# 2020_OS_Fall_HW3: Key-Value Stroages

繳交日期：11/16 ~ 12/14 13:00

## 作業目標

- 請撰寫一支 Key-Value Stroages 的程式，其功能是能夠處理 **PUT**、**GET**、**SCAN** 指令，且處理的資料量必須大於電腦記憶體的容量。
- 請觀察及分析程式執行期間，包括但不限於CPU、Memory、Disk I/O的使用情況，探討作業系統是如何服務我們的程式。
- 請說明你所設計的整理資料作法，並分析這些設計在我們存取資料的時候，如何能提供較佳的存取效率。

## 指令說明

### 指令格式

指令名稱、key、value之間以一個空格分開。

- **PUT** [key] [value]
  - **新增**或是**更新** 特定一個 key 的 value。
- **GET** [key]
  - **取得** 特定一個 key 的對應 value。
  - 如果**沒有取得**則輸出字串 **EMPTY**。
  - **GET** 的時候，必須取得最新的一筆資料。
- **SCAN** [key1] [key2]
  - key1 < key2。
  - **取得** key1 ~ key2 之間的資料（包含 key1、key2）。
  - 相當於執行多個 **GET**。
- **key**
  - 數值大小：8 bytes。
  - 數值範圍：0 ~ 9,223,372,036,854,775,807，即 0 ~ $2^{63}$-1。
    - 410741514603306026
    - 5436506213206988838
- **value**
  - 固定128個字元長度。
  - 由大小寫英文字母及數字0~9組成。
    - pmitlVpSUNVqKpu6zTJNuFEPG5nOXdFAeeiSdeDDRPctjTBZ2rnGV1O9u4UHv0qD4Ek2jfMjbJS9udHpRIcvXQsO9buh6dmbOORtMFTa4cTTMGeDtw5LXIcNYVf6KAGW
    - 2nrMOMu2NQlgkNcuY6JKTXqZG6d51J2qPpfkOObbtmlK4tbjRZIinRm6eKdfBvNEG7gB7Qs9uppUZF9xCeBaYMvqKUHdUREmXoRNOInwKTZtgDU2iwYwUMYpNXOE2h2X

### 指令範例

為了方便閱讀，這裡的 key 用數字，value 用簡單的英文單詞，實際的測資請參閱指令格式的說明

- 輸入檔案

```
PUT 2 first
GET 2
```

- 輸出檔案

```
first
```

**PUT** 執行之後，**Storage** 裡面儲存 key 2 以及其對應的 value，所以當 **GET** 執行，輸出 first。

- 輸入檔案

```
PUT 4 first
PUT 4 second
GET 4
```

- 輸出檔案

```
second
```

第二個 **PUT** 更新了 key 4 的 value。

- 輸入檔案

```
PUT 4 first
PUT 5 second
PUT 10 notes
```

全部都是**PUT**指令，不產生輸出檔。

- 輸入檔案

```
GET 9
```

- 輸出檔案

```
EMPTY
```

輸出 **EMPTY**。
因為沒有執行過 **PUT** 9，**Storage** 裡面還沒有 key 為 9 的資料。

- 輸入檔案

```
GET 8
PUT 8 first
```

- 輸出檔案

```
EMPTY
```

輸出字串 **EMPTY** 。
**GET** 的時候，Storage 裡面還沒有 key 8，所以是**EMPTY**。

- 輸入檔案

```
PUT 3 first
PUT 8 second
PUT 7 third
PUT 5 fourth
SCAN 3 7
```

- 輸出檔案

```
first
EMPTY
fourth
EMPTY
third
```

**SCAN**為取得 key 3 ~ key 7 的 value，**Storage** 裡面沒有 key 4、key 6，所以輸出 **EMPTY**。
**SCAN**取得多筆資料，一筆資料一列。

## 程式說明

**撰寫一隻能夠處理PUT、GET、SCAN指令的程式，由讀取輸入檔案的方式，讀入需要處理的指令，並輸出執行結果。**

1. 程式依照提供的路徑，讀取輸入檔案，檔案內容是要進行處理的指令。

- 輸入檔案的副檔名是 **input**。
- `./P76081043 ./1.input`
- 不得寫死讀取檔案的路徑。
- 從輸入檔案的第一列開始依序讀取執行，下一列為更新的一項指令。
- 一次只會讀取一個輸入檔案。
- 輸入檔案編碼為 `UTF-8` 。
- **請不要更動輸入檔案的資料。**

