JSON解析器json-c

JSON-C实现了一个引用计数对象模型，它允许您轻松地使用C语言来构建JSON对象，将它们输出为JSON格式的字符串，并将JSON格式字符串解析回JSON对象的C语言表示形式。它的目标是符合RFC 7159标准。

二、编译

2.1 automake

使用automake的编译过程如下：

```
$ git clone https://github.com/json-c/json-c.git
$ cd json-c
$ sh autogen.sh
// 标准的三部曲
$ ./configure  # --enable-threading
$ make
$ make install
// 编译并运行测试程序
$ make check
$ make USE_VALGRIND=0 check   # optionally skip using valgrind
```

2.2 cmake

使用cmake编译的过程如下：

```
mkdir build
cd build
cmake ../
make
```

cmake可选的几个编译选项为：

Variable	Type	Description

Variable	Type	Description
CMAKE_INSTALL_PREFIX	String	The install location.
BUILD_SHARED_LIBS	Bool	The default build generates a dynamic (dll/so) library. Set this to OFF to create a static library instead.
ENABLE_RDRAND	Bool	Enable RDRAND Hardware RNG Hash Seed
ENABLE_THREADING	Bool	Enable partial threading support

三、使用

3.1 头文件

要使用json-c，最简单的方式是包含json.h头文件即可，或者最好是下列更具体的头文件之一：

- json_object.h：核心类型和方法；
- json_tokener.h：用于解析和序列化json-c对象树的方法；
- json_pointer.h：用于从JSON-c对象树中检索对象的JSON指针(RFC 6901)实现；
- json_object_iterator.h：用于迭代单个json_object实例的方法；
- json_visit.h：遍历json-c对象树的方法；
- json_util.h：其他混杂工具集函数。

3.2 API介绍

详细且全面的API介绍文档：http://json-c.github.io/json-c/

3.2.1 JSON对象类型

JSON-C支持的JSON对象类型有7种：

```
typedef enum json_type {
  /* If you change this, be sure to update json_type_to_name() too */
  json_type_null,
  json_type_boolean,
  json_type_double,
  json_type_int,
  json_type_object,
  json_type_array,
  json_type_string
} json_type;
```

3.2.2 创建JSON对象

下面系列函数用于创建一个JSON对象：

```
struct json_object *    json_object_new_object (void);
struct json_object *    json_object_new_array (void);
struct json_object *    json_object_new_boolean (json_bool b);
```

```
struct json_object *    json_object_new_int (int32_t i);
struct json_object *    json_object_new_int64 (int64_t i);
struct json_object *    json_object_new_double (double d);
struct json_object *    json_object_new_double_s (double d, const char *ds);
struct json_object *    json_object_new_string (const char *s);
struct json_object *    json_object_new_string_len (const char *s, int len);
```

- json_object_new_object()创建一个新的json对象，引用计数为1，该指针具有唯一的所有权；当使用json_object_object_add()或者json_object_array_put_idx()作用于该对象时，所有权转移到另一方。使用json_object_get作用于该对象的后，必须使用json_object_put释放。
- json_object_new_array()创建一个JSON数组类型JSON对象；
- json_object_new_boolean()创建一个布尔类型JSON对象；
- json_object_new_int()创建32位整形JSON对象；
- json_object_new_int64()创建64位整形JSON对象；
- json_object_new_double()创建double类型JSON对象；
- json_object_new_string()将C字符串转换为JSON字符串格式的对象；

3.2.3 增加/删除/修改

给JSON对象增加字段（不会增加引用计数）：

```
int json_object_object_add (struct json_object *obj, const char *key, struct json_object
int json_object_object_add_ex (struct json_object *obj, const char *const key, struct jso
```

删除json对象的指定字段，被删除的对象引用计数减去1，如果这个val没有更多的所有者，这个key对应的val被free，否则这个val的引用保存在内存中：

```
void json_object_object_del (struct json_object *obj, const char *key);
```

增加一个元素到json数组的末尾，obj引用计数不会增加，增加字段的方式更加紧凑；如果需要获取val的引用，需要用json_object_get()来传递该对象：

```
int json_object_array_add (struct json_object *obj, struct json_object *val);
```

替换json数组中的值：

```
int json_object_array_put_idx (struct json_object *obj, size_t idx, struct json_object *v
```

json数组的排序，这里需要自己写排序函数：

```
void json_object_array_sort (struct json_object *jso, int(*sort_fn)(const void *, const v
```

3.2.4 取值

获取json对象的长度，依据字段的数目：

```
int json_object_object_length (const struct json_object *obj);
```

