**[重复使用bufferevent的一些体会](https://www.cnblogs.com/fgokey/p/5123805.html)**

　　看了一些网上使用bufferevent的例子，一般都是一个连接对应一个bufferevent，连接accept的时候，给bufferevent设置上fd和对应的回调，在连接断开或者发生错误的时候，将bufferevent释放掉。

昨天在使用bufferevent的时候，突发奇想，准备了一个connection对象池，每个connection关联一个bufferevent，然后多线程处理connection的数据收发，每个线程一个connection链表。在connection出错或者关闭的时候，清空connection的相关状态，重新放入队列中。

1.每次有新的连接被accept后取出队列里空闲的connection，设置相关fd和回掉

[复制代码](javascript:void(0);)

1 bool Connection::SetupConnection(evutil\_socket\_t fd)

2 {

3 m\_fd = fd;

4 evutil\_make\_socket\_nonblocking(fd);

5 bufferevent\_setcb(m\_be, DoRead, NULL, DoError, this);

6 bufferevent\_setfd(m\_be, fd);

7 if (bufferevent\_enable(m\_be, EV\_READ|EV\_WRITE) != -1)

8 {

9 m\_bUsed = true;

10 }

11 return m\_bUsed;

12 }

[复制代码](javascript:void(0);)

2.在event\_cb中，如果收到EOF或者Error的时候，关闭connection

[复制代码](javascript:void(0);)

1 void Connection::DoEvent(struct bufferevent \*bev, short what, void \*ctx)

2 {

3 Connection \*pConn = (Connection\*)ctx;

4 if(what & (BEV\_EVENT\_ERROR|BEV\_EVENT\_EOF))

5 {

6 pConn->CloseConnection();

7 }

8 ......

9 }

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

void Connection::CloseConnection()

{

//因为CloseConnection是在libevent线程的Event回掉中调用的，直接

//bufferevent\_setfd就会自动把之前的event给delete掉

//bufferevent\_disable(m\_be, EV\_READ | EV\_WRITE);

bufferevent\_setfd(m\_be, -1);

evutil\_closesocket(m\_fd);

m\_fd = -1;

//当bufferevent\_setfd设置fd是-1的时候，并不会将write和read的event

//加入到event\_base,如果event\_base中监听的event数量为0时候，

//event\_base\_loop会退出，导致libevent线程结束，所以这里重新手动

//开启write,read事件

bufferevent\_enable(m\_be, EV\_READ|EV\_WRITE) ;

m\_bUsed = false;

}

[复制代码](javascript:void(0);)

3.在有新的客户端accept的时候，重新调用SetupConnection建立新的connection连接。

　　总所周知，bufferevent有个管理收发数据的利器evbuffer，一般发送数据都是调用bufferevent\_write，将数据写入发送的evbuffer里，在socket可以发送数据的时候再发送数据，read也是这样，将socket中的数据先读取到接收的evbuffer里，在调用read\_cb通知用户数据可读。那么，问题来了，在CloseConnection里调用的一系列bufferevent接口中，这两个evbuffer的数据并没有清空，这样就会造成在重新SetupConnection的时候，bufferevent会有一部分之前的fd残留下来的数据，导致新的fd在处理接收和发送数据的时候被干扰。

　　第一反应肯定是将evbuffer里的数据清空掉，于是乎就想到evbuffer\_drain接口来清理数据。清理接收evbuffer的时候，没问题，而且一般在socket断开或者出错后，也会一次性将接收evbuffer里的数据都读取出来处理的。但是在清理发送evbuffer的时候，发现evbuffer\_drain失败，于是乎翻了下libevent的源码：

[复制代码](javascript:void(0);)

int

evbuffer\_drain(struct evbuffer \*buf, size\_t len)

{

struct evbuffer\_chain \*chain, \*next;

size\_t remaining, old\_len;

int result = 0;

EVBUFFER\_LOCK(buf);

old\_len = buf->total\_len;

if (old\_len == 0)

goto done;

　　 //在这个地方，直接goto done了

if (buf->freeze\_start) {

result = -1;

goto done;

}

....

}

[复制代码](javascript:void(0);)

buf->freeze\_start为1，直接结束了，并没有清空evbuffer。参考了一下http://blog.csdn.net/zhwei\_87/article/details/43304383的源码分析后发现bufferevent的发送evbuffer只有在真正发送数据的时候才会unfreeze start，其他时间都是freeze的（所以只允许你将新数据写入到发送evbuffer里，因为写是写到end的，drain是由libevent自己处理的），这也很好理解，因为发送的evbuffer里的数据是不允许用户自己随意drain start的（头部的数据是要发送的），关于start跟end，是因为evbuffer是一个链表结构来存放发送数据的。代码如下，这个函数是libevent真正用来发送数据的回掉，用户的回掉在这个函数最后才会执行。

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttps://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

static void

bufferevent\_writecb(evutil\_socket\_t fd, short event, void \*arg)

