

ifm_nettle + libevhttp

[安装libhv](#)

[安装ifm_nettle](#)

[调整libevhttp仓库相关文件](#)

[编译](#)

[测试结果概述](#)

[运行测试程序](#)

1. [example_basic](#)
2. [example_chunked](#)
3. [example_pause.c](#)
4. [example_request_fini](#)
5. [example_vhost](#)
6. [test_basic](#)
7. [test_client](#)
8. [test_extensive](#)
9. [test_perf](#)
10. [test_proxy](#)
11. [test_query](#)
12. [test_vhost](#)

安装libhv

```
cd ~  
git clone https://github.com/ithewei/libhv.git  
cd libhv  
mkdir build  
cd build  
cmake ..  
cmake --build .  
cmake --install .
```

安装ifm_nettle

```
cd ~  
// 注意，由于代码尚未合入，这里的git链接是我个人fork的仓库链接  
git clone git@gitee.com:ptbxzrt/ifm_nettle.git ifm_nettle  
cd ifm_nettle  
mkdir build  
cd build  
cmake ..  
make -j  
sudo make install
```

我在VMware（openEuler-22.03-LTS-SP1-x86_64）上测试时，会出现找不到libhv.so的情况，因此还需要执行以下命令才能确保后面不会出错

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

调整libevhttp仓库相关文件

cd ~

git clone -b 1.2.18 https://github.com/Yellow-Camper/libevhttp.git

修改libevhttp/CMakeLists.txt, 使其不再链接到libevent.so, 而是链接到libifm_libhv2ev.so

```
57- find_package(LibEvent REQUIRED)
58- list(APPEND LIBEVHTTP_EXTERNAL_LIBS ${LIBEVENT_LIBRARIES})
59- list(APPEND LIBEVHTTP_EXTERNAL_INCLUDES ${LIBEVENT_INCLUDE_DIRS})
60- list(APPEND package_deps LibEvent)
→ 57+ # find_package(LibEvent REQUIRED)
58+ # list(APPEND LIBEVHTTP_EXTERNAL_LIBS ${LIBEVENT_LIBRARIES})
59+ # list(APPEND LIBEVHTTP_EXTERNAL_INCLUDES ${LIBEVENT_INCLUDE_DIRS})
60+ # list(APPEND package_deps LibEvent)
61+
62+ list(APPEND LIBEVHTTP_EXTERNAL_INCLUDES /usr/local/include/)
63+ list(APPEND LIBEVHTTP_EXTERNAL_INCLUDES /usr/local/include/hv)
64+ list(APPEND LIBEVHTTP_EXTERNAL_LIBS /usr/local/lib/libifm_libhv2ev.so)
```

删除libevhttp/evhttp.c中libevent相关的头文件

```
29 #include <limits.h>
30- #include <event2/dns.h>
31 → 29 #include <limits.h>
30
```

删除libevhttp/thread.c中libevent相关的头文件

```
12 #include <pthread.h>
13- #include <event2/event.h>
14- #include <event2/thread.h>
15 → 12 #include <pthread.h>
13
```

删除libevhttp/evhttp.h中libevent相关的头文件, 并添加ifm_nettle中的头文件hv2ev.h

```
21 #include <sys/queue.h>
22- #include <event2/event.h>
23- #include <event2/listener.h>
24- #include <event2/buffer.h>
25- #include <event2/bufferevent.h>
26 → 21 #include <sys/queue.h>
22+ #include "hv2ev.h"
23+
24
25 #ifndef EVHTTP_DISABLE_SSL
26- #include <event2/bufferevent_ssl.h>
27- #include <openssl/dh.h>
→ 25 #ifndef EVHTTP_DISABLE_SSL
26 #include <openssl/dh.h>
```

删除libevhttp/thread.h中libevent相关的头文件, 并添加ifm_nettle中的头文件hv2ev.h

```
8 #include <pthread.h>
9- #include <event2/event.h>
10 #include <evhttp/config.h>
11 → 8 #include <pthread.h>
9+ #include "hv2ev.h"
10 #include <evhttp/config.h>
11
```

