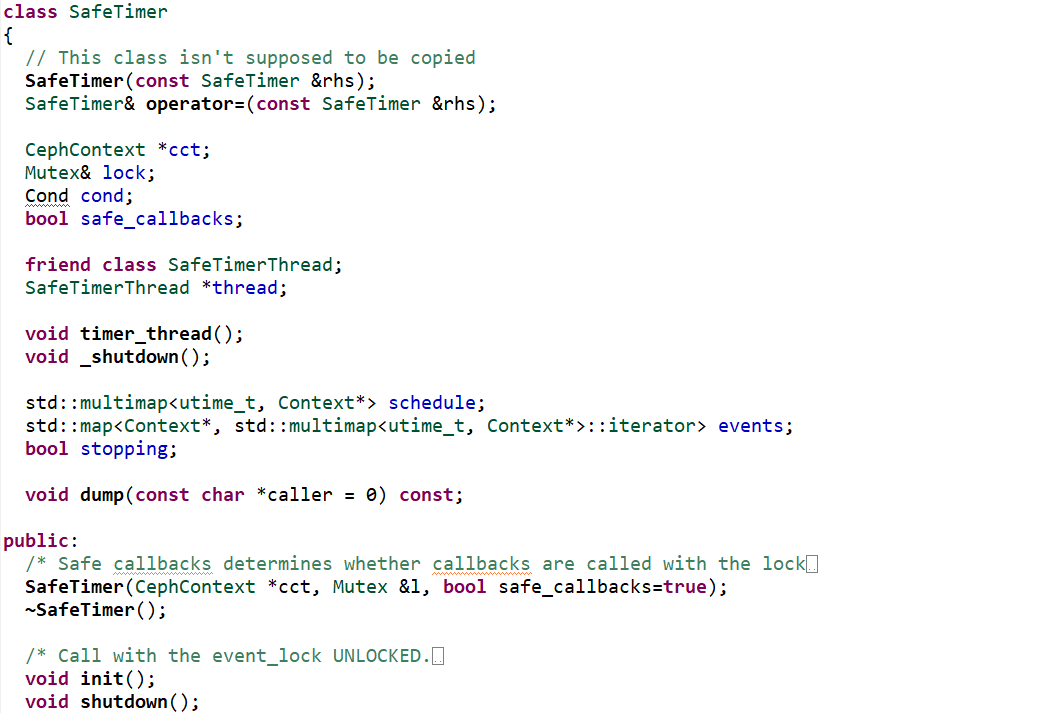
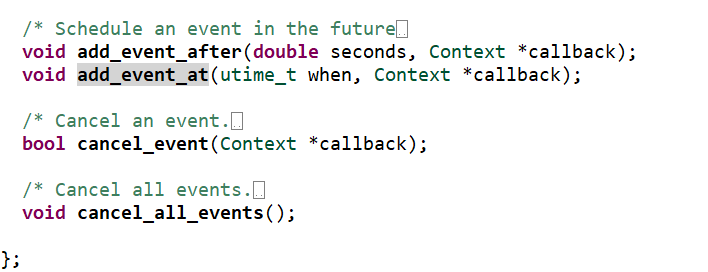
ceph源码中有专门的类封装定时器的功能。

在src/common/Timer.h 中定义了类SafeTimer





分析下这个类的成员变量和成员函数：

bool safe\_callbacks; 是否是安全的定时调用

SafeTimerThread \*thread; 定时器执行线程

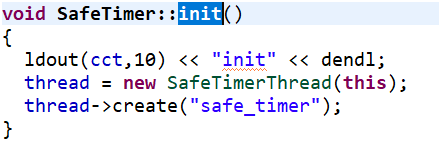
void timer\_thread(); 执行线程执行函数

std::multimap<utime\_t, Context\*> schedule; 目标时间与定时任务执行函数

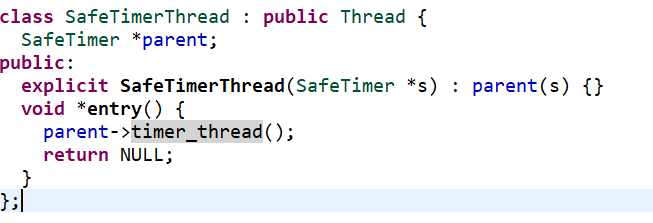
std::map<Context\*, std::multimap<utime\_t, Context\*>::iterator> events; 定时任务与定时任务在schedule中的位置映射。

