資料結構 – HW1 報告 *2024 OCT*

姓名：徐崇恆
學號：112503023
系級：通訊二

1 摘要

本作業的題目是於 Linux 環境以 C 語言開發簡單的 NoSQL 資料庫引擎，實現類似 Redis 的鍵值對 (key-value pair) 資料庫。儲存資料內容為手機通訊錄，將姓名作為 key，通訊資料作為 value，value 以 JSON 形式儲存。

開發環境

硬體：Raspberry Pi 5
作業系統：Raspberry Pi OS Lite (64-bit) (Linux-based OS)
IDE：Visual Studio Code (with SSH Remote)
編譯器：GCC
版本控制：Git, GitHub

2 問題分析

本作業所開發的資料庫需要支援基本的 CRUD (增刪查改) 功能，並在這基礎上再增添額外功能。基於本作業要開發的資料庫屬於 NoSQL 資料庫，採用 JSON 的形式處理資料可以符合 NoSQL 中靈活的資料形態的需求。另外，在程式需確保正確地分配和釋放記憶體，以避免記憶體洩漏 (memory leak) 。

NoSQL

NoSQL (Not Only SQL) 指的是非關聯型資料庫系統，是與關聯型資料庫 (SQL) 對立的資料庫系統。我們可以把 SQL 資料庫看作是欄位固定的試算表，試算表裡的每一行都是一筆資料，試算表的欄位決定資料的屬性；而 NoSQL 資料庫像是資料夾，資料夾裡面的每一頁文件都是一筆資料，每頁文件都有專屬的屬性。NoSQL 的特點是靈活的資料模型與高度可擴展性。

JSON

JSON (JavaScript Object Notation) 是一種用於在不同程式之間傳輸數據的輕量級資料格式，廣泛用於網路程式，也是大多數 NoSQL 資料庫數據交換的標準格式。

Key-value pair

鍵值對是一種資料的儲存形式，key 和 value 之間的關係是一對一映射。在 JSON 中，物件以鍵值對的方式表達。

Redis

Redis (Remote Dictionary Server) 是一個開源和基於記憶體的鍵值型 NoSQL 資料庫，Redis 的讀寫速度遠高於傳統硬碟資料庫，然而，它佔用的記憶體相對較多。

3 設計過程

索引

由於資料庫基於鍵值對，採用雜湊表 (hash table) 作為資料結構會比單一的鏈結串列 (linked list) 的效率更高。程式使用 DJB2 演算法對鍵進行雜湊演算以建立資料庫的索引，DJB2 是一種由 Daniel J. Bernstein 設計的簡單且高效的雜湊算法。資料庫預計儲存 100 筆資料，考慮載荷因子約為 0.7，將雜湊表大小 (table size) 設為 137。程式採用鏈結法 (chaining) 解決雜湊碰撞，雜湊表中每個桶 (bucket) 對應一個鏈結串列作為動態的槽位 (slots)。

持久性

為了實現數據持久性，資料庫需要把資料儲存在硬碟，資料將以 JSON 格式儲存。在程式啟動時，程式會加載前一次儲存的資料。程式結束時會將當前資料儲存。為應對程式長時間運行，程式需支援運行中的資料儲存功能。

互動界面

考慮到這只是一個簡單的程式，因此只以 Terminal 作為與使用者的互動界面。

資料模型

通訊錄的資料模型需要包括人物的姓名、電話號碼、電子郵件地址等。為了驗證程式處理多樣資料型別的能力，本作業的資料模型包含數字、布林值、陣列等型別的屬性。

以下是一份資料樣本:

```
{
    "name": "Alice",
    "jobTitle": "Software Engineer",
    "age": 30,
    "address": "123 Main St",
    "phoneNumbers": ["123-456-7890", "098-765-4321"],
    "emailAddresses": ["alice@example.com", "alice.work@example.com"],
    "isMarried": true,
    "isEmployed": true
}
```

