<https://blog.csdn.net/luotuo44/article/details/38512719>

**转载请注明出处:**[**http://blog.csdn.net/luotuo44/article/details/38512719**](http://blog.csdn.net/luotuo44/article/details/38512719)

        event\_base允许用户对它里面的event设置优先级，这样可以使得有些更重要的event能够得到优先处理。

        Libevent实现优先级功能的方法是：用一个激活队列数组来存放激活event。即数组的元素是一个激活队列，所以有多个激活队列。并且规定不同的队列有不同的优先级。

        可以通过event\_base\_priority\_init函数设置event\_base的优先级个数，该函数实现如下：

**[cpp]** [view plain](https://blog.csdn.net/luotuo44/article/details/38512719) [copy](https://blog.csdn.net/luotuo44/article/details/38512719)

1. //event.c文件
2. **int**
3. event\_base\_priority\_init(**struct** event\_base \*base, **int** npriorities)
4. {
5. **int** i;
7. //由N\_ACTIVE\_CALLBACKS宏可以知道，本函数应该要在event\_base\_dispatch
8. //函数调用前调用。不然将无法设置。
9. **if** (N\_ACTIVE\_CALLBACKS(base) || npriorities < 1
10. || npriorities >= EVENT\_MAX\_PRIORITIES)
11. **return** (-1);
13. //之前和现在要设置的优先级数是一样的。
14. **if** (npriorities == base->nactivequeues)
15. **return** (0);
17. //释放之前的，因为N\_ACTIVE\_CALLBACKS,所以没有active的event。
18. //可以随便mm\_free
19. **if** (base->nactivequeues) {
20. mm\_free(base->activequeues);
21. base->nactivequeues = 0;
22. }
24. //分配一个优先级数组。
25. base->activequeues = (**struct** event\_list \*)
26. mm\_calloc(npriorities, **sizeof**(**struct** event\_list));
27. **if** (base->activequeues == NULL) {
28. event\_warn("%s: calloc", \_\_func\_\_);
29. **return** (-1);
30. }
31. base->nactivequeues = npriorities;
33. **for** (i = 0; i < base->nactivequeues; ++i) {
34. TAILQ\_INIT(&base->activequeues[i]);
35. }
37. **return** (0);
38. }

        从前面一个判断可知，因为event\_base\_dispatch函数会改动激活事件的个数，即会使得N\_ACTIVE\_CALLBACKS(base)为真。所以event\_base\_priority\_init函数要在event\_base\_dispatch函数之前调用。此外要设置的优先级个数，要小于EVENT\_MAX\_PRIORITIES。这个宏是在event.h文件中定义，在2.0.21版本中，该宏被定义成256。在调用event\_base\_new得到的event\_base只有一个优先级，也就是所有event都是同级的。

        上面的代码调用mm\_alloc分配了一个优先级数组。不同优先级的event会被放到数组的不同位置上(下面可以看到这一点)。这样就可以区分不同event的优先级了。以后处理event时，就可以从高优先级到低优先级处理event。

        上面是设置event\_base的优先级个数。现在来看一下怎么设置event的优先级。可以通过event\_priority\_set函数设置，该函数如下：

**[cpp]** [view plain](https://blog.csdn.net/luotuo44/article/details/38512719) [copy](https://blog.csdn.net/luotuo44/article/details/38512719)

1. //event.c文件
2. **int**
3. event\_priority\_set(**struct** event \*ev, **int** pri)
4. {
5. \_event\_debug\_assert\_is\_setup(ev);
7. **if** (ev->ev\_flags & EVLIST\_ACTIVE)
8. **return** (-1);
10. //优先级不能越界
11. **if** (pri < 0 || pri >= ev->ev\_base->nactivequeues)
12. **return** (-1);
14. //pri值越小，其优先级就越高。
15. ev->ev\_pri = pri;
17. **return** (0);
18. }

        在上面代码的第一个判断中，可以知道当event的状态是EVLIST\_ACTIVE时，就不能对这个event进行优先级设置了。因此，如果要对event进行优先级设置，那么得在调用event\_base\_dispatch函数之前。因为一旦调用了event\_base\_dispatch，那么event就随时可能变成EVLIST\_ACTIVE状态。

        现在看一下一个event是怎么插入到event\_base的优先级数组中。

**[cpp]** [view plain](https://blog.csdn.net/luotuo44/article/details/38512719) [copy](https://blog.csdn.net/luotuo44/article/details/38512719)

1. //event.c文件
2. //event\_active\_nolock以event\_queue\_insert(base, ev, EVLIST\_ACTIVE);方式调用
3. **static** **void**
4. event\_queue\_insert(**struct** event\_base \*base, **struct** event \*ev, **int** queue)
5. {
6. ....
8. ev->ev\_flags |= queue; //加入EVLIST\_ACTIVE状态
9. **switch** (queue) {
10. ...
11. **case** EVLIST\_ACTIVE:
12. base->event\_count\_active++;
13. //从优先级数组中找到对应的优先级
14. TAILQ\_INSERT\_TAIL(&base->activequeues[ev->ev\_pri],
15. ev,ev\_active\_next);
16. **break**;
17. ...
18. }
19. }

        最后，我们来看一下默认的event优先级是多少。想必大家都能想到这个默认优先级是在新建event结构体时设置的。不错，看下面的event\_assign函数。

**[cpp]** [view plain](https://blog.csdn.net/luotuo44/article/details/38512719) [copy](https://blog.csdn.net/luotuo44/article/details/38512719)

1. //event.c文件  由event\_new函数调用本函数
2. **int**
3. event\_assign(**struct** event \*ev, **struct** event\_base \*base, evutil\_socket\_t fd,
4. **short** events, **void** (\*callback)(evutil\_socket\_t, **short**, **void** \*), **void** \*arg)
5. {
6. ...
8. ev->ev\_base = base;
9. ev->ev\_callback = callback;
10. ev->ev\_arg = arg;
11. ev->ev\_fd = fd;
12. ev->ev\_events = events;
13. ev->ev\_res = 0;
14. ev->ev\_flags = EVLIST\_INIT;
15. ev->ev\_ncalls = 0;
16. ev->ev\_pncalls = NULL;
18. ....
20. **if** (base != NULL) {
21. /\* by default, we put new events into the middle priority \*/
22. ev->ev\_pri = base->nactivequeues / 2;//默认优先级
23. }
25. ...
26. }

        在这个函数里面，对event的成员变量进行了一些设置。其中，优先级的设置值为优先级数组长度的一半，所以是中间优先级。

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/luotuo44/article/details/38512719