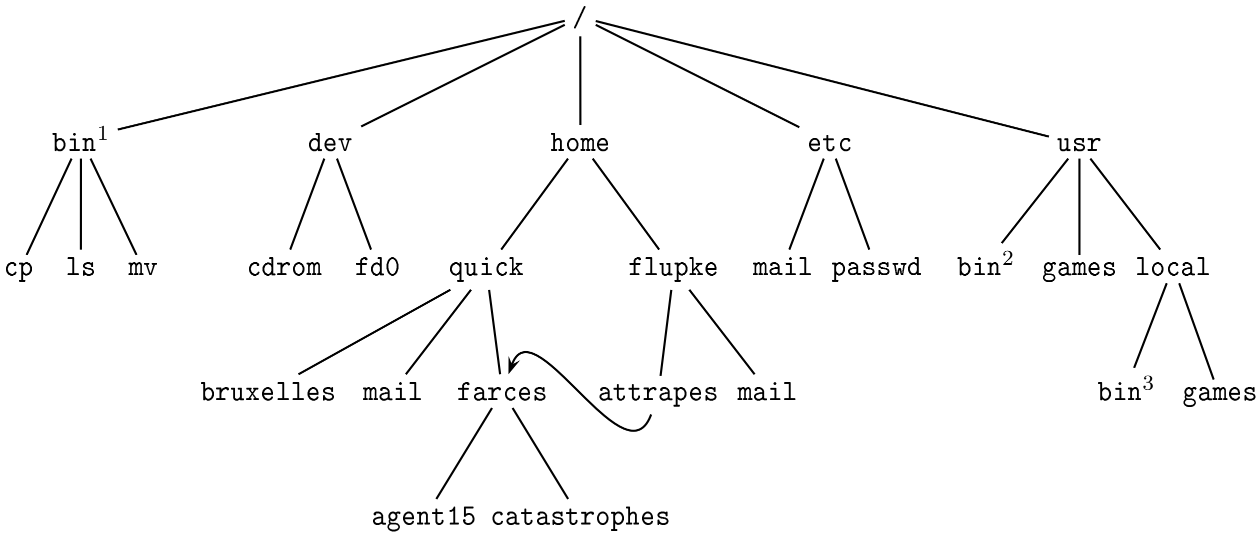
Feuille de TD nº 1 : le système de fichiers d'Unix

Voici un morceau d'arborescence Unix (la flêche représente un lien symbolique) :



La commande `ls -il . flupke quick quick/farces` effectuée depuis le répertoire `/home` de l'arborescence précédente fournit la réponse suivante :

```
.:
total 8
1370580 drwxr-xr-x   3 flupke  gamins   4096 Jul 23 14:32 flupke
6604110 drwxr-xr-x   4 quick   gamins   4096 Nov 14 16:15 quick

flupke:
total 5
1371135 lrwxrwxrwx   1 flupke  gamins     15 Jul 23 10:32 attrapes -> ../quick/farces
1370581 drwx------   2 flupke  gamins   4096 Jul 16 15:30 mail

quick:
total 12
6604111 -rw-r-----   1 quick   gamins   7218 Sep 22 11:35 bruxelles
1524773 drwxrwx---   2 quick   gamins   4096 Nov 14 12:12 farces
1426824 drwx------   2 quick   gamins   4096 Nov 14 10:06 mail

quick/farces:
total 8
1524805 -rw-rw----   1 flupke  gamins   6133 Jul 23 10:23 agent15
1525129 -rw-r-----   1 quick   gamins  12284 Nov 14 12:11 catastrophes
```

Exercice 1 : chemins absolus et relatifs

1. Donnez les chemins d'accès absolus aux différents répertoires `bin`, `games` et `mail`.
2. Donnez des chemins d'accès relatifs aux répertoires $\texttt{bin}^1$ et $\texttt{bin}^3$ depuis $\texttt{bin}^2$.
3. Donnez deux chemins d'accès relatifs différents au fichier `agent15` depuis `flupke`.
4. Créez un répertoire `jeux_interdits` dans le répertoire `flupke` depuis `quick`.
5. Depuis `flupke`, copiez `catastrophes` dans `jeux_interdits` sous le nom `cata1`.
6. Déplacez le fichier `bruxelles` dans `flupke` depuis `flupke`.

Exercice 2 : `chmod`

1. À quels fichiers/répertoires de `quick` `flupke` a-t-il accès en lecture ? en écriture ? Quid de l'utilisateur `agent15` appartenant au groupe `adultes` ?
2. Comment `quick` peut-il interdire à `flupke` d'écrire dans le répertoire `farces` ?
3. Comment peut-il autoriser `agent15` à lire le fichier `bruxelles` ?
4. Comment `flupke` peut-il protéger en lecture l'intégralité de son répertoire principal ?

Exercice 3 : `ln`

La commande `ln` *toto* *tutu* permet de créer un lien appelé *tutu* vers le fichier *toto*. L'option `-s` permet de créer un lien symbolique.

1. Comment `flupke` peut-il créer dans `jeux_interdits` un lien `cata2` et un lien symbolique `cata3` vers `catastrophes` ?
2. Que répond alors la commande `ls -il /home/flupke/jeux_interdits` ? et la commande `ls -il /home/quick/farces` ?
3. Quel est l'effet de `rm /home/quick/farces/catastrophes` ?

Exercice 4 : aller plus loin avec `ls`

Par défaut, `ls` *rep* affiche tous les fichiers du répertoire *rep* (sauf les fichers cachés). On peut vouloir n'afficher que certains fichiers, et pour cela spécifier ce que l'on veut de la manière suivante :

– `*` remplace n'importe quelle suite de caractères (même vide).
– `?` remplace n'importe quel caractère.
– `[a-f]` remplace n'importe quel caractère de l'intervalle délimité par `a` et `f`.
– `[^a-f]` remplace n'importe quel caractère, sauf ceux de l'intervalle `[a-f]`.

Le shell interprète alors l'expression indiquée et la remplace par tous les noms de fichiers qu'elle reconnaît. Dans un répertoire contenant les fichiers `toto, tutu, coucou`, `ls t*` est une instruction équivalente à `ls toto tutu`. Toute autre instruction suivie d'une expression comportant des symboles tels que `*` ou `?` sera traitée de la même manière par le shell, à moins que l'expression ne soit *protégée* par des apostrophes `'` ou des guillemets `"`.

Comment lister les fichiers dont le nom commence par `a` ? finit par `e` ? contient un point ? contient un point d'interrogation ? contient deux `t` ? ne se termine pas par une lettre comprise entre `c` et `g` ? contient un `b` en deuxième position ? contient deux lettres successives ?

Exercice 5 : un peu de `grep`

La commande `grep -E 'regexp' fichier` retourne les lignes d'un fichier contenant l'*expression régulière* `regexp`. Une expression régulière est un mot formé de lettres (qui sont des *éléments*) et des symboles :

- $*$ qui indique que l'élément précédent apparaît un nombre quelconque de fois
- $+$ qui indique que l'élément précédent apparaît au moins une fois
- (et) qui entourent une expression régulière pour former un élément
- | qui indique une disjonction entre deux expressions régulières (on reconnaît l'une ou l'autre)
- ? qui indique que l'élément précédent est facultatif (i.e. il apparaît au plus une fois).

Soit tutu le fichier contenant :

caa
cbbab
cabab
caaabba

Que répond la commande `grep -E 'regexp' tutu` lorsque `regexp` vaut respectivement :

- ca^+
- $c(ab)^+$
- $ca^+(a^*b^*)$
- $(aa^+)|(bb^+)$
- $baa?b$
- a^*

et pourquoi ?