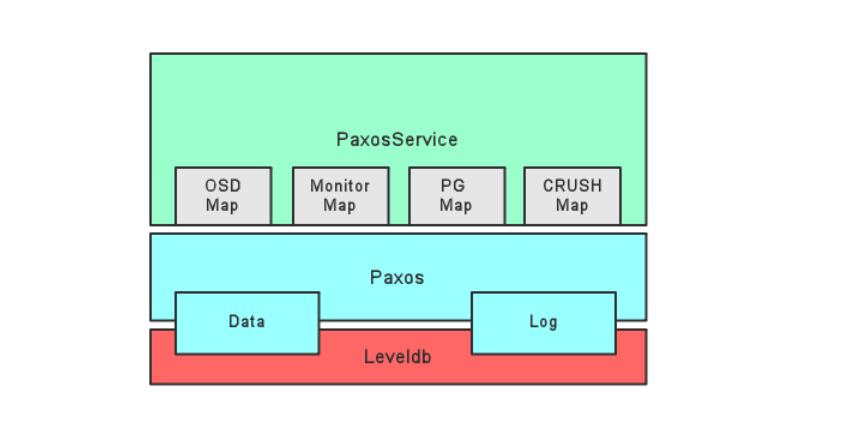
ceph monitor主要负责集群的管理，涉及各个map（monitor map, osd map, pg map crush map）管理，其主要是利用改进的paxos算法管理这些map,对外提供元信息管理服务。

本章主要是分析ceph monitor的启动流程。

ceph monitor 架构如下：

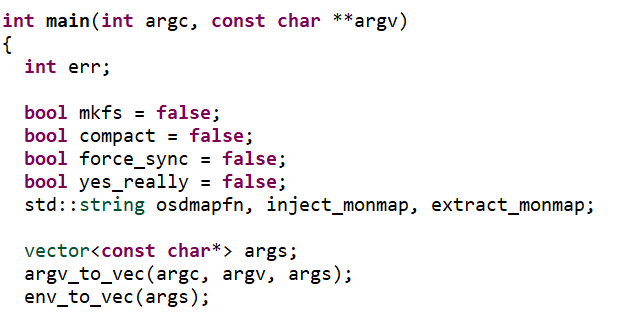


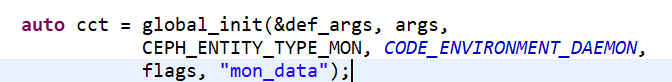
DBStore层：数据的最终存储组件，以leveldb为例。

Paxos层：在集群上对上层提供一致的数据访问逻辑，在这一层看来所有的数据都是kv；上层的多中PaxosService将不同的组件的map数据序列化为单条value，公用同一个paxos实例。

PaxosService层：每个PaxosService代表集群的一种状态信息。对应的，Ceph Moinitor中包含分别负责OSD Map，Monitor Map, PG Map, CRUSH Map的几种PaxosService。PaxosService负责将自己对应的数据序列化为kv写入Paxos层。Ceph集群所有与Monitor的交互最终都是在调用对应的PaxosSevice功能。