2. 依照輸入的測試檔案名稱，在程式執行目錄下產生對應的輸出檔案名稱，副檔名為**output**。
   - 輸出檔案的副檔名是 **output**。
   - 執行： `./P76081043 ./1.input`
   - 程式執行同目錄下輸出： `1.output` 。
   - 輸出檔案編碼為 `UTF-8` 。
   - 因為會測試多個檔案，所以要求對應的檔案名稱，以利批改。
3. 程式必須具備狀態保存的功能。
   - 第二個輸入檔案或之後的輸入檔案中的 **GET**、**SCAN** 指令，必須能夠取得之前輸入檔案中 **PUT** 過的 key。
4. 如果有需要建立暫存檔，請在程式執行目錄下建立一個名為**storage**的資料夾，並將所有的暫存檔放在裡面。
5. 執行程式所需要的輸入參數只允許**一個資料路徑**。
   - `./P7681043 ./1.input`
   - ~~`./P76081043 ./1.input 8`~~
   - ~~`./P76081043 -path ./1.input`~~
   - ~~`./P76081043 -path ./1.input -thread 16`~~
6. 必須能夠正確處理指令說明中說明的各項指令以及規定的格式。
7. 測試的時候，測試資料將會大於實體記憶體。
8. 請不要直接調用外部的函式庫來幫助你完成程式行為。
9. 會提供測試資料，程式必須先通過測試資料的測試，否則後續批改皆不進行。
10. 請勿在程式中，**強制包含**產生測試資料的流程，因為助教會使用自己的測試資料。

11. 執行後目錄結構參考：

```
❯ tree ./HW3_P76081043 -C
./HW3_P76081043
├── 1.output
├── HW3_P76081043_report.html
├── P76081043.class
├── P76081043.java
└── storage
    ├── 1.tmp
    ├── 2.tmp
    └── 3.tmp
```

### 輸入檔案、輸出檔案說明：

- 輸入檔案：
  - 副檔名為 input，例：**1.input**。
  - 檔案編碼為 UTF-8。
  - 一列一個指令，以 **換行字元(LF)** 分隔每個指令。
  - **輸入檔案結尾不會有任何空行。**
  - 
```
❯ tail hw3example.input -n 2
PUT 1600806503768274 57 48AjJsyYD0s9rAkhzEtlVRYXlXFG78
AYjQY2cl7mVVyVn8kjgc2XkqrHGXUxsTv2ioPAbiA4gKr1RhrQfxy
eDyXTjkG689MJGygSnxhkxZUN49ascaBFWXNgw7idzkni
```
  - 
```
dslab@dslab-os02:~$ tail hw3example.input -n 2
PUT 160080650376827457 48AjJsyYD0s9rAkhzEtlVRYXlXFG78AYj
QY2cl7mVVyVn8kjgc2XkqrHGXUxsTv2ioPAbiA4gKr1RhrQfxyeDyXTj
kG689MJGygSnxhkxZUN49ascaBFWXNgw7idzkni
PUT 412406240571828428 mns5NYbz7uEidYNMzg2mUK8jQpoxUFLKe
LKqwvfRVzKoLyDeVt8Eln5YWemC4aU4BowIskhIVNCilg66XZhvUWk9I
BSx5He8ulOzvTG7sSC03onxktHLaYbWG1Sou8ovdslab@dslab-os02:
~$
```
- 輸出檔案：
  - 副檔名為 output，例：**1.output**。
  - 檔案編碼為 UTF-8。
  - 一列為一個輸出結果，以 **換行字元(LF)** 分隔每個結果。
    - **GET** 佔一列。

        - 取得一筆資料。
    - **SCAN** 可能佔多列。
        - **SCAN** 取得多筆資料，一筆資料一列。
- 輸出檔案為 **GET**、**SCAN** 兩個指令的執行結果，如果輸入的測資檔案全為 **PUT** 指令，則不需要產生 output 檔案。

