c一、JOSN的概念：
json指的是JavaScript对象表示法（JavaScript Object Notation）
josn是轻量级的文本数据交换格式
json独立于语言
json具有自我描述性，更易理解
json 使用 JavaScript 语法来描述数据对象，但是 json 仍然独立于语言和平台。json 解析器和 json 库支持许多不同的编程语言。

json的语法规则

json语法简单来说就是四条

数据在名称/值对中
数据由逗号分隔
花括号保存对象
方括号保存数据
json名称/值对

josn数据的书写格式是：名称：值，这样的一对，即名称在前，该名称的值在冒号后面。例如：

“virteNBName”:“virt1”

这里的名称是“virteNBName”，值是“virt1”，他们均是字符串

名称和值的类型可以有以下几种：

数字（整数或浮点数）
字符串（在双引号中）
逻辑值（true或false）
数组（在方括号中）
对象（在花括号中）
null
json数据由逗号分隔

“virteNBName”:“virt1”, “virteNBNum”:5, “begineNBID”:0这几个对象之间就是使用逗号分隔。

数组内的对象之间当然也是要有逗号分隔。只要是对象之间，分隔就是用逗号，但是，要注意，对象结束的时候，不要加逗号。数组内也是，例如：

```
[
    {"eRANName":"eNB1", "eRANID":3002, "ctlPort":36412, "dataPort":2152},
    {"eRANName":"eNB2", "eRANID":10000, "ctlPort":36412, "dataPort":2152},
]

上面这个就是错误的，因为在数组中，两个对象之间需要逗号，但是到这个数组末尾了，不需要加逗号了。
————————————————
版权声明：本文为CSDN博主「makunIT」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。
原文链接：https://blog.csdn.net/makunIT/article/details/107199000
```

josn花括号保存对象

对象可以包含多个名称/值对，如

{“eRANName”:“eNB1”, “eRANID”:3002, “ctlPort”:36412, “dataPort”:2152}

这一点也容易理解，与这条JavaScript语句等价：

“eRANName” = “eNB1”
“eRANID” = 3002
“ctlPort” = 36412
“dataPort” = 2152

josn方括号保存数组

数组可包含多个对象:

```
"eRAN":[
    {"eRANName":"eNB1", "eRANID":3002, "ctlPort":36412, "dataPort":2152},
    {"eRANName":"eNB2", "eRANID":10000, "ctlPort":36412, "dataPort":2152}
]
```

在上面的例子中，对象“eRAN” 是包含个对象的数组，每个对象代表一条基站的记录。

最后我们要知道json文件的类型为“.json”

二、cjosn的介绍
cJSON是一个超轻巧，携带方便，单文件，简单的可以作为ANSI-C标准的JSON解析器。

那么我们为什么选择cJSON来解析JSON字符串那？因为简洁又简单，而且效率又快，cJSON工程文件也非常简单，仅一个.C文件和一个.h文件！并且文件体积大小不到K。源码思路也非常清晰，也非常适合研究。

我们可以通过此链接下载cJOSN：https://sourceforge.net/projects/cjson/

当我们下载好cJSON只需要把.C文件和.h文件包含文件拷贝到我们工程目录下，我们就可以使用了。

cJSON的核心结构体就是一个cJSON，理解了这个结构体，基本上对cJSON的使用就有了个基本概念了，该结构体具体定义如下:

```
typedef struct cJSON {

        struct cJSON*next,*prev;            /* 遍历数组或对象链的前向或后向链表指针*/

        struct cJSON *child;                    /*数组或对象的孩子节点*/

        int type;                                       /* key的类型*/

        char *valuestring;                      /*字符串值*/

        int valueint;                                   /* 整数值*/

        double valuedouble;                     /* 浮点数值*/

        char *string;                                   /* key的名字*/

} cJSON;
```

