CJSON TUTORIAL

程式設計(二)-BOUNS

Due to 06/28 PM 11:59／授課老師：紀博文

一、基本資料

姓名：林育辰
系級：資工 111
學號：40771131H

二、檔案有哪些？

- **CJSON-master (資料夾)**
- **README.pdf**
- **test.c**
- **cJSON.c**
- **cJSON.h**

三、說明

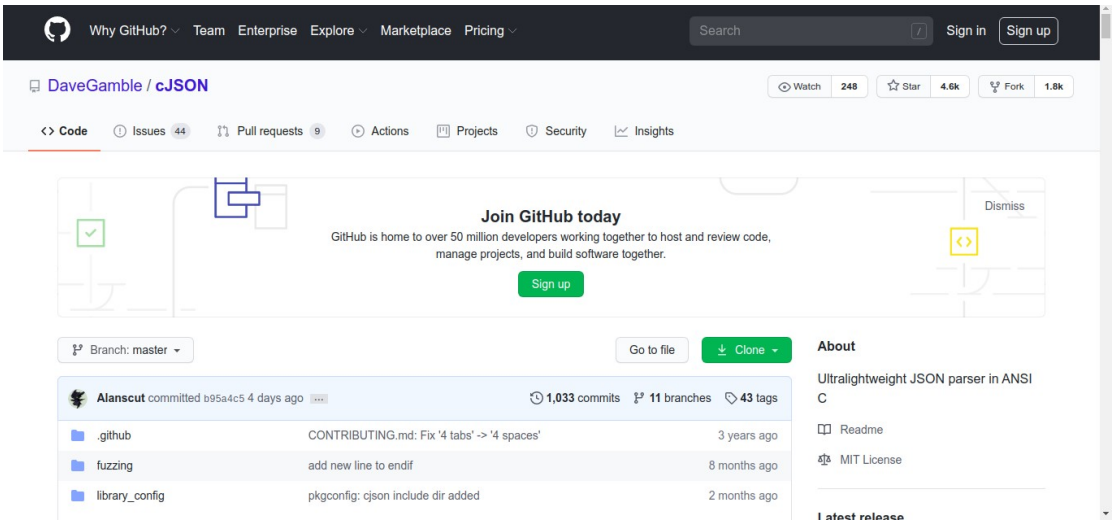
Json 是一種純文字的檔案格式，以文字為其基底儲存、傳送簡單的結構資料，此外還可以透過字串、物件、陣列等格式儲存任何資料，主要特點在於相容性高，格式簡單、好閱讀，且目前許多程式都支援 JSON，也支援了很多種的資料格式，在政府公開資料的網站也常見到他們的蹤跡，例如 PTX 的機場公開資料：https://ptx.transportdata.tw/MOTC/v2/Air/FIDS/Airport/Arrival

