Programmation en C avancée

Feuille de TD6

Exercice 1. Groove en pot !

Le roi Julien est horrifié lorsque la danse des babouins attire plus d'attention que sa petite personne dans le zoo. Après vérification dans son immense discothèque, il constate que les babouins lui ont volé certain de ses CD. Hélas, il n'avait jusqu'à présent aucun moyen de gérer son imposante collection musicale. Nous allons essayer d'aider le roi à y voir plus clair dans sa collection.

► **1.** Julien est un roi moderne. Il possède des vinyles, des cassettes, des CD et des mp3. Proposez un type énumération `support` qui recense l'ensemble des supports utilisés pour sa musique.

► **2.** Proposez une structure de données `chanson` pour représenter une chanson, sachant qu'une chanson se caractérise par un titre, un auteur, un numéro de piste, et enfin le type de support utilisé.

Pour regagner l'intérêt des visiteurs du zoo, il projette de faire une super méga groove partie. Il a besoin pour cela de créer une liste de lecture à partir de ses chansons.

► **3.** Proposez une structure `liste` doublement chaînée qui lui permettrait de naviguer dans ses chansons sélectionnées.

► **4.** Écrire une fonction `create_song` qui prend en paramètre un titre, un auteur, un numéro de piste et le type de support, et qui retourne un pointeur sur la chanson créée. De façon identique, écrire une fonction `delete_song` qui libère l'espace mémoire utilisé par une chanson (attention aux chaînes de caractères).

► **5.** Écrire une fonction `add_song`, qui ajoute une chanson à la fin d'une liste.

► **6.** Écrire une fonction `sort_liste`, qui trie les chansons d'une liste en fonction de leur type (toutes les chansons sur vinyle au début, puis toutes les chansons sur cassettes, ...)

Exercice 2. Compilation séparée et makefile

Un fichier *makefile* (nommé généralement `makefile` ou `Makefile`) est composé d'un ensemble de règles, chacune ayant la forme suivante :

```
1  regle1: dependance1 dependance2 <...>
2  <tabulation> commande1
```

Listing 1 – Forme des règles dans un fichier *makefile*.

Le nom `regle1` est généralement le nom du fichier que l'on veut créer avec la commande `commande1`. `dependance1`, `dependance2`, ...sont les noms des fichiers ou des règles utilisés pour générer le fichier `regle1`.
Pour utiliser un fichier *makefile*, on dispose de la commande `make`. La règle à exécuter doit être passée en paramètre (par exemple, `make regle1`). Si aucun argument n'est passé, par défaut la première règle du fichier sera exécutée.

► **1.** Écrivez un fichier *makefile* simple permettant de compiler le programme de l'exercice 1.

Nous allons maintenant reprendre le programme de l'exercice 1 pour qu'il adhère à certaines bonnes pratiques de programmation telles que la séparation entre l'*API* (l'interface de programmation décrite dans un fichier d'en-têtes), son implémentation (décrite dans un ou plusieurs fichiers de code source C) et le programme principal qui utilise l'*API*.
Dans le cas de l'exercice 1, nous allons regrouper les fonctions qui traitent des chansons et des listes de lecture dans une *API*.

► **2.** Écrivez dans un fichier `music.h` le prototype des fonctions utilisées pour créer et manipuler des chansons et des listes de chansons. De même, écrivez dans un fichier `music.c` la définition de ces mêmes fonctions (sans oublier d'y inclure `music.h`).

► **3.** Maintenant, créez le fichier `test-music.c`. Ce fichier contiendra notamment la fonction `main`. Cette fonction est le point d'entrée du programme qui teste l'*API*. Elle crée quelques chansons, les ajoute à une liste de lecture, trie cette liste puis affiche ses éléments.

Nous allons maintenant automatiser la compilation de ces fichiers.

► **4.** Écrivez le fichier *makefile* contenant notamment :
- les règles pour compiler les deux fichiers C séparément (pour créer les fichiers .o),
- une règle pour réaliser l'édition de lien (à partir des fichiers .o) et créer l'exécutable `music-tool`.

► **5.** Effectuez la compilation avec le fichier *makefile* et exécutez le programme ainsi obtenu.

```
1  make
2  ./music-tool
```

Listing 2 – Commandes pour compiler et exécuter `music-tool`.

► **6.** Dans `test-music.c`, modifiez la fonction main pour afficher la liste de lecture avant et après l'avoir triée.

► **7.** Recompilez le programme avec le fichier *makefile*. Quels fichiers ont été réellement recompilés ?

► **8.** Quels sont, selon vous, les bienfaits de la compilation séparée et de l'utilisation d'un fichier *makefile* ?