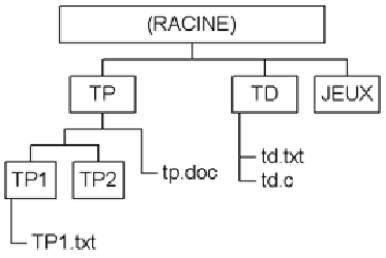
Arborescence, droits d'accès, interpréteur de commandes, shell

Exercice 1 :

Soit l'arborescence de la figure ci-dessous. Si le répertoire courant est JEUX, comment copier tp.doc dans le répertoire TD ? Ecrire la réponse en chemins absolus et en chemins relatifs.

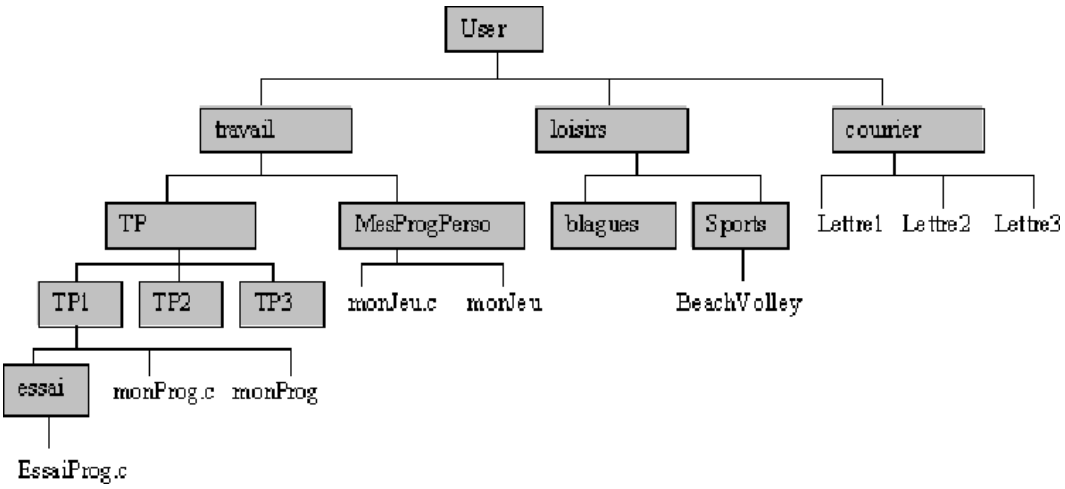


Solution :

`cp /TP/tp.doc /TD`
ou bien
`cp ../TP/tp.doc ../TD`
On peut aussi commencer par `cd ..` pour simplifier l'écriture des chemins.

Exercice 2 :

Soit l'arborescence de la figure ci-dessous. Donner la liste des commandes unix pour faire les actions suivantes :



1. Je suis dans « User », je veux aller dans « essai ».
2. Je suis dans « blagues », je veux aller lire le score de Beach-Volley qui est dans mon fichier « Beach-Volley ».
3. Je suis dans « blagues », je veux lire le score de Beach-Volley qui est dans mon fichier « BeachVolley » sans bouger de mon répertoire courant.
4. Je suis dans « TP », je veux aller créer un fichier « minmax.c » dans le répertoire « MesProgPerso ».

5. Je suis dans « essai », je veux créer un répertoire « TP4 » dans le répertoire « TP » et y mettre le programme « monProg » en le laissant aussi dans « TP1 ».
6. En reprenant l'arborescence initiale, que font ces commandes ?
   — `less travail/` (répertoire courant : User)
   — `less Sports/Beachvolley` (répertoire courant : User)
   — `less /courrier/lettre1` (répertoire courant : User)
   — `cd monJeu` (répertoire courant : MesProgsPerso)
   — `less monProg` (répertoire courant : TP1)
   — `gedit toto.c`
   — `gedit monJeu.c` (répertoire courant : TP1)

Solution :

1. `cd travail/TP/TP1/essai`
2. `cd ../Sports` puis `cat BeachVolley`
3. `cat ../Sports/BeachVolley`
4. `touch ../MesProgPerso/minmax.c` (ou bien en deux fois avec `cd`)
5. `cd ../..` puis `mkdir TP4` et `cp TP1/monProg TP4`
   D'autres variantes sont possibles sans changer de répertoire, ou en se plaçant dans `TP1`
6. — Erreur : `travail` est un répertoire et pas un fichier
   — Erreur : pas de répertoire `Sports` dans `User`
   — Erreur : pas de répertoire `courrier` à la racine (et en plus le nom du fichier `Lettre1` commence par une majuscule)
   — Erreur : `monJeu` n'est pas un répertoire
   — Affiche le contenu de `monProg`
   — Crée un nouveau fichier `toto.c` et l'ouvre dans un éditeur de texte
   — Crée un nouveau fichier `monJeu.c` dans le répertoire TP1 et l'ouvre dans un éditeur de texte

Exercice 3 :

Comment échanger les contenus de deux fichiers nommés `cafe` et `x2.c`,
(a) s'ils sont tous deux dans le même répertoire `TP1` ?
(b) si `cafe` est dans le répertoire personnel et `x2.c` dans le répertoire `TP1` ?

