Pente : DB

D.Moreaux

4 décembre 2024

1 SQLite 3

La base de données utilisée sera en SQLite 3 (https ://www.sqlite.org/). Ce système sauve la totalité de la DB dans un fichier et n'utilise pas de serveur. A la place, tout le traitement est géré par la bibliothèque.

Il faudra installer sqlite3 sur la machine serveur. Un programme `sqlite3` permet d'éditer un fichier DB en fournissant une console SQL. Il suffit de l'appeler avec le nom de la DB (si elle n'existe pas, elle sera créée).

2 CMake

Pour que le programme compilé intègre le système SQLite 3, il faudra éditer le fichier CMakefile.txt.

Avant la création de la cible, on ajoutera une ligne permettant de détecter la présence, les répertoires (include et bibliothèque) et les options de compilateur à ajouter.

```
find_package(SQLite3 REQUIRED)
```

Après la définition de la cible, il faudra indiquer que la bibliothèque SQLite3 doit être ajoutée à la cible en question.

```
target_link_libraries(NomProgramme SQLite::SQLite3)
```

3 API SQLite

Il faudra inclure le header `sqlite3.h` dans chaque fichier utilisant SQLite.

La connexion/déconnexion (ouverture/fermeture) de la DB se fera de la manière suivante :

```
sqlite3 *db;
int rc; // Return Code
rc = sqlite3_open("fichier.sqlite", &db);
if (rc != SQLITE_OK) { // Erreur !!!
...
}
...
sqlite3_close(db);
```

De nombreuses fonctions existent mais on peut se limiter au sous-ensemble suivant (https ://www.sqlite.org/c3ref/funclist.html)

`sqlite3_stmt *stm;` permet de déclarer un "statement"

`rc = sqlite3_prepare_v2(db, sql, -1, &stm, NULL);` où sql est la chaîne contenant la requête SQL, stm la variable qui contiendra le statement préparé.

`rc = sqlite3_bind_double(stm, 1, 3.1415);` permet de lier au premier placeholder (?) la valeur double. `rc = sqlite3_bind_int(stm, 1, 42);` permettra de lier un entier et `rc = sqlite3_bind_text(stm, 1, "Hello", -1, NULL)` permettra de lier une chaîne. Si on désire lier une valeur NULL, on peut soit utiliser `sqlite3_bind_null(stm, 1);` soit utiliser un pointeur NULL au lieu d'une chaîne.

La commande `rc = sqlite3_step(stm);` permettra d'exécuter la requête et, le cas échéant, d'accéder aux résultats. La valeur de retour vaudra `SQLITE_DONE` pour indiquer que la requête est terminée, `SQLITE_ROW` pour indiquer qu'une ligne de résultats est disponible (il faudra répéter le $sqlite3_steppourobtenirlalignesuivante$).

`sqlite3_reset(stm);` permettra de remettre la requête à 0 avant de lier de nouvelles valeurs aux paramètres pour la réexécuter et `sqlite3_finalize(stm);` permettra de libérer la requête. Il est important de libérer les requêtes qui ne sont plus nécessaires [1].

Pour récupérer les résultats, on utilisera les fonctions `sqlite3_column_XXX(stm, 0)` pour récupérer la valeur de la 1ere colonne (et ainsi de suite). XXX vaudra `double`, `int`, `text`. Pour les valeurs de type text, il faudra penser à utiliser/copier la valeur avant de faire le step suivant [2].

1. équivalent de free() quand on utilise malloc()
2. strdup() peut aider