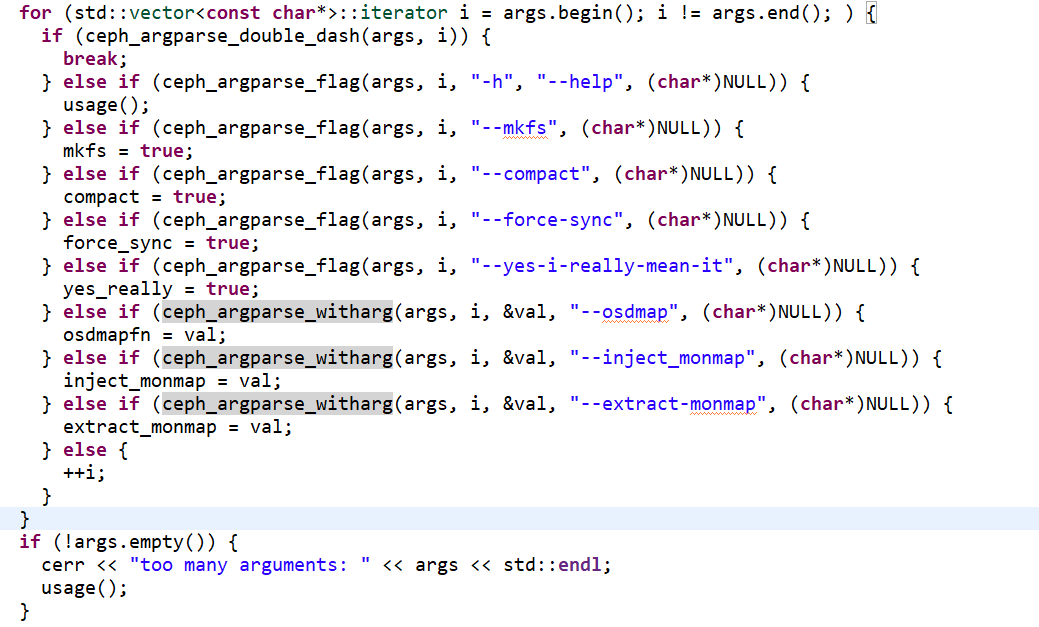
Ceph monitor启动代码在ceph\_mon.cc 中，main函数开始：



