資料結構 – HW5 報告 *2024 DEC*

姓名：徐崇恆
學號：112503023
系級：通訊二

1 摘要

本作業旨在 Linux 環境中以 C 語言實作類 Redis 的 NoSQL 資料庫。本作業重點是為資料庫支援雜湊表 (Hash Table) 的基本操作以及設定鍵值的到期 (Expire) 功能。

2 需求分析

Redis 中的雜湊表是一種用於存儲鍵值對的資料結構，類似於其他資料庫中的物件或字典。每個雜湊表由多個欄位（field）及其對應的值（value）組成，適合儲存結構化且關聯性高的小型資料，例如使用者資訊或配置項。當對非 Hash 資料形態的鍵使用 Hash 指令時，資料庫將會報錯。

在本作業中，需要支援的 Sorted Set 操作包括：

1. **HSET**
   句法：HSET key field value [field value ...]
   將指定 key 的雜湊表的欄位設為指定的值，若雜湊表不存在則創建雜湊表，若欄位已有值則舊值會被覆蓋。回傳成功設定的欄位個數。
2. **HGET**
   句法：HGET key field
   回傳與 key 處儲存的雜湊中的欄位關聯的值。若 key 處或欄位不存在則回傳 NULL。
3. **HDEL**
   句法：HDEL key field [field ...]
   刪除指定 key 的雜湊表的欄位，回傳成功刪除的欄位個數。若雜湊表或欄位不存在則忽略操作。
4. **EXPIRE**
   句法：EXPIRE key seconds [NX | XX | GT | LT]
   對指定 key 的鍵值對設定到期秒數，在指定秒數後，該 key 將被自動清除。

3 設計

雜湊表

雜湊表是一種高效的資料結構，用於以鍵值對的形式儲存和查找資料。它透過雜湊函數將鍵映射到表中的特定槽位，從而快速定位資料。雜湊表的核心特點是操作速度接近常數時間，但在碰撞（不同的鍵映射到相同槽位）發生時需要使用解決方法，例如鏈接法或開放位址法。廣泛應用於快取系統、資料庫和其他需要快速存取的場景中。

本作業採用鏈接法（Chaining） 解決雜湊碰撞，它將映射到同一槽位的元素存入一個鏈結串列中。插入、查找和刪除操作在該槽位的鏈結串列中進行。此方法簡單且有效。

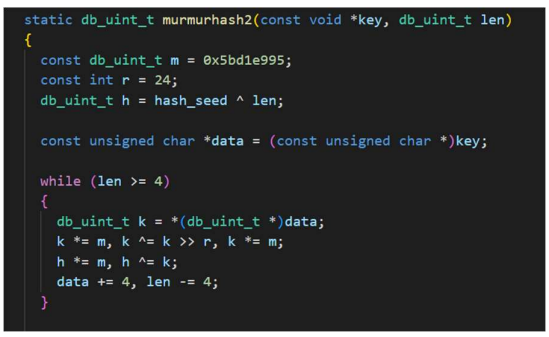
載荷因子與 Rehash 機制

本作業實作的雜湊表具備自動 Rehash 機制。在執行插入、查詢或移除等基本操作時，會自動觸發 Rehash 檢查。當表中元素總數超過表大小的 70% 時，表的大小將擴展為當前大小的兩倍；當元素總數低於表大小的 10% 時，表的大小則縮減為當前大小的一半。

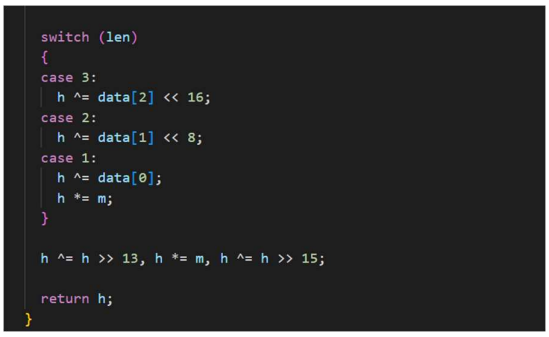
4 實作

雜湊演算法

本作業的雜湊演算法效仿 Redis，採用了 MurmurHash2。這是一種由 Austin Appleby 開發的高效能、非加密雜湊函數，專為快速計算大量資料的雜湊值而設計，常用於雜湊表等資料結構。該函數以運算速度和雜湊值的均勻性為設計重點，適合應用於對安全性要求不高但需要高效雜湊運算的場景，例如資料分片與快取系統。



```c
static db_uint_t murmurhash2(const void *key, db_uint_t len)
{
  const db_uint_t m = 0x5bd1e995;
  const int r = 24;
  db_uint_t h = hash_seed ^ len;

  const unsigned char *data = (const unsigned char *)key;

  while (len >= 4)
  {
    db_uint_t k = *(db_uint_t *)data;
    k *= m, k ^= k >> r, k *= m;
    h *= m, h ^= k;
    data += 4, len -= 4;
  }
```