而 CJSON 則是 C 語言中一種可以用來剖析 JSON、建立 JSON 格式的一種函式庫。

四、下載方式

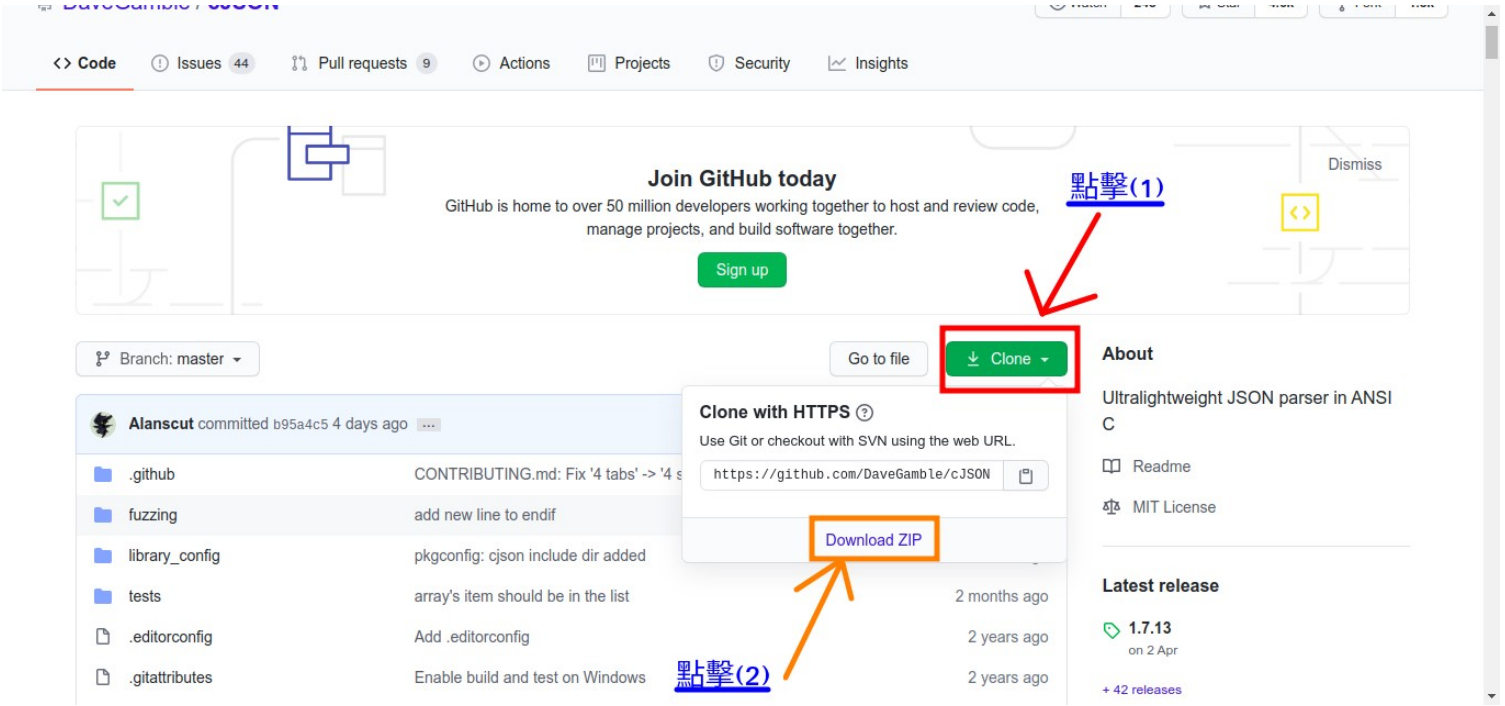
STEP 1: 前往 CJSON 的 github

URL: https://github.com/DaveGamble/cJSON



圖片 1: github 網站畫面

STEP 2: 點擊 Clone → Download ZIP



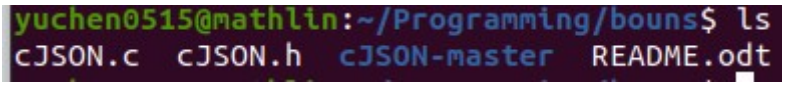
STEP 3: 解壓縮即可，放置你需要使用到他的檔案目錄下

五、Building

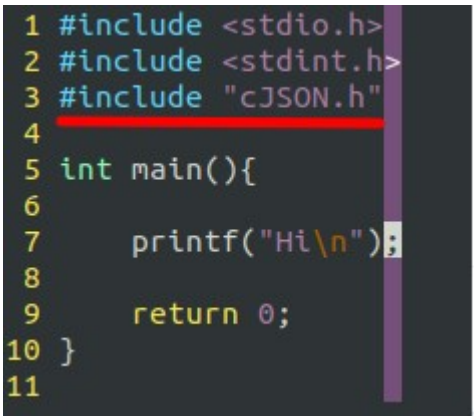
主要有兩種方式，一種是直接使用 CJSON.C 以及 CJSON.H 檔，自行 include 編譯即可，第二種則是進入到解壓縮的附錄執行，以下先示範第一種使用方式。

方法一

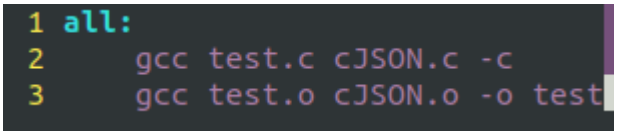
首先，你必須將解壓縮後的資料夾，裡面的 cJSON.c 以及.h 檔複製到需要使用的目錄下：