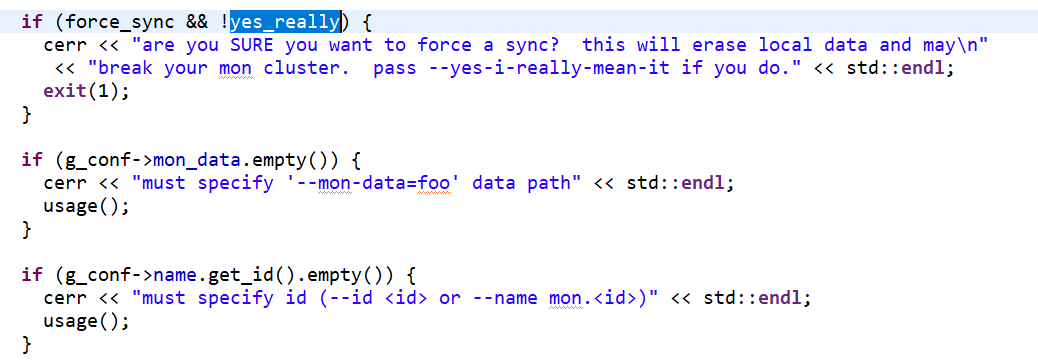


global\_init主要是构造CephContext以及环境变量的解析设置，信号的设置，用户，用户组的设置等。

接下来就是参入参数的解析：

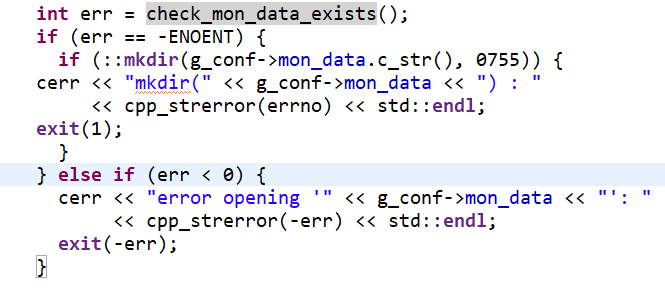


对输入的参数进行检查：

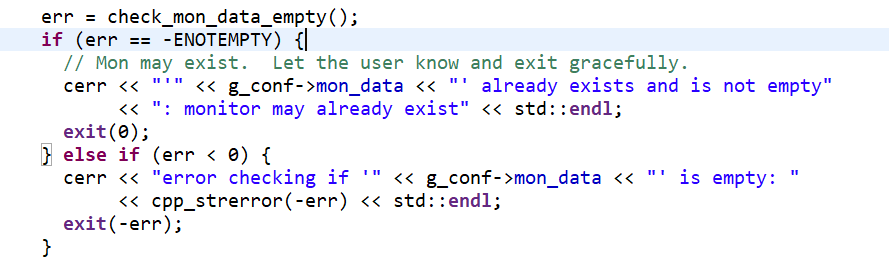


如果输入参数有--mkfs, 在进行ceph文件系统的格式化

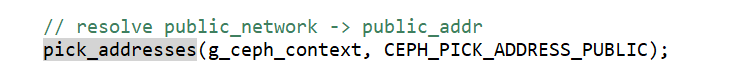
在mkfs{} 中主要执行如下过程：  
1.检查monitor 数据存放目录，没有则创建



检查目录里面是否存在数据，存在说明monitor已经存在了，不是新建



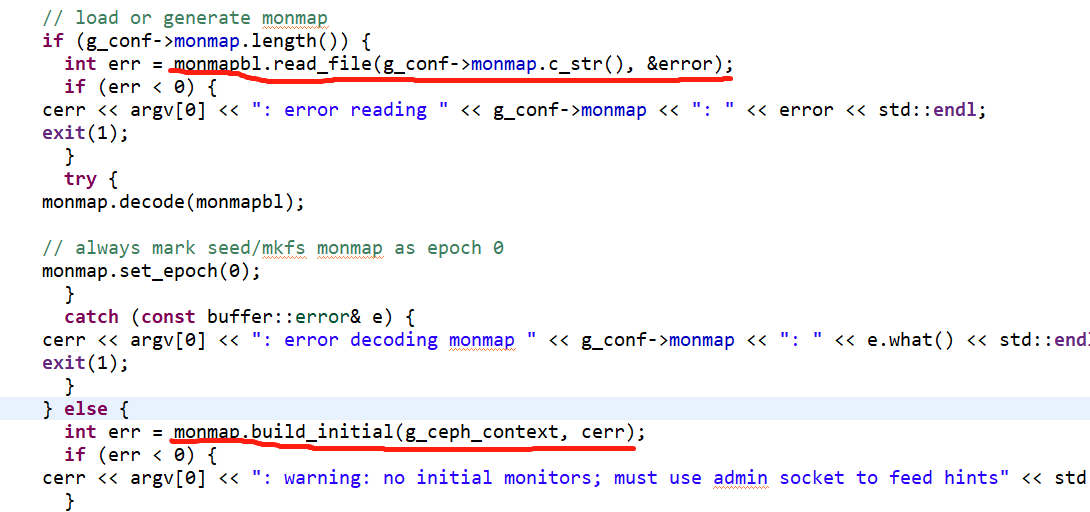
获取public network IP 地址：



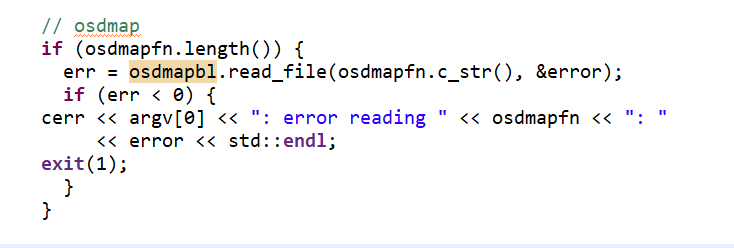
启动CephContextServiceThread以及admin socket：