Solution :

La seule solution qui n'utilise ni de programme externe ni de redirection d'entrée/sortie, consiste à intervertir les fichiers, soit en les renommant avec la commande `mv`, soit avec la commande `cp`. Pour les deux cas, il faudra utiliser un fichier temporaire comme lors de la permutation de valeur entre deux variables.

Solutions pour (a)

avec la commande `mv` :

```
{enseignant} 1 > mv cafe fichier_temporaire
{enseignant} 2 > mv x2.c cafe
{enseignant} 3 > mv fichier_temporaire x2.c
```

ou grâce à la commande `cp` qui écrase les fichiers sans demander confirmation :

```
{enseignant} 1 > cp cafe fichier_temporaire
{enseignant} 2 > cp x2.c cafe
{enseignant} 3 > cp fichier_temporaire x2.c
{enseignant} 4 > rm fichier_temporaire
```

Attention à ne pas écraser le fichier `fichier_temporaire` s'il existait préalablement.

Solution pour (b)

Ce sont les mêmes commandes, sauf que les fichiers sont identifiés avec leur chemin dans la commande : `~/cafe` et `TP1/x2.c`. `fichier_temporaire` peut très bien être dans notre répertoire courant et ne pas être

identifié différemment. Il faudra toujours s'assurer de ne pas l'écraser s'il existait avant.

Exercice 4 :

L'utilisateur Toto tape les commandes suivantes dans un shell :

```
{toto} 1 > ls
TP1     TP2     essai   copiedir.sh
{toto} 2 > cp essai
{toto} 3 > cpessai TP1/truc
{toto} 4 > cp TP1/cafe.data ../TP2/cafe.data
```

Pour chacune des 3 dernières commandes, dîtes si un message d'erreur est affiché, et si oui, lequel et sa signification.

Solution :

```
{enseignant} 1 > ls
TP1     TP2     essai   copiedir.sh
{enseignant} 2 > cp essai
cp: missing destination file operand after 'essai'
Try 'cp --help' for more information.
{enseignant} 3 > cpessai TP1/truc
cpessai: command not found
```

Pour la dernière commande, ne connaissant pas le contenu du répertoire `TP1`, deux situations peuvent arriver suivant si `TP1/cafe.data` existe ou non. S'il existe, il n'y a pas d'erreur, sinon, il se produit l'erreur suivante :

```
{enseignant} 4 > cp TP1/cafe.data ../TP2/cafe.data
cp: cannot stat 'TP1/cafe.data': No such file or directory
```

Si la commande fonctionne, on remarquera que `../TP2/cafe.data` peut exister ou non avant la commande. Il est écrasé par `cp` s'il existait avant.

Exercice 5 :

Écrire des commandes avec **ls** permettant de lister dans un répertoire toutes les entrées dont le nom :

1. commence par la lettre `t` ;
2. comporte au moins un `d` ;
3. commence par `a` ou `t` ;
4. comporte un `a` puis un `b`, dans cet ordre mais pas nécessairement contigus.

Solution :

1. `ls t*`
2. `ls *d*`
3. `ls [at]*`
4. `ls *a*b*`

Exercice 6 :

Voici deux séquences de commandes :

```
{toto} 5 > cd ~/INF123/TP2
{toto} 6 > cp a*.c ..
{toto} 7 > mv * ../TP1
```

et

```
{toto} 5 > cd ~
{toto} 6 > mv INF123/TP2/* INF123
{toto} 7 > cd INF123/TP1
{toto} 8 > mv ../a*.c .
```

Expliquez (avec un exemple) ce que font ces deux séquences. Peuvent-elles avoir exactement le même effet ?

Solution :

On suppose que le répertoire INF123 contient deux répertoires TP1 et TP2.

La première séquence copie tous les fichiers de TP2 dont le nom commence par a et finit par .c dans INF123, et déplace tous les fichiers de TP2 dans TP1. Les fichiers de la forme `a*.c` sont donc dupliqués.

La seconde séquence déplace tous les fichiers de TP2 dans INF123, puis ceux dont le nom commence par a et finit par .c dans TP1. Aucun fichier n'est dupliqué, et s'il y avait des fichiers de la forme `a*.c` dans INF123 ils sont déplacés.

Pour que les deux séquences aient le même effet, il faudrait qu'il n'y ait aucun fichier `a*.c` dans INF123, et aucun fichier (du tout) dans TP2.

Exercice 7 :

Ecrire des commandes permettant de

1. donner tous les droits pour tout le monde au fichier `Tout_est_permis.txt`
2. enlever pour les autres les droits en lecture et écriture aux fichiers en C (`.c`) du répertoire courant
3. donner pour le groupe et les autres les droits en lecture et exécution au répertoire `Consultable` et tout son contenu
4. donner exctement les droits `rw-r-xr--` au fichier `JEVE.sh`

Solution :

1. `chmod a+rwx Tout_est_permis.txt`
2. `chmod o-rw *.c`
3. `chmod go+rx Consultable Consultable/*`
4. `chmod 654 JEVE.sh` (voir le mode octal de `chmod`)