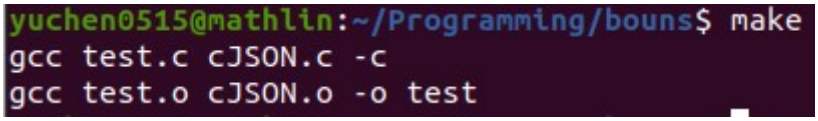
接著，在要使用 CJSON 的程式內，include 這份.h 檔：



這份測試檔案名稱為 test.c，或許你的 Makefile 內容應該如下：



接著輸入 $ make 即可：



方法二

首先，移動到 cJSON 的目錄下：

```
yuchen0515@mathlin:~/Programming/bouns$ ls
cJSON.c   cJSON-master   Makefile    test     test.o
cJSON.h   cJSON.o        README.odt   test.c
yuchen0515@mathlin:~/Programming/bouns$ cd cJSON-master/
yuchen0515@mathlin:~/Programming/bouns/cJSON-master$
```

接著輸入下列指令：

```
yuchen0515@mathlin:~/Programming/bouns/cJSON-master$ mkdir build
yuchen0515@mathlin:~/Programming/bouns/cJSON-master$ cd build
yuchen0515@mathlin:~/Programming/bouns/cJSON-master/build$ cmake ..
```

make 他，並安裝

```
yuchen0515@mathlin:~/Programming/bouns/cJSON-master/build$ make
```

安裝的時候都要注意權限的問題，最前面需要加上 sudo

```
yuchen0515@mathlin:~/Programming/bouns/cJSON-master/build$ sudo make install
```

接著就可以使用了，在需要使用到 cJSON 的程式內，請記得 include：

```
1 #include <stdio.h>
2 #include <stdint.h>
3 #include "cjson/cJSON.h"
4
5 int main(){
6
7     printf("Hi\n");
8
9     return 0;
10 }
11
```

一般的編譯方式即可：

```
yuchen0515@mathlin:~/Programming/bouns$ make
gcc test.c -o test
```

以上為兩種 build 方式的教學，若想要輕鬆簡便的方式可以採用第一種，在下列的範例使用中，都將使用第一種，你也可以選擇使用第二種執行你的程式！

六、CJSON 資料結構

在使用函式庫時，必須密切的去觀看.h 檔內用了什麼主要的結構，包含哪些函式，以及用法，首先我觀看 cJSON.h 的檔案，找到了 cJSON 最主要的結構如下：

```
/* The cJSON structure: */
typedef struct cJSON
{
    /* next/prev allow you to walk array/object chains. Alternatively, use GetArraySize/GetArrayItem/GetObjectItem */
    struct cJSON *next;
    struct cJSON *prev;
    /* An array or object item will have a child pointer pointing to a chain of the items in the array/object. */
    struct cJSON *child;

    /* The type of the item, as above. */
    int type;

    /* The item's string, if type==cJSON_String  and type == cJSON_Raw */
    char *valuestring;
    /* writing to valueint is DEPRECATED, use cJSON_SetNumberValue instead */
    int valueint;
    /* The item's number, if type==cJSON_Number */
    double valuedouble;

    /* The item's name string, if this item is the child of, or is in the list of subitems of an object. */
    char *string;
} cJSON;
```

檢查型態

cJSON 中的型態、狀態，有以下幾種型式：

型式	說明
cJSON_Invalid	代表無效的項目，如果不包含任何值就會遇到了，此外設定全部的 item 都為 0 也會得到一樣的結果。
cJSON_false	僅代表 false 的布林值，如果想要更廣泛的檢查布林值的狀態，你應該看 cJSON_IsBool。
cJSON_true	僅代表 true 的布林值，如果想要更廣泛的檢查布林值的狀態，你應該看 cJSON_IsBool。
cJSON_NULL	代表 null 的值。
cJSON_Number	代表為「數值」的值，如果數值超出的 int 的範圍，能使用 INT_MAX 或是 INT_MIN。

