présent, cela provoque une erreur : le nom de base doit être nouveau dans le répertoire de référence.

```
$ mkdir eclair
$ ls -F
$ cd eclair
$ rmdir ../eclair
$ cd ..
$ rmdir eclair
$ ls -F
```

Question 2 Essayez `rmdir ~` et commentez le résultat obtenu.

1.2 Manipulation de fichiers

1.2.1 Copies

Question 3 Que font les commandes :

```
$ mkdir test
$ cd test
$ touch doc1.html
$ touch doc2.html
$ ls
$ cd
$ cd test
$ cp doc1.html doc2.html ~/tmp
$ cp *.html ~/tmp
$ cd ~/tmp; cp ~/test/* .
```

Quelle est la différence entre les deux premiers `cp` ?

1.2.2 Suppressions.

Question 4 Essayez la suite de commandes :

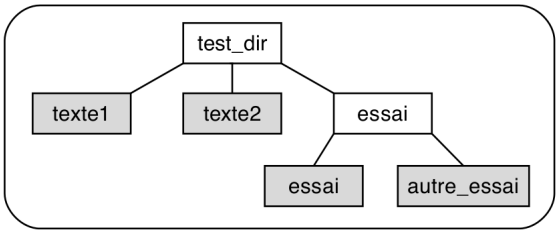
```
$ rm ~/tmp/README.html
$ cd ~/tmp
```

Pour la suppression interactive essayez : `rm -i` $< fichier >$ `...` (répondre `y` pour confirmer la suppression et `n` pour infirmer).

1.2.3 Déplacements et renommages.

```
$ cd ~/test
$ mv doc1.html doc1.html
$ ls -F
$ mv doc1.html ../doc1.html
$ ls -F
$ ls -F ..
$ mv ../doc1.html .
$ ls -F ..
$ ls -F
$ mv *.html ~/tmp
$ ls -F
$ mv ~/tmp/*.html .
```

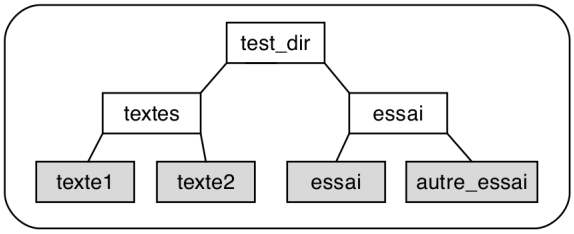
Question 5 Créez à partir de votre répertoire d'accueil l'arborescence :



Les cases blanches sont des répertoires, tandis que les cases grisées sont des fichiers texte à créer. La marche à suivre pour cet exercice est la suivante :

— créez les répertoires ;
— lancez emacs à partir du répertoire d'accueil ;
— éditez les fichiers et sauvegardez-les.

Question 6 Placez-vous dans le répertoire `test_dir`, et ramenez-vous en deux commandes à :



Question 7 Depuis `test_dir` essayez `cp essai textes` ; commentez le résultat. Que fait dans ce même cas `cp *` ? Essayez avec l'option `-R`. Commentez les erreurs possibles lors de l'exercice précédent :
$ mv t* textes
$ cp t* textes

2 Arborescence utilisateur

2.1 Chemins.

On différencie deux types de chemin, absolu et relatif.

Le chemin absolu spécifie un nom de fichier ou de répertoire à partir de la racine. Les chemins d'accès absolus commencent donc par "/" qui est le nom de la racine.

Les chemins d'accès relatifs spécifient un nom de fichier ou de répertoire à partir du répertoire courant. Par conséquent, ces chemins ne commencent pas par "/".

Le chemin relatif évolue bien sûr en fonction de votre navigation dans vos dossiers.

Question 8
— Faites `$ echo ~`
— Par quoi le shell substitue le caractère ~ ?
— Est-ce un chemin absolu ou relatif ?
— Comment s'appelle ce répertoire ?
— Qu'en concluez-vous sur le chemin de la commande `$ cd ~/test` ? Est-il relatif ou absolu ?

Question 9 En partant de l'arborescence de la question 6, placez vous dans le répertoire `essai`. Accédez au répertoire `textes` de deux façons différentes, par un chemin relatif et par un chemin absolu.