将libevhttp/examples/test.c中的libevent相关头文件替换为ifm_nettle中的头文件hv2ev.h

```
8 #include <inttypes.h>
9- #include <event2/event.h>
10 → 8 #include <inttypes.h>
9+ #include "hv2ev.h"
10
```

编译

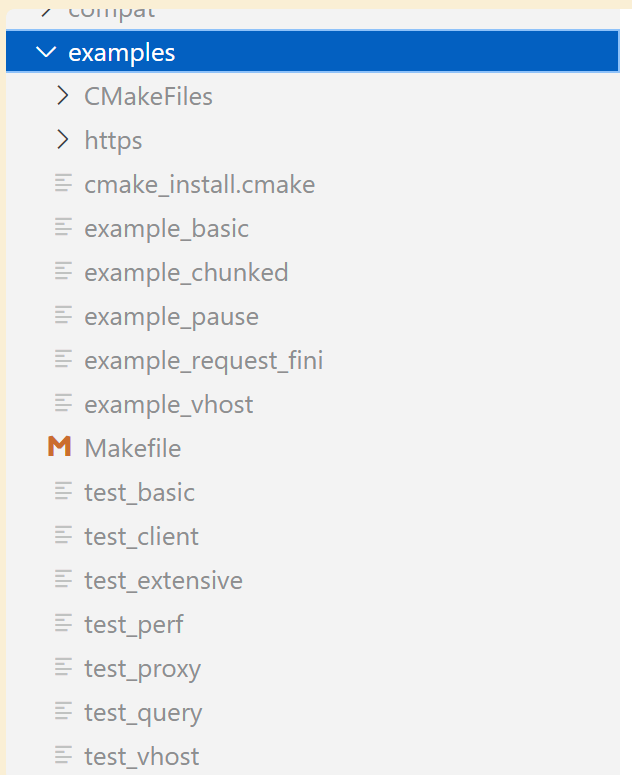
```
mkdir build
```

```
cd build
```

```
cmake ..
```

cmake --build . 该命令编译得到静态库libevhttp/build/libevhttp.a

make examples 该命令编译得到libevhttp/build/examples文件夹下若干二进制测试程序



测试结果概述

- 我用libevhttp提供的示例程序来进行集成测试，同时，我也使用最原始的libevhttp（没有任何改动，底层网络库为libevent）来运行相同的测试程序，以作对比
- 从对比测试的结果来看，基于libhv的libevhttp和基于libevent的libevhttp在测试程序上的行为是一致的。即使有少部分测试程序的运行情况比较奇怪，但两种libevhttp在这些测试下的运行情况是一样的

运行测试程序

1. example_basic

注意，此时是在build目录下。启动example_basic。该程序会启动一个http服务端，响应http客户端的get请求

examples/example_basic

```
[ptbxzrt@localhost build]$ ./examples/example_basic  
[INFO]  example_basic.c:44      Basic server, run: curl http://127.0.0.1:9999/  
|
```

开启另外一个会话窗口，运行以下命令

```
curl http://127.0.0.1:9999/
```

```
curl http://127.0.0.1:9999/
```

```
curl http://127.0.0.1:9999/
```

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:9999/  
[ptbxzrt@localhost build]$ curl http://127.0.0.1:9999/  
[ptbxzrt@localhost build]$ curl http://127.0.0.1:9999/  
[ptbxzrt@localhost build]$ |
```

可以看到，3次httpget指令正常运行并返回

```
[ptbxzrt@localhost build]$ ./examples/example_basic  
[INFO]  example_basic.c:44      Basic server, run: curl http://127.0.0.1:9999/  
127.0.0.1 127.0.0.1:9999 'curl/7.79.1' [23/Oct/2023:01:56:27 +0800] 'GET / HTTP/1.1' 0  
127.0.0.1 127.0.0.1:9999 'curl/7.79.1' [23/Oct/2023:01:56:27 +0800] 'GET / HTTP/1.1' 0  
127.0.0.1 127.0.0.1:9999 'curl/7.79.1' [23/Oct/2023:01:56:28 +0800] 'GET / HTTP/1.1' 0  
|
```

2. example_chunked

启动example_chunked。该程序会启动一个http服务端，且注意启动该程序需要一个参数，这个参数是一个文件名，当收到http客户端请求时，该程序会将参数指代的文件发送给http客户端。这里我直接用的是build目录下的Makefile文件

`./examples/example_chunked Makefile`