4 實作

程式被分為 database 和 interface 兩個主要的功能模塊。database 主要處理資料庫的邏輯功能，interface 主要處理與使用者的互動。

cJSON

為了提高開發效率，程式引入外部函式庫 (external library) `cJSON`。`cJSON` 是一個用於解析和操作 JSON 資料的輕量化函式庫。雖然 `cJSON` 支援從根物件中取得鍵值對的功能，但該功能是透過鏈結串列實現的。因此，程式中採用雜湊表來提升執行效率。

多線程

為了支援多線程環境，程式引入 `pthread` 函式庫。`pthread` 是 Unix 中一套符合 POSIX 標準的 API，提供互斥鎖 (mutex) 和讀寫鎖等功能。程式中使用了互斥鎖確保全域變數的讀寫不衝突。

資料庫

資料庫模塊中實現了基本的 CRUD 功能。在程式執行過程中，資料以人物名稱為鍵儲存在雜湊表中。其中，`get_item` 用於取得資料節點，資料節點包含鍵值對和 next 指位器，節點中的值是 `cJSON` 結構。當需要更新人物屬性時，需要使用 `cJSON` 提供的函式去修改人物資料的 `cJSON` 結構，而在這個過程中會先使用 `get_item` 取得資料節點。此外，`set_item` 用於設定鍵值對，如果當前 key 已被佔用，舊的值將被刪除並替換。而 `rename_item` 用於更新資料鍵值對的 key，`delete_item` 則是用於刪除給定的 key 的資料。

```c
// Struct for storing key-value pairs in the database.
// Each DBItem contains a key (string) and a value (cJSON object), along with a pointer
 to the next item (for linked list chaining).
typedef struct DBItem
{
  char *key;
  cJSON *json;
  struct DBItem *next;
} DBItem;

// Check if an item with the given key exists
bool exists(const char *key);

// Retrieve an item by its key
DBItem *get_item(const char *key);

// Set an item with a given key and cJSON value
// If the key already exists, the old item is deleted and replaced with the new one
DBItem *set_item(const char *key, cJSON *json);

// Rename an item's key
DBItem *rename_item(const char *old_key, const char *new_key);

// Delete an item by its key
bool delete_item(const char *key);
```

Figure 1. database.h 程式碼片段 – DBItem 節點與 CRUD 功能函式

物件模板與定義模型

為了確保程式能適應各種可能的更動或新增的資料模型，程式應避免使用寫死 (hard-coding) 的方式處理物件的創建和編輯。為此，本作業中設計了一種物件模板用於動態地創建與編輯物件，物件模板的設計靈感來自 MongoDB 的 Object-Document Mapper (ODM) 概念。

物件模板將被用於定義該類物件所有鍵的名稱與值的型別，其也支援內嵌物件與陣列屬性的定義。當使用者需要創建物件時，程式能依據物件模板提示使用者逐一輸入對應型別的資料，然後將這些資料組裝成物件。在編輯物件時，則讓使用者選擇需要編輯的屬性，並輸入對應的資料。

本作業在 database 模塊實現了物件模板 `DBModel` 結構與相關應用的函式，而解析物件模板與處理使用者輸入的邏輯則在 interface 模塊實現。這項功能提高了程式的可擴展性與靈活性，也使程式更貼切 NoSQL 的理念。

```c
// Struct for defining the schema of a database model.
typedef struct DBModel
{
  const char *key;
  DBModelType type;
  int intvalue;
  struct DBModel **attributes;
} DBModel;

// Define a model under a parent model
// If creating a root object, set the parent to NULL
DBModel *def_model(DBModel *parent, const char *key, DBModelType type);

// Add an attribute (like length constraints) to a model
DBModel *def_model_attr(DBModel *model, DBModelType attribute, int value);

// Get a specific attribute of a model
DBModel *get_model_attr(DBModel *model, DBModelType type);
```

