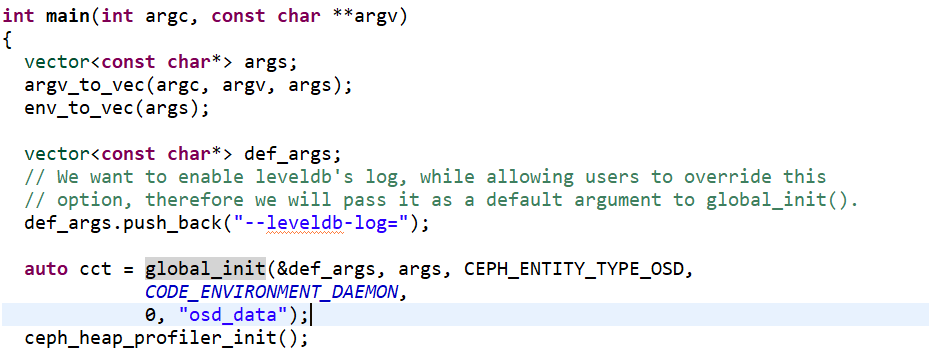
ceph的OSD进程主要负责数据的读写（磁盘的读写）和monitor的通信（状态汇报以及map更新）以及数据一致性保护。本章内容分析OSD进程的启动流程：

ceph osd进程启动代码在ceph\_osd.cc，从main()函数开始：



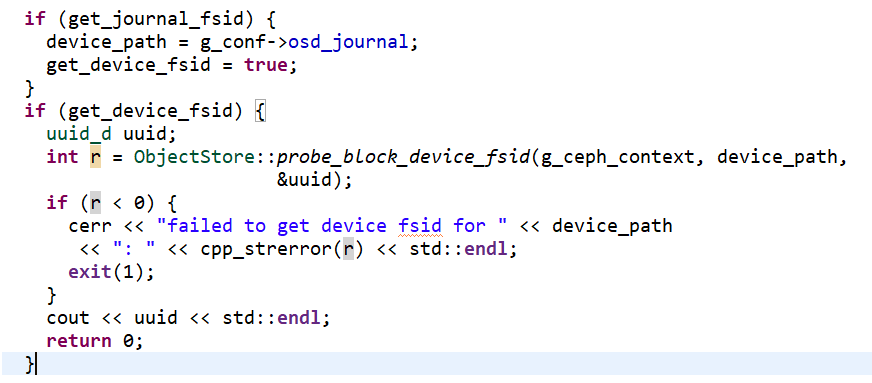
global\_init主要是构造CephContext以及环境变量的解析设置，信号的设置，用户，用户组的设置等。