Question 10 Pour les commandes suivantes déterminez si les chemins donnés en argument sont relatifs ou absolus.

```
$ mkdir ~/tmp
$ mkdir Projets
$ cd ..
$ cd ~/test
```

```
$ rmdir ../eclair
$ mkdir test
$ cd test
$ touch doc1.html
$ cp doc1.html doc2.html ~/tmp
$ cd /usr/bin
```

2.2 Types de fichier.

Sous Unix un fichier est :
— toujours désigné par un nom.
— possède un unique inode (certaines informations concernant le fichier).
— possède les fonctionnalités suivantes : ouverture, fermeture, lecture, écriture.

Un fichier peut être :
— un fichier régulier (-).
— un répertoire (d).
— un lien symbolique (l).
— une socket (s). Sert à communiquer par le réseau.
— un block device (b). Opération sur un périphérique capable de stocker de la donnée (ex. disque).
— un character device (c). Opération sur un périphérique incapable de stocker de la donnée (ex. souris, webcam).
— ...

Question 11 Trouvez l'option de la commande `ls` permettant de visualiser le type des fichiers, et cherchez dans votre arborescence différents types parmi ceux cités.

2.3 Lecture de fichiers

Question 12 Exécutez la ligne de commande suivante :
`man find > man_find.txt`
Les commandes `cat`, `more`, `less` permettent de lire le contenu d'un fichier. Par exemple, tapez la commande : `cat man_find.txt`. Avez-vous pu lire le début du fichier texte ? Tapez ensuite la commande `more man_find.txt`. Pouvez-vous naviguer dans le fichier ? Tapez la commande `less man_find.txt` (tapez `h` pour avoir de l'aide sur la commande `less`).

2.4 Liens

Il existe des liens physiques (hardlinks) et des liens logiciels ou symboliques (softlinks or symlinks). Pour comprendre la différence entre ces liens, il faut détailler le système de fichier. Toute donnée est liée à un identificateur de fichier/dossier (un dossier n'étant qu'un vecteur de fichiers) appelé "inode". Chaque inode contient l'adresse des données, les droits en lecture/écriture/exécution de chaque dossier/fichier, etc... Chaque répertoire ou fichier est lié à un numéro inode visible avec la commande `ls -lia`. Si plusieurs fichiers ont le même inode, ces fichiers sont liés physiquement. On ne peut donc pas créer de lien physique entre deux répertoires. Par exemple, le fichier spécial `.` et `..` ont respectivement le même inode que celui du répertoire courant et parent. Les fichiers spéciaux `.` et `..` sont donc des liens physiques pointant vers les

répertoires courant et parent. Un lien logiciel ou symbolique est un fichier spécial pointant sur un autre inode qui pointe vers des données.

Voici quelques contraintes :

— Les liens physiques doivent relier deux fichiers du même système de fichiers ;
— les liens physiques peuvent lier à une source supprimée ;
— les liens symboliques ne sont pas mis à jour ;

Question 13 Dans un répertoire vide, essayez les commandes suivantes :

```
$ touch fichier
$ ln fichier hardlink
$ ln -s fichier symblink
$ ls -F
$ rm fichier
$ ls -l
$ more hardlink
$ more symblink
```

3 Protections - Les droits d'accès unix

Un fichier appartient :

— à un utilisateur, qui est son propriétaire
— à un groupe (pas nécessairement celui de son propriétaire)

Le mode d'accès définit l'accès en $\left\{ \begin{array}{l} \text{lecture} \\ \text{écriture} \\ \text{exécution/passage} \end{array} \right.$ pour $\left\{ \begin{array}{l} \text{le propriétaire} \\ \text{les membres du même groupe} \\ \text{les autres utilisateurs} \end{array} \right.$.

