TP N°3
Système de fichiers

Préambule

Notions : Manipulation des structures internes du SGF (stat, direct, DIR), récursivité.

Rappel

```
#include <dirent.h>
DIR *opendir(const char *name);
int closedir(DIR *dirp);
```

Avec `opendir`, ouvre le répertoire `name` et retourne un descripteur de répertoire. La fermeture du descripteur de répertoire est assuré avec `closedir`.

```
struct dirent *readdir(DIR *dirp);
```

la fonction `readdir()` retourne l'entrée suivante du répertoire indiqué par `dirp`. Quand toutes les entrées ont été renvoyées, `readdir()` renvoie `NULL`.

```
int stat(const char *path, struct stat *buf);
```

Cette fonction récupère les informations associées au fichier `path` et les stocke dans la structure `buf`. Elle retourne `0` en cas de succès, `-1` et `errno` en cas de problème.

Affichage récursif du contenu d'un répertoire

Exercice 1. Le but de cet exercice est de manipuler en langage C les structures internes du système de fichiers d'UNIX. Pour cela vous écrirez un programme `Rdir.c` qui effectuera un ls récursif en affichant pour chaque élément :

- Nř d'i-nœud
- Nom
- Taille
- UID
- GID
- Type (répertoire ou fichier)

Vous vous inspirerez des fichiers "dir.c" et "status.c" donnés ci-dessous.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
int main(int argc, char *argv[]) {
  /* la structure DIR permet de récupérer des informations sur un répertoire
     */
```

```
  DIR* dir;
  /* la structure "dirent" contient des informations sur un élément du
  répertoire (le détail d'une entrée de répertoire, c'est-à-dire le
  "i-noeud" du fichier et son nom). La structure "dirent" se trouve dans
  le fichier "/usr/include/bits/dirent.h".

  struct dirent {
    #ifndef __USE_FILE_OFFSET64
    __ino_t d_ino;
    __off_t d_off;
    #else
    __ino64_t d_ino;
    __off64_t d_off;
    #endif
    ...
    char d_name[256];
}; */
  struct dirent* dirdata;
  char* path;
  /* controle des parametres d'appel */
  if(argc != 2) {
   printf("erreur de syntaxe d'appel.\n");
   exit(EXIT_FAILURE);
  }
  /* recuperation du nom du repertoire */
  path = (char*)malloc(strlen(argv[1]) +1);
  strcpy(path,argv[1]);
  /* ouverture du repertoire et controle d'existence */
  dir = opendir(path);
  if(dir == NULL) {
   perror("erreur opendir");
   exit(EXIT_FAILURE);
  }
  /* lecture des donnees du repertoire */
  while((dirdata = readdir(dir)) != NULL) {
   printf("%s : N ino = %lu\tNom = %s\n",path,dirdata->d_ino,dirdata->d_name);
  }
  /* liberation memoire */
  free(path);
  /* fermeture du repertoire ouvert */
  closedir(dir);
  exit (EXIT_SUCCESS);}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>

int main(int argc, char *argv[]) {
  /* la structure "stat" permet de récupérer les informations contenues dans
  les entêtes (i-noeud) de fichier. Elle se trouve dans le fichier
  "/usr/include/bits/stat.h"
```

```
10  struct stat {
      unsigned short st_dev;   // identifiant du périphérique contenant le
          fichier
12    unsigned short __pad1;
      unsigned long st_ino;    // i-noeud associé au fichier.
14    unsigned short st_mode;  // mode du fichier (type et droit d'accès).
      unsigned short st_nlink; // nombre de liens absolus vers le fichier.
16    unsigned short st_uid;   // identifiant de l'utilisateur proprietaire
      unsigned short st_gid;   // identifiant du groupe
18    ...
      unsigned long st_size;   // taille du fichier en octets
20    unsigned long st_blksize;// taille des blocs sur le peripherique
      unsigned long st_blocks; // nombre de blocs alloues
22    __time_t st_atime;       // date du dernier accès.
      __time_t st_mtime;       // date de la dernière modification
24    __time_t st_ctime;       // date du dernier changement de status
    }; */
26
    struct stat st;
28  char* path;
    int rep;
30  /* controle des parametres d'appel */
    if(argc != 2) {
32   printf("erreur de syntaxe d'appel.\n");
     exit(EXIT_FAILURE);
34  }
    /* recuperation du nom du repertoire */
36  path = (char*)malloc(strlen(argv[1]) +1);
    strcpy(path,argv[1]);
38  /* recuperation des infos du i_noeud par stat */
    rep = stat(path,&st);
40  if(rep == 0) {
     printf("%s : ino = %d\ttaille = %d\t",path,st.st_ino,st.st_size);
42   if(S_ISDIR(st.st_mode)) printf(" DIRECTORY ");
     if(S_ISREG(st.st_mode)) printf(" FICHIER ");
44   if(S_ISLNK(st.st_mode)) printf(" LIEN ");
     printf("owner = %d\tgrpe = %d\n",st.st_uid,st.st_gid);
46  }
    else {
48   perror("erreur de stat");
     exit(EXIT_FAILURE);
50  }
    /* liberation memoire */
52  free(path);
    exit (EXIT_SUCCESS);
54 }
```