cJSON_String	代表字串的值，在 valuestring 會以 0 作為終止字元。
cJSON_Array	代表陣列的值，使用指標 child 去指向 cJSON 的 linked list 項目，去表示在陣列的值，元素連結的指標(鏈結串列的指標部份)是使用 next 和 prev(類似於 linux 的 linked list 的實作)，元素 prev.next == NULL 以及 next ==NULL。
cJSON_Object	代表物件的值，物件與陣劣是以同樣的方式儲存的，唯一不同的是在物件中的項目，他們的 keys(鍵值)是使用 string。
cJSON_Raw	代表以 valuestring 以 0 為終止字元儲存的任意型態 cJSON，舉例來說，這可以避免印出相同的 static JSON，以增進效能，在 parse 時，不會為其創建型態，同時也不會檢查他是否為有效的 JSON 樣式。
Additional flags	
cJSON_IsReference	具體指出項目中的 child 指標是只可存取其值而不可修改的(類似 const 唯讀)，因此 cJSON_Delete 和其他的函式都只能釋放(deallocate)這個項目(item)，而不是他的 child or valuestring。
cJSON_StringIsConst	意味著 string 指向一個常數的字串，也代表 cJSON_Delete 和其他的函式將不會嘗試用 string 的方式 deallocate 他

- 在使用的時候要注意，如果你已經添加任何東西在 item(項目)或是陣列、項目中，則不能使用 cJSON_Delete 刪除。
- 添加他到陣列或物件中，會轉變他的所有權，讓他能夠在陣列或物件被刪除時，他也跟著被刪除
- 不需要人工 free 掉先前的 valuestring，你能夠使用 cJSON_SetValuestring 去改變 cJSON_String 裡面的 valuestring

檢查型態

- null：可以用 cJSON_CreateNull 建立
- booleans：能夠用 cJSON_CreateTrue 或 cJSON_CreateFalse 或 cJSON_CreateBool 建立

- numbers：能夠使用 cJSON_CreateNumber 建立，這將會同時設定 valuedouble 以及 valueint，如果數值超出範圍，INT_MAX 或 INT_MIN 能使用在 valueint 內
- strings：能夠使用 cJSON_CreateString(複製字串)或 cJSON_CreateStringReference(直接指向字串，表示 valuestring 不能使用 cJSON_Delete 刪除，你無法決定他在程式中的生命周期，這對於固定的數是很有幫助的)

陣列(array)

- cJSON_CreateArray：建造一個空的陣列
- cJSON_CreateArrayReference：建造一個陣列，不過仍然無法直接刪除他，控制他的內容
- cJSON_AddItemToArray：添加 item 在最尾巴
- cJSON_AddItemReferenceToArray：能添加 reference 在另一個 item、陣列或字串中，不過這同時代表 cJSON_Delete 將無法刪除該 item 的 chile 或 valuestring
- cJSON_InsertItemInArray：這可以插入一個 item，基於給定的 index，並將所有存在的 item 全部往右邊移動
- cJSON_DetachItemFromArray：如果你從陣列中提取 item 出來，想要持續使用這個 item 你可以用這個函式，他將回傳這個被提取出來的 item，不過你要記得為他用一個 Point 指向他，否則你可能會發生 memory leak 的慘劇
- cJSON_DeleteItemFromArray：這作用就像 cJSON_DetachItemFromArray 一樣，不過如果要刪除 detached 的 item 是使用 cJSON_Delete
- cJSON_ReplaceItemInArray：用一個 index 或 cJSON_ReplaceItemViaPointer 讓一個指標指向一個元素，如果失敗的話他會回傳 0，他會刪除並新的 item 取代
- cJSON_GetArraySize：可以得到陣列的大小
- cJSON_GetArrayItem：給定一個 index，能得到該 index 的元素

物件(object)

- cJSON_CreateObject：創造一個空的 object
- cJSON_CreateObjectReference：也能創建一個 object，不過你無法「擁有」這個內容的控制權限，此外，你也不能使用 cJSON_Delete 去刪除他(簡單來說就是唯讀的概念)

