

# cJSON使用详细教程 | 一个轻量级C语言JSON解析器

原创Mculover666已于 2022-04-15 14:22:31 修改65454收藏 885

文章标签：cjson

## 1. JSON与cJSON

### JSON —— 轻量级的数据格式

JSON 全称 JavaScript Object Notation，即 JS对象简谱，是一种轻量级的数据格式。

它采用完全独立于编程语言的文本格式来存储和表示数据，语法简洁、层次结构清晰，易于人阅读和编写，同时也易于机器解析和生成，有效的提升了网络

### JSON语法规则

JSON对象是一个无序的"名称/值"键值对的 集合🔗：

- 以"{"开始，以"}"结束，允许嵌套使用；
- 每个名称和值成对出现，名称和值之间使用":"分隔；
- 键值对之间用","分隔
- 在这些字符前后允许存在无意义的空白符；

对于键值，可以有如下值：

- 一个新的json对象
- 数组：使用"["和"]"表示
- 数字：直接表示，可以是整数，也可以是浮点数
- 字符串：使用引号" "表示
- 字面值：false、null、true中的一个(必须是小写)

示例如下：

```
1 {
2     "name": "mculover666",
3     "age": 22,
4     "weight": 55.5,
5     "address":
6     {
7         "country": "China",
8         "zip-code": 111111
9     },
10    "skill": ["c", "Java", "Python"],
11    "student": false
12 }
```

### cJSON

cJSON是一个使用C语言编写的JSON数据解析器，具有超轻便，可移植，单文件的特点，使用MIT开源协议。

cJSON项目托管在Github上，仓库地址如下：

https://github.com/DaveGamble/cJSON

使用Git命令将其拉取到本地：

 Mculover666

关注

👍 261

🗨️

⭐ 885

📄

💬 35

🔗

从Github拉取cJSON源码后，文件非常多，但是其中cJSON的源码文件只有两个：

- cJSON.h
- cJSON.c

使用的时候，只需要将这两个文件复制到工程目录，然后包含头文件 cJSON.h 即可，如下：

```
1 #include "cJSON.h"
```

## 2. cJSON数据结构和设计思想

cJSON的设计思想从其数据结构上就能反映出来。

cJSON使用cJSON结构体来表示一个JSON数据，定义在 cJSON.h 中，源码如下：

```
1  /* The cJSON structure: */
2  typedef struct cJSON
3  {
4      /* next/prev allow you to walk array/object chains. Alternatively, use GetArraySize/GetArrayItem/GetObjectItem */
5      struct cJSON *next;
6      struct cJSON *prev;
7      /* An array or object item will have a child pointer pointing to a chain of the items in the array/object. */
8      struct cJSON *child;
9
10     /* The type of the item, as above. */
11     int type;
12
13     /* The item's string, if type==cJSON_String and type == cJSON_Raw */
14     char *valuestring;
15     /* writing to valueint is DEPRECATED, use cJSON_SetNumberValue instead */
16     int valueint;
17     /* The item's number, if type==cJSON_Number */
18     double valuedouble;
19
20     /* The item's name string, if this item is the child of, or is in the list of subitems of an object. */
21     char *string;
22 } cJSON;
```

cJSON的设计很巧妙。

首先，它不是将一整段JSON数据抽象出来，而是将其中的一条JSON数据抽象出来，也就是一个键值对，用上面的结构体 struct cJSON 来表示，其中用并

- String：用于表示该键值对的名称；
- type：用于表示该键值对中值的类型；
- valuestring：如果键值类型(type)是字符串，则将该指针指向键值；
- valueint：如果键值类型(type)是整数，则将该指针指向键值；
- valuedouble：如果键值类型(type)是浮点数，则将该指针指向键值；

