<https://blog.csdn.net/songfeihu0810232/article/details/54892149>

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/songfeihu0810232/article/details/54892149

# 1.     前言

Linux下的http请求有许多种方式，其中curl库是C语言封装的一个强大的库，使用curl比封装socket更加方便。cJSON是一个小型的json封装库，可以把数据封装成json格式。本文介绍了这两种技术，并通过此技术完成了Linux下的http请求，同时把代码封装到quagga下，quagga运行时可以正常创建数据到ONOS。

# 2.     Linux下http请求实现

## 2.1.    curl简介

curl命令是一个功能强大的网络工具，它能够通过http、ftp等方式下载文件，也能够上传文件。curl命令使用了libcurl库来实现，libcurl库常用在C程序中用来处理HTTP请求，curlpp是libcurl的一个C++封装，这几个东西可以用在抓取网页、网络监控等方面的开发，而curl命令可以帮助来解决开发过程中遇到的问题。本文档就是使用curl完成http请求。

### 2.1.1.    全局初始化

应用程序在使用libcurl之前，必须先初始化libcurl。libcurl只需初始化一次。可以使用以下语句进行初始化：

curl\_global\_init();

curl\_global\_init()接收一个参数，告诉libcurl如何初始化。参数CURL\_GLOBAL\_ALL 会使libcurl初始化所有的子模块和一些默认的选项，通常这是一个比较好的默认参数值。还有两个可选值：

CURL\_GLOBAL\_WIN32

只能应用于Windows平台。它告诉libcurl初始化winsock库。如果winsock库没有正确地初始化，应用程序就不能使用socket。在应用程序中，只要初始化一次即可。

CURL\_GLOBAL\_SSL

如果libcurl在编译时被设定支持SSL，那么该参数用于初始化相应的SSL库。同样，在应用程序中，只要初始化一次即可。

libcurl有默认的保护机制，如果在调用curl\_easy\_perform时它检测到还没有通过curl\_global\_init进行初始 化，libcurl会根据当前的运行时环境，自动调用全局初始化函数。但必须清楚的是，让系统自已初始化不是一个好的选择。

当应用程序不再使用libcurl的时候，应该调用curl\_global\_cleanup来释放相关的资源。在程序中，应当避免多次调用curl\_global\_init和curl\_global\_cleanup。它们只能被调用一次。

### 2.1.2.    easy interface

libcurl中被称为easy interface的api函数，所有这些函数都是有相同的前缀：curl\_easy 。

要使用easyinterface，首先必须创建一个easy handle，easy handle用于执行每次操作。基本上，每个线程都应该有自己的easy handle用于数据通信（如果需要的话）。千万不要在多线程之间共享同一个easy handle。下面的函数用于获取一个easy handle：

CURL \*easy\_handle =curl\_easy\_init();

在easyhandle上可以设置属性和操作(action)。easy handle就像一个逻辑连接，用于接下来要进行的数据传输。

使用curl\_easy\_setopt函数可以设置easy handle的属性和操作，这些属性和操作控制libcurl如何与远程主机进行数据通信。一旦在easy handle中设置了相应的属性和操作，它们将一直作用该easyhandle。也就是说，重复使用easy hanle向远程主机发出请求，先前设置的属性仍然生效。

easy handle的许多属性使用字符串(以\0结尾的字节数组)来设置。通过curl\_easy\_setopt函数设置字符串属性时，libcurl内部会自动拷贝这些字符串，所以在设置完相关属性之后，字符串可以直接被释放掉（如果需要的话）。

    后面章节会根据http的get和post接口对常用的easy handle函数进行说明。

## 2.2.    cJSON简介

cJSON是在C语言中解析JSON的开源库，在cJSON中，一个key-value键值对被解析并存放在一个cJSON结构体变量中，其value取值集为：FALSE，TRUE，NULL，NUMBER，STRING，OBJECT，ARRAY。cJOSN库，仅有两个文件cJSON.c和cJSON.h。

### 2.2.1.       cJSON使用说明

下面使用cJSON组装以下json数据

{

    "mac":  "46:E4:3C:A4:11:12",

    "vlan": "-1",

    "ipAddresses":  ["222.222.233.2"],

    "location": {

        "elementId":    "of:0000001e08000fe3",

        "port": "31"

    }

}

代码如下：

//创建一个object

cJSON \*root =cJSON\_CreateObject();

    cJSON\_AddItemToObject(root,"mac",cJSON\_CreateString("46:E4:3C:A4:13:12"));

    cJSON\_AddStringToObject(root,"vlan","-1");

    cJSON\*array = NULL;

    cJSON\_AddItemToObject(root,"ipAddresses",array=cJSON\_CreateArray());

    cJSON\_AddItemToArray(array,cJSON\_CreateString("192.168.10.2"));

    //创建一个子object，将此object添加到root中

    cJSON\*location = NULL;

    cJSON\_AddItemToObject(root,"location",location=cJSON\_CreateObject());

    cJSON\_AddStringToObject(location,"elementId","of:0000001e08000fe3");

    cJSON\_AddStringToObject(location,"port","31");

    //将json结构格式化到缓冲区

    char\*buf = cJSON\_Print(root);

    //执行http请求函数

http\_client\_thttp\_read\_client;

    http\_client\_read\_init(&http\_read\_client,temp, buf);

    //数据使用完之后，把内存释放掉

cJSON\_Delete(json);

    free(buf);

## 2.3.    http的post请求

int http\_client\_read\_init(http\_client\_t\*http\_client, const char \*url, char \*szJsonData)

{

   if (!url) {

        return -1;