```
[ptbxzrt@localhost build]$ ./examples/example_chunked Makefile
[INFO] example_chunked.c:158 curl http://127.0.0.1:9999/
```

开启另外一个会话窗口，运行以下命令

`curl http://127.0.0.1:9999/`

可以看到，example_chunked程序成功将Makefile文件发送给了http客户端

```
@echo "... log.o"
@echo "... log.i"
@echo "... log.s"
@echo "... numtoa.o"
@echo "... numtoa.i"
@echo "... numtoa.s"
@echo "... parser.o"
@echo "... parser.i"
@echo "... parser.s"
@echo "... thread.o"
@echo "... thread.i"
@echo "... thread.s"
.PHONY : help

#=====
# Special targets to cleanup operation of make.

# Special rule to run CMake to check the build system integrity.
# No rule that depends on this can have commands that come from listfiles
# because they might be regenerated.
cmake_check_build_system:
    $(CMAKE_COMMAND) -S$(CMAKE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR) --check-build-system CMakeFiles/Makefile.c
make 0
.PHONY : cmake_check_build_system

[ptbxzrt@localhost build]$
```

[illegible]

3. example_pause.c

启动example_pause。该程序会启动一个http服务端，在响应http客户端的请求时，会延迟10s再返回

`./examples/example_pause`

```
[ptbxzrt@localhost build]$ ./examples/example_pause
[INFO] example_pause.c:106      response delayed for 10s: curl http://127.0.0.1:9999/
```

开启另外一个会话窗口，运行以下命令

```
curl http://127.0.0.1:9999/
```

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:9999/  
time start 1697999691  
time end 1697999701  
[ptbxzrt@localhost build]$
```

4. example_request_fini

启动example_request_fini。该程序会启动一个http服务端，响应http客户端时，向客户端发送"Hello, world"

`./examples/example_request_fini`

```
[ptbxzrt@localhost build]$ ./examples/example_request_fini
[INFO] example_request_fini.c:56      Simple usage of using request_fini hooks, run: curl http://127.0.0.1:9999/x/v/o
```

开启另外一个会话窗口，运行以下命令

`curl http://127.0.0.1:9999/i/q/z`

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:9999/i/q/z
Hello, world
[ptbxzrt@localhost build]$
```

5. example_vhost

启动example_vhost

./examples/example_vhost

```
[ptbxzrt@localhost build]$ ./examples/example_vhost
[INFO] example_vhost.c:100 [[ try the following commands and you should see 'evhttp.io domains' ]]
[INFO] example_vhost.c:101 =====
[INFO] example_vhost.c:102 curl -H'Host: evhttp.io' http://127.0.0.1:9999/vhost
[INFO] example_vhost.c:103 curl -H'Host: www.evhttp.io' http://127.0.0.1:9999/vhost
[INFO] example_vhost.c:104 curl -H'Host: web.evhttp.io' http://127.0.0.1:9999/vhost
[INFO] example_vhost.c:105 =====
[INFO] example_vhost.c:106 [[ try the following commands and you should see 'google.com domains' ]]
[INFO] example_vhost.c:107 =====
[INFO] example_vhost.c:108 curl -H'Host: google.com' http://127.0.0.1:9999/vhost
[INFO] example_vhost.c:109 curl -H'Host: www.google.com' http://127.0.0.1:9999/vhost
[INFO] example_vhost.c:110 curl -H'Host: web.google.com' http://127.0.0.1:9999/vhost
[INFO] example_vhost.c:111 curl -H'Host: inbox.google.com' http://127.0.0.1:9999/vhost
[INFO] example_vhost.c:112 curl -H'Host: gmail.google.com' http://127.0.0.1:9999/vhost
```

开启另外一个会话窗口，运行以下命令

curl -H'Host: evhttp.io' http://127.0.0.1:9999/vhost

curl -H'Host: www.evhttp.io' http://127.0.0.1:9999/vhost

curl -H'Host: web.evhttp.io' http://127.0.0.1:9999/vhost

curl -H'Host: google.com' http://127.0.0.1:9999/vhost

curl -H'Host: www.google.com' http://127.0.0.1:9999/vhost

curl -H'Host: web.google.com' http://127.0.0.1:9999/vhost

curl -H'Host: inbox.google.com' http://127.0.0.1:9999/vhost

curl -H'Host: gmail.google.com' http://127.0.0.1:9999/vhost