接下来就是传入参数的解析：

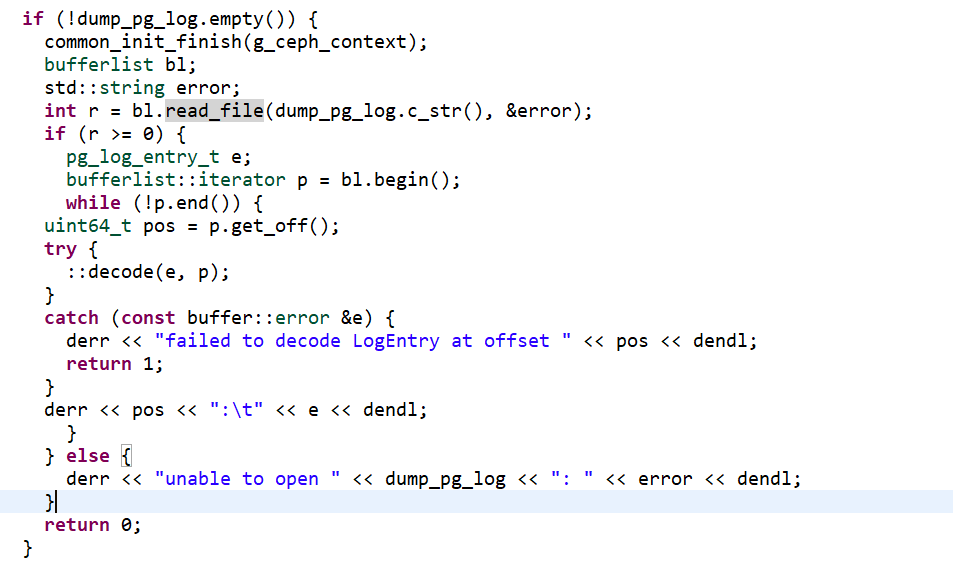


接下来根据输入参数做出相应的执行动作：

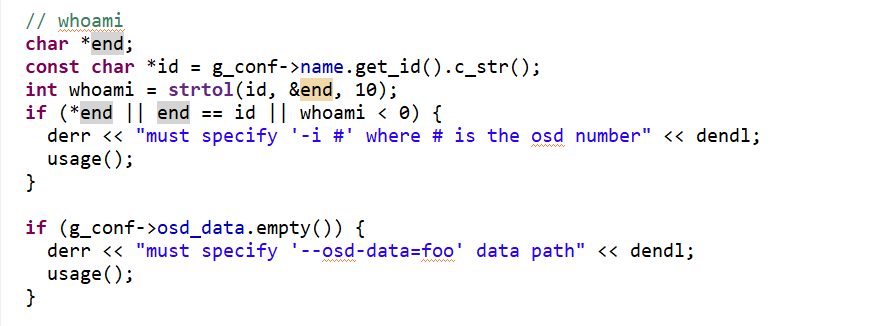
如果输入参数有--get-journal-fsid或者有--get-journal-uuid,说明有已经有journal检查其uuid是否有效，这个不是osd启动选项。



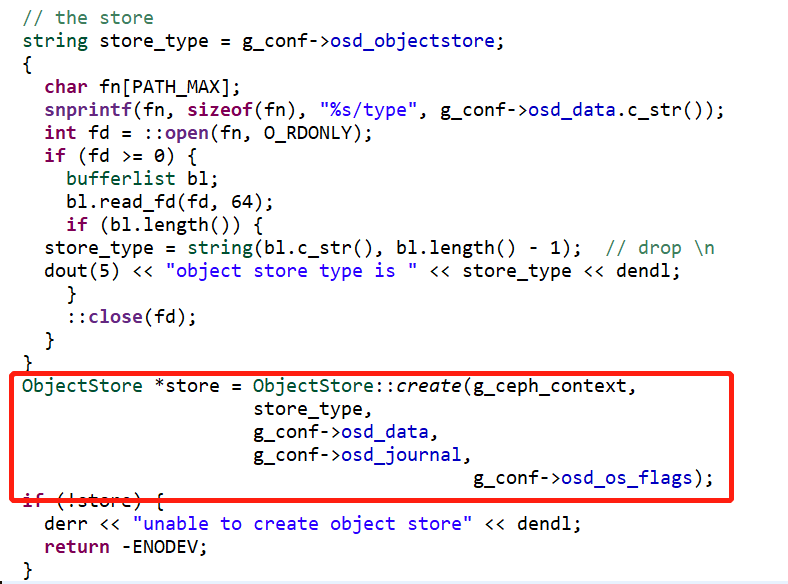
如果输入参数有--dump-pg-log，则检查输入文件是不是pg\_log文件，这个不是启动选项：