说明:

cJSON是使用链表来存储数据的，其访问方式很像一棵树。每一个节点可以有兄弟节点，通过next/prev指针来查找，它类似双向链表；每个节点也可以有孩子节点，通过child指针来访问，进入下一层。只有节点是对象或数组时才可以有孩子节点。

type是键（key）的类型，一共有种取值，分别是：False，True,null，Number，String，Array，Object。
若是Number类型，则valueint或valuedouble种存储着值，如期望的是int，则访问valueint，如期望的是double，则访问valuedouble，可以得到值。
若是String类型的，则valuestring中存储着值，可以访问valuestring得到值。

string中存放的是这个节点的名字，可以理解位key的名称。

三、 cJSON常用函数简介
1、cJSON *cJSON_CreateObject（）；
创建一个json对象，返回一个cJSON结构体类型的指针。

2、cJSON *cJSON_CreateArray();
创建一个数组对象，返回一个cJSON结构体类型的指针。

3、cJSON *cJSON_CreateString(const char *string);
创建一个字符串对象，传入一个char *类型的字符串，返回一个cJSON结构体类型的指针。

4、void cJSON_AddItemToArray(cJSON *array, cJSON *item);
向数组对象中添加一个元素，传入参数array为cJSON *结构体类型的指针，为数组对象，item为添加如数字对象中的对象指针。

5、void cJSON_AddItemToObject(cJSON *object, const char *string, cJSON *item);
向json对象中添加一对元素，object为json对象，string为加入一对元素中的name，item为加入一对元素中的value。

6、char *cJSON_Print(cJSON *item);
将一个cJSON结构体代表的json对象转换为一个json格式的字符串。

7、void cJSON_Delete(cJSON *c)
释放一个cJSON对象占用的内存空间。

四、使用cJSON解析JSON格式
首先我们从简单的开始解析，正所谓万丈高楼起于平地嘛。
1、解析一个键值对/ 名称值对

首先是一个简单的键值对字符串，要解析的目标如下：
{“firstName”:“Brett”}

要进行解析，也就是分别获取到键与值得内容。我们很容易就能看出键为firstName， 值为Brett， 可是，使用cJSON怎么解析呢?

对于这个简单得例子，只需要调用cJSON得三个接口函数就可以实现解析了，这三个函数得原型如下：
cJSON*cJSON_Parse(const char *value);

cJSON*cJSON_GetObjectItem(cJSON *object,const char *string);

void cJSON_Delete(cJSON *c);

下面按解析过程来描述一次：

首先调用cJSON_Parse（）函数，解析JSON数据包，并按照cJSON结构体得结构序列化整个数据包。使用该函数会通过malloc（）函数在内存中开辟一个空间，使用完成需要手动释放。
cJSON*root=cJSON_Parse(json_string);

调用cJSON_GetObjectItem()函数，可从cJSON结构体中查找某个子节点名称（键名称），如果查找成功可把该子节点序列化到cJSON结构体中。
cJSON*item=cJSON_GetObjectItem(root,“firstName”);

如果需要使用cJSON结构体得内容，可以通过cJSON结构体中的valueint和valuestring取出有价值的内容（即键的值）

本例子中，我们直接访问 item->valuestring 就获取到 “Brett” 的内容了。

通过cJSON_Delete()，释放cJSON_Parse()分配出来的内存空间。
cJSON_Delete(root);

这样就完成了一次cJSON接口调用，实现了解析工作。

以上我们用到了cJSON中的常见的函数，我们再来分析以下这三个函数吧。

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

作用：将一个JSON数据包，按照cJSON结构体序列化整个数据包，并在堆中开辟一块内存存储cJSON结构体.。

返回值：成功返回一个指向内存块中的cJSON的指针，失败返回NULL。

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

作用：获取JSON字符串字段值
返回值：成功返回一个指向cJSON类型的结构体指针，失败返回NULL

```
char  *cJSON_Print(cJSON *item);
```

作用：将cJSON数据解析成JSON字符串，并在堆中开辟一块char *的内存空间存储JSON字符串

返回值：成功返回一个char*指针该指针指向位于堆中JSON字符串，失败返回NULL

```
void   cJSON_Delete(cJSON *c);
```

作用：释放位于堆中cJSON结构体内存

返回值：无

2、解析一个结构体

解析的目标如下：

{

```
   "person":
     {
               "firstName":"z",
               "lastName":"jadena",
               "email":"jadena@126.com",
               "age":8,
               "height":1.17
     }
```

}

看起来比一个键值对复杂多了，我们需要学习新的接口函数嘛?

