TP N°2
Système de fichiers

Préambule

Notions : read/write/seek, open/close, argc/argv, struct.

```c
#include <fcntl.h>
#include <unistd.h>

int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

Ouvre le fichier `pathname` et retourne un descripteur de fichier de type `int` ou `-1` sinon. L'argument `flags` inclus un des trois modes d'ouverture possibles : `O_RDONLY` (lecture seule), `O_WRONLY` (écriture seule) ou `O_RDWR` (lecture-écriture). Un mode d'ouverture peut être combiné avec les options suivantes (voir `man open` pour une liste exhaustive) : `O_APPEND` (écriture en fin de fichier), `O_TRUNC` (supprime le contenu du fichier), `O_CREAT` : création si le fichier n'existe pas, ... Quand `O_CREAT` est spécifié, l'argument mode indique les permissions en octal.
Exemple : `int f = open("fichier", O_WRONLY | O_CREAT, 0644);`

```c
int creat(const char *pathname, mode_t mode);
```

est equivalent à `open()` avec les flags positionnés à `O_CREAT | O_WRONLY | O_TRUNC`.

```c
int close(int fd);
```

ferme le descripteur de fichier et renvoie 0.

```c
ssize_t read(int fd, void *buf, size_t count);
```

retourne le nombre de données lues. Le paramètre `count` est le nombre d'octets lus et mis dans le buffer pointé par `buf`.

```c
ssize_t write(int fd, const void *buf, size_t count);
```

retourne le nombre de données écrites.

```c
off_t lseek(int fd, off_t offset, int whence);
```

retourne la position dans le fichier. Le paramétre `offset` est positif ou négatif et correspond au déplacement dans le fichier et `whence` peut prendre les valeurs : `SEEK_SET` (déplacement par rapport au début), `SEEK_CUR` (déplacement par rapport à la position courante) ou `SEEK_END` (déplacement par rapport à la fin).

Manipulation de fichier

Exercice 1. Complétez le programme suivant pour qu'il fonctionne comme indiqué :

```c
#include <stdio.h>
#include <string.h>

#define MAXPERS 3
#define TAILLE_NOM 15
#define TAILLE_ADRESSE 25

struct personne {
  char nom[TAILLE_NOM];
  int  age;
  char adresse[TAILLE_ADRESSE];
};

typedef struct personne Personne;

/* saisie des informations pour les MAXPERS personnes */
void saisiePersonnes(Personne *p) { ... }

/* sauvegarde des informations saisies dans le fichier "personne.data" */
void sauvegardePersonnes(Personne *p){ ... }

/* lecture du fichier "personne.data" entier */
void lecturePersonnes(Personne *p){ ... }

/* lecture du num-éme enregistrement du fichier "personne.data" */
void lireUnePersonne(Personne *p,int num){ ... }

/* fonctionnement du programme :
 * - s'il n'y a pas d'arguments, le programme saisi les personnes et les
    sauvegarde
 *   dans le fichier avant de se terminer ;
 * - si l'argument est 'a', le programme lit le fichier des personnes, l'
    affiche et se
 *   termine ; si le fichier "personne.data" n'existe pas, afficher un message
     d'erreur ;
 * - si l'argument est un numéro entre 0 et MAXPERS, lire l'enregistrement
    correspondant
 *   et l'afficher, sinon erreur.
 */

int main(int argc, char *argv[]) {
  Personne pers[MAXPERS];
  int i;

  if(argc == 1) { /* pas d'arguments */
    /* saisie et enregistrement des personnes */
    ...
```

```
    }
45  else if(strcmp(argv[1],"a") == 0) {
      /* lire tous les elements */
47    ...
    }
49  else {
      /* lire l'element "n" indique en argument */
51    ...
    }
53 }
```

Exercice 2. Créer un programme qui copie les données lues à partir du clavier dans un fichier appelé out.txt. Les Entrées/Sorties doivent être effectuées avec des appels système (de bas niveau) sans utiliser la bibliothèque standard de C stdio.h.