{

struct bufferevent \*bufev = arg;

struct bufferevent\_private \*bufev\_p =

EVUTIL\_UPCAST(bufev, struct bufferevent\_private, bev);

int res = 0;

short what = BEV\_EVENT\_WRITING;

int connected = 0;

ev\_ssize\_t atmost = -1;

\_bufferevent\_incref\_and\_lock(bufev);

if (event == EV\_TIMEOUT) {

/\* Note that we only check for event==EV\_TIMEOUT. If

\* event==EV\_TIMEOUT|EV\_WRITE, we can safely ignore the

\* timeout, since a read has occurred \*/

what |= BEV\_EVENT\_TIMEOUT;

goto error;

}

if (bufev\_p->connecting) {

int c = evutil\_socket\_finished\_connecting(fd);

/\* we need to fake the error if the connection was refused

\* immediately - usually connection to localhost on BSD \*/

if (bufev\_p->connection\_refused) {

bufev\_p->connection\_refused = 0;

c = -1;

}

if (c == 0)

goto done;

bufev\_p->connecting = 0;

if (c < 0) {

event\_del(&bufev->ev\_write);

event\_del(&bufev->ev\_read);

\_bufferevent\_run\_eventcb(bufev, BEV\_EVENT\_ERROR);

goto done;

} else {

connected = 1;

#ifdef WIN32

if (BEV\_IS\_ASYNC(bufev)) {

event\_del(&bufev->ev\_write);

bufferevent\_async\_set\_connected(bufev);

\_bufferevent\_run\_eventcb(bufev,

BEV\_EVENT\_CONNECTED);

goto done;

}

#endif

\_bufferevent\_run\_eventcb(bufev,

BEV\_EVENT\_CONNECTED);

if (!(bufev->enabled & EV\_WRITE) ||

bufev\_p->write\_suspended) {

event\_del(&bufev->ev\_write);

goto done;

}

}

}

atmost = \_bufferevent\_get\_write\_max(bufev\_p);

if (bufev\_p->write\_suspended)

goto done;

//这个地方才会unfree掉，并且发送数据

if (evbuffer\_get\_length(bufev->output)) {

evbuffer\_unfreeze(bufev->output, 1);

res = evbuffer\_write\_atmost(bufev->output, fd, atmost);

evbuffer\_freeze(bufev->output, 1);

if (res == -1) {

int err = evutil\_socket\_geterror(fd);

if (EVUTIL\_ERR\_RW\_RETRIABLE(err))

goto reschedule;

what |= BEV\_EVENT\_ERROR;

} else if (res == 0) {

/\* eof case

XXXX Actually, a 0 on write doesn't indicate

an EOF. An ECONNRESET might be more typical.

\*/

what |= BEV\_EVENT\_EOF;

}

if (res <= 0)

goto error;

\_bufferevent\_decrement\_write\_buckets(bufev\_p, res);

}

if (evbuffer\_get\_length(bufev->output) == 0) {

event\_del(&bufev->ev\_write);

}

/\*

\* Invoke the user callback if our buffer is drained or below the

\* low watermark.

\*/

if ((res || !connected) &&

evbuffer\_get\_length(bufev->output) <= bufev->wm\_write.low) {

\_bufferevent\_run\_writecb(bufev);

}

goto done;

reschedule:

if (evbuffer\_get\_length(bufev->output) == 0) {

event\_del(&bufev->ev\_write);

}

goto done;

error:

bufferevent\_disable(bufev, EV\_WRITE);

\_bufferevent\_run\_eventcb(bufev, what);

done:

\_bufferevent\_decref\_and\_unlock(bufev);

}

View Code

所以，如果需要清空bufferevent的发送evbuffer里的数据，必须手动调用evbuffer\_unfreeze将start unfreeze掉后才能drain。当然，一般bufferevent\_write都是在其他线程中做的，所以要确保drain的时候没有线程在进行bufferevent\_write，我这里先不处理多线程write这块了，或者直接在SetupConnection的时候去drain。

[复制代码](javascript:void(0);)

void Connection::CloseConnection()

{

//需要先unfreeze output evbuffer

evbuffer\_unfreeze(bufferevent\_get\_output(m\_be), 1);

if (evbuffer\_drain(bufferevent\_get\_input(m\_be), evbuffer\_get\_length(bufferevent\_get\_input(m\_be))))

{

...

}

size\_t length = evbuffer\_get\_length(bufferevent\_get\_output(m\_be));

if (evbuffer\_drain(bufferevent\_get\_output(m\_be), length))

{

...

}

bufferevent\_setfd(m\_be, -1);

evutil\_closesocket(m\_fd);

m\_fd = -1;

bufferevent\_enable(m\_be, EV\_READ|EV\_WRITE) ;

m\_bUsed = false;

}

[复制代码](javascript:void(0);)

这样bufferevent就可以重复使用了，不知道这样是不是正确的做法，有点麻烦，还是重新分配bufferevent来得方便感觉。