```
[ptbxzrt@localhost build]$ curl -H'Host: evhttp.io' http://127.0.0.1:9999/vhost
vhost_1__callback_ = host:evhttp.io, arg:evhttp.io domains
[ptbxzrt@localhost build]$ curl -H'Host: www.evhttp.io' http://127.0.0.1:9999/vhost
vhost_1__callback_ = host:www.evhttp.io, arg:evhttp.io domains
[ptbxzrt@localhost build]$ curl -H'Host: web.evhttp.io' http://127.0.0.1:9999/vhost
vhost_1__callback_ = host:web.evhttp.io, arg:evhttp.io domains
[ptbxzrt@localhost build]$ curl -H'Host: google.com' http://127.0.0.1:9999/vhost
vhost_2__callback_ = host:google.com, arg:google.com domains
[ptbxzrt@localhost build]$ curl -H'Host: www.google.com' http://127.0.0.1:9999/vhost
vhost_2__callback_ = host:www.google.com, arg:google.com domains
[ptbxzrt@localhost build]$ curl -H'Host: web.google.com' http://127.0.0.1:9999/vhost
vhost_2__callback_ = host:web.google.com, arg:google.com domains
[ptbxzrt@localhost build]$ curl -H'Host: inbox.google.com' http://127.0.0.1:9999/vhost
vhost_2__callback_ = host:inbox.google.com, arg:google.com domains
[ptbxzrt@localhost build]$ curl -H'Host: gmail.google.com' http://127.0.0.1:9999/vhost
vhost_2__callback_ = host:gmail.google.com, arg:google.com domains
[ptbxzrt@localhost build]$ |
```

6. test_basic

启动test_basic

```
[ptbxzrt@localhost build]$ ./examples/test_basic
```

开启另外一个会话窗口，运行以下命令

```
curl http://127.0.0.1:8081/simple/
```

```
curl http://127.0.0.1:8081/1/ping
```

```
curl http://127.0.0.1:8081/1/ping.json
```

```
curl http://127.0.0.1:8081/issue161
```

客户端得到的结果似乎有点奇怪，因此我还使用了最原始的、链接到libevent.so的libevhttp来进行测试。对比二者的结果，发现是一致的

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/simple/  
simple[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/1/ping  
one[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/1/ping.json  
one[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/issue161  
[ptbxzrt@localhost build]$ |
```

7. test_client

这里需要修改一下test_client.c中的源码。test_client程序会启动1个http客户端，向指定IP地址发送http请求。test_client.c中原本给出的ip地址无法使用，因此我通过Windows上的nslookup命令获取了www.bilibili.com的IP地址121.194.11.73，并将test_client.c中指定的IP地址替换为该地址

```
54     evbase = event_base_new();
55     conn    = evhttp_connection_new(evbase, "121.194.11.73", 80);
56     request = evhttp_request_new(request_cb, evbase);
57
```

重新编译测试文件

make examples

启动test_client，它会立即返回

./examples/test_client

```
[ptbxzrt@localhost build]$ ./examples/test_client
Got 9 bytes
hi 9
[ptbxzrt@localhost build]$ |
```