其次，一段完整的JSON数据中由很多键值对组成，并且涉及到键值对的查找、删除、添加，所以使用链表来存储整段JSON数据，如上面的代码所示：

- next 指针：指向下一个键值对
- prev 指针指向上一个键值对

最后，因为JSON数据支持嵌套，所以一个键值对的值会是一个新的JSON数据对象（一条新的链表），也有可能是一个数组，方便起见，在cJSON中，数组

在键值对结构体中，当该键值对的值是一个嵌套的JSON数据或者一个数组时，由 child 指针指向该条新链表。



Mculover666

关注

👍 261



🌟 885



💬 35



## 封装方法

封装JSON数据的过程，其实就是创建链表和向链表中添加节点的过程。

首先来讲述一下链表中的一些术语：

- 头指针：指向链表头结点的指针；
- 头结点：不存放有效数据，方便链表操作；
- 首节点：第一个存放有效数据的节点；
- 尾节点：最后一个存放有效数据的节点；

明白了这几个概念之后，我们开始讲述创建一段完整的JSON数据，即如何创建一条完整的链表。

- ① 创建头指针：

```
1 | cJSON* cJSON_test = NULL;
```

- ② 创建头结点，并将头指针指向头结点：

```
1 | cJSON_test = cJSON_CreateObject();
```

- ③ 尽情的向链表中添加节点：

```
1 | cJSON_AddNullToObject(cJSON * const object, const char * const name);
2 |
3 | cJSON_AddTrueToObject(cJSON * const object, const char * const name);
4 |
5 | cJSON_AddFalseToObject(cJSON * const object, const char * const name);
6 |
7 | cJSON_AddBoolToObject(cJSON * const object, const char * const name, const cJSON_bool boolean);
8 |
9 | cJSON_AddNumberToObject(cJSON * const object, const char * const name, const double number);
10 |
11 | cJSON_AddStringToObject(cJSON * const object, const char * const name, const char * const string);
12 |
13 | cJSON_AddRawToObject(cJSON * const object, const char * const name, const char * const raw);
14 |
15 | cJSON_AddObjectToObject(cJSON * const object, const char * const name);
16 |
17 | cJSON_AddArrayToObject(cJSON * const object, const char * const name);
```

## 输出JSON数据

上面讲述，一段完整的JSON数据就是一条长长的链表，那么，如何打印出这段JSON数据呢？

cJSON提供了一个API，可以将整条链表中存放的JSON信息输出到一个字符串中：

```
1 | (char *) cJSON_Print(const cJSON *item);
```

使用的时候，只需要接收该函数返回的指针地址即可。

## 封装数据和打印数据示例

单纯的讲述方法还不够，下面用一个例子来说明，封装出开头给出的那段JSON数据：

```
1 | #include <stdio.h>
2 | #include "cJSON.h"
```



