TD 2

Primitives de gestion de fichiers – partie 2
Programmation Sys. en C

Objectifs du TD :

L'objectif de ce TD est de permettre aux étudiants d'apprendre les ***primitives*** *ou les* ***fonctions*** de ***gestion*** *des* ***répertoires*** via l'écriture et l'exécution des programmes en C.

Travail à effectuer :

Ce TD consiste à :

1) Apprendre avec l'aide du chargé de TD les primitives pour créer/supprimer un répertoire (*mkdir*/*rmdir*), ouvrir/fermer un répertoire (*opendir/closedir*) et parcourir ou lire un répertoire (*readdir*).
2) Il s'agit maintenant d'écrire un programme en C permettant de parcourir un répertoire en récupérant des informations sur les fichiers présents à la manière d'un ***ls***. Pour cela, nous allons utiliser les primitives contenues dans la bibliothèque standard (fichier d'en-tête dirent.h), présentées dans 1).

1) Les **primitives de manipulation d'un répertoire** sont décrites en détail dans ce qui suit :

   a. Créer un nouveau répertoire :
      i. **Syntaxe :**

```
#include <sys/stat.h>
#include <sys/types.h>
```

int **mkdir**(const char *dirname, mode_t mode) ; dans *mode* l'utilisateur peut spécifier les permissions d'accès. L'argument *mode* peut prendre une des valeurs suivantes :

S_ISUID	04000	bit set-UID
S_ISGID	02000	bit set-GID
S_ISVTX	01000	bit « sticky »
S_IRWXU	00700	lecture/écriture/exécution du propriétaire
S_IRUSR	00400	le propriétaire a le droit de lecture
S_IWUSR	00200	le propriétaire a le droit d'écriture
S_IXUSR	00100	le propriétaire a le droit d'exécution
S_IRWXG	00070	lecture/écriture/exécution du groupe
S_IRGRP	00040	le groupe a le droit de lecture
S_IWGRP	00020	le groupe a le droit d'écriture
S_IXGRP	00010	le groupe a le droit d'exécution
S_IRWXO	00007	lecture/écriture/exécution des autres
S_IROTH	00004	les autres ont le droit de lecture
S_IWOTH	00002	les autres ont le droit d'écriture
S_IXOTH	00001	les autres ont le droit d'exécution

ii. **Exemples** :
mkdir(''rep1'', 0664)
iii. **Valeur de retour** : 0 si succès et -1 si erreur

b. Supprimer un répertoire
i. **Syntaxe** :
#include <sys/stat.h>
int **rmdir**(const char *dirname); pour que l'utilisateur puisse supprimer le répertoire, ce dernier doit être vide !!
ii. **Valeur de retour** : 0 si succès et -1 si erreur

c. Ouvrir un répertoire :
i. **Syntaxe** :
#include <sys/types.h>
#include <dirent.h>
DIR * **opendir**(const char *dirname) ➔ ouvrir un *directory stream*.
ii. **Exemple** :
- DIR * dir1 ;
- dir1 = opendir(''rep1'') ;

iii. **Valeur de retour** : un pointeur vers un *directory stream* si succès et -1 si erreur

d. Lire dans un répertoire :
   i. **Syntaxe :**
   #include <sys/types.h>
   #include <dirent.h>
   struct dirent ***readdir**(DIR *dir) ; *readdir* lit dans le répertoire l'élément à la position courante et se positionne sur le prochain élément pour la prochaine lecture.
   ii. **Exemple** :

```
DIR *dir1;
struct dirent *entry;
dir1 = opendir("rep1");
entry = readdir(dir1) ;
```

   iii. **Valeur de retour** : un pointeur sur une structure de type ***dirent***, contenant des données, si succès et -1 si erreur. La structure ***dirent*** contient les données ci-dessous :

```
struct dirent {
    ino_t d_ino;                    /* inode number */
    off_t d_off;                    /* offset to the next dirent */
    unsigned short int d_reclen;    /* length of this record */
    unsigned char d_type;           /* type of file */
    char d_name[256];               /* We must not include limits.h! */
};
```

Le type du fichier peut être : inconnu, Fifo, dispositif à caractères (ex. /dev/sda), répertoire, dispositif à blocks, fichier normal, lien symbolique, socket, …

DT_UNKNOWN 0
DT_FIFO 1
DT_CHR 2
DT_DIR 4
DT_BLK 6
DT_REG 8
DT_LNK 10
DT_SOCK 12
DT_WHT 14

e. Fermer un répertoire :
   i. Syntaxe :
   #include <sys/types.h>
   #include <dirent.h>
   int **closedir**(DIR * dir)
   ii. Exemple : int d = closedir(dir1);
   iii. Valeur de retour : 0 si succès et -1 si erreur

Pour obtenir plus de détails sur les primitives présentées ci-dessus utilisez ***man nom_primitive*** ➔ man opendir, man closedir, man readdir …

2) Il s'agit maintenant d'écrire un programme en C permettant de parcourir un répertoire en récupérant des informations sur les fichiers présents à la manière d'un *ls*. Pour cela, nous allons utiliser les primitives contenues dans la bibliothèque standard (fichier d'en-tête dirent.h), présentées dans 1).

a. Ecrire une fonction qui liste les noms des fichiers et des répertoires dans un répertoire donné. Pour rappel, les fonctions *stat* et *fstat* vues dans la partie 1 du cours fournissent le statut du fichier (ses caractéristiques). Vous trouvez la syntaxe de ces 2 fonctions et la structure ***stat*** dans le support du cours (pages 20-23).
b. Améliorez la fonction précédente en indiquant le type du fichier. Nous ne testerons que les fichiers ordinaires, les répertoires et les liens symboliques.
c. Modifiez le programme précédent pour afficher la taille de chaque entrée du répertoire étudié.
d. Affichez aussi les droits associés à chaque fichier.