8. test_extensive

test_extensive程序是由test.c编译生成的。启动test_extensive

`./examples/test_extensive`

```
[ptbxzrt@localhost build]$ ./examples/test_extensive
```

开启另外一个会话窗口，运行以下命令

`curl http://127.0.0.1:8081/ref`

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/ref
print_path() full      = '/ref'
               path    = '/'
               file     = 'ref'
               match start = '/ref'
               match_end = ''
               methno   = '0'
print_kv() key = 'Host', val = '127.0.0.1:8081'
print_kv() key = 'User-Agent', val = 'curl/7.79.1'
print_kv() key = 'Accept', val = '*/*'
print_kvs() key = 'Host', val = '127.0.0.1:8081'
print_kvs() key = 'User-Agent', val = 'curl/7.79.1'
print_kvs() key = 'Accept', val = '*/*'
test_default_cb
```

`curl http://127.0.0.1:8081/foo`

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/foo
print_path() full      = '/foo'
               path    = '/'
               file     = 'foo'
               match start = '/foo'
               match_end = ''
               methno   = '0'
print_kv() key = 'Host', val = '127.0.0.1:8081'
print_kv() key = 'User-Agent', val = 'curl/7.79.1'
print_kv() key = 'Accept', val = '*/*'
print_kvs() key = 'Host', val = '127.0.0.1:8081'
print_kvs() key = 'User-Agent', val = 'curl/7.79.1'
print_kvs() key = 'Accept', val = '*/*'
test_foo_cb
```

`curl http://127.0.0.1:8081/foo/`

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/foo/
print_path() full      = '/foo/'
               path     = '/foo/'
               file     = '(null)'
               match start = '/foo/'
               match_end = ''
               methno    = '0'
print_kv() key = 'Host', val = '127.0.0.1:8081'
print_kv() key = 'User-Agent', val = 'curl/7.79.1'
print_kv() key = 'Accept', val = '*/*'
print_kvs() key = 'Host', val = '127.0.0.1:8081'
print_kvs() key = 'User-Agent', val = 'curl/7.79.1'
print_kvs() key = 'Accept', val = '*/*'
test_foo_cb
```

curl http://127.0.0.1:8081/bar

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/bar
print_path() full      = '/bar'
               path     = '/'
               file     = 'bar'
               match start = '/bar'
               match_end = ''
               methno    = '0'
print_kv() key = 'Host', val = '127.0.0.1:8081'
print_kv() key = 'User-Agent', val = 'curl/7.79.1'
print_kv() key = 'Accept', val = '*/*'
print_kvs() key = 'Host', val = '127.0.0.1:8081'
print_kvs() key = 'User-Agent', val = 'curl/7.79.1'
print_kvs() key = 'Accept', val = '*/*'
test_bar_cb
```

curl http://127.0.0.1:8081/500

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/500
print_path() full      = '/500'
               path     = '/'
               file     = '500'
               match start = '/500'
               match_end = ''
               methno    = '0'
print_kv() key = 'Host', val = '127.0.0.1:8081'
print_kv() key = 'User-Agent', val = 'curl/7.79.1'
print_kv() key = 'Accept', val = '*/*'
print_kvs() key = 'Host', val = '127.0.0.1:8081'
print_kvs() key = 'User-Agent', val = 'curl/7.79.1'
print_kvs() key = 'Accept', val = '*/*'
test_500_cb
```

curl http://127.0.0.1:8081/pause

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/pause
print_kvs() key = 'Host', val = '127.0.0.1:8081'
print_kvs() key = 'User-Agent', val = 'curl/7.79.1'
print_kvs() key = 'Accept', val = '*/*'
test_pause_cb
```