答案是不需要！还是那三个函数。只是解析的步骤多了一些，解析的过程：

1. 根据JSON串中的对象，我们定义一个相应的结构体如下：

typedefstruct

{

```
    char firstName[32];

      char lastName[32];

      char email[64];

      int age;

      float height;
```

} PERSON;

具体的对应关系，一目了然

调用cJSON_Parse()函数，解析JSON数据包。
cJSON*root=cJSON_Parse(json_string);

调用cJSON_GetObjectItem()函数，获取到对象person。
cJSON *object=cJSON_GetObjectItem(root,“person”);

对我们刚取出来的对象person，多次调用cJSON_GetObjectItem()函数，来获取对象的成员。此时要注意，不同的成员，访问的方法不一样：
cJSON*item;
PERSONperson;

item=cJSON_GetObjectItem(object,“firstName”);

memcpy(person.firstName,item->valuestring,strlen(item->valuestring));

item=cJSON_GetObjectItem(object,“lastName”);

memcpy(person.lastName,item->valuestring,strlen(item->valuestring));

item=cJSON_GetObjectItem(object,“email”);

memcpy(person.email,item->valuestring,strlen(item->valuestring));

item=cJSON_GetObjectItem(object,“age”);

person.age=item->valueint;

item=cJSON_GetObjectItem(object,“height”);

person.height=item->valuedouble
这样，就获取到了对象的全部内容了。

通过cJSON_Delete()，释放cJSON_Parse()分配出来的内存空间。
cJSON_Delete(root);

至此，我们就使用cJSON接口完成了基于结构体的解析工作。

3、解析结构体数组的JSON串；

最后，我们来个更复杂一些的，来解析一个数组，并且数组的成员是结构体！要解析的JSON串如下：

{
“people”:[

{“firstName”:“z”,“lastName”:“Jason”,“email”:“bbbb@126.com”,“height”:1.67},

{“lastName”:“jadena”,“email”:“jadena@126.com”,“age”:8,“height”:1.17},

{“email”:“cccc@126.com”,“firstName”:“z”,“lastName”:“Juliet”,“age”:36,“height”:1.55}

]

}

此时，我们真的又需要学习新的接口了，一个是获取数组长度，一个是取数组成员，函数原型如下：

int cJSON_GetArraySize(cJSON array);
cJSONcJSON_GetArrayItem(cJSON *array,int item);