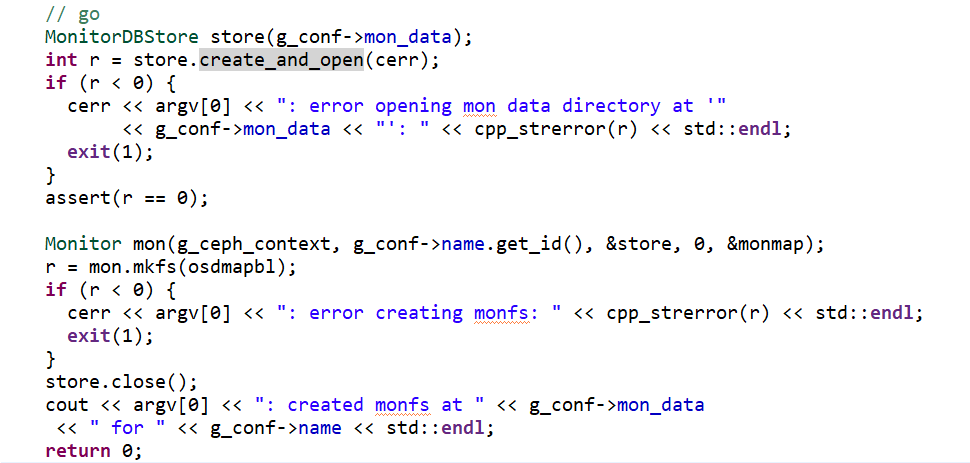
取得原有的monmap还是产生新的monmap:



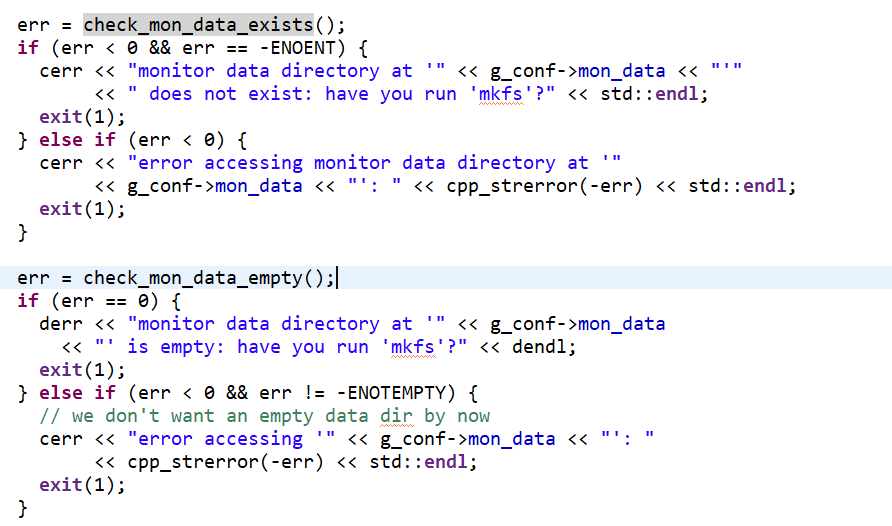
获取osdmap:



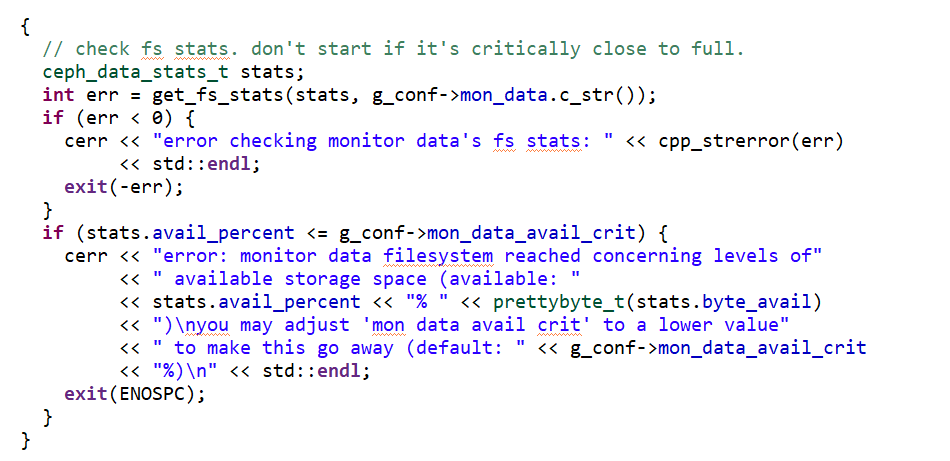
创建MonitorDBStore以及新的monitor，并且通过mkfs接口保存monmap,osdmap,keyring：