> 為了方便閱讀，這裡的 key 用數字，value 用簡單的英文單詞，實際要用的測資請參閱指令格式的說明

測試資料範例如下：

- 1.input

```
PUT 1 memory
PUT 2 page
PUT 8 word
GET 2
GET 5
PUT 2 disk
GET 2
PUT 6 gpu
PUT 1 monitor
PUT 4 section
PUT 5 tea
SCAN 4 7
...(略)
```

程式讀取並處理完成之後，
產生一個輸出檔案，內容如下：

- 1.output

```
page
EMPTY
disk
section
tea
gpu
EMPTY
```

## 如何開始

1. 請撰寫一隻 Key-Value Storages 程式，其功能符合程式說明並能夠正確處理指令說明之中的指令行為。
2. 請使用任意工具或方法分析、觀察你所撰寫的程式，並優化你的程式（例如：降低執行時間）。
    - 紀錄如何最佳化程式的過程，會有加分。
3. 將你所觀察到的現象，試著思考作業系統背後的行為，撰寫出一份完整的分析報告。
4. 請說明你所設計的整理資料作法，並分析這些設計在我們存取資料的時候，如何能提供較佳的存取效率。
5. 將你撰寫的程式碼及說明文件，依照作業繳交的規定，於期限內上傳到Moodle平台。

# 說明文件

說明文件的格式限定為 **PDF**、HackMD產生的**HTML**。
在撰寫此份說明文件時，必須要包含下列基本內容。

## 基本資訊

**學號：**
**姓名：**
**系級：**

## 開發環境：

- OS: Ubuntu 20.04.1
- CPU: Intel® Core™ i7-10700 CPU @ 2.90GHz × 16
- Memory: 32GB
- Programming Language(version): Java 1.8.0_261
  - 必須包含版本資訊

## 程式執行時間：

- 請在你的程式中加入量測執行時間的程式碼，以精準的獲取此數值。
- 助教測試程式的時候，會利用 Linux 提供的 time (https://man7.org/linux/man-pages/man1/time.1.html) 指令輔助驗證。
  - `time ./P76081043 ./1.input`

## 程式開發與使用說明：

- 你是如何開發這支程式，程式在處理資料的流程及邏輯為何？
- 你的程式該如何使用，請詳細說明編譯並執行的步驟。

```
# （重要）請確保助教能夠按照此步驟編譯並執行你的程式。
# 在程式執行的指令中，請提供可設定測試資料路徑的參數，且只能有資料路徑這一個參數。

# Java Example
# Compile
❯ javac ./YourSourceCode.java
# Run
❯ java ./YourSourceCode [data path]
```

## 分析報告：

分析報告的內容建議包含以下內容，但你也可以自由發揮。
此部分的內容將會是作業評分的重點，盡你所能說明的越詳盡越好。

- 請說明你所設計的整理資料作法，並分析這些設計在我們存取資料的時候，如何能提供較佳的存取效率。
- **在你開發的程式執行下：**

    - 請觀察系統效能以及OS是如何服務我們的程式。
- 可以搭配圖片、圖表或外部資料來說明。

# 作業繳交

## 繳交日期

> 繳交日期：11/16 ~ 12/14 13:00

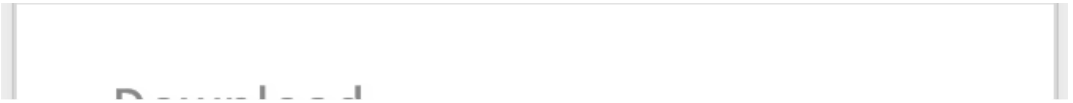
- 逾期繳交將按以下規則**採連續扣分**。
    - 逾期一日：得分扣10分。
    - 逾期二日：再扣20分。
    - 逾期三日：再扣30分。
    - 逾期四日以上：得分以0分計算。

## 繳交方法

- 請將你的「程式原始碼」、「說明文件」打包成ZIP壓縮檔，。
    - 檔案名稱請命名為 **HW3_你的學號.zip**。
        - **請不要試圖更換壓縮檔名字，違者一概視同缺交作業。**
    - 不需要繳交測試所使用的**測試檔案**、**輸出檔案**以及**storage資料夾**。
    - 目錄結構應該會如下圖：

```
❯ tree ./HW3_P76081043 -C
./HW3_P76081043
├── HW3_P76081043_report.html
└── P76081043.java
```