bool stopping; 是否停止定时器的执行。

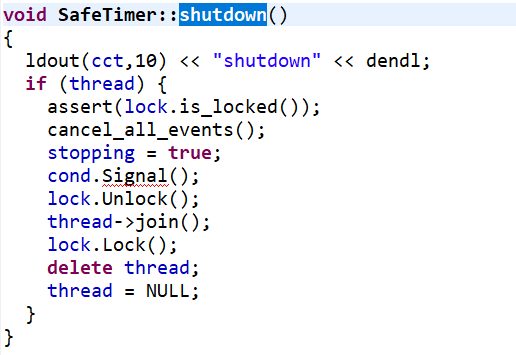
void init(); 初始化定时器，其实就是创建定时线程。



定时线程执行接口就是执行定时器的timer\_thread()函数。

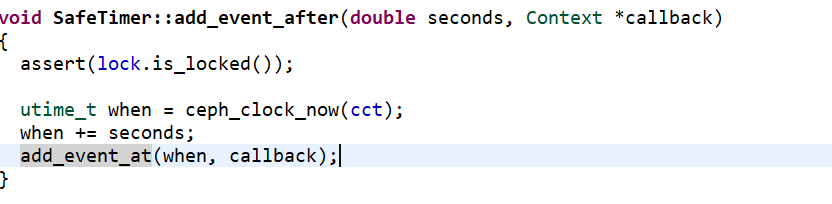


void shutdown(); 关闭定时器。其实就是结束所有定时活动以及停止删除线程。



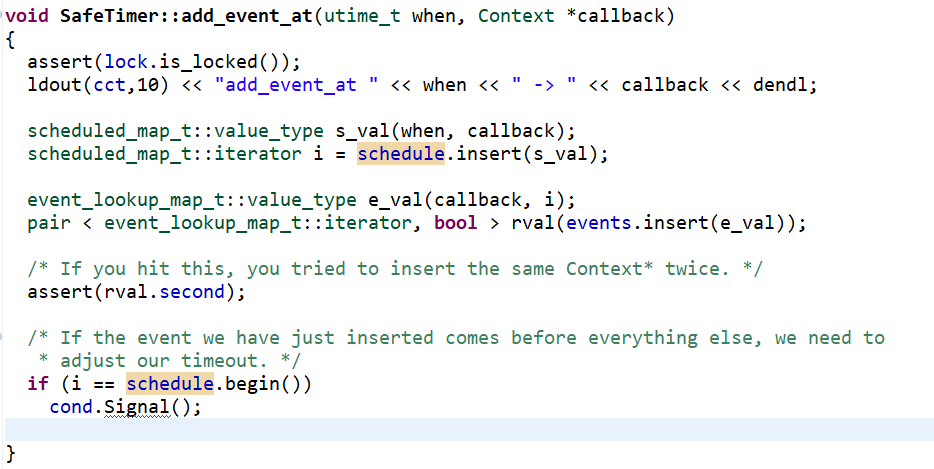
**void** **add\_event\_after**(**double** seconds, Context \*callback);

添加定时任务，设定当前时间多少秒后执行的定时任务，最后也调用add\_event\_at函数。



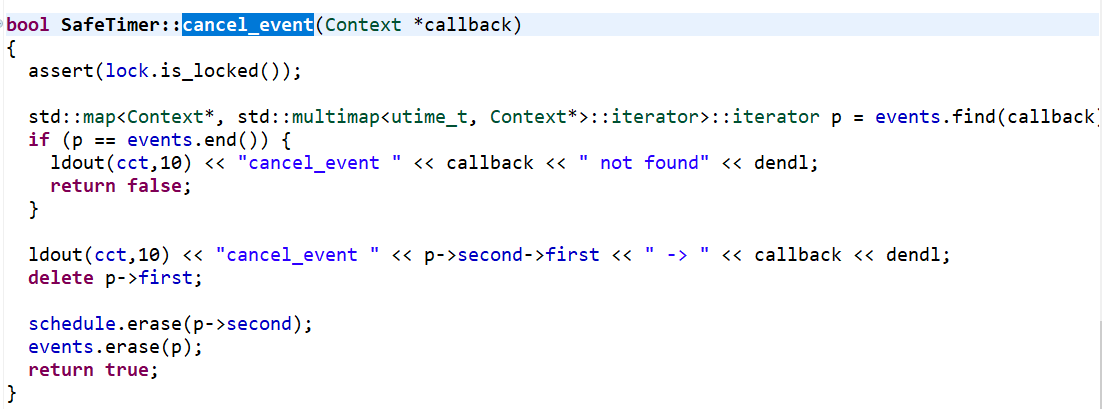
**void** **add\_event\_at**(utime\_t when, Context \*callback);

