(1条消息)基于libevent库实现的http server示例 - 李清的博客 - CSDN博客

最近在工作当中接触到libevent库，用于http server端功能还是比较强大，特在此记录一笔，以备后面查漏补缺。首先是下载安装，直接去官网下载对应版本的安装包，解压安装即可，这里就不啰嗦了。

#tar
#./configure
#make
#make install

完成安装之后，就可以开始编写自己的示例程序了，先上代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <evhttp.h>
#include <event.h>
#include <string.h>
#include "event2/http.h"
#include "event2/event.h"
#include "event2/buffer.h"
#include "event2/bufferevent.h"
#include "event2/bufferevent_compat.h"
#include "event2/http_struct.h"
#include "event2/http_compat.h"
#include "event2/util.h"
#include "event2/listener.h"

#define BUF_MAX 1024*16


void get_post_message(char *buf, struct evhttp_request *req)
{
        size_t post_size = 0;

        post_size = evbuffer_get_length(req->input_buffer);
        printf("====line:%d,post len:%d\n",__LINE__,post_size);
        if (post_size <= 0)
        {
                printf("====line:%d,post msg is empty!\n",__LINE__);
                return;
        }
        else
        {
                size_t copy_len = post_size > BUF_MAX ? BUF_MAX : post_size;
                printf("====line:%d,post len:%d, copy_len:%d\n",__LINE__,post_size,copy_len);
                memcpy(buf, evbuffer_pullup(req->input_buffer,-1), copy_len);
                buf[post_size] = '\0';
                printf("====line:%d,post msg:%s\n",__LINE__,buf);
        }
}


char *find_http_header(struct evhttp_request *req,struct evkeyvalq *params,const char *query_char)
{
        if(req == NULL || params == NULL || query_char == NULL)
        {
                printf("====line:%d,%s\n",__LINE__,"input params is null.");
                return NULL;
        }

        struct evhttp_uri *decoded = NULL;
        char *query = NULL;
        char *query_result = NULL;
        const char *path;
        const char *uri = evhttp_request_get_uri(req);
```

```
        if(uri == NULL)
        {
                printf("====line:%d,evhttp_request_get_uri return null\n",__LINE__);
                return NULL;
        }
        else
        {
                printf("====line:%d,Got a GET request for <%s>\n",__LINE__,uri);
        }


        decoded = evhttp_uri_parse(uri);
        if (!decoded)
        {
                printf("====line:%d,It's not a good URI. Sending BADREQUEST\n",__LINE__);
                evhttp_send_error(req, HTTP_BADREQUEST, 0);
                return;
        }


        path = evhttp_uri_get_path(decoded);
        if (path == NULL)
        {
                path = "/";
        }
        else
        {
                printf("====line:%d,path is:%s\n",__LINE__,path);
        }


        query = (char*)evhttp_uri_get_query(decoded);
        if(query == NULL)
        {
                printf("====line:%d,evhttp_uri_get_query return null\n",__LINE__);
                return NULL;
        }


        evhttp_parse_query_str(query, params);
        query_result = (char*)evhttp_find_header(params, query_char);

        return query_result;
}


void http_handler_testget_msg(struct evhttp_request *req,void *arg)
{
        if(req == NULL)
        {
                printf("====line:%d,%s\n",__LINE__,"input param req is null.");
                return;
        }

        char *sign = NULL;
        char *data = NULL;
        struct evkeyvalq sign_params = {0};
        sign = find_http_header(req,&sign_params,"sign");
        if(sign == NULL)
        {
                printf("====line:%d,%s\n",__LINE__,"request uri no param sign.");
        }
        else
        {
                printf("====line:%d,get request param: sign=[%s]\n",__LINE__,sign);
        }

        data = find_http_header(req,&sign_params,"data");
        if(data == NULL)
        {
                printf("====line:%d,%s\n",__LINE__,"request uri no param data.");
        }
```

```
        else
        {
                printf("====line:%d,get request param: data=[%s]\n",__LINE__,data);
        }
        printf("\n");


        struct evbuffer *retbuff = NULL;
        retbuff = evbuffer_new();
        if(retbuff == NULL)
        {
                printf("====line:%d,%s\n",__LINE__,"retbuff is null.");
                return;
        }
        evbuffer_add_printf(retbuff,"Receive get request,Thamks for the request!");
        evhttp_send_reply(req,HTTP_OK,"Client",retbuff);
        evbuffer_free(retbuff);
}


void http_handler_testpost_msg(struct evhttp_request *req,void *arg)
{
        if(req == NULL)
        {
                printf("====line:%d,%s\n",__LINE__,"input param req is null.");
                return;
        }

        char buf[BUF_MAX] = {0};
        get_post_message(buf, req);
        if(buf == NULL)
        {
                printf("====line:%d,%s\n",__LINE__,"get_post_message return null.");
                return;
        }
        else
        {

