Interrogation sur machine

Ecrivez une nouvelle version du programme de gestion d'une bibliothèque travaillé pendant ces dernières semaines : `biblioNew`.

L'usage du programme sera :

```
biblioNew [fichierTextIn] fichierEnreg
```

Si un seul nom de fichier est passé en argument au programme, c'est le nom du fichier (`fichierEnreg`) qui contient des enregistrements de type *Livre*, si deux arguments sont présents, le premier est celui d'un fichier texte (`fichierTextIn`) et le second celui du fichier d'enregistrements de type *Livre*.

Le fichier `fichierEnreg` contient des enregistrements de type *Livre* dont la structure est :

```
typedef struct {
    char titre[128];
    char auteur[80];
    long isbn;
    char editeur[50];
    int an;
    enum Genre genre;
} Livre;
```

Le fichier `fichierTextIn` contient des lignes de texte dont le format est :

1. Une chaîne de 128 caractères pour le titre
2. Une chaîne de 80 caractères pour le(s) auteur(s)
3. Un entier long pour l'ISBN (International Standard Book Number ou, en français, numéro international normalisé du livre)
4. Une chaîne de 50 caractères pour l'éditeur
5. Un entier pour l'année d'édition
6. Un genre sachant que les genres à prendre en considération pour cette application sont : « BandeDessinée », « Poésie », « Théâtre », « Roman », « RomanHistorique », « LittératureFrançaise », « LittératureEtrangère », « Sciences », « Informatique », « Science-fiction », « Santé », « Histoire » ; ce genre doit être converti en `enum Genre`.

Le programme doit

1. lire le fichier `fichierEnreg` s'il existe, charger son contenu dans une liste chaînée triée
2. lire, dans le fichier `fichierTextIn` s'il est précisé ou à l'entrée standard dans le cas contraire, des lignes d'au plus 256 caractères qui contiennent les informations concernant un livre, les décomposer, les stocker dans une structure de type *Livre* et les insérer dans la liste chaînée qui doit toujours rester triée.
3. écrire tous les éléments de la liste chaînée dans le fichier `fichierEnreg`

Contraintes de programmation :

- L'application doit être découpée en modules :

  listeLivre : gère la liste chaînée triée (le fichier listeLivre.o est fourni et ne peut pas être modifié, il contient les fonctions de traitement de la structure *Livre* et de la liste).

  fichierLivre :gère la lecture et l'écriture du fichier `fichierEnreg`

  biblioNew : gère la bibliothèque

  Recopiez dans votre répertoire les fichiers qui vous sont fournis dans le répertoire `/usr/local/I2010_langC/interro`

- ***Avant tout***, dans **chaque fichier de type texte** recopié, remplacer NOM par *votre* nom de famille et PRENOM par *votre* prénom.

- Complétez le Makefile.

- Les fonctions de gestion de la liste chaînée triée sont données dans le fichier `listeLivre.o`. Le fichier `listeLivre.h` vous précise les entêtes de ces fonctions.

- Complétez le module fichierLivre en écrivant les fonctions `lireFichier` et `ecrireFichier`
  - `lireFichier` reçoit deux paramètres, la liste chaînée à remplir et le pointeur vers le stream (`FILE *`) du fichier `fichierEnreg` ouvert en lecture, lit les enregistrements contenus dans le fichier et les ajoute à la liste chaînée et renvoie un booléen (entier) à vrai si le traitement s'est fait complètement sans erreur et faux sinon. Le fichier est trié, chaque enregistrement lu doit être simplement placé à la fin de la liste, réfléchissez à optimiser votre algorithme en évitant de parcourir toute la liste chaînée à chaque ajout, évitez d'utiliser la fonction `inserer` .
  - `ecrireFichier` reçoit deux paramètres, la liste chaînée à écrire et le pointeur vers le stream (`FILE *`) du fichier `fichierEnreg` ouvert en écriture, parcourt la liste chaînée et écrit chaque structure dans le fichier et renvoie un booléen (entier) à vrai si le traitement s'est fait complètement sans erreur et faux sinon. Toutes les zones mémoires allouées dynamiquement doivent être libérée et la liste réinitialisée.

- Complétez le module biblioNew en y ajoutant les instructions nécessaires au traitement demandé ci-dessus

    a. Si le programme est appelé avec moins d'un argument ou plus de deux, il affiche l'usage sur la sortie d'erreur et s'arrête avec un code d'erreur égal à 1.

    b. Si le `fichierEnreg` n'existe pas (`errno` positionné à `ENOENT`), le premier traitement est ignoré.

    c. Si le `fichierEnreg` ne peut pas être ouvert en lecture, un message d'erreur spécifique est affiché à la sortie d'erreur et le programme se termine avec un code d'erreur égal à 11, de même s'il n'est possible de l'ouvrir en écriture mais dans ce cas le code erreur devra valoir 12.

    d. Si le `fichierETextIn` ne peut pas être ouvert en lecture, un message d'erreur spécifique est affiché à la sortie d'erreur et le programme se termine avec un code d'erreur égal à 13.

Dans toute l'application, si des problèmes sont rencontrés pour lire ou pour écrire dans un fichier, l'utilisateur doit être averti par un message qui précise le type d'erreur avant que le programme ne se termine prématurément avec un code d'erreur à 21 (pour un problème en lecture) ou 22 (pour un problème en écriture).

Les éventuels problèmes d'allocation de la mémoire doivent aussi être signalés et provoquer l'arrêt du programme avec un code d'erreur à 30.

Les lignes lues dans le fichier `fichierTextIn` ou à l'entrée standard sont supposées correctes et conformes au format attendu, le séparateur est le caractère ' :' .