Système d'Exploitation
Travaux Pratiques (2), Licence 2 Informatique
Commandes de Base de Gestion des Fichiers

Les exercices suivants ont pour but de vous familiariser avec les commandes de base de la gestion des fichiers.
Il vous est recommandé de consulter les pages `man` de ces primitives pour de plus amples informations sur leur syntaxe, leur sémantique et les éventuelles options qu'elles offrent.

Les instructions des exercices se repèrent par des icônes, qui sont les suivantes :

	Information	Information concernant l'usage ou le rôle d'une commande, par exemple. Dans certains cas, il s'agit d'une information sur ce que vous êtes en train de faire ou sur ce qui se passe.
	Exemple	Exemple d'utilisation.
	Contrôle	Vérifier le résultat d'une (ou plusieurs) action(s).
	Action	Effectuer la ou les action(s) décrite(s).
	Question	Questions auxquelles vous devez répondre.

De plus, un `texte en police courier` correspond soit à une sortie écran soit à des noms spécifiques (menus, fenêtre, icône, processus, commandes…).
Un **texte en police times gras** correspond à ce que l'utilisateur doit introduire comme valeur de paramètre, ou encore, est utilisé pour attirer l'attention de l'utilisateur.

La commande `mkdir`

La commande `mkdir` permet de créer les répertoires passés en paramètres (références absolues ou relatives de répertoires).

Consultez le manuel de la commande `mkdir` et prenez connaissance des différentes options qu'elle accepte.

```
$ mkdir TESTS
$ ls –l TESTS
total 1
drwxrwxr-x   2 pierre      users          1024 oct  6 04:00 TESTS
```

Créez un répertoire `TP0`.

La commande `ls`

La commande `ls` permet de visualiser le contenu d'un ou plusieurs répertoires et les caractéristiques de fichiers ou répertoires donnés en arguments au moyen de diverses options.

```
$ ls
ADMIN  COURSUNIX  a.out  fichier  temporaire
$ ls –l
total 9
drwxrwxr-x   2 pierre users   1024 oct  6 03:53 ADMIN
drwxrwxr-x   3 pierre users   1024 oct  6 04:00 COURSUNIX
-rwxr-xr-x   1 pierre users   4119 oct  6 03:55 a.out
-rw-r--r--   1 pierre users   33   oct  4 19:09 fichier
drwxrwxr-x   2 pierre users   1024 oct  6 03:55 temporaire
```

Consultez le manuel de la commande `ls` et prenez connaissance des différentes options qu'elle accepte (notamment celle qui permet l'affichage des numéros d'i-nodes).

La commande `cd`

La commande `cd` est une commande de navigation qui permet de se déplacer d'un répertoire à un autre du système de fichiers. Le répertoire courant est représenté par '.', le répertoire parent par '..' et le répertoire personnel (*home directory*) par '~'. La commande `pwd` vous indique le répertoire courant (sur lequel vous êtes positionné).

```
$ cd COURSUNIX
$ pwd
/home/pierre/COURSUNIX
$ cd
$ pwd
/home/pierre
$
```

Consultez le manuel de la commande `cd` et prenez connaissance des différentes options qu'elle accepte.

Déplacez-vous dans le répertoire `TP0`.

La commande `touch`

La commande `touch` permet de modifier le *timestamp* (heure) de dernier accès et de dernière modification d'un fichier. Cette commande permet également de créer un fichier vide (s'il n'existe pas déjà).


Consultez le manuel de la commande `touch` et prenez connaissance des différentes options qu'elle accepte.


Créez un fichier `file0`.


Vérifiez que le fichier a bien été créé et notez l'heure de sa création.


Exécutez à nouveau `touch` avec le fichier `file0` en argument.


Le fichier a-t-il toujours la même date que précédemment.

La commande `cp`


La commande `cp` permet de copier des fichiers et des répertoires.


Consultez le manuel de la commande `cp` et prenez connaissance des différentes options qu'elle accepte.


Copiez le fichier `file0` en un second fichier `file1`.


Vérifiez que le fichier a bien été créé.


Positionnez-vous sur votre répertoire personnel et copiez le répertoire `TP0` en un second répertoire `TP00`.


Vérifiez que le répertoire et les fichiers contenus ont bien été créés.

La commande `cat`


La commande `cat` permet de concaténer des fichiers. Cette commande permet également d'afficher le contenu d'un fichier.


Consultez le manuel de la commande `cat` et prenez connaissance des différentes options qu'elle accepte.


Positionnez-vous sur le répertoire `TP0`. Ouvrez les fichiers `file0` et `file1` avec un éditeur, introduisez-y du texte dans chacun et sauvegardez-les.


Exécutez la commande `cat` avec `file0` en argument. Exécutez la commande `cat` avec `file1` en argument. Exécutez la commande `cat` avec `file0` et `file1` en arguments.