                printf("====line:%d,request data:%s",__LINE__,buf);
        }


        struct evbuffer *retbuff = NULL;
        retbuff = evbuffer_new();
        if(retbuff == NULL)
        {
                printf("====line:%d,%s\n",__LINE__,"retbuff is null.");
                return;
        }
        evbuffer_add_printf(retbuff,"Receive post request,Thamks for the request!");
        evhttp_send_reply(req,HTTP_OK,"Client",retbuff);
        evbuffer_free(retbuff);
}

int main()
{
        struct evhttp *http_server = NULL;
        short http_port = 8081;
        char *http_addr = "0.0.0.0";


        event_init();

        http_server = evhttp_start(http_addr,http_port);
        if(http_server == NULL)
        {
                printf("====line:%d,%s\n",__LINE__,"http server start failed.");
                return -1;
        }


        evhttp_set_timeout(http_server,5);
```

```
        evhttp_set_cb(http_server,"/me/testpost",http_handler_testpost_msg,NULL);
        evhttp_set_cb(http_server,"/me/testget",http_handler_testget_msg,NULL);


        event_dispatch();

        evhttp_free(http_server);

        return 0;
}
```

- 63
- 64
- 65
- 66
- 67
- 68
- 69
- 70
- 71
- 72
- 73
- 74
- 75
- 76
- 77
- 78
- 79
- 80
- 81
- 82
- 83
- 84
- 85
- 86
- 87
- 88
- 89
- 90
- 91
- 92
- 93
- 94
- 95
- 96
- 97
- 98
- 99
- 100
- 101
- 102
- 103
- 104
- 105
- 106
- 107
- 108
- 109
- 110
- 111
- 112
- 113
- 114
- 115
- 116
- 117
- 118
- 119
- 120
- 121
- 122
- 123
- 124
- 125
- 126
- 127
- 128
- 129
- 130
- 131
- 132
- 133
- 134
- 135

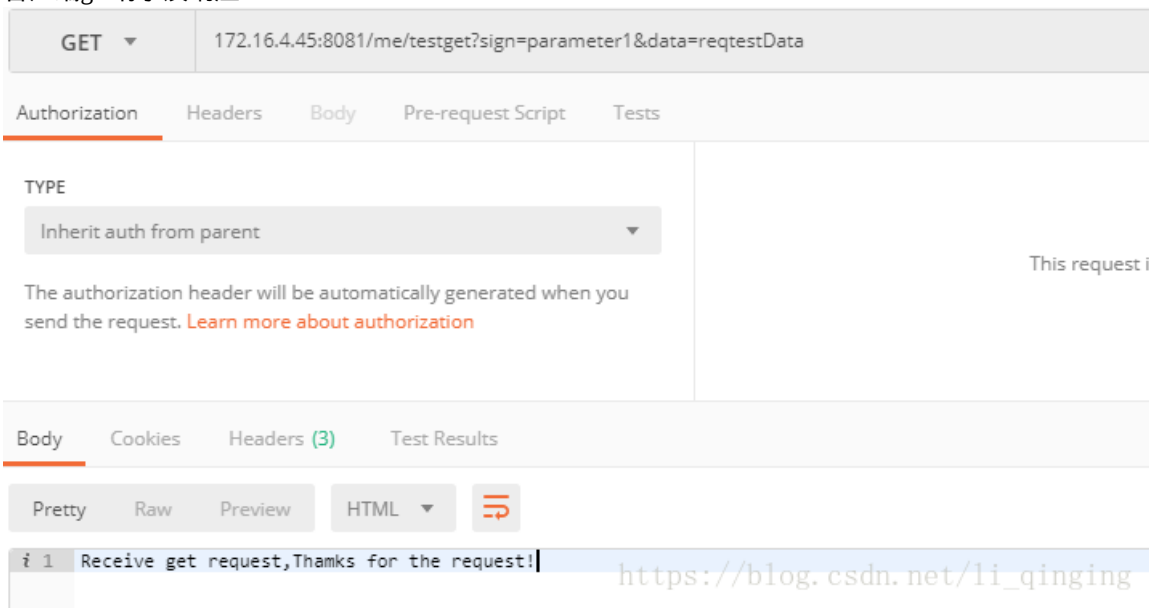