添加定时任务，设定未来某一时刻执行的定时任务

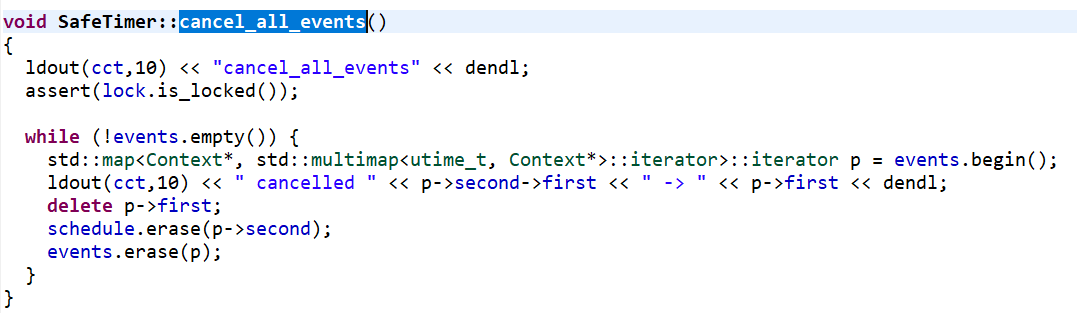


bool cancel\_event(Context \*callback);