Figure 2. database.h 程式碼片段 - DBModel struct 宣告與相關函式

```
DBModel *person_model = def_model(NULL, "Person", DBModelType_Object);
def_model(person_model, "name", DBModelType_String);
def_model(person_model, "jobTitle", DBModelType_String);
def_model(person_model, "age", DBModelType_Number);
def_model(person_model, "address", DBModelType_String);
def_model(def_model(person_model, "phoneNumbers", DBModelType_Array), DBModel_ArrayType
Symbol, DBModelType_String);
def_model(def_model(person_model, "emailAddresses", DBModelType_Array), DBModel_ArrayTy
peSymbol, DBModelType_String);
def_model(person_model, "isMarried", DBModelType_Boolean);
def_model(person_model, "isEmployed", DBModelType_Boolean);
```

Figure 3. interface.c 程式碼片段 - DBModel 的使用

使用者介面

使用者界面 (interface 模塊) 主要處理提示與輸入。`main_menu` 函式為主選單功能，以一個 while true 迴圈支持使用者連續操作，程式將等待使用者輸入，並依據使用者的選項執行相應的功能。

```
#ifndef CCH137_INTERFACE_H
#define CCH137_INTERFACE_H

#include "./cJSON.h"
#include "./database.h"

// Error handler for memory allocation issues.
// It prints the file name, line number, and function name where the error occurred.
void memory_error_handler(const char *filename, int line, const char *funcname);

// Print the details of a person (DBItem).
void print_person(DBItem *item);

// Create a new person using the given model
void create_person(DBModel *person_model);

// Find a person in the database
void find_person();

// Update a person's information using the given model
void update_person(DBModel *person_model);

// Delete a person from the database
void delete_person();

// Capture user input and create a cJSON object based on a model
cJSON *input_cjson_with_model(DBModel *model, int tab_depth);

// Edit an existing cJSON object based on a model
bool edit_cjson_with_model(DBModel *model, cJSON *json, int tab_depth);

// Display the main menu and handle user input
void main_menu();

#endif
```

Figure 4. interface.h 程式碼片段

5 測試與結果

為了確保程式的各項功能正確，本作業在 test.c 中對資料庫程式進行簡單的單元測試。此外，本作業也對從 Terminal 操作資料庫的功能進行測試。所有的測試結果均符合預期，程式的功能都能夠正常執行，沒有明顯的錯誤。

```
################ Main Menu ################
Welcome to CCH's address book!!!
Choose an option:
C - Create a new person
R - Find a person
U - Update a person
D - Delete a person
K - List keys
S - Save database
X - Exit
Your choice: c
<Object> Person:
- <String> name: cch137
- <String> jobTitle: Student
- <Number> age: 20
- <String> address: 320, Zhong Li
- <Array> phoneNumbers
  length: 1
  - <String> 1: 12345678
- <Array> emailAddresses
  length: 1
  - <String> 1: cheechorngherng@gmail.com
- <Boolean> isMarried (y/n): n
- <Boolean> isEmployed (y/n): y
Person has been successfully created.
```

Figure 5. 創建一個聯絡人

```
################ Main Menu ################
Welcome to CCH's address book!!!
Choose an option:
C - Create a new person
R - Find a person
U - Update a person
D - Delete a person
K - List keys
S - Save database
X - Exit
Your choice: u
Enter the name of the person to update: cch137
Object fields:
1 - name
2 - jobTitle
3 - age
4 - address
5 - phoneNumbers
6 - emailAddresses
7 - isMarried
8 - isEmployed
Select a field of <Object> Person (1~8): 3
Selected key: age
- Enter a number value: 21
Person has been successfully updated.
```

Figure 6. 更新一個聯絡人

```
################ Main Menu ################
Welcome to CCH's address book!!!
Choose an option:
C - Create a new person
R - Find a person
U - Update a person
D - Delete a person
K - List keys
S - Save database
X - Exit
Your choice: r
Enter the name of the person: cch137
--------------------------------------------------
Name            : cch137
Job Title       : Student
Age             : 20
Address         : 320, Zhong Li
Phone Number    : 12345678
Email Address   : cheechorngherng@gmail.com
Married         : NO
Employed        : YES
--------------------------------------------------
```