由于前面已经实现了结构体的解析，这里我们只需要关注下数组的相关调用即可。

调用cJSON_Parse()函数，解析JSON数据包。

调用一次cJSON_GetObjectItem()函数，获取到数组people。

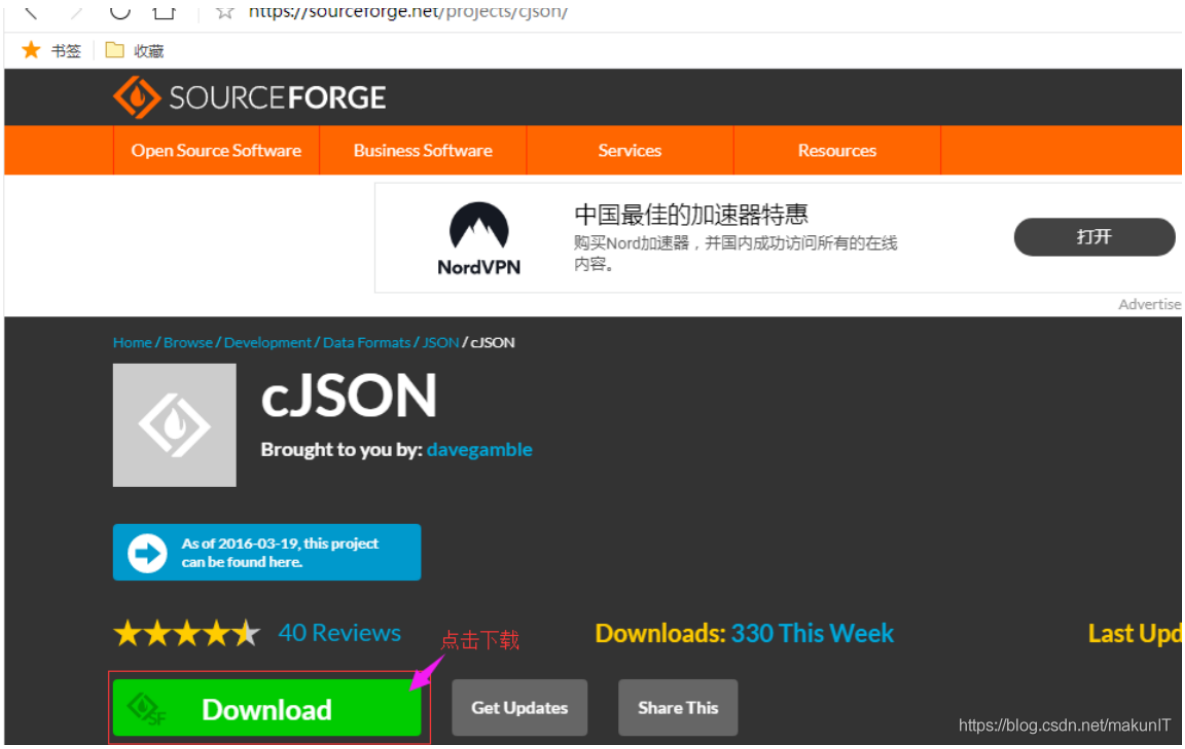
对我们刚取出来的数组people，调用cJSON_GetArraySize()函数，来获取数组中对象的个数。然后，多次调用cJSON_GetArrayItem()函数，逐个读取数组中对象的内容。

通过cJSON_Delete()，释放cJSON_Parse()分配出来的内存空间。

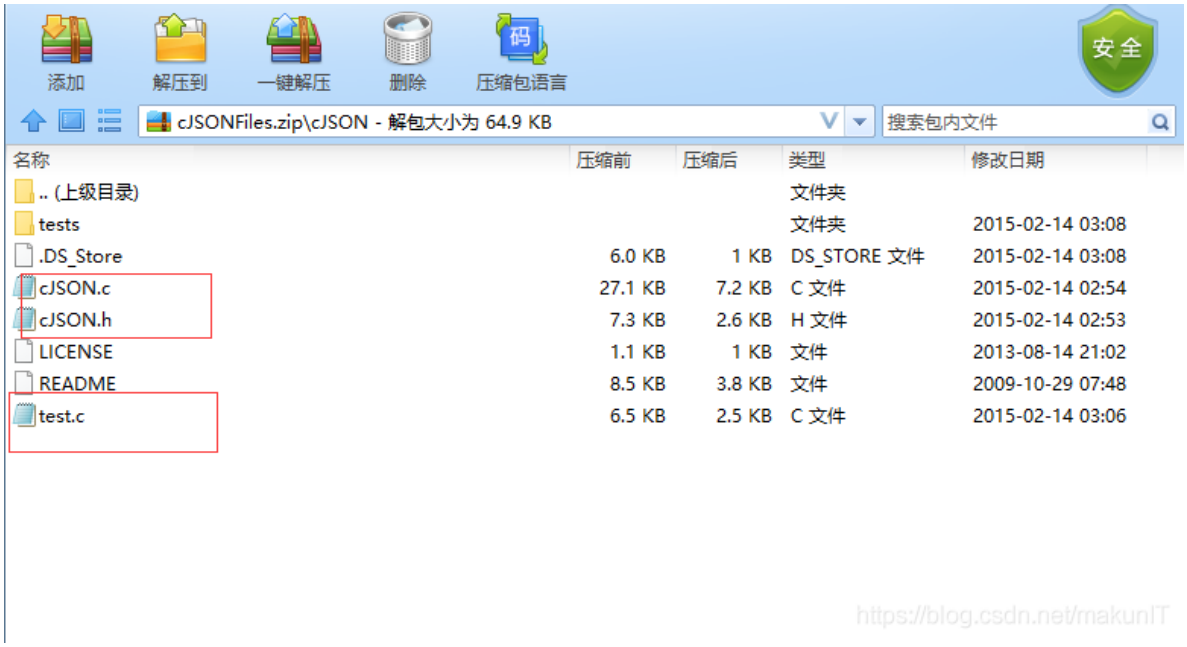
这样，我们就使用cJSON接口完成了结构体数组的解析工作。
————————————————————

