TD 1

Primitives de gestion de fichiers
Programmation Sys. en C

Objectifs du TD :

L'objectif de ce TD est, d'une part, de se familiariser avec la programmation en C et acquérir des connaissances en prog. C nécessaires pour réaliser facilement les TDs et les TPs à venir. D'autre part, ce TD permettra aux étudiants d'apprendre les primitives ou les fonctions de gestion de fichiers via l'écriture d'un programme en C.

Travail à effectuer :

Ce TD consiste à :

1) Apprendre avec l'aide du chargé de TD les primitives pour ouvrir/fermer un fichier (open/close), se positionner dans un fichier (lseek) et lire/écrire dans un fichier (read/write).

2) Ecrire un programme C nommé « ***init.c*** » qui initialise la valeur **100** dans un fichier nommé **NOMBRE**

3) Ecrire un programme C nommé « ***ajout.c »*** qui :
   - i) **Ouvre** le fichier **NOMBRE** ;
   - ii) effectue une **boucle de 10 tours** permettant de :
     - a) **lire** dans le **fichier NOMBRE** et sauvegarder la valeur lue dans un compteur,
     - b) augmenter de 1 à chaque tour la valeur du compteur et
     - c) écrire la valeur du compteur dans le fichier **NOMBRE**.
   - iii) Affiche la dernière valeur sauvegardée dans le fichier.

1) Les **primitives de gestion d'un fichier** sont décrites en détail dans ce qui suit :
   a. Ouvrir un fichier :
      i. **Syntaxe** :
      #include <sys/types.h>
      #include <sys/stat.h>
      #include <fcntl.h>
      int **open**(const char *pathname, int flags)
      int **open**(const char *pathname, int flags, mode_t mode) ; flags : O_RDONLY, O_WRONLY et O_RDWR
      ii. **Exemple** : pour ouvrir le fichier NOMBRE en écriture seulement et en mode d'accès 0664, nous devons taper :
      int df = open ("NOMBRE",O_WRONLY|O_CREAT,0664) ; O_CREAT : si le fichier n'existe pas, il sera créé.
      iii. **Valeur de retour** : le descripteur du fichier (une valeur entière) si succès et -1 si erreur

   b. Se positionner dans un fichier :
      i. Syntaxe :
      #include <sys/types.h>
      #include <unistd.h>
      off_t **lseek**(int fd, off_t offset, int whence) ➔ à *whence*, il faut ajouter l'offset, qui peut être positif ou négatif, pour obtenir la nouvelle position. Le premier paramètre ***fd*** est le descripteur du fichier
      ii. Exemples :
         1. Pour se positionner au début du fichier : lseek(df_n,(off_t)0,SEEK_SET)
         2. Pour se positionner à la fin du fichier : lseek(df_n,(off_t)0,SEEK_END)
         3. Pour se positionner à la position courante : lseek(df_n,(off_t)0,SEEK_CUR)
      iii. **Valeur de retour** : la position courante (nombre d'octets à partir du début du fichier) si succès et -1 si erreur

   c. Lire dans un fichier :
      i. **Syntaxe :**
      #include <unistd.h>
      ssize_t **read**(int fd, void * buf, size_t count)
      ii. Exemple : nbo_lus=read(df_n,&nombre,sizeof(int))
      iii. Valeur de retour : le nombre d'octets lus dans le fichier si succès et -1 si erreur

   d. Ecrire dans un fichier :
      i. Syntaxe :
      #include <unistd.h>
      ssize_t **write**(int fd, void * buf, size_t count)
      ii. Exemple : nbo_ecrits=write(df_n,&nombre,sizeof(int))

iii. **Valeur de retour** : le nombre d'octets écrits dans le fichier si succès et -1 si erreur

e. Fermer un fichier :
   i. **Syntaxe :**
      #include <unistd.h>
      int close(int fd)
   ii. **Exemple : close**(df_n);
   iii. **Valeur de retour** : 0 si succès et -1 si erreur

2)
Ayant vu ensemble la syntaxe et des exemples d'utilisation des primitives de gestion de fichiers, nous allons maintenant répondre à la deuxième question : nous allons écrire le programme *init.c* qui permet d'initialiser une valeur entière (par exemple 100) dans un fichier nommé NOMBRE. Si le fichier NOMBRE n'existe pas, dans le programme nous allons forcer sa création (flag : O_CREAT).

```
/* Programme d'initialisation init.c */

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <unistd.h>
#include <errno.h>


A compléter …
```

3) L'étudiant a une demie-heure à disposition pour répondre à cette question en tenant compte de tous les points spécifiés ci-dessus dans la question 3) et en écrivant un programme C, nommé *ajout.c*. Ensuite, la solution ci-après est présentée et expliquée par le chargé du TD.

```
/*Programme d'ajout ajout.c*/

#include <sys/types.h>
#include <errno.h>
#include <sys/file.h>
#include <unistd.h>

#define N_Boucles     10

A compléter …
```

A venir
Primitives de gestion des répertoires (opendir, readdir, mkdir, …)