    }

    //初始化libcurl，设置默认参数

   CURLcode return\_code = curl\_global\_init(CURL\_GLOBAL\_ALL);

   if (CURLE\_OK != return\_code) {

        printf("initlibcurl failed.\n");

       return -1;

    }

    //获取easy handle

   http\_client->handle = curl\_easy\_init();

   if (!http\_client->handle) {

       return -1;

    }

    //通过CURLOPT\_URL属性设置url

   curl\_easy\_setopt(http\_client->handle, CURLOPT\_URL, url);

    //通过CURLOPT\_HTTPHEADER定义http消息的header

   struct curl\_slist \*plist = NULL;

    plist = curl\_slist\_append(plist,

                 "Content-Type:application/json");

   curl\_easy\_setopt(http\_client->handle, CURLOPT\_HTTPHEADER, plist);

    printf("Thejson is: %s\n", szJsonData);

    //通过CURLOPT\_POSTFIELDS设置要POST的数据

curl\_easy\_setopt(http\_client->handle,CURLOPT\_POSTFIELDS, szJsonData);

    //通过CURLOPT\_USERPWD属性来设置用户名与密码。参数是”user:password “的字符串

    curl\_easy\_setopt(http\_client->handle,CURLOPT\_USERPWD, "karaf:karaf");

    //使用curl\_easy\_perform执行上传数据

    curl\_easy\_perform(http\_client->handle);

    //任务执行结束使用curl\_easy\_cleanup把内存释放

curl\_easy\_cleanup(http\_client->handle);

return 0;

}

## 2.4.    http的get请求

int http\_client\_init(http\_client\_t\*http\_client, const char \*url, write\_cb\_t \*write\_data, void \*userp)

{

   if (!url) {

       return -1;

    }

    //初始化libcurl，设置默认参数

   CURLcode return\_code;

    return\_code= curl\_global\_init(CURL\_GLOBAL\_ALL);

   if (CURLE\_OK != return\_code) {

        printf("initlibcurl failed.\n");

       return -1;

    }

    //获取easy handle

   http\_client->handle = curl\_easy\_init();

   if (!http\_client->handle) {

       return -1;

}

    //通过CURLOPT\_URL属性设置url

curl\_easy\_setopt(http\_client->handle,CURLOPT\_URL, url);

//注册回调函数write\_cb，回调函数将会在接收到数据的时候被调用

   curl\_easy\_setopt(http\_client->handle, CURLOPT\_WRITEFUNCTION, write\_data);     //通过CURLOPT\_USERPWD属性来设置用户名与密码。参数是”user:password “的字符串

    curl\_easy\_setopt(http\_client->handle,CURLOPT\_USERPWD, "karaf:karaf");

if (userp) {

//设置写数据的变量

       curl\_easy\_setopt(http\_client->handle, CURLOPT\_WRITEDATA, userp);

}

//使用curl\_easy\_perform执行上传数据

    curl\_easy\_perform(http\_client->handle);

    //任务执行结束使用curl\_easy\_cleanup把内存释放

curl\_easy\_cleanup(http\_client->handle);

   return 0;

}

//回调函数，将接收到的数据保存到本地文件中，同时显示在控制台上。

static size\_t write\_data(void \*buf, size\_tsize, size\_t nmemb, void \*userp)

{

   FILE \*fp = (FILE \*)userp;

   size\_t return\_size = fwrite(buf, size, nmemb, fp);

   printf("write\_data: %ld, return\_size: %ld\n", nmemb,return\_size);

   return return\_size;

}

## 2.5.    http的del请求

//获取easy handle

   http\_client->handle = http\_client\_init();

   if (!http\_client->handle) {

       return -1;

    }

    // 通过CURLOPT\_URL属性设置url

   curl\_easy\_setopt(http\_client->handle, CURLOPT\_URL, url);

    // 设置http发送的内容类型为JSON

   struct curl\_slist \*plist = NULL;

    //plist = curl\_slist\_append(plist,

        //"Content-Type:application/json");

    curl\_easy\_setopt(http\_client->handle,CURLOPT\_HTTPHEADER, plist);

    // 设置要POST的JSON数据

   //curl\_easy\_setopt(http\_client->handle, CURLOPT\_POSTFIELDS,szJsonData);

    curl\_easy\_setopt(http\_client->handle,CURLOPT\_CUSTOMREQUEST, "DELETE");

    curl\_easy\_setopt(http\_client->handle,CURLOPT\_USERPWD, "karaf:karaf");

## 2.6.    编译并执行

代码完成之后，使用gcc运行时需要带如下参数：

gcc -o http\_client http\_client.ccJSON.c -lcurl –lm

编译之后开始运行

./http\_client

通过抓包可以看到http报文

注意事项：

1.  -lcurl是链接curl库，如果运行报如下错误：

mlogc.c:32:23: error: curl/curl.h: No such fileor directory

mlogc.c:1091: error: expected ‘)’ before ‘\*’token

mlogc.c: In function ‘logc\_init’:

 则需要安装如下依赖包：libcurl-dev, libcurl-devel

 centOS上安装依赖包：

yum install libcurl-dev libcurl-devel

2.   –lm是链接math的库，由于cJSON需要调用math库，如果不添加会报如下错误：

3.   如果有多个方法例如：get/post/delete等用到的URL一样，需要把如下代码，包含URL的头部注释掉，否则报文发布出去。

    // 设置http发送的内容类型为JSON

   struct curl\_slist \*plist = NULL;

    //plist = curl\_slist\_append(plist,

        //"Content-Type:application/json");

    curl\_easy\_setopt(http\_client->handle,CURLOPT\_HTTPHEADER, plist);