五、Linux下用编程实现用cJSON解析JSON字符

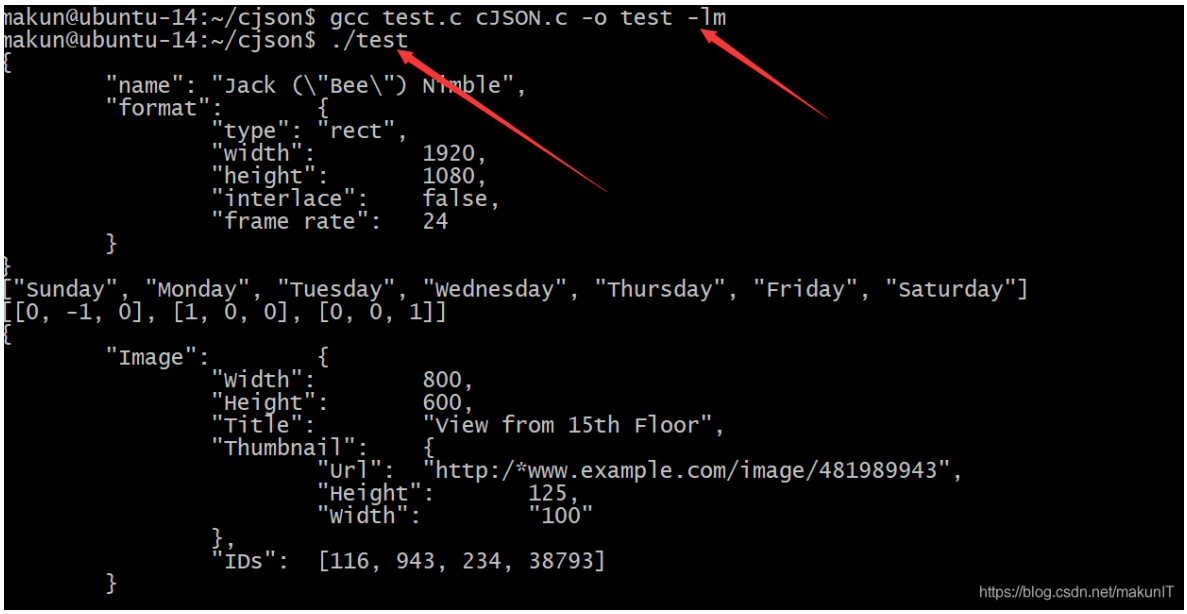
码下载：https://sourceforge.net/projects/cjson/