La commande `mv`


La commande `mv` permet de déplacer des fichiers et des répertoires. Cette commande permet également de renommer des fichiers et des répertoires.


Consultez le manuel de la commande `mv` et prenez connaissance des différentes options qu'elle accepte.


Déplacez les deux fichiers `file0` et `file1` du répertoire `TP0` vers le répertoire `TP00`.


L'exécution s'est-elle bien déroulée ? Quelle en est la cause ?


Copiez les fichiers `file0` et `file1` du répertoire `TP0` vers le répertoire `TP00` en leur donnant comme nom respectif `file00` et `file11`.

Les commandes `rm` et `rmdir`


La commande `rm` permet de supprimer des fichiers et des répertoires. La commande `rmdir` permet de supprimer des répertoires.


```
$ rmdir COURSUNIX
rmdir: COURSUNIX: Directory not empty
```


Consultez le manuel des commandes `rm` et `rmdir` et prenez connaissance des différentes options qu'elle accepte.


Supprimez les fichiers `file0` et `file1` du répertoire `TP0`. Puis, supprimez le répertoire `TP0`.

Les commandes `ln`


La commande `ln` permet de créer des liens physiques vers des fichiers. Avec l'option `-s` c'est un lien symbolique qui est créé.


Consultez le manuel des commandes `ln` et prenez connaissance des différentes options qu'elle accepte.


À partir de votre répertoire personnel, créer un lien (physique) `lienP_file00` vers le fichier `file00` du répertoire `TP00`.


Affichez le contenu du fichier `lienP_file00`.


Quel est le contenu de `lienP_file00` ?


Renommez le fichier `file00` en `00file`.


Quel est à nouveau le contenu de `lienP_file00` ? Qu'en déduisez-vous ?


Supprimez le fichier `00file`.


Quel est à nouveau le contenu de `lienP_file00` ? Qu'en déduisez-vous ?


À partir de votre répertoire personnel, créer un lien (physique) `lienP_TP00` vers le répertoire `TP00`.


L'exécution s'est-elle bien déroulée ? Quelle en est la cause ?


À partir de votre répertoire personnel, créer un lien symbolique `lienS_file11` vers le fichier `file11` du répertoire `TP00`.


Affichez le contenu du fichier `lienS_file11`.


Quel est le contenu de `lienP_file11` ?

Renommez le fichier `file11` en `11file`.

Quel est à nouveau le contenu de `lienS_file11` ? Qu'en déduisez-vous ?

Renommez à nouveau le fichier `11file` en `file11`.

Quel est à nouveau le contenu de `lienS_file11` ? Qu'en déduisez-vous ?

Supprimez le fichier `file11`.

Quel est à nouveau le contenu de `lienS_file11` ? Qu'en déduisez-vous ?

À partir de votre répertoire personnel, créer un lien symbolique `lienS_TP00` vers le répertoire `TP00`.

Affichez le contenu du fichier `lienS_TP00`.

Modification des caractéristiques d'un répertoire ou fichier : `chmod`**,** `chown`**,** `chgrp`

La commande `chmod` permet de modifier les droits d'accès d'un répertoire ou fichier.

```
$ cd
$ ls –l COURSUNIX
total 1
drwxrwxr-x   3 pierre users   1024 oct  6 04:00 COURSUNIX
$ chmod 777 COURSUNIX
$ ls –l COURSUNIX
total 1
drwxrwxrwx   3 pierre users   1024 oct  6 04:02 COURSUNIX
$ chmod u-x,go-rwx COURSUNIX
$ ls –l COURSUNIX
total 1
drw-------   3 pierre users   1024 oct  6 04:03 COURSUNIX
```

Consulter les pages `man` de la commande `chmod` pour prendre connaissance des différentes options et tester certaines.

Changer les droits du propriétaire (aucun accès autorisé, puis droit de lecture seulement, puis droit d'écriture seulement, …) d'un répertoire vous appartenant et tester (à nouveau) la commande `cd` sur ce répertoire et les commandes `mkdir` et `rmdir` à l'intérieur de ce même répertoire.

En fonction des droits possédés sur ce répertoire, en déduite les opérations permises.

Essayer de changer les droits d'un fichier vous appartenant puis ceux d'un fichier ne vous appartenant pas.

Quel résultat obtenez-vous ?

La commande `chown` permet de modifier le propriétaire d'un répertoire ou d'un fichier.

```
$ ls –l fichier
total 1
-rw-r--r--   1 pierre users   33   oct  4 19:09 fichier
$ chown paul fichier
$ ls –l fichier
total 1
-rw-r--r--   1 paul users     33   oct  4 19:09 fichier
$
```

Changer les droits d'accès d'un de vos répertoires de façon que seul le propriétaire ait des droits de lecture/écriture dessus et uniquement la lecture pour les autres utilisateurs.

Pouvez-vous voir le contenu de ce répertoire ?