Mculover666

关注

👍 261



🌟 885



💬 35

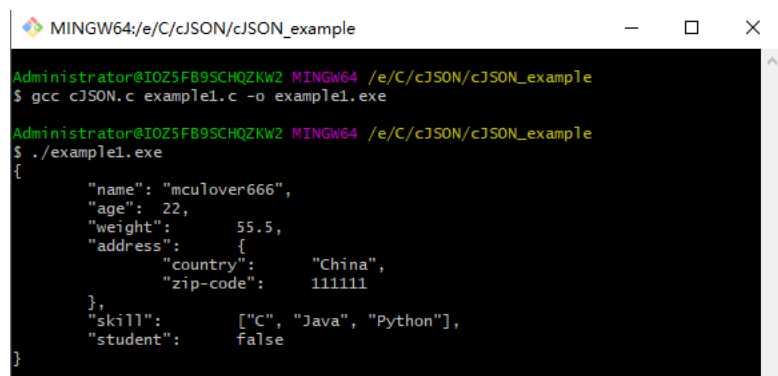


```
5 {
6     cJSON* cJSON_test = NULL;
7     cJSON* cJSON_address = NULL;
8     cJSON* cJSON_skill = NULL;
9     char* str = NULL;
10
11     /* 创建一个JSON数据对象(链表头结点) */
12     cJSON_test = cJSON_CreateObject();
13
14     /* 添加一条字符串类型的JSON数据(添加一个链表节点) */
15     cJSON_AddStringToObject(cJSON_test, "name", "mculover666");
16
17     /* 添加一条整数类型的JSON数据(添加一个链表节点) */
18     cJSON_AddNumberToObject(cJSON_test, "age", 22);
19
20     /* 添加一条浮点类型的JSON数据(添加一个链表节点) */
21     cJSON_AddNumberToObject(cJSON_test, "weight", 55.5);
22
23     /* 添加一个嵌套的JSON数据(添加一个链表节点) */
24     cJSON_address = cJSON_CreateObject();
25     cJSON_AddStringToObject(cJSON_address, "country", "China");
26     cJSON_AddNumberToObject(cJSON_address, "zip-code", 111111);
27     cJSON_AddItemToObject(cJSON_test, "address", cJSON_address);
28
29     /* 添加一个数组类型的JSON数据(添加一个链表节点) */
30     cJSON_skill = cJSON_CreateArray();
31     cJSON_AddItemToArray(cJSON_skill, cJSON_CreateString( "C" ));
32     cJSON_AddItemToArray(cJSON_skill, cJSON_CreateString( "Java" ));
33     cJSON_AddItemToArray(cJSON_skill, cJSON_CreateString( "Python" ));
34     cJSON_AddItemToObject(cJSON_test, "skill", cJSON_skill);
35
36     /* 添加一个值为 False 的布尔类型的JSON数据(添加一个链表节点) */
37     cJSON_AddFalseToObject(cJSON_test, "student");
38
39     /* 打印JSON对象(整条链表)的所有数据 */
40     str = cJSON_Print(cJSON_test);
41     printf("%s\n", str);
42
43     return 0;
44 }
45
```