curl http://127.0.0.1:8081/glob/

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/glob/
print_path() full      = '/glob/'
           path       = '/glob/'
           file       = '(null)'
           match start = '/glob/'
           match_end   = ''
           methno      = '0'
print_kv() key = 'Host', val = '127.0.0.1:8081'
print_kv() key = 'User-Agent', val = 'curl/7.79.1'
print_kv() key = 'Accept', val = '*/*'
print_kvs() key = 'Host', val = '127.0.0.1:8081'
print_kvs() key = 'User-Agent', val = 'curl/7.79.1'
print_kvs() key = 'Accept', val = '*/*'
test_glob_cb
```

curl http://127.0.0.1:8081/max_body_size

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/max_body_size
print_path() full      = '/max_body_size'
           path       = '/'
           file       = 'max_body_size'
           match start = '/max_body_size'
           match_end   = ''
           methno      = '0'
print_kv() key = 'Host', val = '127.0.0.1:8081'
print_kv() key = 'User-Agent', val = 'curl/7.79.1'
print_kv() key = 'Accept', val = '*/*'
test_max_body
```

curl http://127.0.0.1:8081/chunkme

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/chunkme
print_path() full      = '/chunkme'
           path       = '/'
           file       = 'chunkme'
           match start = '/chunkme'
           match_end   = ''
           methno      = '0'
print_kv() key = 'Host', val = '127.0.0.1:8081'
print_kv() key = 'User-Agent', val = 'curl/7.79.1'
print_kv() key = 'Accept', val = '*/*'
print_kvs() key = 'Host', val = '127.0.0.1:8081'
print_kvs() key = 'User-Agent', val = 'curl/7.79.1'
print_kvs() key = 'Accept', val = '*/*'
I give you the light of Eärendil,
our most beloved star.
May it be a light for you in dark places,
when all other lights go out.
```

curl http://127.0.0.1:8081/ownme

这条curl命令会一直卡住，不会返回，看起来很奇怪。因此我又使用最原始的libevhttp做了实验，发现的现象也是一直卡住，也就是说，在该命令下，二者的表现情况是一样的

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/ownme
```

9. test_perf

test_perf程序有一段复杂的处理命令行参数的代码，为了方便测试，我将该段代码直接注释掉，这样test_perf程序在启动时会使用默认的参数设置


```

40 // while ((c = getopt(argc, argv, "t:a:p:b:ndrs:")) != -1)
41 // {
42 //     switch (c) {
43 //         case 't':
44 //             num_threads = atoi(optarg);
45 //             break;
46 //         case 'a':
47 //             baddr       = strdup(optarg);
48 //             break;
49 //         case 'p':
50 //             bport       = atoi(optarg);
51 //             break;
52 //         case 'b':
53 //             backlog     = atoll(optarg);
54 //             break;
55 //         case 'n':
56 //             nodelay     = 1;
57 //             break;
58 //         case 'd':
59 //             defer_accept = 1;
60 //             break;
61 //         case 'r':
62 //             reuse_port  = 1;
63 //             break;
64 //         case 's':
65 //             payload_sz  = atoll(optarg);
66 //             break;
67 //         default:
68 //             fprintf(stdout, "Usage: %s [flags]\n", argv[0]);
69 //             fprintf(stdout, "  -t <n> : number of worker threads\n");
70 //             fprintf(stdout, "  -a <s> : bind address\n");
71 //             fprintf(stdout, "  -p <n> : bind port\n");
72 //             fprintf(stdout, "  -b <b> : listen backlog\n");
73 //             fprintf(stdout, "  -s <n> : size of the response\n");
74 //             fprintf(stdout, "  -n      : disable nagle (nodelay)\n");
75 //             fprintf(stdout, "  -d      : enable deferred accept\n");
76 //             fprintf(stdout, "  -r      : enable linux reuseport\n");
77 //             exit(EXIT_FAILURE);
78 //     } /* switch */
79 // }
80

```

`./examples/test_perf`

```
[ptbxzrt@localhost build]$ ./examples/test_perf
```

开启另外一个会话窗口，运行以下命令

`curl http://127.0.0.1:8081/data`

```
[ptbxzrt@localhost build]$ curl http://127.0.0.1:8081/data
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB[ptbxzrt@lo
calhost build]$ |
```

10. test_proxy

test_proxy会作为1个代理，转发http客户端的请求。但test_proxy.c的代码中，将远端服务器写死为本地地址，因此需要将其替换为1个可用的地址才行

