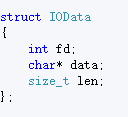
WriteQueue和ReadQueue是提供来与通信机通信的。压入的事件是IODataPtr类型，在utils.h中（其他队列压入的是MsgReq等三种事件）。但是要注意的是，目前调度器并没有对WriteQueue进行检测，因此需要自己另外处理这个队列。



首先，这里clientsocket和client是不同的，client是用来与UI、Trace进行通信的，而clientsocket是用来与通信机进行通信的。

例如在qpsk中，初始化函数Init()中会构造一个ClientSocket对象。（GetDap()是Driver类中的，Dap对象也是Driver中的）

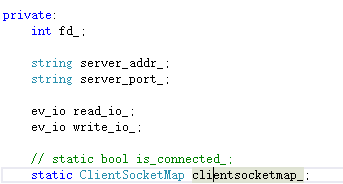


在ClientSocket类中，获得对象的函数。如果在映射中找不到该IP和端口对应的对象，就创建一个，同时调用这个类中的初始化函数Init()。

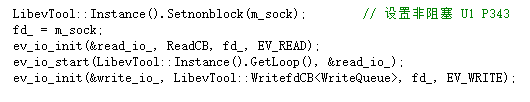
这里除了初始化这个对象以外，还在clientsocketmap\_这个映射对象中插入：套接字、文件描述符、ClientSocket的指针。（插入以后，可以用ClientSocketMap类中的Find函数，通过套接字找到对应的ClientSocket对象，返回指针。如果没有这个对象就返回NULL）



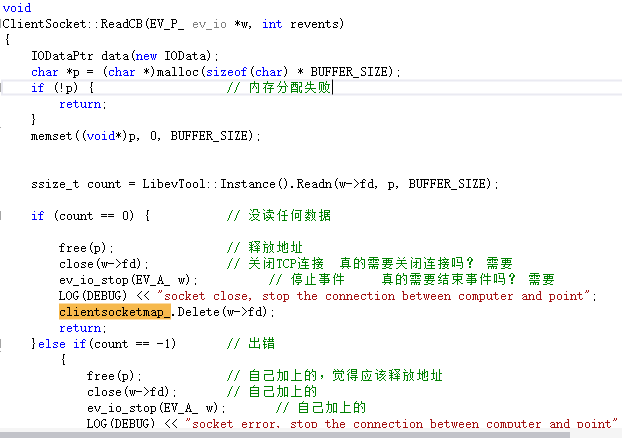
在这里，clientsocketmap\_是一个静态成员，因此对于所有ClientSocket对象都只有一个映射。（它的初始化在类外。）

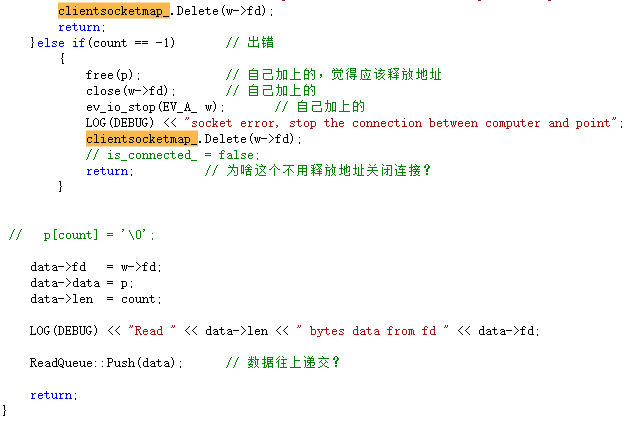


**ClientSocket类中**的初始化函数Init会对传入的IP地址和端口，创建一个tcp连接，并且设置为非阻塞，然后监听该端口。当可读的时候，就调用**LibevTool类中**的写回调函数ReadCB。（监听写的事件只是初始化完成了，如果要写数据要手动操作，即调用ClientSocket类中的Write()函数。其中写的回调函数是LibevTool中的WritefdCB，需要传入事件队列，而读的回调函数使用ClientSocket自身类中的ReadCB。）

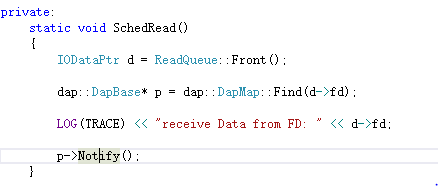


回调函数先创建一个IOData对象，然后读出1032个字节的数据，对对象的赋值，压入全局的读事件队列中，最后由调度器来调用。（IOData事件包含了数据和文件描述符，调度器通过文件描述符找到Dap，从而通知该协议收到了数据。）

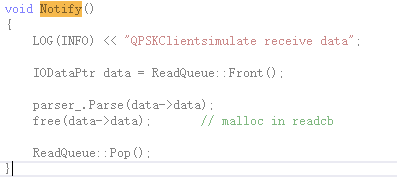




调度器会检测全局的读事件是否为空，然后调用处理函数。此时不是Sap而是Dap，因为全局的读事件队列都是通信机给物理层发消息用的。然后调用物理层协议的处理消息函数。（DapMap中的映射是从套接字描述符到Dap的映射。）



例如**QPSK协议中**的Notify函数：



**MyParser类中**的Parse（解析）函数，对这些指令解析，然后发布相应的事件。

