Stage de rentrée de C

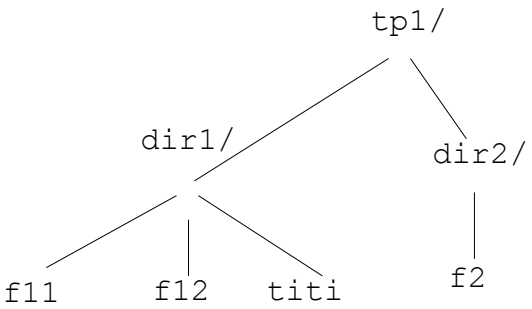
TP journée 1

Commandes Linux

Les exercices 1 à 3 sont à faire sans la machine.

Exercice 1

a) Donner la suite de commandes permettant de générer l'arborescence suivante:



b) Mettre les droits suivants:

- **`tp1`**: tous les droits pour tout le monde
- **`dir1`**: tous les droits pour tout le propriétaire et le groupe
- **`dir2`**: accès restreint au propriétaire
- **`f11`**: exécutable par tous
- **`f12`**: lisible par tous, modifiable par le propriétaire
- **`f2,titi`**: lecture et écriture pour le propriétaire

c) Donner la suite de commandes correspondant aux actions suivantes:

- aller dans **`dir1`**
- sans changer de répertoire, copier **`f11`** vers **`tp1`**
- renommer cette copie en **`tutu`**
- aller dans **`dir2`**
- déplacer ici tous les fichiers de **`dir1`** commençant par '**`f`**'

Exercice 2

Expliquer ce que fait la suite de commandes suivantes:

```
cd
mkdir ./tutu
mkdir tutu/titi
cd tutu
chmod u=r ../tutu/titi
```

mkdir ~/tutu/titi/toto

Exercice 3

a) Avec la commande **echo**, créez un fichier nommé **titi** contenant **Bonjour!**

b) En vous aidant de la commande **cat**, créez un fichier **toto** contenant deux fois le contenu de **titi**. Donnez deux façons de faire.

c) Que fait la commande **cat z >> z** ?

ATTENTION: NE SURTOUT PAS TESTER SUR MACHINE LA REPONSE A LA QUESTION SUIVANTE SI VOUS AVEZ LES DROITS ROOT DANS UNE SALLE RESEAU

d) Tentez de détruire tout le répertoire **/boot** en stockant les messages d'erreurs dans un fichier nommé **err**. Affichez les 10 dernières lignes de **err**.

e) Donnez des commandes permettant de connaître la taille en octets de la page **man** de **mv** (indice: **ls** donne cette taille).

* *

Les exercices suivants sont à faire sur machine.

Exercice 4

En utilisant la page **man** de la commande **cut** et en vous rappelant que la sortie de **ls -l** donne quelque chose comme:

```
-rw-r--r--  1 paumier igm    372 jui 11 02:48 test.c
```

donnez les commandes utilisant des *pipes* permettant d'afficher les choses suivantes:

- juste la liste des droits de ce qu'il y dans le répertoire (on ne veut pas la première ligne avec "total ...")
- les droits, propriétaires et noms des fichiers avec l'extension .c. Attention, comme **cut** est sensible aux espaces multiples que produit **ls -l**, veillez à les supprimer avec la commande **tr**.
- uniquement les droits en lecture des fichiers

Récupérez avec la commande **wget** le fichier suivant:
http://igm.univ-mlv.fr/~paumier/C/passwd.txt
Sachant que les lignes de ce fichier sont de la forme:

```
talon:x:635:201:Marie-Pierre Talon:/home/compta/talon:/bin/bash
```

donnez une commande permettant de sauver la liste des noms et prénoms dans un fichier **tutu**.

En utilisant **grep**, donnez une commande permettant d'afficher les logins des gens prénommés **Pierre**.

Exercice 5

a) Lancer en simultané plusieurs instances du programme **xeyes**.
b) En utilisant des *pipes*, n'afficher que les numéros de processus correspondant aux **xeyes**.
c) Ajouter un *pipe* avec la commande **kill** pour tuer ces processus. Que se passe-t-il ?