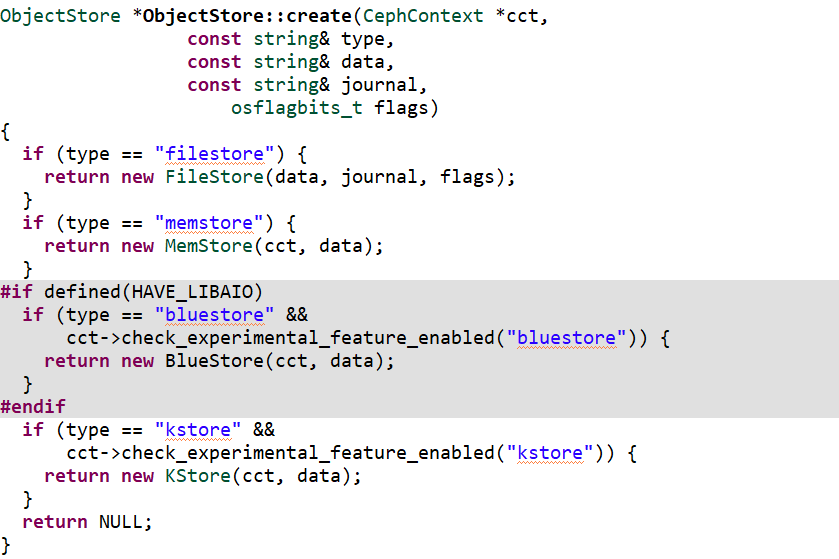
检查osd编号以及数据存放目录



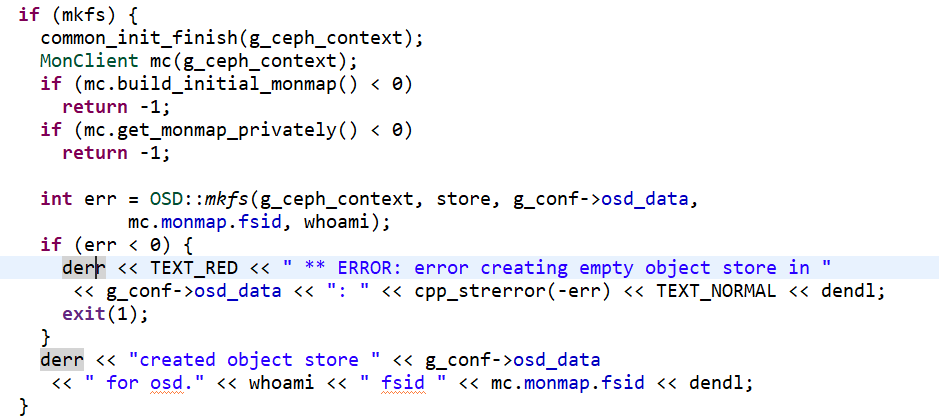
根据存储类型（filestore还是bluestore）创建ObjectStore:



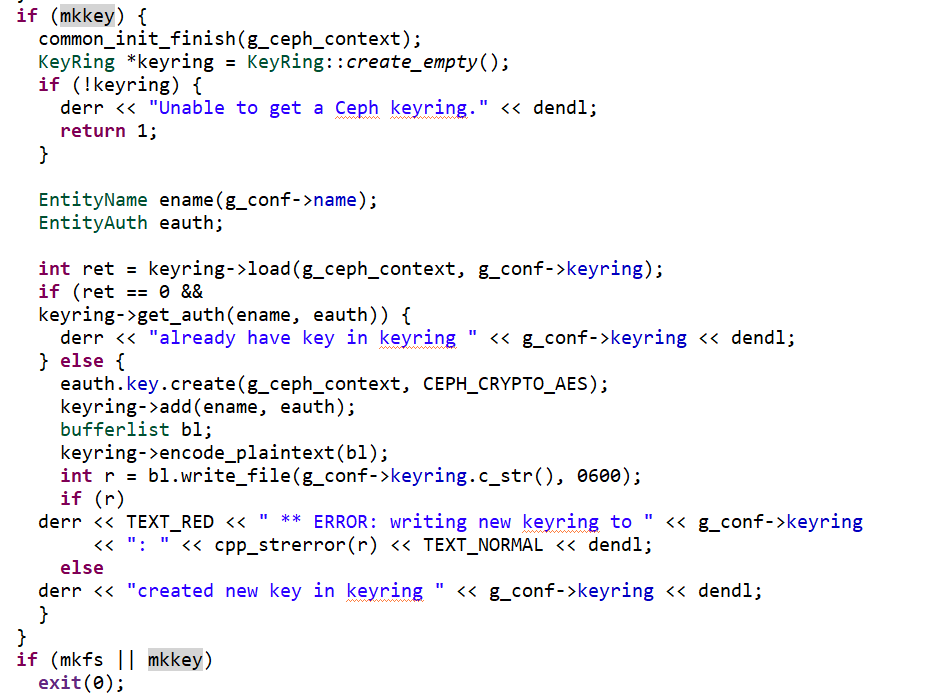
默认是filestore, 然后从/var/lib/ceph/osd/ceph-$i/type读出其type类型，根据类型创建不同的ObjectStore对象存储接口：



创建ObjectStore实例后，如果传入参数有--mkfs就创建相关的系统信息：



如果传入参数有--mkkey，则需要创建相应的keyring:



