TP N°1 : Appels système & fichiers – 3exercices (+1 optionnel) Durée approx. ~ 1h45min.

A l'aide du support de cours et des éventuels mémentos réalisez les exercices suivants:
(Tout code ou implémentation compilant ou non sera étudié)
Le présent sujet de TP comporte 1 page

*I – **Copie de fichiers***

Écrivez un programme qui recopie un fichier **f1** vers un fichier **f2** à créer, à l'aide des primitives système et du skeleton (*cf archive **01_skeleton.tar.gz***) fourni.

- Vous ne chercherez pas à créer le nouveau fichier avec les permissions du fichier original.
- Les noms des fichiers sont à passer en paramètre sur la ligne de commande selon le code existant.

*II – **Print « reverse »***

A l'aide de la fonction ***lseek*** et du skeleton (*cf archive **01_skeleton.tar.gz***), écrivez un programme qui affiche un fichier texte à l'envers (affichage du dernier caractère, de l'avant-dernier, etc.)

- Vous ne chercherez pas à prendre en charge les fichiers autres que textuels.
- Le nom de fichier d'entré est à passer en paramètre sur la ligne de commande selon le code existant.
- L'affichage du résultat du fichier se fera sur la sortie standard ***STDOUT****(1).*

*III – **ls « like »***

Implémentez une nouvelle version de la commande « ***ls*** » listant les méta-données des fichiers au sein d'une arboressance. A l'aide des primitives système, des structures usuelles et du skeleton (*cf archive **01_skeleton.tar.gz***).

Pour simplifier le code, on ne passera ni option, ni expression régulière (RegExp)

- Vous chercherez à vous appuyer les structures suivantes : ***dirent*** / ***stat*** / ***passwd*** / ***group*** / ***tm*** ;
- Efforcez-vous à gérer au maximum les erreurs via les valeurs ***errno***.
- L'affichage sur la sortie standard est libre, néanmoins vous devez faire apparaitre les informations suivantes : **Nom** / **Permissions** / **Propriétaire** / **Groupe** / **Taille** / **Date de dernière màj**

  Exemple :

  # Prompt > ***ls /root/my_folder***

  File1 – rwxrwxrw- root : root – 4096 – 010916 @ 16h24
  File2 – rwxrwx--- root : root – 0785 – 020916 @ 10h22
  Myfil3 – rwx------ toto : users – 4096 – 010916 @ 16h24
  […]

*IV – **I/O « bufferisées »** (optionnel)*

On désire implémenter une nouvelle version de la librairie standard d'entrées/sorties à l'aide des primitives système.

- Donnez une définition du type FICHIER (cf : FILE) **/!** N'oubliez de prévoir la bufférisassions des entrées/sorties.
- Programmez la fonction ***my_open***, analogue à ***fopen***. Pour simplifier, on ne considérera que les modes d'ouverture "**r**" et "**w**".
- Programmez les fonctions ***my_getc*** et ***my_putc***, analogues à ***fgetc*** et ***fputc*** : fonctions bufferisées en entrée (*my_getc*) ou en sortie (*my_putc*).
- Programmez la fonction ***my_close***, analogue à ***fclose***.

*IV – **Rappels***

Approche incrémentale du développement

Pour obtenir les résultats attendus aux différents exercices, veuillez toujours appliquer une approche incrémentale en termes d'ajout de code/fonctionnalité.

Par exemple: une approche incrémentale pour ce type d'exercice "ls-like » serait :

1. La récupération des paramètres
2. Tester fichier/répertoire
3. Parcourir les éléments du répertoire en affichant leur nom
4. Alimenter chaque fichier avec une information supplémentaire : permission / taille / propriétaire…

Documentation

Pour obtenir des informations ou de la documentation ayez le réflexe d'utiliser les pages du manuel.

Par exemple:

- man 3 stat / man 2 open / man 2 readdir / man errno

Gestion des erreurs

Afin d'avoir une gestion des erreurs la plus précise possible ayez le réflexe d'utiliser les codes retours ***ERRNO*** spécifiés dans les pages de manuel

Par exemple:

- **EEXIST** File exists (POSIX.1)
- **EFAULT** Bad address (POSIX.1)
- **EISDIR** Is a directory (POSIX.1)
- **ENOTDIR** Not a directory (POSIX.1)
- **ELOOP** Too many levels of symbolic links (POSIX.1)