73	make_request(evthr_get_base(req->conn->thr	73	make_request(evthr_get_base(req->conn->
74	req->conn->thread,	74	req->conn->thread,
75-	"127.0.0.1", 80,	→ 75+	"123.234.3.168", 80,
76	req->uri->path->full,	76	req->uri->path->full,
77	req->headers_in, backend_cb,	77	req->headers_in, backend_c
78		78	

启动test_proxy，注意重新编译一下

make examples

./examples/test_proxy

```
[ptbxzrt@localhost build]$ ./examples/test_proxy
Spinning up a thread: 1
Spinning up a thread: 2
Spinning up a thread: 3
Spinning up a thread: 4
Spinning up a thread: 5
Spinning up a thread: 6
Spinning up a thread: 7
Spinning up a thread: 8
|
```

开启另外一个会话窗口，运行以下命令

curl -x http://127.0.0.1:8081 http://www.baidu.com

```
[ptbxzrt@localhost ifm_nettle]$ curl -x http://127.0.0.1:8081 http://www.baidu.com
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
<head><title>400 Bad Request</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
  Sorry for the inconvenience.<br/>
  Please report this message and include the following information to us.<br/>
  Thank you very much!</p>
<table>
<tr>
<td>URL:</td>
<td>http://localhost/</td>
</tr>
<tr>
<td>Server:</td>
<td>cn-sdqd-cu-w-v6-03</td>
</tr>
</table>
```

11. test_query

test_query和底层网络IO没有关系，它主要是测试http头文件解析的功能

./examples/test_query

```
OK      a=b;key=val                {"a": "b;key=val"}
OK      a;b=val                    {"a;b": "val"}
OK      end_empty_string=          (error) <error>
OK      end_null                   (error) <error>
OK      hexa=some%20&hexb=bla%0    {"hexa": "some%20", "hexb": "bla%0"}
OK      hexa=some%20;hexb=bla      {"hexa": "some%20;hexb=bla"}
OK      hexa%z=some               {"hexa%z": "some"}
OK      aaa=some%az               (error) <error>
- ignore_hex_tests
OK      hexa=some%20&hexb=bla%0&hexc=% {"hexa": "some%20", "hexb": "bla%0", "hexc": "%"}
OK      hexa%z=some               {"hexa%z": "some"}
OK      aaa=some%zz               {"aaa": "some%zz"}
- allow_empty_tests
OK      end_empty_string=          {"end_empty_string": ""}
- allow_null_tests
OK      end_null                   {"end_null": (null)}
- treat_semicolon_as_sep_tests
OK      a=b;key=val                {"a": "b", "key": "val"}
OK      a;b=val                   (error) <error>
- lenient_tests
OK      a=b;key&c=val              {"a": "b", "key": (null), "c": "val"}
OK      a=b;key=val                {"a": "b", "key": "val"}
OK      end_empty_string=          {"end_empty_string": ""}
OK      end_null                   {"end_null": (null)}
OK      hexa=some%a;hexb=bl%0&hexc=%az {"hexa": "some%a", "hexb": "bl%0", "hexc": "%az"}

[ptbxzrt@localhost build]$ |
```

12. test_vhost

启动test_vhost

`./examples/test_vhost`

```
[ptbxzrt@localhost build]$ ./examples/test_vhost
```

开启另外一个会话窗口，运行以下命令

```
curl -i -X POST http://127.0.0.1:8081/host1
```

```
curl -i -X POST http://127.0.0.1:8081/localhost
```

返回404，奇怪的结果，因此我又使用最原始的libevhttp做了实验，发现也是返回404

```
[ptbxzrt@localhost build]$ curl -i -X POST http://127.0.0.1:8081/host1
HTTP/1.1 404 Not Found
Content-Length: 0
Content-Type: text/plain
```

```
[ptbxzrt@localhost build]$ curl -i -X POST http://127.0.0.1:8081/localhost
HTTP/1.1 404 Not Found
Content-Length: 0
Content-Type: text/plain
```