TP 3 – Fichiers et répertoires

On souhaite réaliser en C un programme qui liste les fichiers contenus dans un répertoire. Le TP est progressif, il faut réaliser les exercices dans l'ordre. Vous ferez un fichier par exercice.

1 Propriétés d'un fichier : *stat*

- En utilisant la fonction *stat* et la structure *stat*, réalisez un programme qui teste si le fichier existe et qui affiche si le fichier est un fichier régulier (normal) ou un répertoire. Il affichera le numéro i-node du fichier (ou du répertoire). Le nom du fichier à tester sera donné en argument de votre programme.

  Exemple :

```
etu@fst-ped-mia-vm-l02$ ./LectureFichier MyLS.c
<MyLS.c> est un fichier (i-node 20)
```

Aide C :

- À l'aide du *manuel*(`man 2 stat`), recherchez les champs de la structure *stat*, qui vous seront utiles.
- Exemple d'utilisation de la fonction *stat* :

```
struct stat s;
char ref[1024];

...(*)..
if(stat(ref, &s)==-1) {
       perror("Le fichier/répertoire n'existe pas");
       exit(1);
}
/* utilisation de la structure s */
...
```

- (*)Pour copier un argument (argv[1]) dans une chaine de caractère :
  ~~`char ref[1024] = argv[1] ;`~~
  `char ref[1024];`
  `strcpy(ref, argv[1]);`
- Pour tester si le fichier est un fichier régulier ou un répertoire, utilisez la fonction `S_ISREG(s.st_mode)` et `S_ISDIR(s.st_mode)` (vraie si ok).

- Modifiez le programme pour qu'il affiche aussi (si c'est un fichier régulier) sa taille et la date du dernier accès et de la dernière modification.

  Exemple :

```
etu@fst-ped-mia-vm-l02$ ./LectureFichier MyLS.c
<MyLS.c> est un fichier (i-node 20)
--> taille : 2 Ko
--> modif le Wed Feb 11 17:15:40 2015
--> acces le Wed Feb 11 16:41:24 2015
```

  Aide C :

  - ☞ Pour convertir des secondes (depuis 1970) en date, vous utiliserez `ctime(&var)`. *var* est une variable de type time_t, comme le champs de *stat* que vous utilisez. N'oubliez d'inclure <time.h>
  - ☞ Inconvénient de `ctime` : il renvoie un chaine de caractère avec un retour à la ligne (\n). Il serait alors possible d'utiliser `localtime(&var)` qui renvoie un pointeur sur une structure *tm*, avec de nombreux champs (jour, mois, année...), voir `man 3 localtime`. Mais ne l'utilisez pas ici (pour ne pas perdre de temps).

- Si l'argument donné en entrée est un fichier, affichez les 100 premiers caractères du fichier (pour donner un extrait du fichier). Il faut ouvrir le fichier avec `open`, et lire **caractère par caractère** avec `read` (jusqu'au 100 caractères ou jusqu'à la fin s'il y a moins de 100 caratères). **Pour cet excercice nous ne voulons pas lire directement 100 caractères avec** `read` (nous aurions pu évidemment). N'oubliez pas de fermer l'utilisation du fichier avec `close`.

  Exemple :

```
etu@fst-ped-mia-vm-l02$ ./LectureFichier MyLS.c
<MyLS.c> est un fichier (i-node 20)
--> taille : 2 Ko
--> modif le Wed Feb 11 17:15:40 2015
--> acces le Wed Feb 11 16:41:24 2015
=====Extrait=======
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <un
===================
```

  (avec un fichier binaire on obtient aussi les premiers caractères)

Aide C :

- La fonction `open` renvoie le descripteur associé au fichier ouvert. (-1 si l'ouverture n'a pas fonctionné). Pour une lecture d'un fichier, le deuxième argument de la fonction est O_RDONLY. Il n'y a pas de troisième argument ici (qui sont les droits en écriture lors d'une création de fichier).
- La fonction `read` renvoie 0 lorsqu'elle arrive à la fin du fichier (c-a-d qu'elle n'a pu lire que 0 caractère).

