# 2014/1/6

## 修改

1. 服务器进程退出时，采取释放线程池🡪删除客户连接列表->删除监听对象列表

**原因**：

服务器进程主动退出时，如果采取以下释放顺序

CNetServer

1. 删除客户连接列表（此时对象未被释放，引用计数不为0），closesocket，将唤醒投递的I/O操作，根据错误将I/O操作对象释放，全部释放后，客户连接对象引用计数也为0
2. 删除监听对象列表（此时对象也未被释放，引用计数不为0），closesocket，将唤醒投递的I/O操作，根据错误将I/O操作对象释放，全部释放后，客户连接对象引用计数也为0
3. 释放线程池对象，让线程退出

但是有个问题：如何保证1、 2步骤执行完之后才退出线程池，如果1 、2步骤的I/O操作未被处理，线程池就退出，那么就会有内存泄露的问题，虽然整个程序都退出，但是系统回收资源不知道是不是实时的。

所以最好是3、1、2的顺序

2.将CNetServer中客户端连接对象set容器修改成map

~~std::set<IAsyncIO\*> m\_setRemoteClient;~~

std::map<IAsyncIO\*, std::set<LPPER\_IO\_CONTEXT>> m\_mapRemoteClient2IOSet;

**原因**：

本设计采用命令模式，I/O操作携带执行对象，执行对象的生命周期与I/O操作相关，只有全部的I/O操作都被释放了，执行对象才能被释放。

现客户端连接对象创建后，投递I/O Recv操作，该操作对象不保存，直接投递到队列，采用了3、1、2的步骤之后，I/O Recv对象就无法被释放，本地有没有保存对象指针

3. 观察者模式中IObserver不在维护一个被观察者对象列表

~~std::set<ISubject\*> m\_setSubject;~~

**原因**：

当初添加列表的原因是：ISubject维护一个观察者列表，当状态更新时，就会遍历列表区通知观察者，那这里就有一个问题，如果Obsever1对象被释放了，ISubject这边没有得到通知，遍历的时候就会出现访问已经释放的内存，所以就考虑IObserver也保存ISubject对象指针，当自己被释放时反过来通知ISubject，更新列表。代码实现起来有点乱。。。有点循环通知的可能。

改进：

删除IObserver中的ISubject列表。

## 添加