TD 1 : Gestion de fichiers

Exercice 1:

1. Nommez quelques primitives système. Est-ce que **fopen** est une primitive système ?
2. Quelles sont les différences et ressemblances entre primitives système et fonctions de bibliothèque ? Comment justifier ces différences ?
3. Illustrez ces différences et ressemblances sur les fonctions et primitives d'entrée / sortie.

Exercice 2:

Pourquoi la fonction ci-dessous ne peut pas fonctionner ? Détaillez toutes les fautes. On ne demande pas de corriger cette fonction.

```
int faux (char *nom)
{
    FILE *fp ;
    int c ;

    fp = open (nom, "r") ;
    read (fp, &c, 1) ;
    fclose (fp) ;
    return c ;
}
```

Exercice 3:

Écrivez un programme qui recopie un fichier **toto** vers un fichier **titi** à créer, à l'aide des primitives système.
Vous ne chercherez pas à créer le nouveau fichier avec les permissions du fichier original.

Exercice 4:

Écrivez la fonction **getchar** qui renvoie un caractère lu sur l'entrèe standard, ou la constante **EOF** en fin de fichier.

Exercice 5:

Écrivez une version bufferisée de **getchar**.

Exercice 6:

Écrivez un programme qui affiche en clair le type du fichier demandé (répertoire, fichier ordinaire, etc.), ainsi que ses permissions (lecture, écriture et exécution, sous la même forme que la commande **ls** avec l'option **-l**).

Exercice 7:

On désire implémenter une nouvelle version de la librairie standard d'entrées/sorties à l'aide des primitives système.

1. Donnez une définition du type **FICHIER**. N'oubliez pas de prévoir la bufferisation des entrées/sorties.
2. Programmez la fonction *my_open*, analogue à **fopen**.
3. Reprenez l'exercice 3 pour programmer *my_getc*, analogue à **getc**.
4. Programmez la fonction *my_putc*, analogue à **putc**.
5. Programmez la fonction *my_close*, analogue à **fclose**.

Exercice 7:

Écrivez une commande qui prend en paramètre un nom de répertoire, et qui affiche tous les objets contenus dans ce répertoire. On prendra les mêmes conventions de restriction d'affichage que la commande **ls** (pas d'affichage des noms commençant par un point).

Exercice 8:

Reprenez le programme de l'exercice 3 pour recopier toute une arborescence.

Exercice 9:

Reprenez le programme de l'exercice précédent pour restaurer dans les copies les dates d'accès et modification ainsi que les permissions des fichiers originaux.

Exercice 10:

La variable Shell `PATH` contient une liste de répertoires séparés par des caractères "`:`".
Par exemple : **/bin:/usr/bin:/usr/local/bin**.

1. Écrivez une fonction qui prenne en paramètre une chaîne et sépare dans un tableau de chaînes les différents constituants.
2. Écrivez un programme qui lit le contenu de la variable **PATH** (avec la fonction **getenv**), sépare les différents constituants à l'aide de la fonction que vous venez de programmer, et qui affiche ces différents constituants.
3. Écrivez un programme **which** afin de chercher où une commande est trouvée. Par exemple, "**which ls**" doit donner : **/bin/ls**.

Exercice 11:

Le shell **ksh** dispose d'une variable **CDPATH**. Celle-ci spécifie un certain nombre de répertoires de recherche. Lorsqu'on utilise **cd** avec un argument (nom de chemin relatif), celui est cherché dans les différents répertoires indiqués par **CDPATH** et le changement de répertoire est effectué s'il est trouvé.

1. Programmez une commande **chdir**.
2. Pourquoi cette commande ne peut pas fonctionner ?