



1 Présentation

Votre travail consistera à écrire un programme permettant de gérer une collection de CD, en profitant des tables de hachages pour accélérer les traitements. Ce travail donnera lieu à un programme et à un compte-rendu qui sera noté.

Pour le compte-rendu, vous devrez mettre en avant les concepts du cours que vous aurez appliqué ici. Vous indiquerez également les structures de donnée utilisées. La clarté, la pertinence et les justifications serviront de base à la notation.

Pour les programmes, vous devrez être le plus fidèle possible aux spécifications énoncées ci-dessous. Vous ferez attention à la cohérence entre les éléments décrits dans le compte-rendu et les éléments de votre programme. La clarté, la propreté et le fonctionnement de votre programme serviront de base à la notation. Les programmes devront impérativement être écrit en langage C et compatible avec le système d'exploitation Linux.

Au moment de rendre votre travail, créer une archive au format zip (nom binôme 1 – nom binôme 2.zip) contenant :

- Le compte-rendu au format DOC ou PDF (Tout compte-rendu utilisant un autre format ne sera pas lu)
- Le code source commenté.

2 Gestion d'une CDthèque

Il s'agit de gérer une collection de CD au moyen d'un programme.

Afin de vérifier l'efficacité des différentes structures de données que l'on a pu voir en cours, vous utiliserez des listes simples et des tables de hachages permettant de faire des index.

Lorsqu'un élément sera ajouter ou supprimer de la CDthèque, vous mettrez à jour à la fois les listes simples et les index.

Pour remplir votre CDthèque, vous pouvez utiliser les informations provenant du site web suivant : <http://www.freedb.org>

2.1 Structures de données

Les données à gérer par le programme concernent le CD et les pistes du CD.

Voici les information que l'on souhaite conservé pour un CD :

- Nom de l'artiste
- Nom de l'album

- Durée
- Nombre de pistes
- Disc-ID
- Liste des pistes

Voici les informations que l'on souhaite conserver pour une piste d'un CD :

- Numéro de la piste
- Nom de la piste
- Durée

2.2 Les listes simples

Vous utiliserez des listes simples (triées ou non, à vous de choisir) pour stocker la liste des CDs et la liste des pistes. Vous avez le choix dans l'organisation des structures (deux listes séparées, une liste dans une liste...)

Vous indiquerez dans le compte-rendu quels sont vos choix en les justifiant.

2.3 Les index

Vous créerez quatre index se basant sur des structures de données de type tables de hachages :

- Index sur les pistes
- Index sur les « disc-ID »
- Index sur les noms de CD
- Index sur la durée totale des CD

Vous avez le choix l'implémentation de la table de hachage (une ou plusieurs fonctions de hachage, gestion des collisions directe ou indirecte...).

Vous indiquerez dans le compte-rendu quels sont vos choix en les justifiant.

2.4 Fonctionnalités offertes à l'utilisateur

Vous pouvez constituer votre base de données « en dur » grâce à une initialisation statique des différentes structures de données. Si vous optez pour cette solution, veillez à réaliser cette initialisation dans un fichier C à inclure au début du programme principal afin de ne pas nuire à la lisibilité de votre code.

Optionnellement, vous pouvez également proposer à l'utilisateur d'ajouter ou de supprimer des éléments de la base de données (CD ou pistes).

Les fonctions à réaliser obligatoirement sont les fonctions de recherche.

Vous proposerez à l'utilisateur grâce à un menu interactif plusieurs types de recherche :

- Recherche à partir d'un nom de piste la liste des CD contenant cette piste dans leur liste (Par exemple, si je recherche le mot « of », je récupérerai la liste des CD dont au moins une piste contient le mot « of »).
- Rechercher un CD en fonction de son disc-ID.
- Rechercher un CD en fonction de son nom.
- Rechercher un CD en fonction d'une durée maximale (Par exemple si j'effectue une recherche avec comme paramètre 10, j'obtiendrai les CD dont la durée ne dépasse pas 10 minutes).

Vous devez donc avoir dans votre code source 8 fonctions de recherche, 4 exploitant vos listes simples, et 4 exploitant vos index.

2.5 Comparaison des performances

Vous comparerez les performances lors de la recherche d'un élément dans la CDthèque. Vous pouvez utiliser la fonction `gettimeofday()` (Voir la page de manuel sur <http://www.linux-kheops.com/doc/man/manfr/man-html-0.9/man2/gettimeofday.2.html>) afin d'obtenir une mesure du temps précise.

Vous conserverez le temps minimal, maximal et moyen pour les recherches effectuées sur les listes simples et sur les tables de hachages (Vous conserverez donc 6 statistiques). Lorsque de l'utilisateur désire faire une recherche, le programme effectuera une première recherche dans vos listes simples puis une seconde recherche en se basant sur les index. Ces deux recherches doivent se faire de manière complètement indépendante afin de pouvoir mesurer avec précision les performances de ces deux types de structures de données.

Vous afficherez ensuite le résultat des mesures de performances à chaque recherche.

Vous pouvez également ajouter dans le menu une option permettant à l'utilisateur d'afficher les statistiques que vous avez enregistré à chaque recherche.

3 Question ouverte

Quelle structures de données semble la plus efficace pour effectuer des recherches dans une grande quantité de données ?

Cette efficacité se vérifie-t-elle dans tous les cas ?

Justifiez vos réponses en apportant vos arguments.

4 Indications supplémentaires

Ne négliger pas le rapport, un bon rapport indique un bon programme. Cela fonctionne aussi dans l'autre sens.

Il est interdit de s'échanger du code ou des bouts de code entre binôme.