Linux json-c 的使用

——编写者:草根老师(程姚根)

一、JSON介绍

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。 易于人阅读和编写。同时也易于机器解析和生成。 它基于JavaScript Programming Language,Standard ECMA-262 3rd Edition - December 1999的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C, C++, C#, Java, JavaScript, Perl, Python等）。 这些特性使JSON成为理想的数据交换语言。

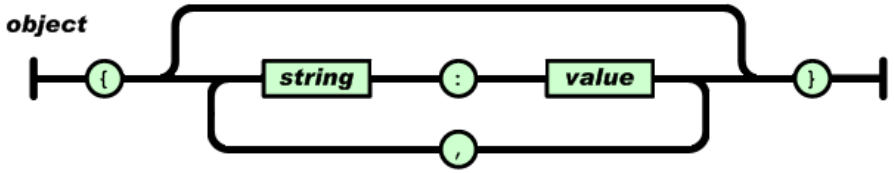
JSON建构于两种结构：

- “名称/值”对的集合（A collection of name/value pairs）。不同的语言中，它被理解为*对象（object）*，纪录（record），结构（struct），字典（dictionary），哈希表（hash table），有键列表（keyed list），或者关联数组 （associative array）。
- 值的有序列表（An ordered list of values）。在大部分语言中，它被理解为数组（array）。

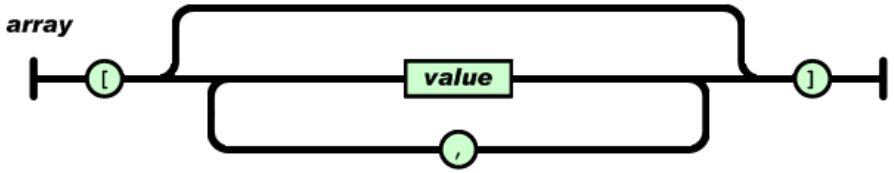
这些都是常见的数据结构。事实上大部分现代计算机语言都以某种形式支持它们。这使得一种数据格式在同样基于这些结构的编程语言之间交换成为可能。

JSON具有以下这些形式：

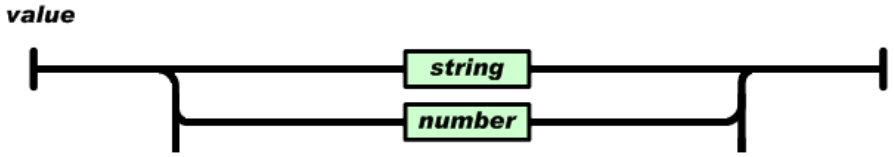
对象是一个无序的“‘名称/值’对”集合。一个对象以“{”（左括号）开始，“}”（右括号）结束。每个“名称”后跟一个“:”（冒号）；“‘名称/值’ 对”之间使用“,”（逗号）分隔。