Cela fait 9 informations élémentaires (3 champs de 3 flags), que l'on note :

$$\underbrace{\texttt{rwx}}_{u}\underbrace{\texttt{rwx}}_{g}\underbrace{\texttt{rwx}}_{o}$$

On utilise un - pour indiquer qu'un accès n'est pas autorisé :

```
rw-r-r-
rwxr-xr-x
```

3.1 Modification des accès

`chmod` $< champs >^{+}_{-} < accs >$ $< chemin >$

`chmod` $< champs >= < accs >$ $< chemin >$

`chmod` $< codeoctale >$ $< chemin >$

Exemple :

`chmod o-r` $< fichier >$
`chmod +w,o-x` $< chemin >$
`chmod 744` $< chemin >$

En mode octale, 1 à 4 chiffres de 0 à 7 (seulement 3 utilisés ici) sont utilisés. Tout chiffre manquant est considéré comme nul. Chaque chiffre fait référence à un champ. Vous pouvez convertir ce chiffre octale en 3 chiffres binaires avec des poids respectifs :

$$\underbrace{\texttt{r}}_{2^2} + \underbrace{\texttt{w}}_{2^1} + \underbrace{\texttt{x}}_{2^0} \tag{1}$$

Question 14 Quels sont les droits donnés au fichier `Test` à la fin de chaque commande :

```
$ touch Test
$ chmod 444 Test
$ chmod +w,o-x Test
```

Question 15 Que permettent de faire les commandes `chgrp` et `chown`. Donnez des exemples.

Question 16 Faites `ls -l` dans votre répertoire d'accueil. Commentez l'affichage.

Question 17 Pourquoi la commande `ls -l` ne liste pas les droits du répertoire courant (rechercher au moyen de `man` la façon de faire).

3.2 Droit d'examiner/modifier le contenu d'un fichier, de lancer un fichier exécutable

Question 18 Modifiez les droits des répertoires parents pour identifier ceux nécessaires pour la lecture, pour l'exécution et la modification d'un fichier.

Question 19 Modifiez les droits d'un fichier pour identifier ceux nécessaires pour la lecture, pour l'exécution et la modification d'un fichier.

3.3 Droit de lister le contenu d'un répertoire

Question 20

- Depuis votre répertoire d'accueil, listez le contenu du répertoire "Documents".
- Sur le répertoire parent du répertoire "Documents", enlevez les droits en exécution.
- Tentez de lister le contenu du répertoire "Documents".
- Restaurez les droits.

Question 21 Sur le répertoire "Documents" essayez successivement de lister son contenu après avoir enlevé sur ce répertoire :

- le droit en lecture
- le droit en écriture
- le droit en exécution

Question 22 D'après les deux questions précédentes qu'en concluez vous sur le droit de listage d'un répertoire ?

3.4 Droit de créer, de renommer ou d'effacer un fichier

Question 23
- Sur un répertoire parent du répertoire "Documents", enlevez les droits en exécution.
- Tentez de créer un fichier dans le répertoire "Documents".
- Restaurez les droits.

Question 24 Dans le répertoire "Documents" essayez successivement de créer/renommer/effacer un fichier après avoir enlevé sur ce répertoire :
- le droit en lecture
- le droit en écriture
- le droit d'exécuter

Question 25 Vérifiez que l'on peut avoir le droit de détruire un fichier sans avoir le droit de le lire (ou même n'ayant aucun droit sur le fichier lui-même).

Question 26 D'après les questions précédentes qu'en concluez-vous sur le droit de créer, de renommer ou d'effacer un fichier ?

Question 27 Dans quel cas peut-on détruire un fichier qui ne nous appartient pas ?

4 Visite de l'arborescence du système Unix

Question 28 Présentez les principaux répertoires du système Unix.
- `cd /; ls -l`
- observez les droits d'accès
- donnez le rôle de `/bin` ; ...
- donnez le rôle de `/lib` ou `/usr/lib` ;
- donnez le rôle de `/tmp`, `/dev`, `/dev/null`
- visitez et commentez `usr`, en particulier `include` et `bin` (`gcc, java`).
- testez la commande `which` < *commande* > qui permet d'identifier le chemin de la commande.

Question 29 Essayez la commande `find /usr/lib -name *.a` .

Question 30 Utilisez l'option `-type` de `find` pour lister les répertoires et sous-répertoires de `/dev`.

Question 31 Donnez la commande permettant de trouver des fichiers chez vous modifiés depuis hier.

Question 32 Donnez la commande permettant de trouver chez vous les fichiers modifiés il y a plus de deux jours mais moins de 10 jours.