Ajouter le droit d'exécution pour le propriétaire puis changer le propriétaire de ce même répertoire et essayer de changer les droits d'accès d'un fichier qu'il contient.

Quel est le résultat ?

Essayer ensuite de supprimer ce même fichier.

Quelle interprétation en faites-vous ?

La commande `chgrp` permet de changer le groupe propriétaire d'un répertoire ou d'un fichier.

```
$ ls –l fichier
total 1
-rw-r--r--   1 pierre users   33   oct  4 19:09 fichier
$ chgrp users2 fichier
$ ls –l fichier
total 1
-rw-r--r--   1 pierre users2  33   oct  4 19:12 fichier
$
```

Refaire les mêmes exercices que pour la commande `chown`.

Quelle interprétation en faites-vous ?

Les fichiers `/etc/passwd` et `/etc/group` donnent des indications sur les noms de propriétaires (utilisateurs) et de groupes existants.

Droits d'endossement : `set-uid bit`

Soit le programme suivant qui ouvre en lecture/écriture le fichier dont le nom est passé en paramètre de ligne de commande, le lit et affiche son contenu sur la sortie standard.

Vous pouvez faire un copier/coller à partir du sujet présent sur Celene.

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

int main(int argc, char **argv) {

printf("Mon UID réel est : %d\n", getuid() );// Affiche l'UID réel du processus
printf("Mon UID effectif est : %d\n", geteuid() );  // Affiche l'UID effectif du processus
    int fd = open( argv[1], O_RDWR );  // Ouvre en L/E le fichier passé en paramètre
    if ( fd == -1) {                   // open retourne -1 si erreur d'ouverture
        perror("Problème ouverture");
        return -1;
    }
    else {
        printf("Ouverture réussie, contenu du fichier :\n");
        char c[1];          // Déclaration d'un tableau constitué d'un seul élément (type caractère)
        while ( read(fd, c, 1) == 1 )  // Lecture d'un caractère depuis le fichier, stocké dans c
        write(0, c, 1);                // Écriture sur sortie standard du caractère stocké dans c

        return -1;
    }
}
```

Recopiez-le dans un fichier que vous appellerez prog.c et créez un fichier qui sera nommé Donnees qui contiendra un texte quelconque. Seul le propriétaire du fichier Donnees devra avoir des droits de lecture et d'écriture dessus.

Compilez le programme et appelez l'exécutable prog (gcc prog.c –o prog).

Exécutez le programme en lui passant comme paramètre le fichier Donnees (. /prog Donnees).

Quel est le résultat de son exécution ?

Connectez-vous sous un autre nom d'utilisateur et exécutez à nouveau le programme.

Quel est le résultat de son exécution ?

Positionnez le set-uid bit de l'exécutable (fichier prog) et exécutez-le à nouveau sous les deux comptes.

Quel est le résultat de chacune des exécutions ?

Comparaison de fichiers : `cmp, diff`

La commande `cmp` permet de comparer deux fichiers de tous types. Si les deux fichiers sont différents, les numéros d'octet et de ligne de la première différence détectée sont affichés.

Créer une copie, grâce à la commande `cp` vue précédemment, d'un de vos fichiers existants. Appliquer la commande `cmp` à ces deux fichiers.

Quel est le résultat de la commande ?

Ouvrir le fichier correspondant à la copie précédente et procéder à la modification (ou suppression) d'un quelconque caractère.

Appliquer de nouveau la commande `cmp` aux deux fichiers.

Quel est cette fois-ci le résultat de l'exécution ?

La commande `diff` permet de trouver les différences entre deux fichiers. Contrairement à `cmp`, elle ne s'arrête pas à la première différence. De plus, l'un ou les deux arguments de la commande peuvent être des répertoires. Dans ce dernier cas, `diff` compare les fichiers correspondants dans les deux répertoires, dans l'ordre alphabétique.

Tester la commande `diff`. Tester également certaines options.

Découper un fichier en plusieurs parties : `split`

A partir d'un fichier, la commande `split` permet de produire plusieurs fichiers contenant les parties consécutives du fichier d'entrée. La taille des fichiers de sortie peut être exprimée en nombre de lignes ou en nombre d'octets. Un préfixe peut être associé aux fichiers de sortie.

A l'aide de la commande `split`, découpez un fichier texte source en `N` fichiers dont le nom de chacun aura comme préfixe "`partie`". La taille des fichiers sera déterminée en fonction du nombre `N` et de la taille du fichier source.

Vérifier que les fichiers sont bien créés et que chacun d'eux contient une **partie distincte** du contenu du fichier source.

Procéder à la même opération que précédemment mais sur un fichier source exécutable.

Concaténer (avec la commande `cat`) les `N` fichiers obtenus précédemment en un seul nouveau fichier (nom différent du fichier source découpé).

Exécuter le nouveau fichier pour vérifier l'absence d'altération durant le processus découpage/concaténation.

Autres commandes :

`head, tail, file, find, grep, more, less, …`