终止某一定时任务。

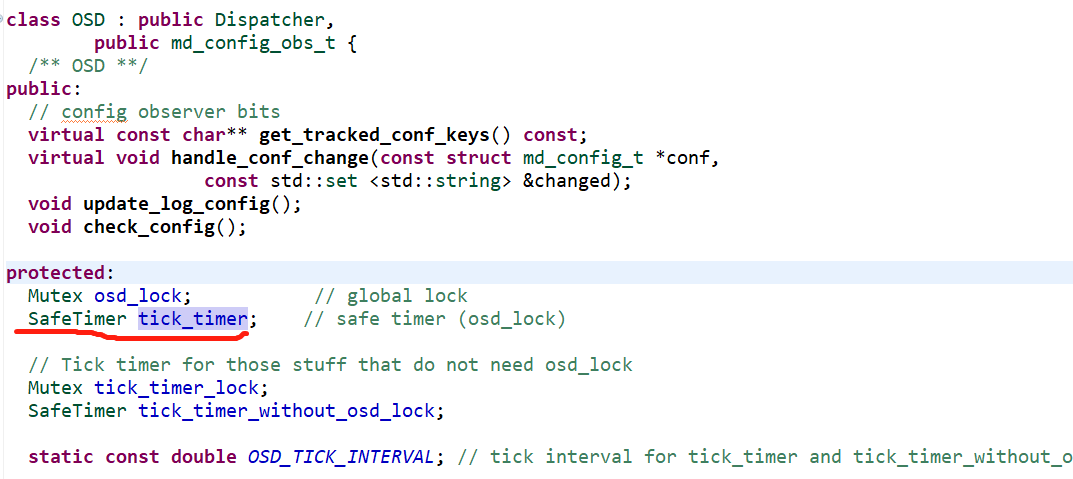


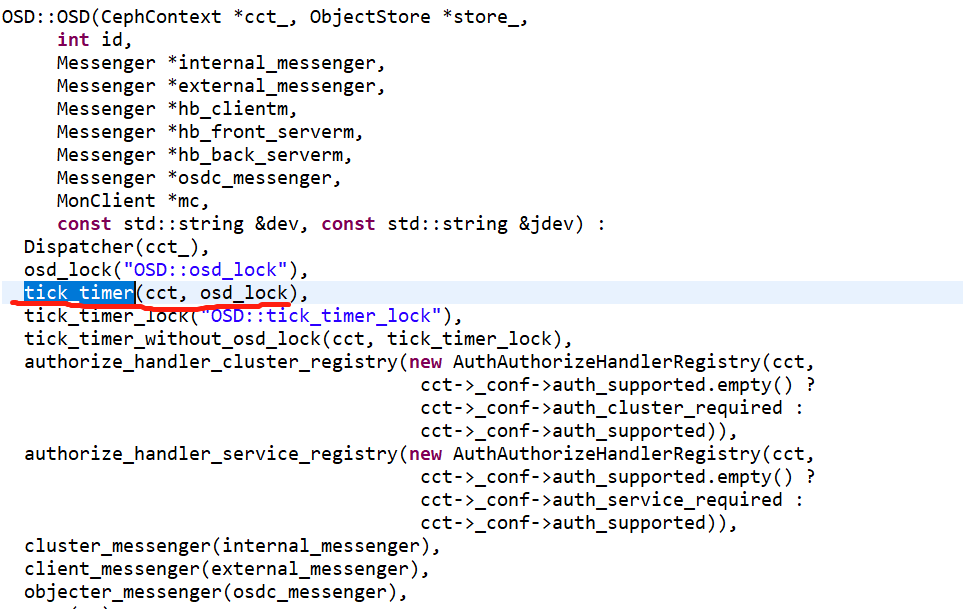
void cancel\_all\_events();终止所有定时任务



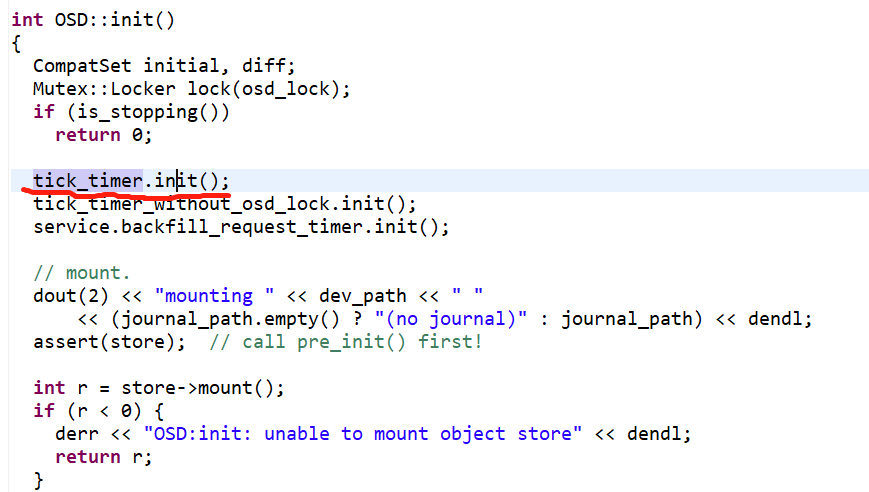
我们整理下如何使用定时器：

在OSD之间心跳检测中就采用了定时器机制：  
1.首先构造定时器对象，在构造osd对象时

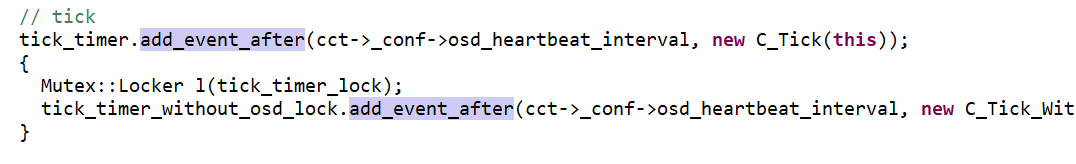




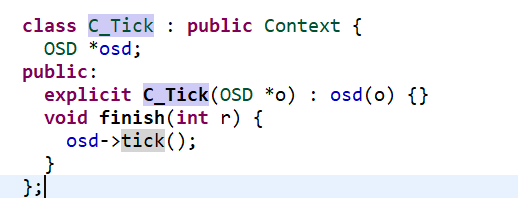
2． 然后初始化定时器，在osd init时



3.添加定时任务，在osd init时



其定时任务就是调用osd::tick()



4. 关闭定时器，在osd shutdown时