- 說明文件只接受 **pdf**、**html** 兩種格式。
    - 如果是使用 Markdown 撰寫，請利用 HackMD 轉換成 html。
    - 不得只放 HackMD 的連結網址。
    - 只接受由 HackMD 轉換出來的 html。



- 如果是用其他文件格式撰寫，一律轉換為 pdf。
- **請再次確認你的程式能夠正常執行，並能正確執行測試資料，且說明文件中有包含指定的內容。**
  - 程式無法編譯成功、執行失敗或是與測試輸出檔案驗證失敗，後續批改皆不進行，此作業以零分計算。
- 請將打包好的壓縮檔，上傳到Moodle平台，即可完成此次作業的繳交。

## 評分項目

### 程式部分

- **是否滿足上述的程式行為以及要求，若無法完全達成則本次作業以0分計算。**
- 程式執行的效率以及正確性。

### 報告部分

- 文件是否有包含指定的內容。
- 報告內容的完整度及正確性。
- 對系統資源觀察的程度以及呈現的說明內容。
- 說明整理資料作法的完整度及為什麼要這樣做。
- 可以寫出改進程式的過程，會有加分。

## 測試方法

本次測試助教會準備三種檔案，每個檔案大小都會超過4G，以此測試程式的各種行為並評定分數。
使用的測試環境中，會將實體記憶體限制在4G，而資料量將會大於4G。

- 第一個檔案之中，所有的指令皆為 **PUT**，助教會依照第一個檔案執行完成的時間，評估程式的新增、更新資料以及整理資料的效率。
  - `time ./P76081043 ./1.input`
  - 全部 **PUT** 指令的測試資料，**不需要產生輸出檔案**。
- 第二個檔案之中，所有的指令皆為 **GET**，助教會依照第二個檔案執行完成的時間，評估程式讀取資料的效率。
  - `time ./P76081043 ./2.input`
  - 產生：**2.output**
- 第三個檔案之中，包含所有的指令 **PUT**、**GET**、**SCAN**，助教會依照第三個檔案執行完成的時間，評估程式整體執行的效率。
  - `time ./P76081043 ./3.input`
  - 產生：**3.output**

第二個檔案以及第三個檔案中的指令，會包含取得第一個檔案之中，新增或是更新的資料，所以應該會有儲存暫存資料的地方。

## 範例測試資料

包含輸入檔案、輸出檔案。
實際進行測試時，使用的輸入檔案會大於實體記憶體。

https://drive.google.com/drive/folders/17B28qesYY9vLT0DK6UdEoWG-kTgqzP-3?usp=sharing
(https://drive.google.com/drive/folders/17B28qesYY9vLT0DK6UdEoWG-kTgqzP-3?usp=sharing)

交上來的程式碼，在編譯成功之後，助教會先使用範例檔案測試，如果執行失敗或是與輸出檔案驗證失敗，後續批改皆不進行，直接以零分計算。

## 注意事項

- 你可以使用任意程式語言撰寫作業，但助教只會用Ubuntu環境執行你的程式，因此建議使用Linux OS來撰寫此份作業。
- 你可以用任何OS（Windows、Windows Subsystem for Linux、Virtual Machine等）開發，但如同前項所述，助教只會在Ubuntu中執行你的程式。
- 你可以在虛擬機器（Virtual Machine）上撰寫程式及分析效能，但請在報告中註記你是使用虛擬機器，並提供關於該虛擬機器的基本資訊（像是：OS、vCore、Memory size等）。
- **請不要直接調用外部的函式庫來幫助你完成程式行為。**
- **助教在測試程式時，使用的測試資料會超過電腦實體記憶體大小。**
- **嚴格禁止互相抄襲程式碼，助教會進行程式碼比對，違者此次作業以零分計算。**
- **若有較高比例參考網路上的程式碼，請務必於報告中註明出處，並說明你是如何使用它以及對它做了多少修改。若此部分說明不完整，助教可能會請你親自來實驗室說明，你所撰寫之程式背後的行為。**