mkfs和mkkey是单独的osd命令，启动osd前的准备操作。

接下来就是journal的一些操作，这些操作都是针对不同的输入参数执行不同的操作，非启动流程。

--mkjournal创建journal

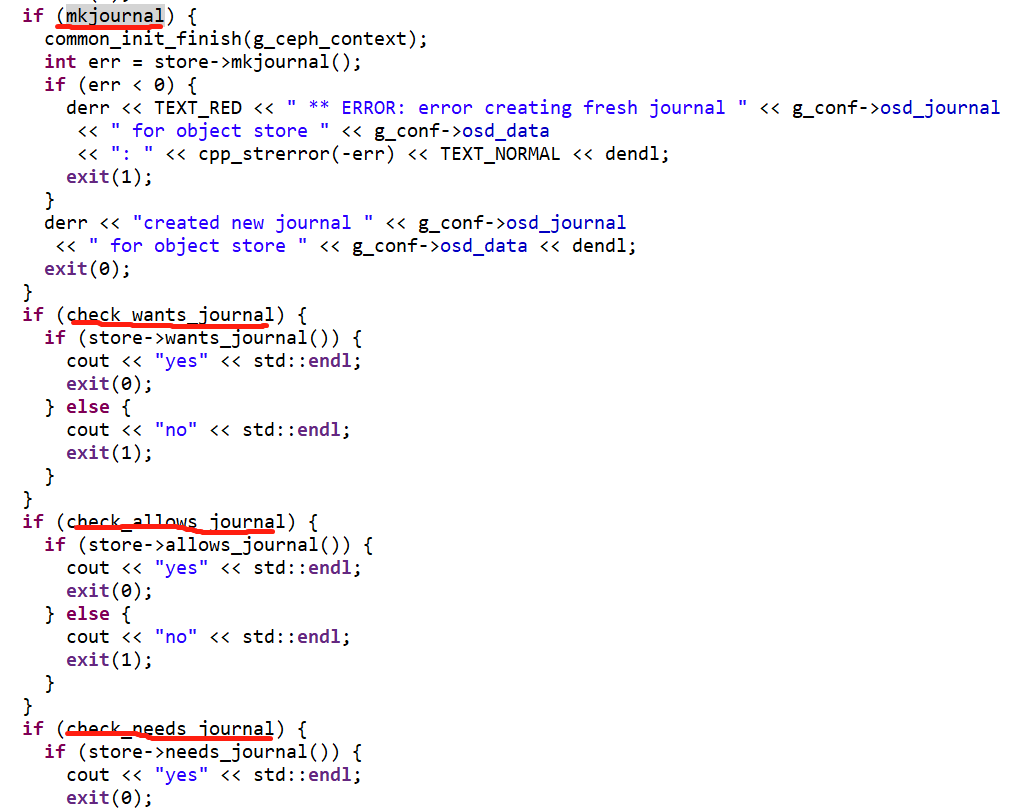
--check-allows-journal

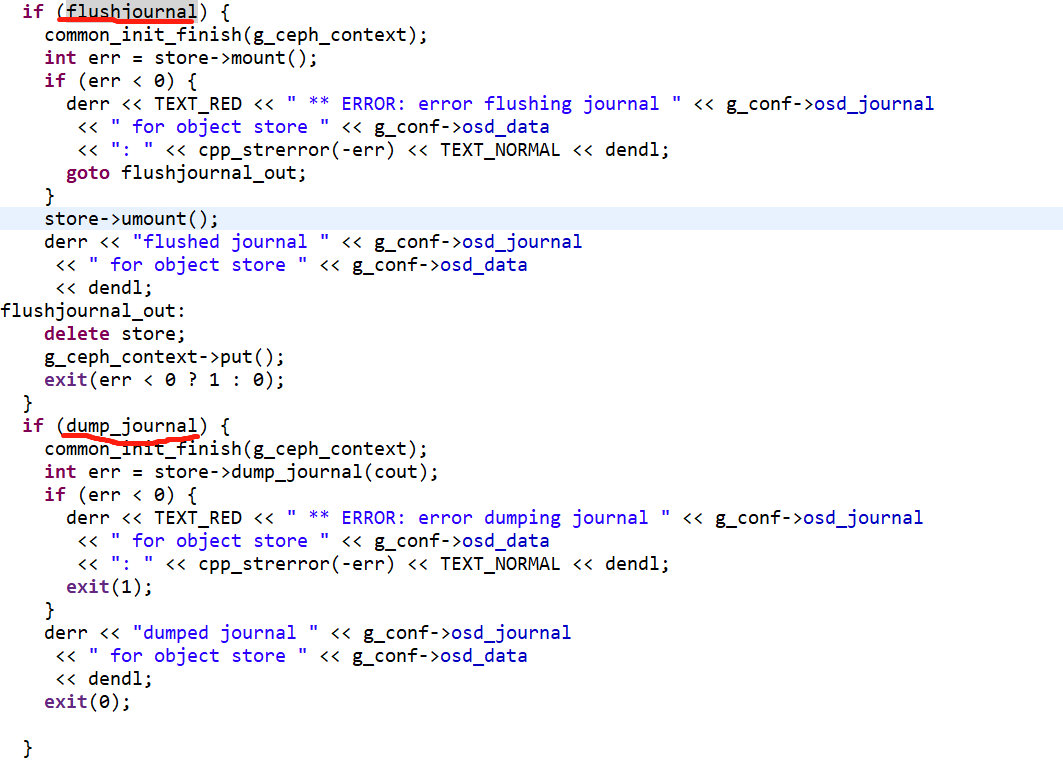
--check-wants-journal

--check-needs-journal

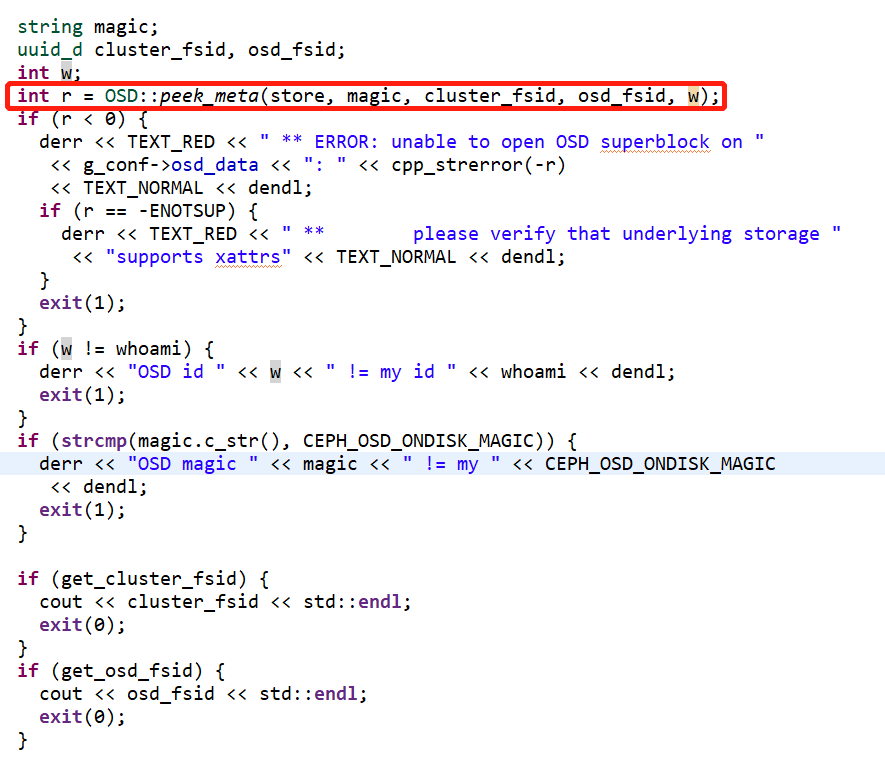