编译运行：

```
1 | gcc cJSON.c example1.c -o example1.exe
```

实验结果如图：



该JSON数据链表的关系如图：



Mculover666

关注

261

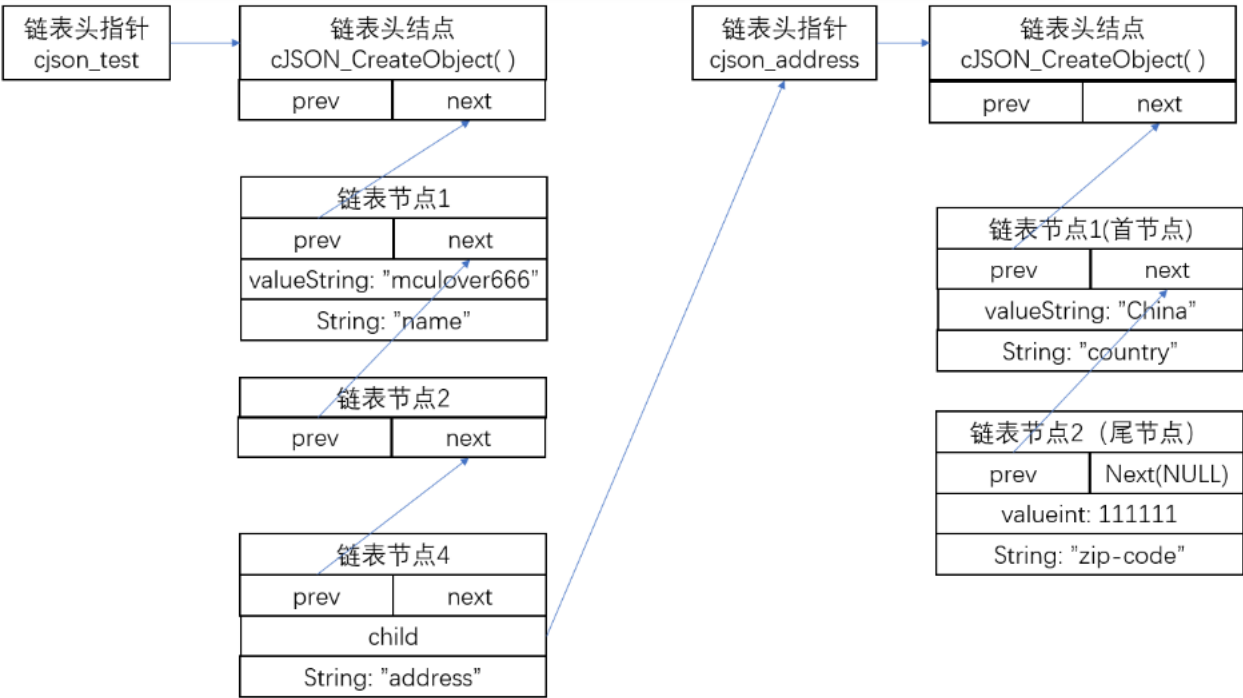


885



35





## 4. cJSON数据解析

### 解析方法

解析JSON数据的过程，其实就是剥离一个一个链表节点(键值对)的过程。

解析方法如下：

- ① 创建链表头指针：

```
1 | cJSON* cjson_test = NULL;
```

- ② 解析整段JSON数据，并将链表头结点地址返回，赋值给头指针：

解析整段数据使用的API只有一个：

```
1 | (cJSON *) cJSON_Parse(const char *value);
```

- ③ 根据键值对的名称从链表中取出对应的值，返回该键值对(链表节点)的地址

```
1 | (cJSON *) cJSON_GetObjectItem(const cJSON * const object, const char * const string);
```

- ④ 如果JSON数据的值是数组，使用下面的两个API提取数据：

```
1 | (int) cJSON_GetArraySize(const cJSON *array);
2 | (cJSON *) cJSON_GetArrayItem(const cJSON *array, int index);
```

### 解析示例

下面用一个例子来说明如何解析出开头给出的那段JSON数据：

```
1 | #include <stdio.h>
2 | #include "cJSON.h"
```

Mculover666 关注

👍 261

👎

🌟 885

📧

💬 35

🔗

```
5  "{
6  \ "name\ ": \ "mculover666\ ", \
7  \ "age\ ": 22, \
8  \ "weight\ ": 55.5, \
9  \ "address\ ": \
10     { \
11         \ "country\ ": \ "China\ ", \
12         \ "zip-code\ ": 111111 \
13     }, \
14     \ "skill\ ": [ \ "c\ ", \ "Java\ ", \ "Python\ " ], \
15     \ "student\ ": false \
16 }";
17
18 int main(void)
19 {
20     cJSON* cJSON_test = NULL;
21     cJSON* cJSON_name = NULL;
22     cJSON* cJSON_age = NULL;
23     cJSON* cJSON_weight = NULL;
24     cJSON* cJSON_address = NULL;
25     cJSON* cJSON_address_country = NULL;
26     cJSON* cJSON_address_zipcode = NULL;
27     cJSON* cJSON_skill = NULL;
28     cJSON* cJSON_student = NULL;
29     int skill_array_size = 0, i = 0;
30     cJSON* cJSON_skill_item = NULL;
31
32     /* 解析整段JSON数据 */
33     cJSON_test = cJSON_Parse(message);
34     if(cJSON_test == NULL)
35     {
36         printf("parse fail.\n");
37         return -1;
38     }
39
40     /* 依次根据名称提取JSON数据（键值对） */
41     cJSON_name = cJSON_GetObjectItem(cJSON_test, "name");
42     cJSON_age = cJSON_GetObjectItem(cJSON_test, "age");
43     cJSON_weight = cJSON_GetObjectItem(cJSON_test, "weight");
44
45     printf("name: %s\n", cJSON_name->valuelstring);
46     printf("age:%d\n", cJSON_age->valueint);
47     printf("weight:%.1f\n", cJSON_weight->valuedouble);
48
49     /* 解析嵌套json数据 */
50     cJSON_address = cJSON_GetObjectItem(cJSON_test, "address");
51     cJSON_address_country = cJSON_GetObjectItem(cJSON_address, "country");
52     cJSON_address_zipcode = cJSON_GetObjectItem(cJSON_address, "zip-code");
53     printf("address-country:%s\naddress-zipcode:%d\n", cJSON_address_country->valuelstring, cJSON_address_zipcode->valueint);
54
55     /* 解析数组 */
56     cJSON_skill = cJSON_GetObjectItem(cJSON_test, "skill");
57     skill_array_size = cJSON_GetArraySize(cJSON_skill);
58     printf("skill:[");
59     for(i = 0; i < skill_array_size; i++)
60     {
61         cJSON_skill_item = cJSON_GetArrayItem(cJSON_skill, i);
62         printf("%s,", cJSON_skill_item->valuelstring);
63     }
64     printf("\b]\n");
65
66     /* 解析布尔型数据 */
67     cJSON_student = cJSON_GetObjectItem(cJSON_test, "student");
68     if(cJSON_student->valueint == 0)
```