- cJSON_AddItemToObject：添加一個 item 到 object 中
- cJSON_AddItemToObjectCS：添加一個 item 到 object(透過 name，可能是一個常數或 reference(item 的鍵值、在 cJSON 結構中的 string)，這不能使用 cJSON_Delete free(釋放)
- cJSON_AddItemReferenceToArray：一個元素能被當成 reference 加入到另一個 object、陣列或字串中，看到 reference 就表示他同時不能使用 cJSON_Delete 他 item 中的 chile 或 valuestring
- cJSON_DetachItemFromObjectCaseSensitive：想要從 object 中取出 item，可以使用這個，他將回傳取出的 item，同時要注意，記得使用一個指標指向他，否則仍會發生 memory leak 的慘劇
- cJSON_DeleteItemFromObjectCaseSensitive：如果要刪除 item 可使用，他的運作方式就像 cJSON_DetachItemFromObjectCaseSensitive 隨著 cJSON_Delete
- cJSON_ReplaceItemInObjectCaseSensitive：透過鍵值或 cJSON_ReplaceItemViaPointer 得到一個指向元素的指標，如果失敗會回傳 0，他將把原始的東西丟掉，插入新的東西取代
- cJSON_GetArraySize：你能使用這個去得到 object 的大小(之所以函式名稱也叫 Array 是因為 object 的運作其實跟 Array 沒兩樣，上述說明中有提到。
- cJSON_GetObjectItemCaseSensitive：如果想要存取一個在 object 中的 item 可以使用這個

七、剖析 JSON

- 如果給定一個字串(以\0 作為終止字元)，你可以使用：
  *cJSON *json = cJSON_Parse(sring);*
- 給定一個字串，無論是不是以\0 作為終止字元都可以使用
  *cJSON *json = cJSON_ParseWithLength(string, buffer_length);*
- 當你 parse 一個 JSON 以及分配一段記憶體給 cJSON 的 item 時，一旦成功並得到回傳的指標時，你也可以使用 cJSON_Delete 釋放掉這塊記憶體
- 如果輸入字串時指向錯誤的位置，能夠使用 cJSON_GetErrorPtr 存取
- 需要更多選項，可以使用
  *cJSON_ParseWithOpts(const char *value, const char **return_parse_end, cJSON_bool require_null_terminated)* 其

中 return_parse_end 回傳一個指標指向 JSON 的末端，或是發生錯誤的位置(也因此他可以直接取代 cJSON_GetErrorPtr 的功能)，此外 require_null_terminated，如果設為 1 代表錯誤

- 如果想要更多 options 你要多給 buffer length 的參數，並使用：*cJSON_ParseWithLengthOpts(const char *value, size_t buffer_length, const char **return_parse_end, cJSON_bool require_null_terminated)*

八、印出 JSON

詳細使用方式，其實觀察 example_code 會理解的比較快，若想執行看看可輸入以下指令： [ps. 範例檔名稱為 test.c]

```
yuchen0515@mathlin:~/Programming/bouns$ make
gcc test.c cJSON.c -c
gcc test.o cJSON.o -o test
```

執行結果，裡面只是放了自己的名字資訊
第一條是專題的 deadline，第二第三則是實際 presentation 的日期與時間
不過這邊只是要展示如何用 CJSON 創建 JSON 的物件，以及印出而已

```
yuchen0515@mathlin:~/Programming/bouns$ ./test
{
        "name": "Yu-Chen Lin",
        "project_time": [{
                        "date": 628,
                        "time": 2359
                }, {
                        "date": 629,
                        "time": 1615
                }, {
                        "date": 629,
                        "time": 1630
                }]
}
```

關於函式的說明可見檔案內容~

在 test.c 的內容中，create_project_with_deadline 函式是示範創建 JSON 並印出，其中包含很多檢查錯誤的環節，goto 像是組合語言中的 label 只是拿來省行數的。

supports 則拿來檢查 JSON 的元素有沒有符合的內容，如果 Yu-Chen Lin 的專題報告時間，是在 6/29 的 16:15，就輸出狀態 1，若 date 以及 time 任一個不為數字，則為狀態 0，其他的狀態因為到了第 98 行後沒有檢查了，則也會進入到 end label，傳回狀態 0