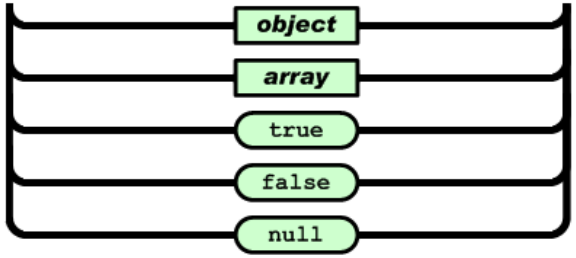


数组是值（value）的有序集合。一个数组以“[”（左中括号）开始，“]”（右中括号）结束。值之间使用“,”（逗号）分隔。



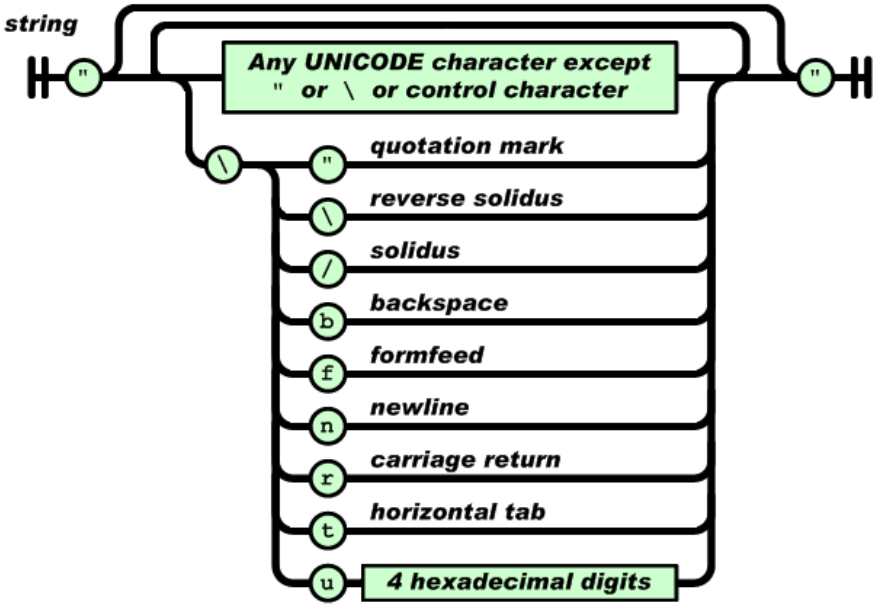
值（*value*）可以是双引号括起来的字符串（*string*）、数值(number)、`true`、`false`、 `null`、对象（object）或者数组（array）。这些结构可以嵌套。



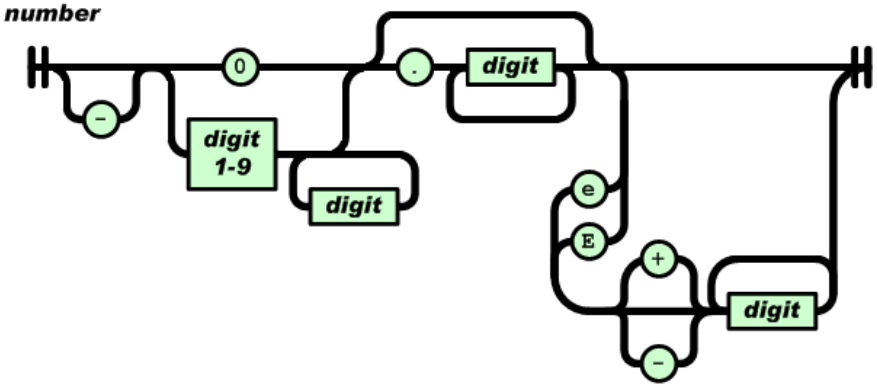


字符串（*string*）是由双引号包围的任意数量Unicode字符的集合，使用反斜线转义。一个字符（character）即一个单独的字符串（character string）。

字符串（*string*）与C或者Java的字符串非常相似。



数值（*number*）也与C或者Java的数值非常相似。除去未曾使用的八进制与十六进制格式。除去一些编码细节。



空白可以加入到任何符号之间。 以下描述了完整的语言。

二、JSON对象结构

JSON结构共有2种：

- **（1）对象结构；**
- **（2）数组结构；**

JSON，简单来说就是JavaScript中的对象或数组，所以这两种结构就是对象和数组。通过这两种结构就可以表示各种复杂的结构。

1、JSON对象结构

对象结构是使用大括号“{}”括起来的，大括号内是由0个或多个用英文逗号分隔的“关键字:值”对（**key:value**）构成的。

语法：

```
varjsonObj =
{
    "键名1":值1,
    "键名2":值2,
    ……
    "键名n":值n
}
```

说明：

jsonObj指的是**json**对象。对象结构是以“{”开始，到“}”结束。其中“键名”和“值”之间用英文冒号构成对，两个“键名:值”之间用英文逗号分隔。

注意，这里的键名是字符串，但是值可以是数值、字符串、对象、数组或逻辑**true**和**false**。

举例：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
        varobj =
        {
            "name":"helicopter",
            "age":23,
            //JSON对象内部也有一个JSON对象
            hobby:
            {
                "first":"swimming",
                "second":"singing",
                "third":"dancing"
            }
        }
        //读取JSON数据
```