Lorsqu'on met une commande entre quotes inversées (**`toto`**) cette expression est équivalente au résultat affiché par **toto** et va à son tour être évaluée. Exemple:

```
$>echo find me
find me
$>`echo find me`
find: me: No such file or directory
```

d) En utilisant des quotes inversées, modifier la commande précédente pour obtenir une ligne de commande qui tue tous les processus **xeyes**.

Exercice 6

Un script est un fichier texte contenant des commandes destinées à être interprétées. La première doit contenir un dièse suivi du nom du programme qui va interpréter les commandes. Pour un script shell, cela sera de la forme:

```
#!/bin/sh
```

a) Créer un script **control.sh** qui affiche "bonjour" quand on le lance.
b) Exécuter **./control.sh**. Que se passe-t-il? Corriger.

En shell, on peut affecter une expression à une variable. Par exemple:

```
cmd='ps'
res=`$cmd`
echo -e "$cmd produit le resultat:\n$res"
```

donne comme résultat affiché à l'écran:

```
ps produit le resultat:
  PID TTY          TIME CMD
17607 pts7     00:00:00 bash
28329 pts7     00:00:00 control.sh
28330 pts7     00:00:00 ps
```

Lorsqu'une variable contient plusieurs éléments séparés par des espaces, des tabulations ou des retours à la ligne, on peut parcourir ces éléments au moyen d'une boucle **for** selon le schéma suivant:

```
cmd='ps'
res=`$cmd`
for i in $res
```

```
do
        echo "*** $i ***"
done
```

Ce qui donne à l'exécution:

```
*** PID ***
*** TTY ***
*** TIME ***
*** CMD ***
*** 17607 ***
*** pts7 ***
*** 00:00:00 ***
*** bash ***
*** 28536 ***
*** pts7 ***
*** 00:00:00 ***
*** control.sh ***
*** 28537 ***
*** pts7 ***
*** 00:00:00 ***
*** ps ***
```

c) En utilisant ce qui précède, écrire un script avec une boucle qui affiche tous les fichiers du répertoire courant comme **`ls -l`**, mais en séparant chaque ligne par des dièses:

```
$>ls -l
total 2
-rwx------  1 paumier igm    69 sep  5 14:14 control.sh
-rw-r--r--  1 paumier igm 10915 aoû 30 11:21 toto
$>./control.sh
###################################
-rwx------  1 paumier igm    69 sep  5 14:14 control.sh
###################################
-rw-r--r--  1 paumier igm 10915 aoû 30 11:21 toto
###################################
```

Un script shell peut faire appel aux variables spéciales suivantes:

$#	nombre de paramètres du script
$0	nom du script
$1	1er paramètre
$2	2eme paramètres
$3	etc
$*	liste de tous les paramètres

d) Ecrire un script **`param.sh`** qui prend deux arguments **`a`** et **`b`** qui met dans le fichier **`b`** le résultat de la commande **`a`**. Exemple:

```
$>./param.sh ps toto
$>cat toto
  PID TTY          TIME CMD
17607 pts7     00:00:00 bash
```

```
28329 pts7     00:00:00 param.sh
28330 pts7     00:00:00 ps
```

Exercice 7

a) En utilisant l'éditeur **gedit**, recopier le fichier **uppercase.c** suivant:

```
#include "work.h"

int main(int argc,char* argv[]) {
   work();
   return 0;
}
```

Puis le fichier **work.c**:

```
#include <stdio.h>
#include <ctype.h>

void work(void) {
   int c;
   while ((c=fgetc(stdin))!=EOF) {
      fputc(toupper(c),stdout);
   }
}
```

Et enfin le fichier **work.h**:

```
#ifndef work_H
#define work_H

void work(void);

#endif
```

b) Compiler le programme.
c) Rédiger un **Makefile** qui compile ce programme.
d) Deviner ce que fait ce programme (vous pouvez vous aider de **man**).
e) Ajouter à **work.h** un commentaire qui explique ce que fait la fonction **work**.
f) Appliquer le programme pour transformer la page man de **gcc**.

Exercice 8

Modifiez le programme précédent pour qu'il se comporte comme la commande **tr -s**.

Exemple:

```
$>echo 'coool    !!!' | ./a.out 'o !'
col !
```