Figure 7. 顯示已創建的聯絡人資料

```
################ Main Menu ################
Welcome to CCH's address book!!!
Choose an option:
C - Create a new person
R - Find a person
U - Update a person
D - Delete a person
K - List keys
S - Save database
X - Exit
Your choice: r
Enter the name of the person: cch137
--------------------------------------------------
Name            : cch137
Job Title       : Student
Age             : 21
Address         : 320, Zhong Li
Phone Number    : 12345678
Email Address   : cheechorngherng@gmail.com
Married         : NO
Employed        : YES
--------------------------------------------------
```

Figure 8. 顯示更新後的聯絡人資訊

```
################ Main Menu ################
Welcome to CCH's address book!!!
Choose an option:
C - Create a new person
R - Find a person
U - Update a person
D - Delete a person
K - List keys
S - Save database
X - Exit
Your choice: d
Enter the name of the person to delete: cch137
Person deleted successfully.

################ Main Menu ################
Welcome to CCH's address book!!!
Choose an option:
C - Create a new person
R - Find a person
U - Update a person
D - Delete a person
K - List keys
S - Save database
X - Exit
Your choice: r
Enter the name of the person: cch137
Person not found.
```

Figure 9. 將聯絡人刪除並檢查是否成功刪除

```
get_item(Alice) PASS
get_item(Unknown) PASS
get_item((null)) PASS
set_item(Person1, 0x5555b2b1c990) PASS
set_item((null), 0x5555b2b1ca20) PASS
set_item((null), (nil)) PASS
rename_item(Alice, Alex) PASS
rename_item(Bob, Bob) PASS
rename_item(NotInDBName1, NotInDBName2) PASS
rename_item(Bob, (null)) PASS
rename_item((null), Bob) PASS
rename_item((null), (null)) PASS
delete_item(Alex) PASS
delete_item(Unknown) PASS
get_database_keys() PASS
get_cjson_keys() PASS

total PASS: 16
total FAIL: 0
```

Figure 10. test.c 執行結果

6 討論

在開發本作業的過程中，我發現自己其實是在開發一個簡易版的 Redis。透過查閱資料，我了解到 Redis 可以依靠網路功能來實現伺服器節點間的資料傳輸和同步，並支援分片技術來分散資料，從而實現分散式集群的高併發和容錯。然而，在我實作的資料庫中，並沒有實現作業額外要求中的網路功能，因此目前只能在單機上運行。

此外，程式採用了同步持久化儲存方式，這與 Redis 採用的異步儲存方式不同。同步持久化儲存會阻塞主線程，這可能會影響整體的效能。同時，作為一個資料庫若擁有自動備份的功能會更符合實際業務需求，也會進一步提升系統的可靠性與資料安全性。以上都是可以改進的地方。

在這一次開發過程中，我首次在 Linux 環境中進行 C 語言開發。這使我學會了使用終端機進行編譯和調試，並對 Linux 系統的命令行操作有了初步的了解。透過這次經驗，我對 Linux 環境下的開發流程有了更深入的理解，也對開源世界有了進一步的探索興趣。

7 結論

程式最終實現了 CRUD 基本功能，並支援以雜湊表實現的索引、JSON 檔的持久化儲存以及多線程功能，本作業中也對程式進行了簡單的單元測試。在這份作業中，我學習了 NoSQL、JSON 語法等，也對指位器和記憶體的分配與釋放進行充分的運用和練習。總的來說，程式達到了作業題目預期目標，我也學習到了關於資料庫開發的經驗。

8 參考文獻與資料

1. djb2：一個產生簡單的隨機分佈的雜湊函數 - 範加索爾拉 - 博客園
   https://www.cnblogs.com/vancasola/p/9951686.html
2. POSIX 執行緒 - 維基百科，自由的百科全書
   https://zh.wikipedia.org/zh-tw/POSIX%E7%BA%BF%E7%A8%8B
3. Redis 中的資料是如何持久化的儲存？
   https://youtu.be/ENP9PpV6Bw0

9 附錄

1. 程式碼 GitHub 倉庫
   https://github.com/113NCUCE/hw1-cch137
2. 函式庫 cJSON GitHub 倉庫 - DaveGamble/cJSON: Ultralightweight JSON parser in ANSI C
   https://github.com/DaveGamble/cJSON