```
        document.write("名字是："+obj.name+"<br>");
        document.write("第一项爱好是："+obj.hobby.first);
    </script>
</head>
<body>
</body>
</html>
```

在浏览器预览效果如下：

2、JSON对象结构

JSON数组结构是用中括号“[]”括起来，中括号内部由0个或多个以英文逗号“,”分隔的值列表组成。

语法：

```
vararr =
[
    {
        "键名1":值1,
        "键名2":值2
    },
    {
        "键名3":值3,
        "键名4":值4
    },
    ……
]
```

说明：

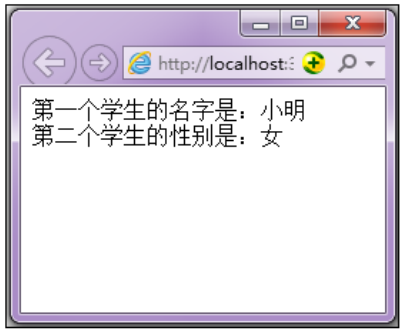
arr指的是json数组。数组结构是以“[”开始，到“]”结束，这一点跟JSON对象不同。不过在JSON数组结构中，每一对“{}”相当于一个JSON对象，大家看看像不像？而且语法都非常类似。

注意，这里的键名是字符串，但是值可以是数值、字符串、对象、数组或逻辑true和false。

举例：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
        vararr =
        [
            {
                "name":"小明",
                "age":23,
                "gender":"男"
            },
            {
                "name":"小红",
                "age":21,
                "gender":"女"
            }
        ]
        //读取JSON数据
        document.write("第一个学生的名字是："+arr[0].name+"<br>");
        document.write("第二个学生的性别是："+arr[1].gender+"<br>")
;
    </script>
</head>
<body>
</body>
</html>
```

在浏览器预览效果如下：



三、普通字符串、JSON字符串和JSON对象

初学者经常有一个困惑，就是分不清普通字符串、JSON字符串和JSON对象。其实这3者是非常容易区分的。

1、普通字符串

字符串嘛，大家都知道是使用单引号或双引号括起来的一串字符。

举例：

```
varstr = "绿叶学习网json教程";
```

2、JSON对象

JSON对象我们在“JSON对象结构”这一节已经详细为大家讲解了。JSON对象，指的是符合JSON格式要求的JavaScript对象。

举例：

```
varjsonObj = {"name":"helicopter","age":23,"gender":"男"};
```

3、JSON字符串

JSON字符串，指的是符合“JSON格式”的字符串。

JSON字符串要求两点：

（1）必须是字符串，也就是要用单引号或双引号括起来；

（2）必须符合“JSON”格式。

举例：

```
varjsonStr = '{"name": "helicopter", "age":23, "gender": "男"}
';
```

JSON字符串说白了就是在JSON对象外面加一对单引号。

大家好好对比一下，就知道普通字符串、JSON对象和JSON字符串的区别了。

四、Linux JSON库的安装

1.下载json-c的源码

JSON下载地址:https://github.com/json-c/json-c/wiki

2.解压json-c的源码包

tar -xvf json-c-0.10.tar.gz

3.配置json-c的源码包

(1)在x86平台配置

./configure --prefix=/usr

(2)在ARM平台配置

./configure --prefix=安装路径 --host=交叉开发工具链前缀

例如安装路径为"/home/linux/json/install" 交叉开发工具链前缀为arm-none-linux-gnueabi-,则指定configuer参数如下:

./configure --prefix=/home/linux/json/install --host=arm-none-linux-gnueabi-

4.编译

make

5.安装

make install

注意:

在x86平台安装的时候，由于我们指定的/usr目录，普通用户对于这个目录是没有写权限的，所以在安装的时候会提示"Permission denied"，解决方法是以超级用户身份执行。

五、Linux JSON库的使用

```c
#include <stdio.h>
#include <json/json.h>

/*********************************************************************************************
 *作者: 草根老师(程姚根)
 *
 *json-test.c测试程序功能如下:
 *
 *1.创建一个json对象
 *2.给json对象加入成员, 并且赋值
 *3.以字符串的形式输出json对象
 *4.将json格式的字符串转成json对象
 *5.获取json对象的成员值
 *
 *编译:
 *x86平台编译 :gcc json-test.c -o  json-test -ljson
 *ARM平台编译 :需要指定交叉编译,通过 -I 指定头文件所在路径 -L 指定库所在路径  -l指定库名
 *
 */

void print_json_object_member(json_object *new_object,const char *field)
{

    //根据指定对象的成员名, 获取这个成员对象的json对象
    struct json_object *object = json_object_object_get(new_object, field);
    //获取json对象的类型
    enum json_type object_type = json_object_get_type(object);
```

```
    switch(object_type){
        case json_type_int:
            printf("new_obj.%s json_object_get_type()=%s\n", field,json_type_to_name(object_type));
            printf("new_obj.%s json_object_get_int()=%d\n", field,json_object_get_int(object));
            break;

        case json_type_string:
            printf("new_obj.%s json_object_get_type()=%s\n", field,json_type_to_name(object_type));
            printf("new_obj.%s json_object_get_string()=%s\n", field,json_object_get_string(object))
            break;
    }
    printf("\n-----------------------------------------------------\n");

    //释放json对象
    json_object_put(object);

    return;
}

void print_json_object(json_object *j_object)
{
    print_json_object_member(j_object,"temp");
    print_json_object_member(j_object,"humidity");
    print_json_object_member(j_object,"triaxialX");
    print_json_object_member(j_object,"triaxialY");
    print_json_object_member(j_object,"triaxialZ");
    print_json_object_member(j_object,"userName");
    print_json_object_member(j_object,"passWord");

    return;
}


int main(int argc, const char *argv[])
{
    json_object *new_obj;
    json_object *my_object;
    unsigned char *my_object_string;
    unsigned char json_fromat_string[1024];

    //创建一个空的json对象
    my_object = json_object_new_object();

    //以key-value的形式添加json对象的成员
    json_object_object_add(my_object, "temp", json_object_new_int(20));
    json_object_object_add(my_object, "humidity", json_object_new_int(10));
    json_object_object_add(my_object, "triaxialX", json_object_new_int(10));
```

```
    json_object_object_add(my_object, "triaxialY", json_object_new_int(20));
    json_object_object_add(my_object, "triaxialZ", json_object_new_int(30));
    json_object_object_add(my_object, "userName", json_object_new_string("cyg"));
    json_object_object_add(my_object, "passWord", json_object_new_string("123456"));

    //将json对象内容，转成json格式的字符串
    my_object_string = (char *)json_object_to_json_string(my_object);
    printf("my_object.to_string()=%s\n", my_object_string);

    //将json格式的字符串转成json对象
    new_obj = json_tokener_parse(my_object_string);
    printf("new_obj.to_string()=%s\n", json_object_to_json_string(new_obj));

    //输出json对象的成员
    print_json_object(new_obj);

    //释放json对象
    json_object_put(my_object);
    json_object_put(new_obj);

    return 0;
}
```

注意:

在编译的时候，会提示"/usr/include/json/json.h:27:34: fatal error: json_object_iterator.h: No such file or directory"，原因是我们在JSON库安装的时候，没有将json源码下的" json_object_iterator.h"头文件安装到/usr/include/json目录下，解决的方法很简单，我们只需要将json源码树下的" json_object_iterator.h"拷贝到/usr/include/json目录下就可以了。

参考文献:

http://www.lvyestudy.com/json/json.aspx

http://json.org/json-zh.html