L3i – projet “Systèmes d’Exploitation”

C. Béranger / S. Berri / J.L. Bourdon / P. Laroque

janvier 2024

DÉPARTEMENT
Sciences Informatiques

Présentation du sujet

Il s’agit d’implémenter une bibliothèque de fonctions de gestion de fichiers simulant celle d’UNIX.

Après formatage d’une “partition” (un gros fichier UNIX classique), les fonctions devront permettre la création, la lecture / écriture, la recherche d’informations dans un fichier ainsi que sa destruction.

Travail à fournir

La bibliothèque à construire devra contenir au minimum un type *file* et les fonctions suivantes:

- **int myFormat(char* partitionName);** la fonction retourne 0 si le “formatage” a réussi.
- **file * myOpen(char* fileName);** la fonction “ouvre” ou crée un fichier de ce nom.
- **int myWrite(file* f, void* buffer, int nBytes);** la fonction retourne le nombre d’octets écrits.
- **int myRead(file* f, void* buffer, int nBytes);** la fonction retourne le nombre d’octets lus.
- **void mySeek(file* f, int offset, int base);** la fonction déplace le pointeur de lecture / écriture du fichier.

Attention: il est évidemment exclu d’utiliser les fonctions UNIX (sauf pour la création de la “partition”), puisque vos fonctions doivent travailler à l’intérieur du gros fichier / partition que vous aurez créé grâce à *myFormat*.

Programme minimal (“noyau dur”)

Dans la version minimale, il n’est pas demandé de gestion particulière de l’espace géré par la “partition”. La bibliothèque sera cependant fournie avec un programme de test permettant de vérifier que chacune des fonctions se comporte comme attendu. Ce programme prendra la forme d’un menu permettant à l’utilisateur de tester chacune des fonctions comme il le souhaite.

Extensions possibles:

- gestion de “blocs disque” afin de résoudre les problèmes de fragmentation (extension fortement recommandée!)
- visualisation de la partition avec les parties réservées et libres évoluant en cours de programme
- fonctions supplémentaires (ex: **int size(file* f);**)
- ...

Rapports

En ce qui concerne la documentation, on demande de rédiger deux (petits) documents PDF:

- Un guide de l’utilisateur, où le fonctionnement et l’utilisation du programme (paramètres, fichier(s) de configuration, ...) sont décrits. Le but de ce premier document est d’aider un programmeur système à utiliser ce programme (et, le cas échéant, les fonctions de la bibliothèque).
- Un guide du développeur, où l’algorithme utilisé (ainsi que vos structures de données) est précisé. Le but de ce second document est d’aider un programmeur qui souhaiterait améliorer / modifier la bibliothèque et / ou le programme de test à s’y retrouver dans le code de vos fonctions.

Commentaires

On demande enfin un bon niveau de commentaires dans les programmes sources. L'idéal serait d'utiliser la syntaxe doxygen (assez proche de javadoc) qui permettra d'extraire de vos commentaires une doc HTML. Une entrée du makefile pourrait permettre de réaliser cette extraction.

Délivrables, évaluation et échéances

Le travail à rendre est à faire par groupes de deux ou trois étudiants. Dès qu'un groupe est constitué, il doit envoyer le nom des membres, par mail, à l'adresse laroque@u-cergy.fr. La date limite d'envoi de ce mail d'information est donnée sur la page moodle du projet.

Le rendu final sera présenté sous la forme d'une archive au format tar (compressé ou non) ou zip, à l'exclusion de tout autre format.

Cette archive contiendra tous les fichiers décrits ci-dessus (source commenté, makefile, les 2 guides), et RIEN D'AUTRE!!!!

L'archive à rendre doit être postée sur moodle (date limite de remise indiquée sur la page moodle). *Les retards (concernant le choix des binômes comme la fourniture des délivrables) seront sanctionnés à hauteur de 1pt par jour, sans excéder 5 points.*

La grille d'(auto-)évaluation jointe sera utilisée pour évaluer les travaux remis