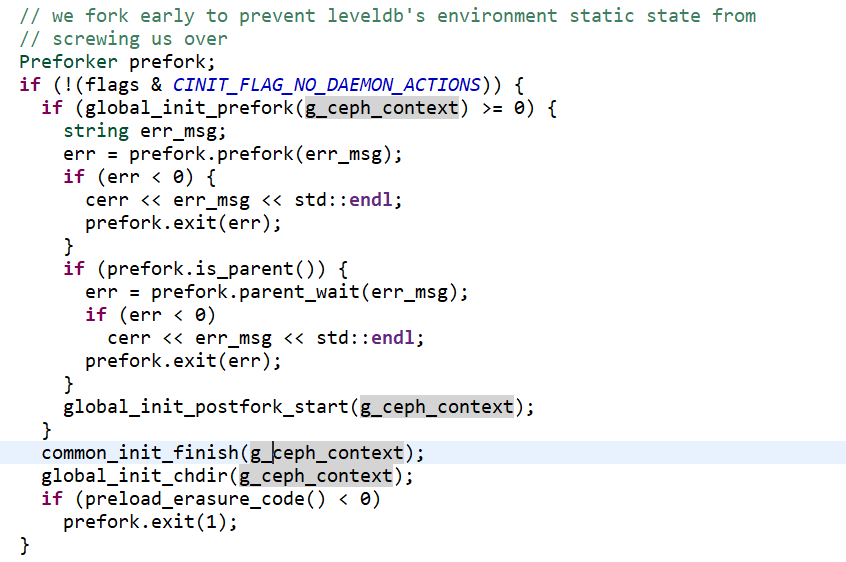
上面是针对mkfs选项的，对于没有mkfs选项的，同样要检查其数据存放目录以及目录数据：



