C TD/TP 5: Arborescence des fichiers, suite.

Enoncé

Ce TP se base sur le TP3 "Arborescence de fichiers". Pour rappel, le TP3 proposait de construire à l'aide de pointeurs une structure arborescente servant à stocker des fichiers texte en mémoire vive [1]. Cette organisation en arbre s'inspirait de l'organisation des fichiers sur disque dur. L'avantage d'une telle organisation est de pouvoir insérer/retrouver très rapidement des données en mémoire. Il existe cependant un gros inconvenient : la structure étant stockée en mémoire vive, toutes les données sont perdues lorsque le programme se termine.

Dans le TP5, on propose de remédier à cet inconvenient : on souhaite pouvoir sauvegarder sur disque dur la structure arborescente et le contenu des fichiers. La structure pourra également être restaurée en mémoire vive si nécessaire.

La sauvegarde de la structure se fera dans un fichier sur le disque, qu'on appellera `structure.sav`. Il est nécessaire de pouvoir lire/écrire des données à partir du fichier de sauvegarde. Pour cela, vous utiliserez les fonctions vues dans le TP précedent :

- `FILE *fopen(char *fic, char *type)` pour ouvrir le fichier.
- `int fread(char *ptr, int taille, int nitem, FILE *f)` pour lire à partir du fichier.
- `int fclose(FILE *f)` pour fermer le fichier.

Vous utiliserez également la fonction `int fwrite(char *ptr, int taille, int nitem, FILE *f)` pour écrire dans le fichier de sauvegarde. Pour plus d'informations sur ces fonctions, voir le TP4, ou taper `man <nom_fonction>` dans votre terminal linux.

Question 1

Si ce n'est déjà fait, terminer le TP3.

Question 2

Coder la fonction `void sauvegarder_fichier(fichier *fich, FILE *fichier_sauvegarde)` qui prend en paramètre un fichier à sauvegarder et l'écrit dans le fichier de sauvegarde (pour cela, il écrit d'abord la taille de la chaîne de caractères correspondant au nom du fichier, suivie de la chaîne elle-même. Puis il écrit la taille de la chaîne de caractères qui correspond à son contenu suivie de la chaîne elle-même).

Question 3

Coder la fonction `fichier *restaurer_fichier(FILE *fichier_sauvegarde)` qui lit un fichier à partir du fichier de sauvegarde donné en paramètre.

1. Une petite erreur s'est glissée dans l'énoncé du TP3 : dans la définition de la structure `repertoire`, il faut rajouter une variable entière qui contient le nombre de sous-répertoires, et une variable entière qui contient le nombre de fichiers contenus dans le répertoire.

Question 4

Coder la fonction `void sauvegarder_repertoire(repertoire *rep, FILE *fichier_sauvegarde)` qui prend en paramètre un répertoite à sauvegarder et l'écrit dans le fichier de sauvegarde. **Attention, puisqu'un répertoire peut contenir d'autres répertoires, cette fonction est *récursive*, c'est-à-dire qu'elle est susceptible de contenir un appel à elle-même.** Plus précisement, les actions effectuées par la fonction sont :

1. Écrire dans le fichier la taille de la chaîne de caractères correspondant au nom du répertoire suivi du nom lui-même.
2. Écrire *le nombre* de fichiers que contient le répertoire.
3. Parcourir tous les fichiers du répertoire en appelant à chaque fois `sauvegarder_fichier()`.
4. Écrire *le nombre* de sous-répertoires que contient le répertoire.
5. Parcourir tous les sous-répertoires en appelant à chaque fois `sauvegarder_repertoire()`.

Question 5

Coder la fonction `repertoire *restaurer_repertoire(FILE *fichier_sauvegarde)` qui lit un répertoire à partir du fichier de sauvegarde donné en paramètre. Cette fonction à un comportement similaire à la fonction `sauvegarder_repertoire`, sauf qu'elle lit à partir du fichier de sauvegarde au lieu d'écrire dedans.

Question 6

En vous aidant des fonctions codées précedemment, coder la fonction `void sauvegarder_systeme_de_fichiers()`. Cette fonction effectue les actions suivantes :

1. Ouvrir le fichier de sauvegarde en écriture.
2. Appeler sauvegarder_repertoire() sur le répertoire racine.
3. Fermer le fichier de sauvegarde.

Question 7

En vous aidant des fonctions codées précedemment, coder la fonction `void restaurer_systeme_de_fichiers()`. Cette fonction effectue les actions suivantes :

1. Ouvrir le fichier de sauvegarde en lecture.
2. Appeler restaurer_repertoire() sur le répertoire racine.
3. Fermer le fichier de sauvegarde.