Mculover666

关注

261



885



35

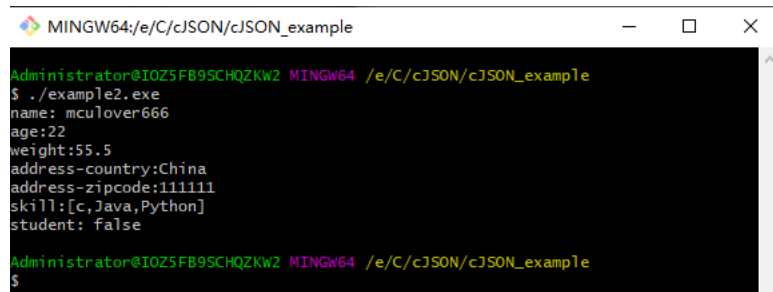


```
71     }
72     else
73     {
74         printf("student:error\n");
75     }
76
77     return 0;
78 }
```

编译：

```
1 | gcc cJSON.c example2.c -o example2.exe
```

运行结果如图：



```
Administrator@IO25FB9SCHQZKw2 MINGW64 /e/C/cJSON/cJSON_example
$ ./example2.exe
name: mculover666
age:22
weight:55.5
address-country:China
address-zipcode:111111
skill:[c,Java,Python]
student: false
Administrator@IO25FB9SCHQZKw2 MINGW64 /e/C/cJSON/cJSON_example
$
```

## 注意事项

在本示例中，因为我提前知道数据的类型，比如字符型或者浮点型，所以我直接使用指针指向对应的数据域提取，在实际使用时，如果提前不确定数据类型，再从对应的数据域中提取数据。

## 5. cJSON使用过程中的内存问题

### 内存及时释放

cJSON的所有操作都是基于链表的，所以cJSON在使用过程中大量的使用 malloc 从堆中分配动态内存的，所以在用完之后，应当及时调用下面的函数，；数也可用于删除某一条数据：

```
1 | (void) cJSON_Delete(cJSON *item);
```

注意：该函数删除一条JSON数据时，如果有嵌套，会连带删除。

### 内存钩子

cJSON在支持自定义malloc函数和free函数，方法如下：

- ① 使用 cJSON\_Hooks 来连接自定义malloc函数和free函数：

```
1 typedef struct cJSON_Hooks
2 {
3     /* malloc/free are CDECL on Windows regardless of the default calling convention of the compiler, so ensure the hooks allow pass
4     void *(CJSON_CDECL *malloc_fn)(size_t sz);
5     void (CJSON_CDECL *free_fn)(void *ptr);
6 } cJSON_Hooks;
```

- ② 初始化钩子cJSON\_Hooks

```
1 | (void) cJSON_InitHooks(cJSON_Hooks* hooks);
```



Mculover666

关注

261



885



35



🔗 文章知识点与官方知识档案匹配，可进一步学习相关知识

C技能树 > 首页 > 概览 90406 人正在系统学习中



Mculover666

微信公众号 >

凭着与生俱来对嵌入式的热情专注于这个领域

“相关推荐”对你有帮助？

-  非常有帮助
-  有帮助
-  一般
-  没帮助
-  非常没帮助

关于我们 招贤纳士 商务合作 寻求报道

400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心 家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 账号管理规范 版权与免责声明 版权申诉 出版物许可证 营业执照

©1999-2022北京创新乐知网络技术有限公司



Mculover666

关注

👍 261



★ 885



💬 35