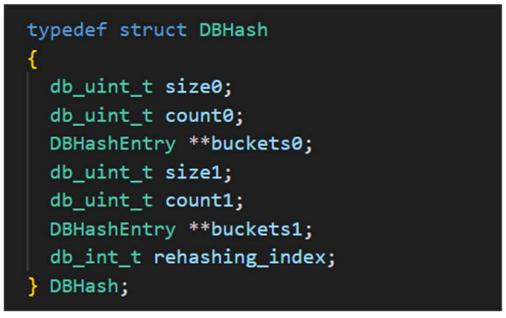


```c
  switch (len)
  {
  case 3:
    h ^= data[2] << 16;
  case 2:
    h ^= data[1] << 8;
  case 1:
    h ^= data[0];
    h *= m;
  }

  h ^= h >> 13, h *= m, h ^= h >> 15;

  return h;
}
```

結構體 (struct) 與 Rehash 機制

雜湊表相關的結構體分為雜湊條目與雜湊表本身。在本作業中，雜湊表實際由兩個部分組成：表 0 作為主表，表 1 用於 rehash 過程中。當 rehash 發生時，表 1 會被創建，並逐步將條目從表 0 遷移至表 1。當所有條目遷移完成後，表 1 將取代表 0 成為新的主表。雜湊表的條目包含鍵值對以及用於鏈接法的 next 指標，其中值為一個 obj 結構體。因此，雜湊表能夠支援多種類型的資料，而不僅限於字串。

```c
typedef struct DBHashEntry
{
  char *key;
  struct DBHashEntry *next;
  DBObj *data;
} DBHashEntry;
```



```c
typedef struct DBHash
{
  db_uint_t size0;
  db_uint_t count0;
  DBHashEntry **buckets0;
  db_uint_t size1;
  db_uint_t count1;
  DBHashEntry **buckets1;
  db_int_t rehashing_index;
} DBHash;
```

5 測試與結果

在編寫完程式後我們對相關函式進行了測試，測試結果正常。

1. HSET, HGET, HDEL 的測試:

```
Welcome to cch137's database!
Please use commands to interact with the database.
> hset k1 k1k1 a k1k2 b k1k3 c
(uint) 3
> hget k1 k1k1
"a"
> hget k1 k1k2
"b"
> hget k1 k1k3
"c"
> hset k2 k2k1 d k2k2 e
(uint) 2
> hget k2 k2k2
"e"
> hdel k1 k1k3
(uint) 1
> hget k1 k1k3
(nil)
```

2. EXPIRE 的測試:

   第一次 get 是馬上輸入，第二次 get 是在 10 秒後才輸入。

```
Welcome to cch137's database!
Please use commands to interact with the database.
> set k1 v1
"OK"
> get k1
"v1"
> expire k1 10
(int) 1
> get k1
"v1"
> get k1
(nil)
```

6 討論

事件觸發導向

在本作業中，我們未選用 libev 等函式庫，而是自主實現了事件驅動模型，旨在效仿 Redis 的設計理念，維持程式的簡潔性與高效性。本程式分為兩個執行緒（thread）：主執行緒與資料庫執行緒。主執行緒負責提供使用者介面，而資料庫執行緒則專注於資料庫的核心操作。

指令會被解析為 Request 結構體並提交給資料庫執行緒處理。在提交過程中，主執行緒會上鎖全域互斥鎖（mutex），並在提交完成後解鎖。資料庫執行緒則不斷進行上鎖與解鎖操作，用於檢查請求，並將接收到的請求分配給相應的函式處理。當資料庫操作完成後，結果會以 Reply 結構體的形式回傳給主執行緒。

定時任務

當資料庫啟動時，會創建一個執行緒，執行無窮迴圈。每次迴圈結束後，會根據當前的系統繁忙程度，決定下一次迴圈啟動的時間間隔。當資料庫接收到關閉指令時，該無窮迴圈將終止。

在這個無窮迴圈中，可以設置定時任務，例如在資料庫空閒時檢查是否需要執行 Rehash 或清除到期資料，可以使到期的資料及時清除，從而有效提升程式效能。

7 結論

本次作業透過 C 語言實作類 Redis 的 NoSQL 資料庫，成功完成雜湊表 (Hash Table) 核心功能以及 EXPIRE 指令的實現。透過 MurmurHash2 雜湊演算法、鏈接法與動態 Rehash 機制，確保資料庫高效且穩定運作。事件驅動模型與雙執行緒設計有效分工，並結合定時任務進一步提升了系統效能。

測試結果顯示，HSET、HGET、HDEL 和 EXPIRE 指令均正常執行。本作業實作了雜湊表資料結構與事件驅動設計，為未來可能的功能擴展奠定了良好基礎。

8 參考文獻與資料

1. Redis hashes | Docs
   https://redis.io/docs/latest/develop/data-types/hashes/
2. Redis 哈希 (Hash) | 菜鸟教程
   https://www.runoob.com/redis/redis-hashes.html
3. Redis Expire 命令 | 菜鸟教程
   https://www.runoob.com/redis/keys-expire.html

9 附錄

1. 函式庫 cJSON GitHub 倉庫 - DaveGamble/cJSON: Ultralightweight JSON parser in ANSI C
   https://github.com/DaveGamble/cJSON