获取json对象的哈希表：

```
struct lh_table * json_object_get_object (const struct json_object *obj);
```

获取对象的数组列表：

```
struct array_list * json_object_get_array (const struct json_object *obj);
```

获取json的类型：

```
enum json_type json_object_get_type (const struct json_object *obj);
```

获取json数组对象的长度：

```
size_t json_object_array_length (const struct json_object *obj);
```

获取json对象的bool值，int和double对象是0转换为FALSE，否则返回TRUE；非0长度的字符串返回TRUE；其他对象非空的话，返回TRUE：

```
json_bool json_object_get_boolean (const struct json_object *obj);
```

获取json对象的长度,如果参数不是string类型的json，返回0：

```
int json_object_get_string_len (const struct json_object *obj);
```

按照索引获取json数组的对象：

```
struct json_object * json_object_array_get_idx (const struct json_object *obj, size_t idx
```

3.2.5 类型转换

转换json对象到c字符串格式：

```
const char * json_object_to_json_string (struct json_object *obj);
```

获取JSON中指定类型的数值：

```
json_bool json_object_get_boolean (const struct json_object *obj);
int32_t json_object_get_int (const struct json_object *obj);
int64_t json_object_get_int64 (const struct json_object *obj);
```

```
double json_object_get_double (const struct json_object *obj);
const char * json_object_get_string (struct json_object *obj);
```

将字符串转换为json对象：

```
struct json_object * json_tokener_parse (const char *str)
```

3.2.6 JSON对象释放

以下两个函数配合使用，前者获取该对象指针的所有权，引用计数加1，如果对象已经被释放，返回NULL;后者引用计数减1，如果对象已经被释放，返回1：

```
json_object * json_object_get (struct json_object *obj);
int json_object_put (struct json_object *obj);
```

3.2.7 其他方法

类型判断：

```
int json_object_is_type (const struct json_object *obj, enum json_type type);
```

3.3 json_util.h

json_util.h提供了有关文件读写操作的函数，这个文件的内容是json格式的：

```
/* utility functions */
extern struct json_object* json_object_from_file(const char *filename);
extern struct json_object* json_object_from_fd(int fd);
extern int json_object_to_file(const char *filename, struct json_object *obj);
extern int json_object_to_file_ext(const char *filename, struct json_object *obj, int fla
extern int json_object_to_fd(int fd, struct json_object *obj, int flags);
const char *json_util_get_last_err(void);
extern int json_parse_int64(const char *buf, int64_t *retval);
extern int json_parse_double(const char *buf, double *retval);
extern const char *json_type_to_name(enum json_type o_type);
```

"感谢！个人愚见能给大家带来帮助是我坚持分享的最大动力"

赞赏支持 还没有人赞赏，支持一下

konishi5202

资深架构师，PMP获得者，擅长于系统建模和系统分析设计。从事过汽车安全系统、汽车娱乐系统、消

总资产9 共写了12.8W字 获得877个赞 共1,082个粉丝

关注

被以下专题收入，发现更多相似内容

 程序员  读书  工具癖  Linux三方库

推荐阅读更多精彩内容 >

面试题

Swift1> Swift和OC的区别1.1> Swift没有地址/指针的概念1.2> 泛型1.3> 类型严谨 对...

 cosWriter 阅读 9,237 评论 1 赞 27

h5常见的面试题

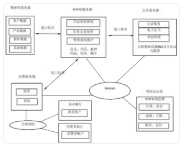
第一部分 HTML&CSS整理答案 1. 什么是HTML5？ 答:HTML5是最新的HTML标准。注意:讲述HT...

 kismetajun 阅读 24,154 评论 1 赞 45

电子商务网站开发与建设

概要 64学时 3.5学分 章节安排 电子商务网站概况 HTML5+CSS3 JavaScript Node 电子...

 阿啊阿吖丁 阅读 5,337 评论 0 赞 2



快过年了，为过完年跳槽的人准备一份面试题

1.设计模式是什么？ 你知道哪些设计模式，并简要叙述？设计模式是一种编码经验，就是用比较成熟的逻辑去处理某一种类型...

 龍蠡 阅读 1,604 评论 0 赞 12

喜一场

你是我曾跨跃的山峰，也是将我淹没的河流，你是我梦中的蝴蝶 也曾是我迷离的梦幻 你是我开启幸福的动力 也是埋葬我...

麦积梦境 阅读 137 评论 7 赞 3

konishi5202

关注

总资产9

个人简历编写注意事项

阅读 1,935

常见工作方法总结(SWOT分析

写下你的评论...

评论1

赞17