--flush-journal

--dump-journal

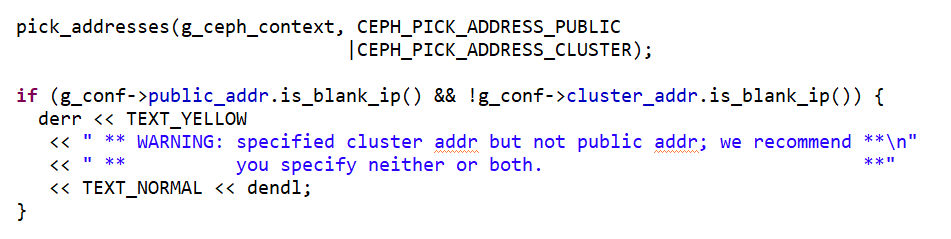




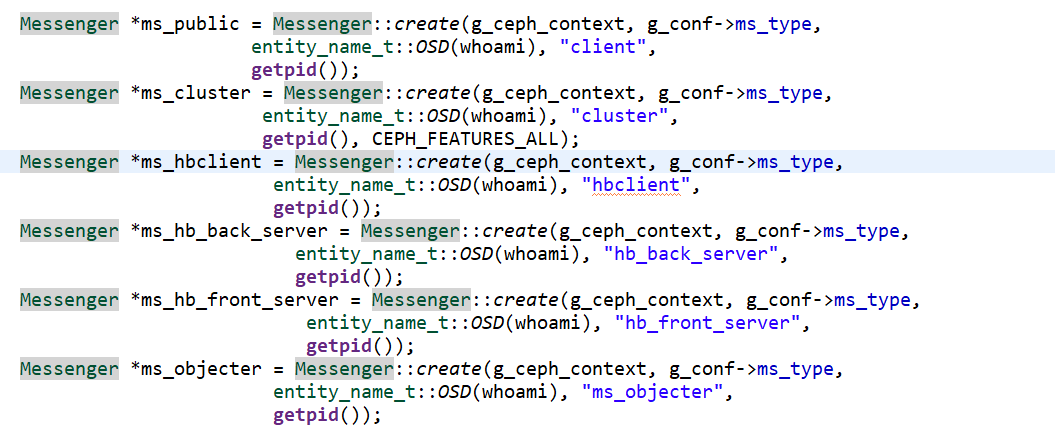
下面就是启动的流程了，从osd目录中分别读取magic, whoami, ceph\_fsid,fsid



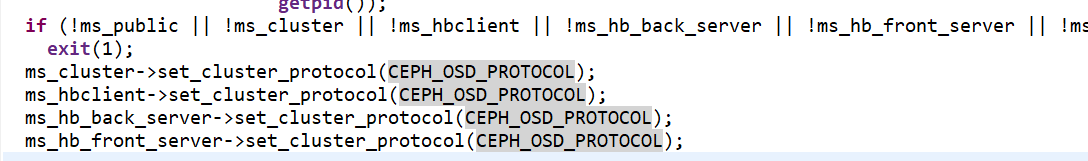
获取检查cluster\_network 和public\_network:



加下来就是创建Messeger了就是SimpleMessenger,网络通信模块,



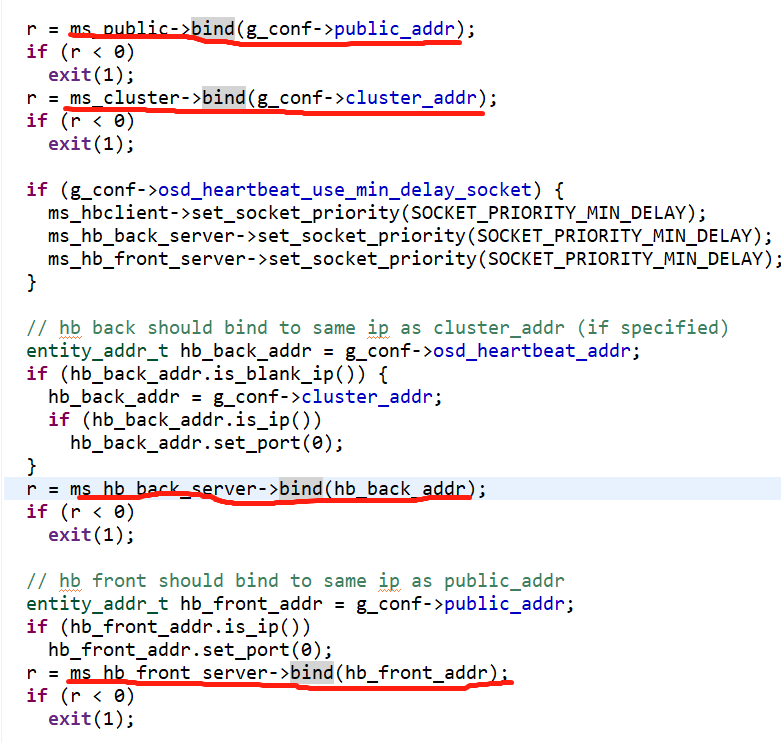
设置协议，CEPH\_OSD\_PROTOCOL表示是集群内部通信协议



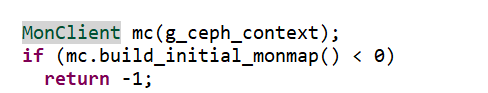
设置策略，保证通信双方的兼容与认证



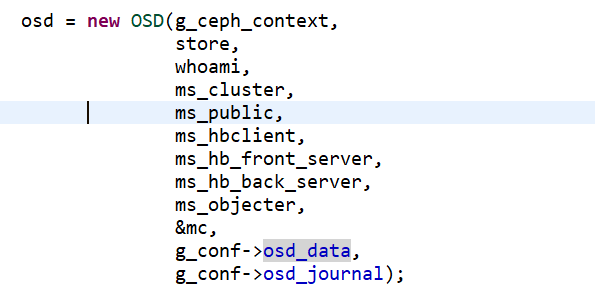
接下来就是绑定网络地址了：



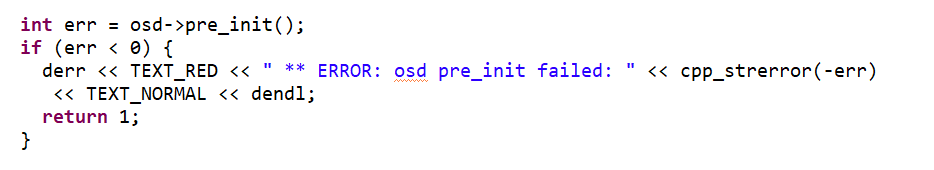
创建monmap:



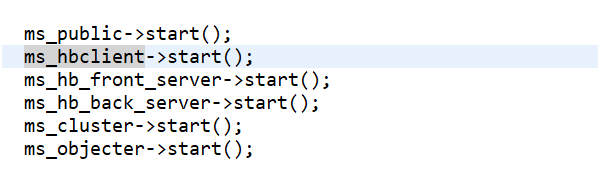
接着就是构建OSD对象了：



然后进行初始化准备工作：



启动Messenger实例

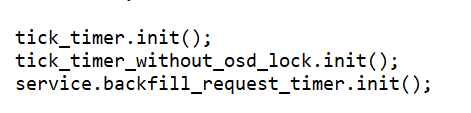


加下来就是OSD初始化了，这个函数比较长，涉及的内容有：

OSD::init()函数动作如下：

初始化心跳定时器；

初始化backfill请求定时器；



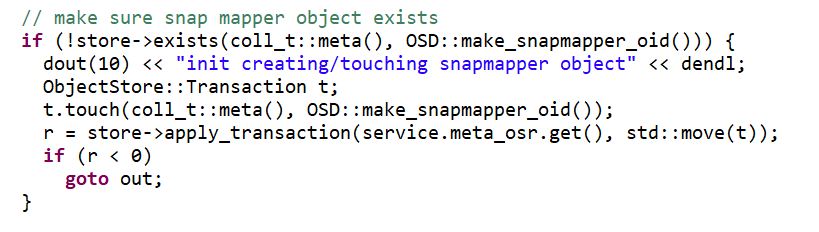
挂载文件系统；



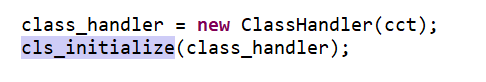
从集群中读取superblock信息且保存到OSDSuperblock类对象中；