2 Parcours d'un répertoire : `DIR` et `dirent`

Les fonctions POSIX que nous allons utiliser pour lire et parcourir les entrées d'un répertoire sont `opendir`, `readdir` et `closedir`. Elles manipulent l'objet de type `DIR` et la structure `dirent`.

- Réalisez un programme qui ouvre un répertoire (avec la fonction *opendir*) et qui parcourt le répertoire entrée par entrée (avec la fonction *readdir*) et affiche le nom et le numéro i-node. À chaque appel, la fonction *readdir* remplit une structure *dirent* contenant le nom du fichier (ou répertoire) dans le champ *d_name* (et son i-node *d_ino*). Pour parcourir toutes les entrées du répertoire, il faut appeler plusieurs fois *readdir* jusqu'à ce qu'elle renvoie NULL. À la fin du parcours, fermez la lecture/le parcours avec la fonction *closedir*.

Aide C :

- Voici les différents prototypes des fonctions utilisées (regardez le *man* pour plus de détails)
  - #include <dirent.h>
  - `DIR * opendir(const char* reference)` renvoie NULL si pas trouvée.
  - `struct dirent * readdir(DIR* pDir)` renvoie NULL si il n'y a plus d'entrée à traiter.
  - `int closedir(DIR* pDir)`
- Les appels successifs de *readdir* avancent entrée par entrée du répertoire.
- Rappel sur la notation pour accéder aux champs d'une structure :
  - *nomvariable.nomduchamp* si *nomvariable* est une **structure**.
  - *nomvariable->nomduchamp* si *nomvariable* est un **pointeur sur une structure**.

- Maintenant, en plus d'afficher le nom et l'i-node, affichez pour chaque entrée si c'est un fichier ou un répertoire. Il suffit de réutiliser la fonction et la structure *stat* comme dans l'exercice précédent. Attention, il faut cependant faire précéder le nom du fichier par le nom du répertoire (pour que la fonction *stat* trouve le fichier).

Aide C :

- Pour formater une chaine de caractères, vous utiliserez la fonction *sprintf*. Ex : `sprintf(chaine,"%s/%s",nomrep, nomfichier)` ;

- Modifiez le programme pour qu'il affiche la taille du répertoire (en sommant la taille des fichiers contenus dans le répertoire, sans considérer les autres répertoires).

3 Parcours des sous-répertoires

- Réalisez maintenant le programme final qui parcourt un répertoire et affiche la taille de chaque répertoire contenu dans ce dernier. La taille d'un répertoire est la taille cumulée de ses différents sous-répertoires et de ses fichiers.
  - Reprenez ce que vous avez fait à la question précédente sous forme d'une fonction (si cela n'avait pas été fait) qui renvoie la taille du répertoire.*(*)*
  - Si l'entrée du répertoire est un répertoire, la fonction pourra s'appeler elle-même pour gérer ce sous-répertoire.
  - la taille d'un répertoire est donc la somme des fichiers et de ses sous-répertoires.
  - Attention, si vous rencontrez les répertoires . et .., ne lancez pas la lecture de ces répertoires, sinon votre programme va boucler!!

(*) Conseil :

☞ Pour rendre lisible l'affichage des répertoires, on peut ajouter quelques espaces/tabulations en début de ligne en fonction du niveau du sous-répertoire pour créer une sorte d'arborescence.
Pour savoir à quel niveau on se situe il est alors conseillé d'ajouter le niveau comme paramètre de la fonction.
Exemple d'affichage :

```
============
R Repertoire1
        F fichier1 (123 o)
        F fichier2 (456 o)
        R Repertoire2
                F fichier1 (45 o)
                F fichier2 (156 o)
                (Taille : 201 o)
        F fichier3 (130 o)
        (Taille : 810 o)
============
```

Aide en C :

☞ Pour comparer deux chaines de caractères, utilisez la fonction *strcmp*. Elle renvoie 0 en cas d'égalité. Ex : si chaine1 égale chaine 2 faire...⇒ `if (!strcmp(chaine1, chaine2)) { faire...}`