- 136
- 137
- 138
- 139
- 140
- 141
- 142
- 143
- 144
- 145
- 146
- 147
- 148
- 149
- 150
- 151
- 152
- 153
- 154
- 155
- 156
- 157
- 158
- 159
- 160
- 161
- 162
- 163
- 164
- 165
- 166
- 167
- 168
- 169
- 170
- 171
- 172
- 173
- 174
- 175
- 176
- 177
- 178
- 179
- 180
- 181
- 182
- 183
- 184
- 185
- 186
- 187
- 188
- 189
- 190
- 191
- 192
- 193
- 194
- 195
- 196
- 197
- 198
- 199
- 200
- 201
- 202
- 203
- 204
- 205
- 206
- 207
- 208

- 209
- 210

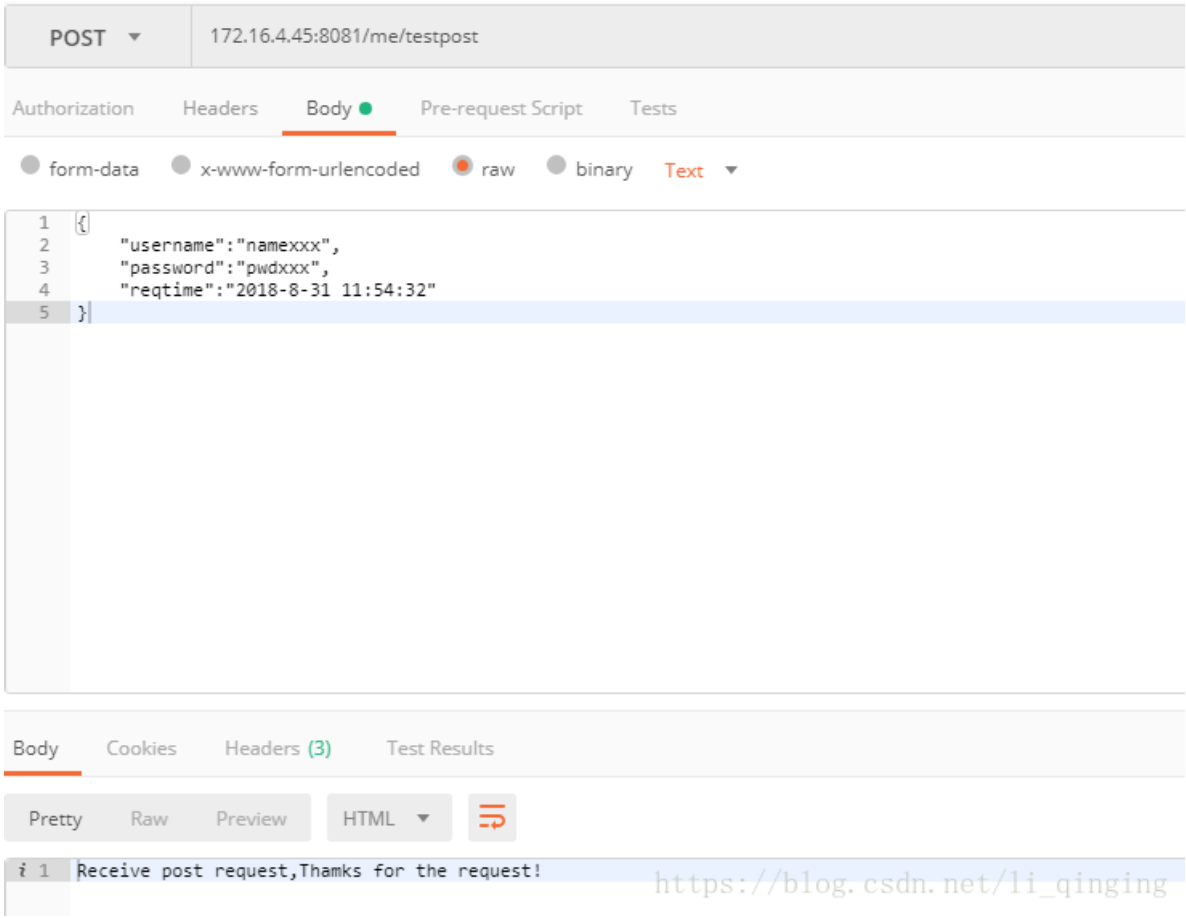
关于代码，相应的注释已经很清楚了，这里主要说明一下编译有关的问题，编译的时候需要引入libevent库相关的头文件目录，动态库路径和名称。
编译命令：gcc -o http_server http_server.c -I/usr/local/include/ -L/usr/local/lib/ -levent
-I/usr/local/include/ ：头文件路劲
-L/usr/local/lib/ ：动态库路劲
-levent ：动态库名称（libevent.so）

附get请求和post请求及处理结果，使用postman工具模拟发送http命令，
下载地址：postman下载地址下载后一键安装即可用。

客户端get请求及响应：



客户端post请求及响应：



服务端处理get请求：

```
[root@localhost test]# ./http_server
====line:62,Got a GET request for </me/testget?sign=parameter1&data=reqtestData>
====line:82,path is:/me/testget
====line:119,get request param: sign=[parameter1]
====line:62,Got a GET request for </me/testget?sign=parameter1&data=reqtestData>
====line:82,path is:/me/testget
====line:129,get request param: data=[reqtestData]
```

服务端处理post请求：

```
[root@localhost test]# ./http_server
====line:24,post len:80
====line:33,post len:80, copy_len:80
====line:36,post msg:{
        "username":"namexxx",
        "password":"pwdxxx",
        "reqtime":"2018-8-31 11:54:32"
}
====line:165,request data:{
        "username":"namexxx",
        "password":"pwdxxx",
        "reqtime":"2018-8-31 11:54:32"
```