IN405 – Système d'exploitation
TD 2 – Système de fichiers (1/2)

S. Gougeaud

2017/2018

L'archive `code-td2.tar` *contient trois fichiers :*

- `se_fichier.h` – *fichier d'en-têtes contenant les prototypes de la 'bibliothèque' à écrire ;*
- `main.c` – *fichier principal de l'exercice 2, contenant différents scénarios utilisant la 'bibliothèque' ;*
- `main-opt.c` – *fichier principal de l'exercice optionnel 3, contenant différents scénarios utilisant les fonctions avancées de la 'bibliothèque'.*

Exercice 1 – Bibliothèque de fonctions d'E/S

L'objectif de ce premier exercice est de créer une bibliothèque d'entrée/sorties utilisant les appels systèmes liés au système de fichier. A cet effet, il vous est demandé d'écrire le contenu du fichier `se_fichier.c`, composé du corps des fonctions énoncées dans `se_fichier.h`. **Attention : vous devez respecter les prototypes du fichier d'en-tête.**

Pour compiler le fichier `se_fichier.c` et en obtenir une bibliothèque, vous devez utiliser les commandes suivantes :

```
$ gcc -c -fPIC se_fichier.c
$ ar rcs libsef.a se_fichier.o
```

Les fonctions de la bibliothèque sont alors contenues dans l'archive `libsef.a`. Pour utiliser les fonctions d'une bibliothèque dans un autre programme, le compilateur doit avoir accès à deux ressources : les fonctions compilées (ici `libsef.a`) et les prototypes de fonctions apportées par le fichier d'en-tête (ici `se_fichier.h`). En supposant que ces ressources sont situées dans le même répertoire que votre programme issu de `test.c`, sa compilation se fait à l'aide de la commande suivante :

```
$ gcc test.c -L. -lsef
```

L'option `-L` permet d'indiquer au compilateur un chemin supplémentaire pour la recherche de bibliothèque ; le chemin indiqué est le répertoire courant. L'option `-l` lie la bibliothèque sef (`libsef.a`) au programme.

1. Ecrivez la fonction `SE_ouverture()` qui ouvre le fichier dont le chemin est donné en paramètre. Attention à bien créer le fichier s'il n'existe pas.
2. Ecrivez la fonction `SE_fermeture()` qui ferme le fichier donné en paramètre.
3. Ecrivez la fonction `SE_suppression()` qui supprime le fichier donné en paramètre.
4. Ecrivez la fonction `SE_lectureCaractere()` qui lit le prochain caractère du fichier donné en paramètre. Vérifiez si le fichier a été ouvert avec les bonnes permissions.
5. Ecrivez la fonction `SE_ecritureCaractere()` qui écrit le caractère dans le fichier donné en paramètre. Vérifiez si le fichier a été ouvert avec les bonnes permissions.

Exercice 2 – Utilisation de la bibliothèque

A l'aide des fonctions de votre bibliothèque, remplissez les corps des fonctions suivantes du fichier `main.c`. Le programme issu de la compilation consiste en l'exécution de tests vérifiant le bon fonctionnement de ces fonctions.

1. Ecrivez la fonction `affichage()` ayant le même comportement que la commande `cat`.
2. Ecrivez les fonctions `copie()` et `deplacement()` ayant respectivement le même comportement que les commandes `cp` et `mv`.
3. Ecrivez la fonction `sontIdentiques()` qui retourne `1` si les fichiers sont identiques, et `0` sinon.

Exercice 3 – Optionnel : Amélioration de la bibliothèque

Vous allez maintenant ajouter des fonctions de lecture/écriture à votre bibliothèque. Pour ceci, décommentez les prototypes restants de `se_fichier.h`, et écrivez les corps correspondants dans `se_fichier.c`. Ces fonctions peuvent être testées grâce au programme issu de `main-opt.c`.

1. Ecrivez les fonctions de lecture/écriture d'une chaîne de caractère.
2. Ecrivez les fonctions de lecture/écriture d'un entier (comportement de scanf avec %d).
3. Ecrivez la fonction de lecture d'un mot (comportement de scanf avec %s)