Main

1. 调用log\_init()，对日志部分初始化

日志部分是基于log4cplus网络库实现的，初始化了日志的一些格式。

1. 调用signal()，对一些信号进行处理。接收到一些特殊信号时对某些模块进行重新加载
2. 调用init\_config()，初始化配置

* 对access.conf配置文件进行了初始化，初始化各个模块需要的一些值
* 初始化了许多键值，两个关键的类为SDConfiguration类和config\_singletion，后续的使用基本是获取已初始化的实例，即可以获取之前全部初始化好的值，
* 接入层任务里面的ReloadAccess没有执行，不太理解，ReloadAccess的作用具体又是什么，初步看来是更新注册地址
* 接下来是进入ServiceStart函数，进入函数的主逻辑

ServiceStart

主要有三大部分的初始化以及启动，分别是CommandDispatch、RegisterClient、pMyServer，三个类的实现都有点类似，都是先初始化然后启动，目前简单略过一遍以及查看了相应的函数跳转之后初步判断CommandDispatch是命令分发（任务分发），即接收到设备的请求之后分发给业务层处理；RegisterClient是负责设备注册的的模块，目前未发现有listen相关的函数，估计是pMyServer处统一处理；pMyServer是实现服务器的主要代码，主要是处理IP连接。

serve

myserver类的serve函数是服务器启动的主要函数，里面包含了不同功能线程的初始化以及线程的处理函数。首先是创建监听套接字createAndListenOnSocket，然后生产一个监听线程和多个读写IO线程，最后对线程资源进行回收。

其中核心在线程的处理函数中，处理函数主要是基于libevent库进行一个事件的注册以及执行事件的循环。在处理函数中，若线程为监听线程，则处理监听事件，若不是监听线程，则处理IO读写，其中IO事件由监听线程分发给剩余的IO线程，然后进行读写，类似muduo库那样处理。读写之后的信息通过kafka传输给业务层，与客户端的连接就是TCP连接。

明天再详细阅读接受连接建立过程所提供的函数，以及记录一些比较关键的步骤。

Run

程序从run函数开始了线程的运行处理函数，基于libevent框架实现事件循环，事件循环函数中区分监听线程和读写处理线程，监听线程通过listenhandler对客户端进行accept，然后建立连接，通过管道把该连接分发给读写处理线程。然后读写处理线程通过notifyHandler接收和处理读写事件，至此整个程序的大概框架已完成建立。

notifyHandler

notifyHandler函数是处理主线程分发过来的读写事件的主要函数，主要功能是从管道获取连接，然后处理连接的读写。作为服务器，写函数一般是次要，写一般是回复型的，使用会较少，暂不深究。然后是读函数，读函数主要是将连接里读到的数据缓存起来，紧接着进行数据分析，数据分析之后加入任务队列等待执行。

readHandler+

readHandler函数先通过参数建立一个连接，读取数据之后返回数据给该链接的m\_recv\_raw\_buffer变量中，然后进行进一步的数据分析。接着是进入dataParse，选择是否开启加密数据。

nonSstDataParse+

首先执行**m\_msmart\_parser.parse**校验数据包长度、类型等是否正确，然后进入数据预处理过程**preMsgProcess**，处理完毕之后提交到任务队列。在nonSstDataParse中，校验时会检测数据包预期长度与实际长度是否符合，若实际长度较长，则循环处理，分成一个一个包，再进入数据预处理过程**preMsgProcess**中。紧接着进入更详细的预处理函数**pre\_process\_msg。**

pre\_process\_msg

疑问：

6.22

1. resetConnection函数具体作用未详细看

2. parse\_pkg函数的判断与返回值处理不是很理解，还有结构体的转换

3.这个dataParse最后函数又会执行到dataParse，有点费解，走sstDataParse也一样

4. automic exchange函数的具体作用比较难理解，网上搜了也讲不清楚的

6.23

1.数据头的同步标志synctag的用处是什么

2.为什么要用hex2printstr转换不是很理解

4. update\_status为何更新状态

5. msg\_parse\_impl\_msmart::pre\_process\_msg函数中检查操作权限m\_msmartinfo->auth、验证模块类型verify\_2g\_moduletype、更新状态update\_status、验证原型版本verify\_proto\_version等未深入看