<http://rtxbc.iteye.com/blog/1121419>

**任务:**

**1.制作http服务器，读取url提交的相关数据.**

**2.把读到的数据推入到队列中.**

条件:

使用libevent的类库,所以先安装libevent

**Sh代码  [收藏代码](javascript:void())**

1. tar zxvf libevent-2.0.12-stable.tar.gz
2. cd libevent-2.0.12-stable/
3. ./configure --prefix=/usr/local/libevent-2.0.12-stable/
4. make
5. make install
6. cd ../

**1.服务端简易代码如下**

**C代码  [收藏代码](javascript:void())**

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include <err.h>
5. #include <event.h>
6. #include <evhttp.h>
8. **void** http\_handle(**struct** evhttp\_request \*req, **void** \*arg); /\*  HTTP Request Handle  \*/
10. **int** main(){
11. **struct** evhttp \*httpd;
12. event\_init();
13. httpd = evhttp\_start("0.0.0.0", 2345);
14. **if** (httpd == NULL) {
15. fprintf(stderr, "Error: Unable to listen on %s:%d\n\n");
16. exit(1);
17. }
18. evhttp\_set\_timeout(httpd, 2000);
19. evhttp\_set\_gencb(httpd, http\_handle, NULL);
20. event\_dispatch();
21. evhttp\_free(httpd);
23. **return** 0;
24. }
26. **void** http\_handle(**struct** evhttp\_request \*req, **void** \*arg){
27. **struct** evbuffer \*buf;
28. buf = evbuffer\_new();
30. /\*  Response the client  \*/
31. evhttp\_send\_reply(req, HTTP\_OK, "OK", buf);
33. //evbuffer\_add\_printf(buf, "%s", "HTTPSQS\_AUTH\_FAILED");
35. /\*  Release the memory  \*/
36. evbuffer\_free(buf);
37. fprintf(stderr,"Send \n");
38. }

编译：(编译时把libevent的类库中的.so文件和.h文件**连接** 进来)

**Sh代码  [收藏代码](javascript:void())**

1. gcc http.c -L/usr/local/libevent-2.0.12-stable/lib/ -levent -I/usr/local/libevent-2.0.12-stable/include/

测试

在服务器端，执行编译后的文件a.out

**Java代码  [收藏代码](javascript:void())**

1. ./a.out

客户端进行访问(也可以使用浏览器访问，但是要打开端口号2345——vi /etc/sysconfig/iptables)

**Java代码  [收藏代码](javascript:void())**

1. [www@zhoubc data]$ curl http://127.0.0.1:2345

此时再看服务端,变成如下状态

**Java代码  [收藏代码](javascript:void())**

1. [www@zhoubc queue]$ ./a.out
2. Send

**2.处理http请求,重写htt\_handle方法**

**C代码  [收藏代码](javascript:void())**

1. **void** http\_handle(**struct** evhttp\_request \*req, **void** \*arg){
2. **struct** evbuffer \*buf;
3. buf = evbuffer\_new();
5. /\*  Analyst the URI  \*/
6. **char** \*decode\_uri = strdup((**char**\*) evhttp\_request\_uri(req));
7. **struct** evkeyvalq http\_query;
8. evhttp\_parse\_query(decode\_uri, &http\_query);
9. free(decode\_uri);
11. /\*  URI Parameter  \*/
12. **const** **char** \*http\_input\_opt = evhttp\_find\_header (&http\_query, "opt"); /\* Operation Type \*/
13. **const** **char** \*http\_input\_name = evhttp\_find\_header (&http\_query, "name"); /\* Queue Name \*/
14. **const** **char** \*http\_input\_data = evhttp\_find\_header (&http\_query, "data"); /\* Data With GET \*/
16. /\*  header  \*/
17. evhttp\_add\_header(req->output\_headers, "Content-Type", "text/plain");
18. evhttp\_add\_header(req->output\_headers, "Connection", "keep-alive");
19. evhttp\_add\_header(req->output\_headers, "Cache-Control", "no-cache");
20. evhttp\_add\_header(req->output\_headers, "author", "Dennis .Z Ritchie");
22. **if**(http\_input\_opt != NULL && http\_input\_name != NULL && strlen(http\_input\_name) < 300){
23. /\*  GET Method,OUT The Queue  \*/
24. **if**(strcmp(http\_input\_opt,"put") == 0){
25. **int** buffer\_data\_len = EVBUFFER\_LENGTH(req->input\_buffer);
26. **if**(buffer\_data\_len > 0){ /\* POST METHOD \*/
27. **char** \*input\_value\_data = EVBUFFER\_DATA(req->input\_buffer); /\* Submited Data \*/
28. fprintf(stderr,"%s \n",input\_value\_data);
29. }**else** **if**(http\_input\_data != NULL){
30. fprintf(stderr,"%s \n",http\_input\_data);
31. }
32. }**else** **if**(strcmp(http\_input\_opt,"get") == 0){
33. }
34. }
36. /\*  Response the client  \*/
37. evhttp\_send\_reply(req, HTTP\_OK, "OK", buf);
39. //evbuffer\_add\_printf(buf, "%s", "HTTPSQS\_AUTH\_FAILED");
41. /\*  Release the memory  \*/
42. evhttp\_clear\_headers(&http\_query);
44. evbuffer\_free(buf);
45. }