检查存储空间是否满：



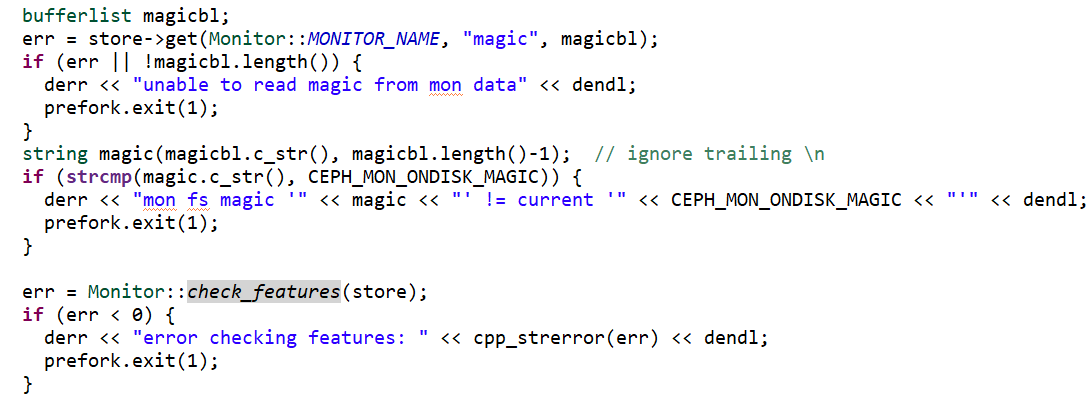
Monitor作为后台daemon：



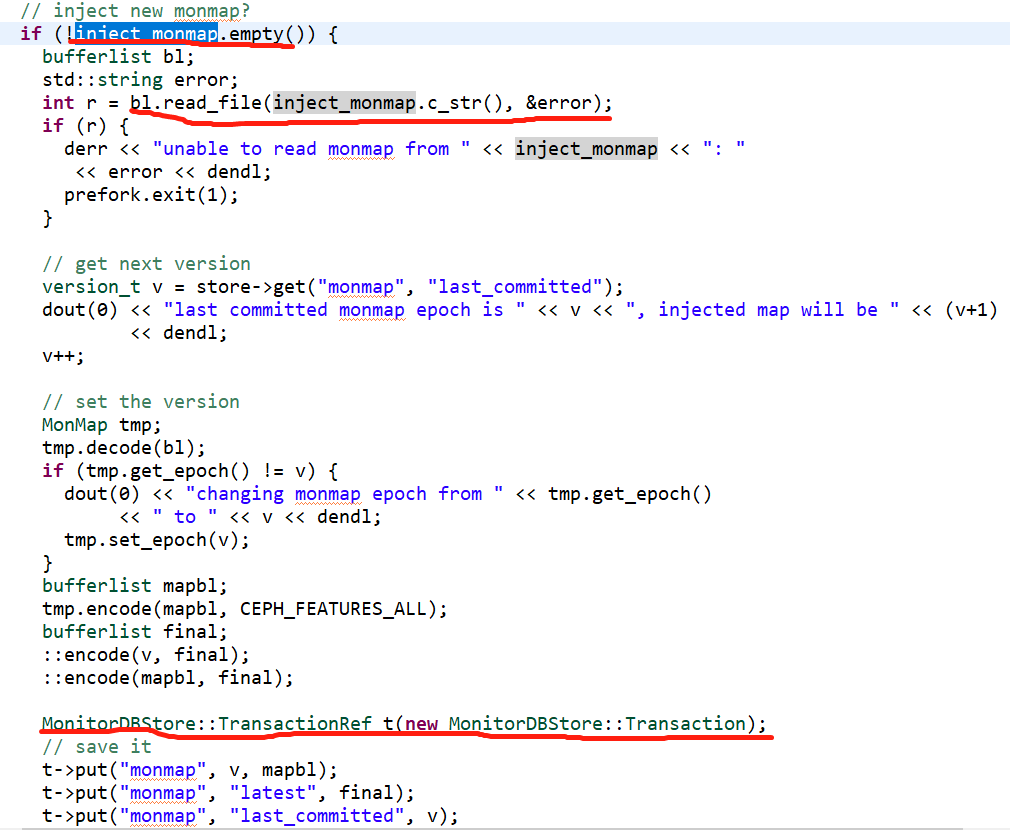
创建MonitorDBStore，并且启动 DB



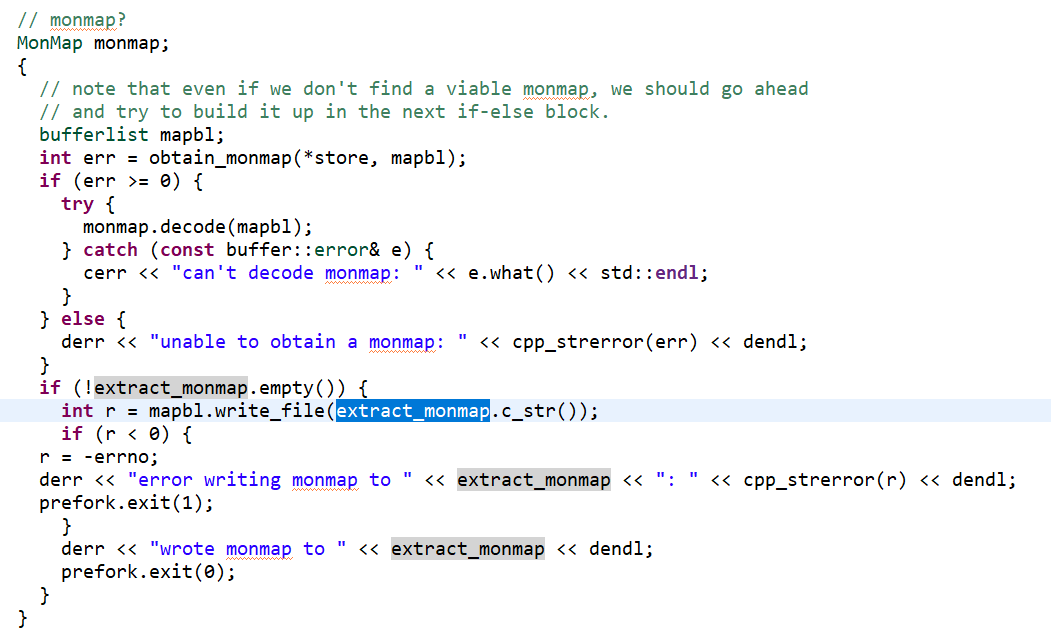
读取magic,并且检查，检查check\_features



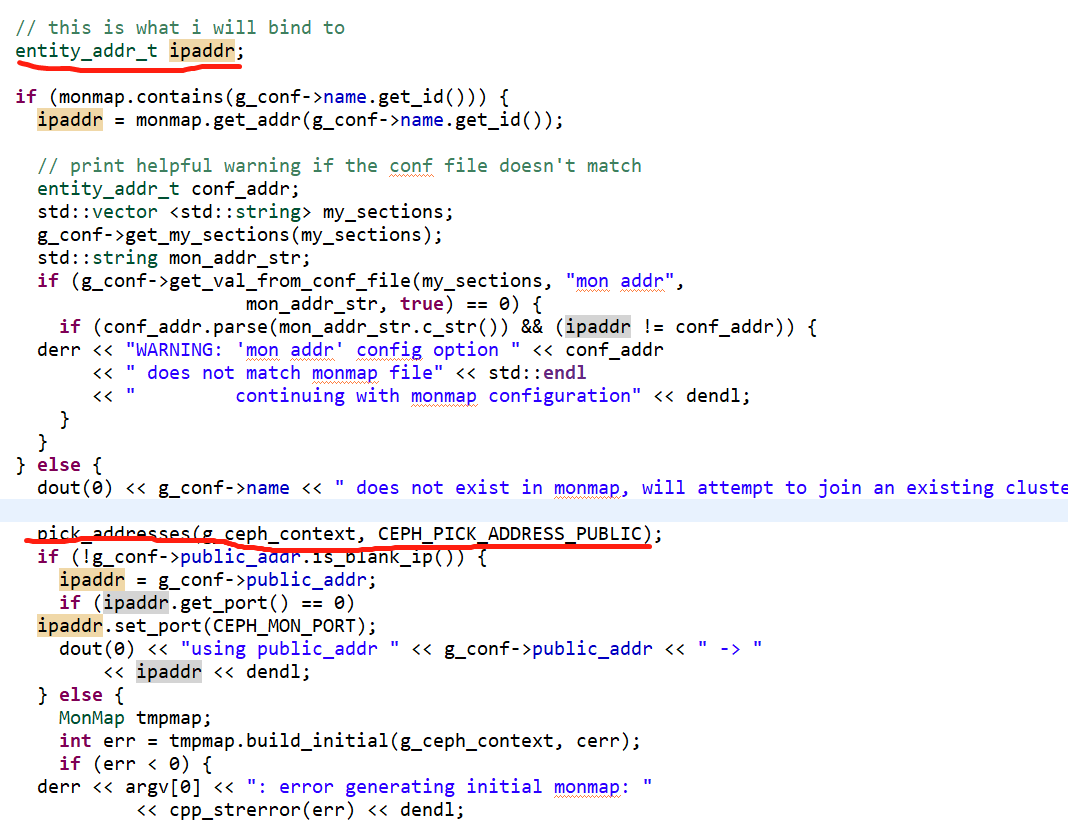