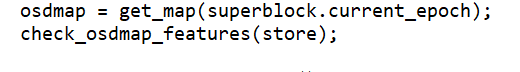
确保snapmapper对象存在于”meta”中；



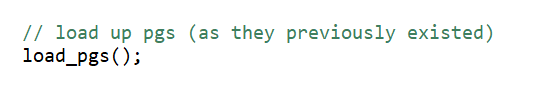
创建ClassHandler类实例；



根据superblock中记录的当前epoch值获取osdmap，若osdmap没有在map\_cache中则从”meta”的osdmap.epoch中读取osdmap且同步到map\_cache中；



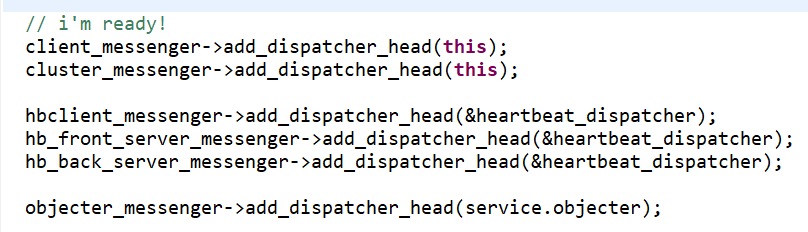
读取OSD上所有的pg信息；



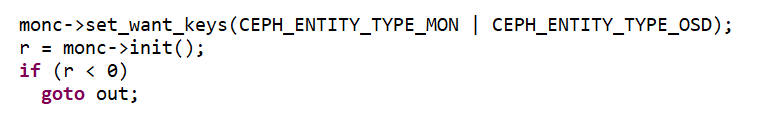
创建OSD logger



所有的messenger注册dispatcher



初始化MonClient类实例



启动osd\_tp/osd\_op\_tp/recovery\_tp/disk\_tp/command\_tp线程池

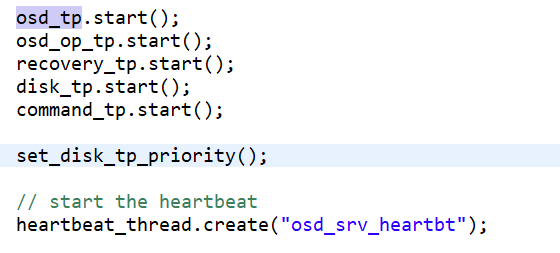
有4种OSD线程池:

op\_tp: 处理ops和sub ops

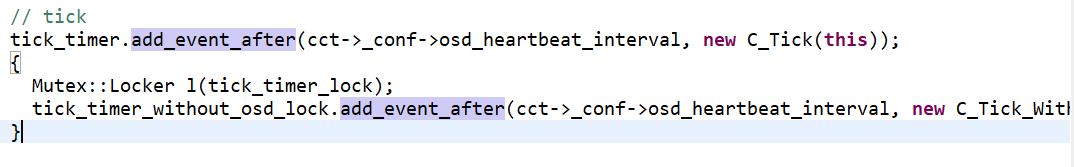
recovery\_tp:处理修复任务

disk\_tp: 处理磁盘密集型任务

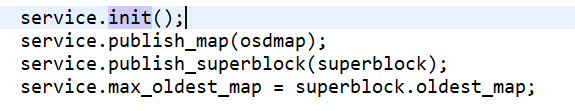
command\_tp: 处理命令



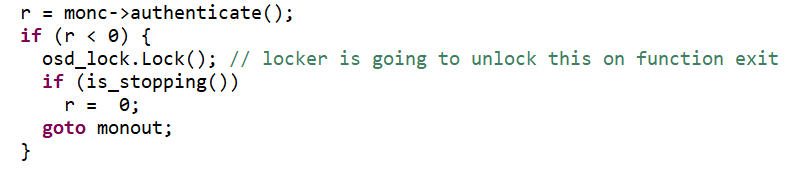
定时器的设定：



初始化OSDService

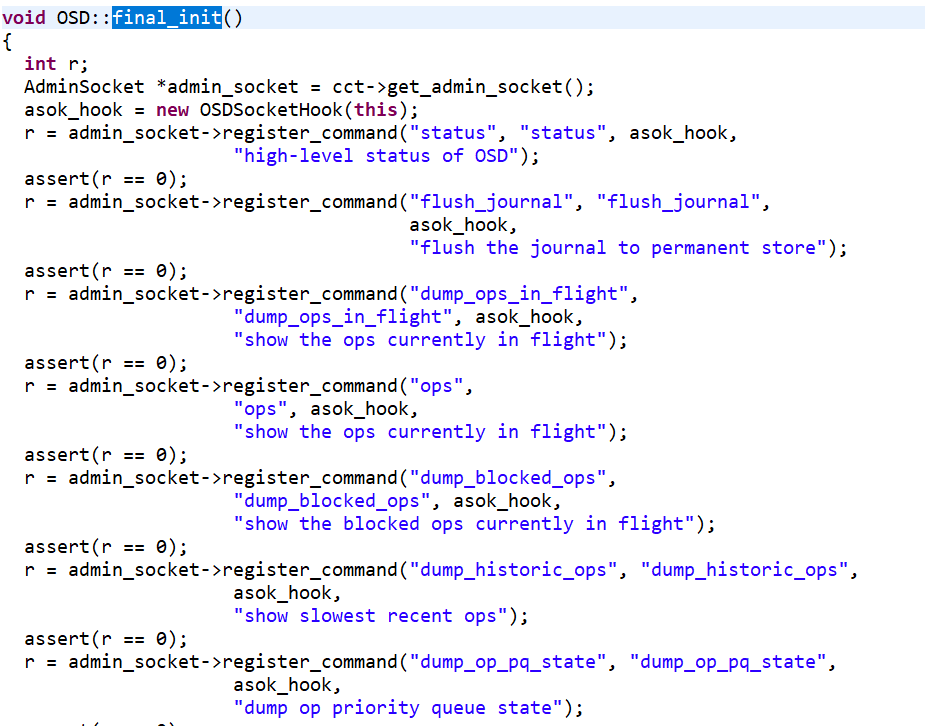


做Monitor认证



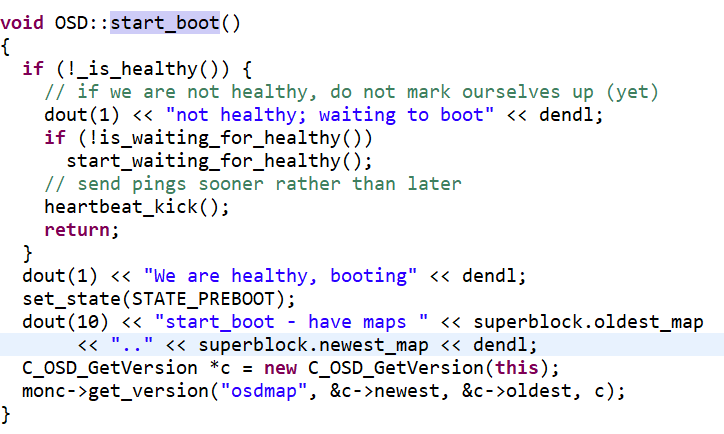
service.final\_init();

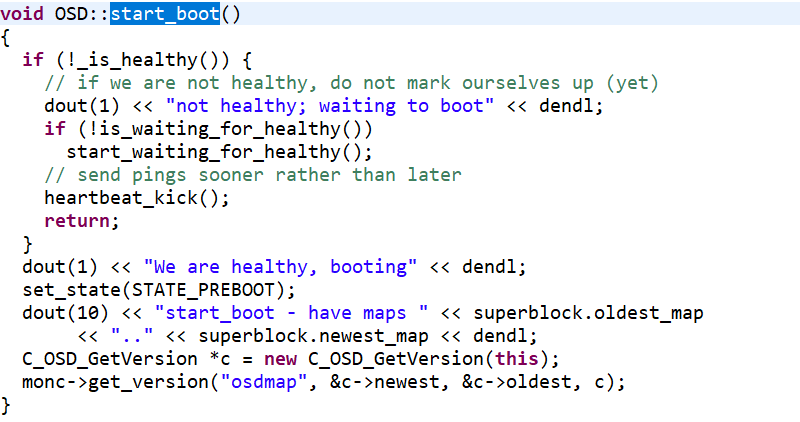
注册一系列通过socket进行操作的命令：



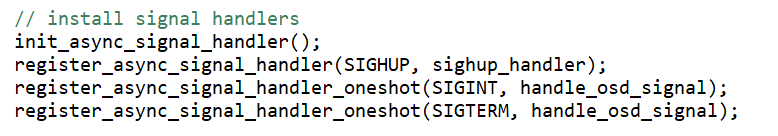
start\_boot();

与monitor通信，获取发送osdmap version





接下来就是注册信号处理函数了：



至此OSD进程启动了。