TD de Systeme n° 6

On rappelle les structures de données utiles définies dans la librairie standard :

```
struct dirent {
  ino_t         d_ino;      /* inode number */
  char          d_name[256]; /* filename */
};

struct stat {
  dev_t    st_dev;    /* ID of device containing file */
  ino_t    st_ino;    /* inode number */
  mode_t   st_mode;   /* protection */
  nlink_t  st_nlink;  /* number of hard links */
  uid_t    st_uid;    /* user ID of owner */
  gid_t    st_gid;    /* group ID of owner */
  dev_t    st_rdev;   /* device ID (if special file) */
  off_t    st_size;   /* total size, in bytes */
  blksize_t st_blksize; /* blocksize for file system I/O */
  blkcnt_t st_blocks; /* number of 512B blocks allocated */
  time_t   st_atime;  /* time of last access */
  time_t   st_mtime;  /* time of last modification */
  time_t   st_ctime;  /* time of last status change */
};
```

Ainsi que les fonctions :

```
DIR *opendir(const char *name);
struct dirent *readdir(DIR *dirp);
int stat(const char *path, struct stat *buf);
int closedir(DIR *dirp);
int open(const char *path, int oflag, ...);
int close(int fildes);
size_t read(int fildes, void *buf, size_t size);
size_t write(int fildes, void *buf, size_t size);
off_t lseek(int fildes, off_t offset, int whence);
struct group *getgrgid(gid_t gid);
struct passwd *getpwuid (uid_t uid);
int chmod(const char *pathname, mode_t mode);
```

Exercice 1 : tail

La commande "`tail`" permet, par défaut, d'afficher les 10 dernières lignes d'un fichier.

1. Écrire une commande "`tail`" qui prend en argument le chemin d'un fichier et affiche les 10 dernières lignes de ce fichier.
2. Modifier le programme pour prendre en compte l'option -n permettant de préciser le nombre de lignes à afficher.
3. Modifier le programme pour prendre en compte l'option -c permettant de préciser le nombre d'octets à afficher.

Exercice 2 : ls

Dans cet exercice on veut réécrire la commande ls.

1. Écrire une commande qui prend en argument le chemin d'un répertoire et affiche son contenu. Exemple d'affichage :

```
SYSTEME
| TD7
| TP7
| readme.txt
```

2. Modifier votre programme pour prendre en compte l'option -R. Le programme affiche alors la structure de l'arborescence des fichiers à partir du répertoire donné. Exemple d'affichage :

```
SYSTEME
| TD7
  | td7.pdf
  | td7.tex
| TP7
  | tp7.c
  | tp7.pdf
  | tp7.tex
| readme.txt
```

3. Modifier votre programme pour prendre en compte l'option -l, qui premet en plus de donner le propriétaire, le groupe et la taille en octets.

Exercice 3 : `\filtremod"`

On souhaite écrire une commande permettant de donner aux différents types d'utilisateurs des droits « raisonnables » sur les fichiers d'une arborescence donnée, pour éviter par exemple des aberrations du style :

- droits plus restrictifs pour le propriétaire que pour son groupe ou pour les autres ;
- fichier ou répertoire autorisé en lecture à certains utilisateurs dans un répertoire qui leur est inaccessible ;
- fichier de type « exécutable » interdit en exécution à des utilisateurs ayant par ailleurs le droit de le lire...

1. Écrire une fonction `int elargir_droits(char *fichier)` qui accorde à chaque catégorie d'utilisateurs les droits accordés à une catégorie « inférieure ».
2. Écrire une fonction `int restreindre_droits(char *fichier)` qui retire à chaque catégorie d'utilisateurs les droits qui ne sont accordés à aucune catégorie « supérieure ».
3. Écrire un programme permettant d'appliquer l'une des deux fonctions précédentes à tous les fichiers d'un répertoire, en fonction d'un paramètre de la ligne de commande.
4. Sous Unix, les fichiers binaires exécutables sont au format ELF, et commencent donc [1] tous par les quatre caractères suivants : le caractère de code ASCII 127 (7f en hexadécimal, 177 en octal), puis les caractères `'E'`, `'L'` et `'F'`.

   Écrire une fonction `int est_de_type_executable(char *fic)` qui détermine si un fichier a vocation à être exécuté, puis modifier votre programme pour qu'il puisse accorder des droits en exécution pertinents sur les fichiers d'un répertoire.

Exercice 4 : wc

Écrire un programme "`monwc`" tel que `monwc` *-opt toto* affiche le nombre de :

- lignes du fichier *toto* si *opt* est l,
- mots du fichier *toto* si *opt* est w,
- caractères du fichier *toto* si *opt* est m,
- octets du fichier *toto* si *opt* est c.

1. pour en savoir plus, `man magic` ou `man file`