如果输入参数有monmap,则需要向DB中注入monmap：



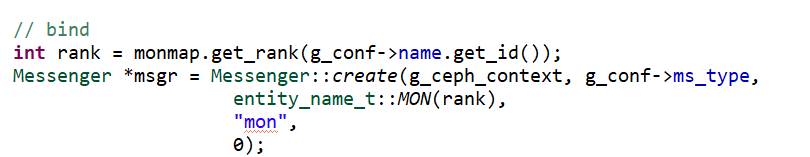
如果选项有提取monmap。则从DB中读取出来并且解码输出：



解析出monitor messenger需要绑定的IP地址：



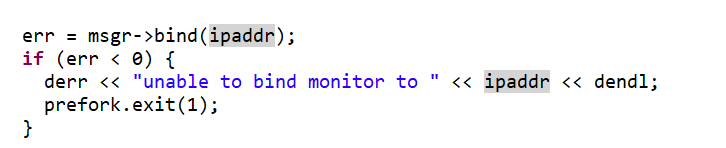
创建messenger:



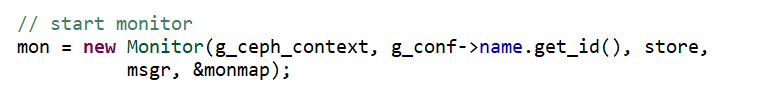
设定策略，主要是做通信的兼容性处理，应为不同版本的通信模块需要兼容：