程式相關內容如附圖：

```c
#include <stdio.h>
#include <stdint.h>
#include "cJSON.h"


//示範函式
char *create_project_with_deadline(){

    const uint32_t deadline_and_presentation_time[3][2] = {
        {628, 2359},
        {629, 1615},
        {629, 1630}
    };

    //最終回傳的指標位置
    char *string = NULL;
    size_t index = 0;
    cJSON *project_time = NULL;


    //創建一個object，用一個cJSON結構的指標去接他
    cJSON *project= cJSON_CreateObject();

    //加入名字的資訊在object內
    //如果失敗就回傳(加上刪除)
    if (cJSON_AddStringToObject(project , "name", "Yu-Chen Lin") == NULL)
        goto end;

    //第2層資訊，整體為project的時間和截止日期，簡單來說就是第2層元素
    project_time = cJSON_AddArrayToObject(project , "project_time");
    if (project_time == NULL)
        goto end;

    for (index = 0; index < (sizeof(deadline_and_presentation_time) / (2 * sizeof(int32_t))); index++){
        cJSON *project_time_ = cJSON_CreateObject();

        if (cJSON_AddNumberToObject(project_time_, "date", deadline_and_presentation_time[index][0]) == NULL)
            goto end;

        if (cJSON_AddNumberToObject(project_time_, "time", deadline_and_presentation_time[index][1]) == NULL)
            goto end;

        cJSON_AddItemToArray(project_time, project_time_);
    }

    //回傳
    string = cJSON_Print(project);
    if (string == NULL)
        fprintf(stderr, "Failed to print monitor.\n");

end:
    cJSON_Delete(project);
    return string;
}

/* return 1 if the project supports presentation time at 16:15, 0 otherwise */
int32_t supports(const char* const project){

    const cJSON *project_time_= NULL;
    const cJSON *project_time = NULL;
    const cJSON *name = NULL;
    int32_t status = 0;

    //先把傳入的JSON字串剖析程cJSON的結構
    cJSON *project_json = cJSON_Parse(project);

    //當然要檢查有沒有成功，失敗就傳出錯誤訊息
    if (project_json == NULL){
        const char *error_ptr = cJSON_GetErrorPtr();
        if (error_ptr != NULL){
            fprintf(stderr, "Error before: %s\n", error_ptr);
        }
        status = 0;
        goto end;
```

```c
    }

    //檢查project的名字
    name = cJSON_GetObjectItemCaseSensitive(project_json, "name");
    if (cJSON_IsString(name) && (name->valuestring != NULL)){
        printf("Checking project \"%s\"\n", name->valuestring);
    }

    project_time = cJSON_GetObjectItemCaseSensitive(project_json, "project_time");

    //尋訪陣列中每一個
    cJSON_ArrayForEach(project_time_, project_time){
        cJSON *date = cJSON_GetObjectItemCaseSensitive(project_time_, "date");
        cJSON *time = cJSON_GetObjectItemCaseSensitive(project_time_, "time");

        if (!cJSON_IsNumber(date) || !cJSON_IsNumber(time)){
            status = 0;
            goto end;
        }

        if ((date->valuedouble == 629) && (time->valuedouble == 1615)){
            status = 1;
            goto end;
        }
    }

end:
    cJSON_Delete(project_json);
    return status;
}

int main(){

    char *ptr = create_project_with_deadline();
    printf("%s\n", ptr);

    int32_t status = supports(ptr);
     printf("status: %d\n", status);

     return 0;
}
```

右圖為執行結果→

```
yuchen0515@mathlin:~/Programming/bouns$ ./test
{
        "name": "Yu-Chen Lin",
        "project_time": [{
                        "date": 628,
                        "time": 2359
                }, {
                        "date": 629,
                        "time": 1615
                }, {
                        "date": 629,
                        "time": 1630
                }]
}
Checking project "Yu-Chen Lin"
status: 1
```

事實上使用方式淺顯易懂
在實際印出、創建的部份
則使用 example_code 代替

相信透過註解，讀者能輕鬆學會
CJSON 的用法！