<https://www.cnblogs.com/charlesblc/p/6286654.html>

之前有一篇文章描述过一些对比：

http://www.cnblogs.com/charlesblc/p/6078029.html

这里在代码和应用方面再说一下。

看一下两边的Helloworld基本就了解个大概。

libevent的hello world

[复制代码](javascript:void(0);)

#include <sys/signal.h>

#include <event.h>

void signal\_cb(int fd, short event, void \*arg) {

event\_base \*base = (event\_base\*) arg;

timeval delay = {2, 0};

printf("Caught an interrupt signal, exit in 2 sec...\n");

event\_base\_loopexit(base, &delay);

}

void timeout\_cb(int fd, short event, void \*arg) {

printf("timeout\n");

}

int main() {

event\_base \*base = event\_init();

event \*signal\_event = evsignal\_new(base, SIGINT, signal\_cb, base);

event\_add(signal\_event, NULL);

timeval tv = {1, 0};

event \* timeout\_event = evtimer\_new(base, timeout\_cb, NULL);

event\_add(timeout\_event, &tv);

event\_base\_dispatch(base);

event\_free(timeout\_event);

event\_free(signal\_event);

event\_base\_free(base);

}

[复制代码](javascript:void(0);)

上面Libevent最基本的事件驱动器是 event\_base \*base = event\_init(); 下面libev最基本的事件驱动器是 struct ev\_loop \*main\_loop = ev\_default\_loop(0);

上面Libevent的事件，在初始化的时候，会把驱动器加进去，event \*signal\_event = evsignal\_new(base, SIGINT, signal\_cb, base); 然后直接event\_add就行了，这时候不用加驱动器；

下面libev的事情，在初始化的时候，不涉及驱动器，先关联事件和回调函数：ev\_init(&io\_w,io\_action); 再绑定事件源：ev\_io\_set(&io\_w,STDIN\_FILENO,EV\_READ);  最后再向驱动器注册：ev\_io\_start(main\_loop,&io\_w);

上面Libevent开始事件循环用 event\_base\_dispatch(base); 下面libev开始事件循环用 ev\_run(main\_loop,0);

libev的hello world

[复制代码](javascript:void(0);)

#include<ev.h>

#include <stdio.h>

#include <signal.h>

#include <sys/unistd.h>

ev\_io io\_w;

ev\_timer timer\_w;

ev\_signal signal\_w;

void io\_action(struct ev\_loop \*main\_loop,ev\_io \*io\_w,int e)

{

int rst;

char buf[1024] = {'\0'};

puts("in io cb\n");

read(STDIN\_FILENO,buf,sizeof(buf));

buf[1023] = '\0';

printf("Read in a string %s \n",buf);

ev\_io\_stop(main\_loop,io\_w);

}

void timer\_action(struct ev\_loop \*main\_loop,ev\_timer \*timer\_w,int e)

{

puts("in tiemr cb \n");

ev\_timer\_stop(main\_loop,timer\_w);

}

void signal\_action(struct ev\_loop \*main\_loop,ev\_signal signal\_w,int e)

{

puts("in signal cb \n");

ev\_signal\_stop(main\_loop,signal\_w);

ev\_break(main\_loop,EVBREAK\_ALL);

}

int main(int argc ,char \*argv[])

{

struct ev\_loop \*main\_loop = ev\_default\_loop(0);

ev\_init(&io\_w,io\_action);

ev\_io\_set(&io\_w,STDIN\_FILENO,EV\_READ);

ev\_init(&timer\_w,timer\_action);

ev\_timer\_set(&timer\_w,2,0);

ev\_init(&signal\_w,signal\_action);

ev\_signal\_set(&signal\_w,SIGINT);

ev\_io\_start(main\_loop,&io\_w);

ev\_timer\_start(main\_loop,&timer\_w);

ev\_signal\_start(main\_loop,&signal\_w);

ev\_run(main\_loop,0);

return 0;

}