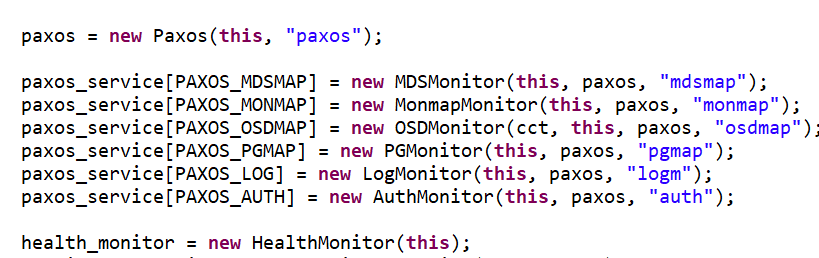
Messenger绑定IP：



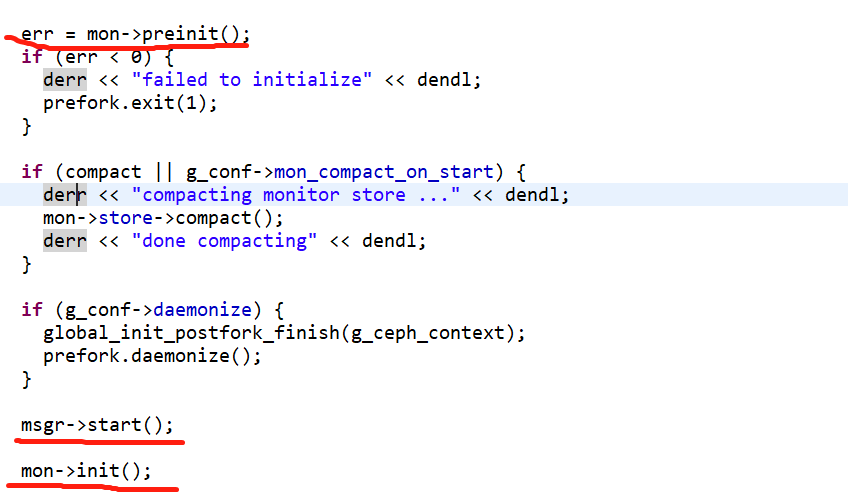
创建monitor：



在这个初始化过程中，创建paxos,并且注册不同map的monitor。



然后就是启动messenger和初始化monitor,在init函数中会注册messenger以及进入monitor Bootstrap过程，这个过程会通信其他的monitor信息，同步数据，选举leader。



至此monitor服务就启动完毕。