2、解压，进入cJSON文件夹，将其中的cJSON.c,cJSON.h,test.c拷贝到Ubuntu下测试



3、编译并运行



以上是官方自带的，下面我们来看一下自己写的程序：

1、简单的解析

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

int main(int argc, char* argv[])
{
    char buf[1024] = {" {\"date\":\"20181128\"} "};//要解析的json数据

    cJSON * root = cJSON_Parse(buf);//将字符串格式的json数据转化为JSON对象格式
    if(root == NULL)
    {
        printf("parse error\n");
        return -1;
    }

    cJSON *value = cJSON_GetObjectItem(root, "date");//根据键"date"获取其对应的值
    if(value == NULL)
    {
        printf("getvalue error\n");
        return -1;
    }

    char *data = cJSON_Print(value);//将获取值转化为字符串格式
    if(data == NULL)
    {
        printf("printf error\n");
        return -1;
    }

    printf("data=%s\n", data);//打印获取到的json数据

    return 0;
}
```

```
makun@ubuntu-14:~/cjson$ gcc cJSON.c test1.c -o test1.c -lm
makun@ubuntu-14:~/cjson$ ./test1.c
data="20181128"
makun@ubuntu-14:~/cjson$ vim test1.c
```

2、用cJSON解析JSON

①我们先写在linux上写一个josn文件， 如下：

```
data.json
 1 {
 2    "date": "29日星期四",
 3    "sunrise": "07:08",
 4    "high": "高温 9.0℃",
 5    "low": "低温 0.0℃",
 6    "sunset": "16:50",
 7    "aqi": 50.0,
 8    "fx": "东风",
 9    "fl": "4-5级",
10    "type": "晴",
11    "notice": "愿你拥有比阳光明媚的心情"
12 }
~
                                                    https://blog.csdn.net/makunIT
makun@ubuntu-14:~/cjson$ ls
1  cJSON.c  cJSON.h  data.json  test  test1.c  test2  test2.c  test.c
makun@ubuntu-14:~/cjson$
makun@ubuntu-14:~/cjson$
```

②然后我们使用cJSON来解析此JSON，代码如下：

```c
#include <stdio.h>
#include "cJSON.h"
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main()
{
    //打开JSON数据文件
    int fd = open("data.json",O_RDWR);
    if(fd < 0)
    {
        perror("open fail\n");
        return -1;
    }

    //读取文件中的数据
    char buf[2048]={0};
    int ret = read(fd, buf, sizeof(buf));
    if(ret == -1)
    {
        perror("read error");
        return -1;
    }

    //关闭文件
    close(fd);
```

```
    //把该字符串数据转换成JSON对象
    cJSON *root=cJSON_Parse(buf);
    if(root == NULL)
    {
        printf("parse error\n");
        return -1;
    }

    //根据key值去获取对应的value
    cJSON *value = cJSON_GetObjectItem(root,"date");
    if(value == NULL)
    {
      printf("GetObjec error\n");
      return -1;
    }

    //把数据转成 字符串输出
    char  *date = cJSON_Print(value);
    printf("date=%s\n",date);

    value = cJSON_GetObjectItem(root,"fx");
    if(value == NULL)
    {
        printf("GetObjec error\n");
        return -1;
    }

    //把数据转成 字符串输出
    date = cJSON_Print(value);
    printf("notice=%s\n",date);


    //根据key值去获取对应的value
    value = cJSON_GetObjectItem(root,"notice");
    if(value == NULL)
    {
        printf("GetObjec error\n");
        return -1;
    }

    //把数据转成 字符串输出
    date=cJSON_Print(value);
    printf("notice=%s\n",date);

    return 0;
}
```

运行结果如下

```
makun@ubuntu-14:~/cjson$ gcc cJSON.c test2.c -o test2 -lm
makun@ubuntu-14:~/cjson$ ./test2
notice="29日星期四"
notice="愿你拥有比阳光明媚的心情"
makun@ubuntu-14:~/cjson$ ls
1  cJSON.c  cJSON.h  data.json  test  test1.c  test2  test2.c  test.c
```