# 带你读源码之MySQL连接管理

环境说明

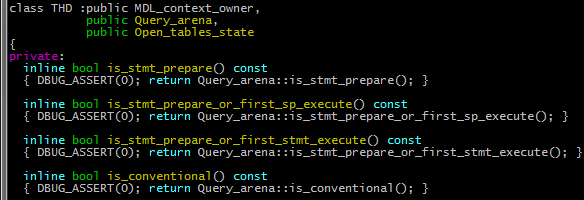
MySQL 5.7.17

Centos 6.4

## 一、MySQL连接

此处首先定义本文中MySQL连接这个概念。MySQL连接这个概念承载着如下两个最为重要的概念，一是事务，一是通讯。事务就是我们常说的那个具有ACID属性的事务，通讯是指服务器和客户端进行信息交互。

如下就是MySQL连接对象的类定义

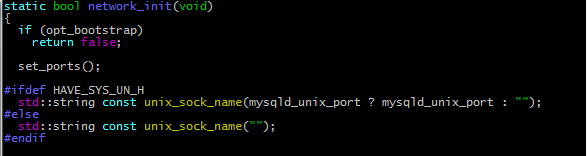


本篇文章将重点讲解与通讯相关的，以及MySQL管理连接的相关机制。事务相关的部分将不做介绍。讲解将按照服务器的启动-运行-关闭这个顺序进行，不过运行部分实际包括端口监听，以及连接建立两个部分。

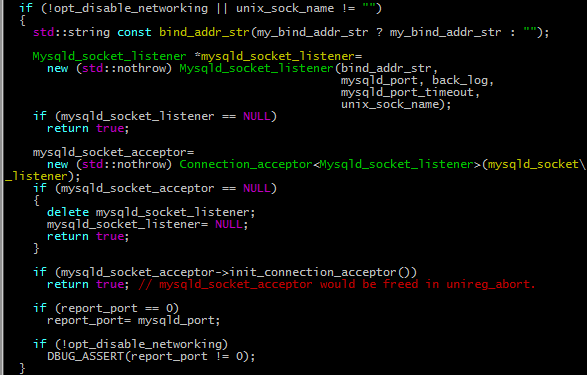
## 启动

针对不同操作系统MySQL有不同的通讯方式支持，Linux系统下仅支持socket的通讯方式；Windows下MySQL支持socket、命名管道（named pipe）和共享内存（shared memory）3种通讯方式。MySQL用户可以根据所在的操作系统和使用需求，在MySQL的配置文件中设置通讯方式，以满足使用的需求。

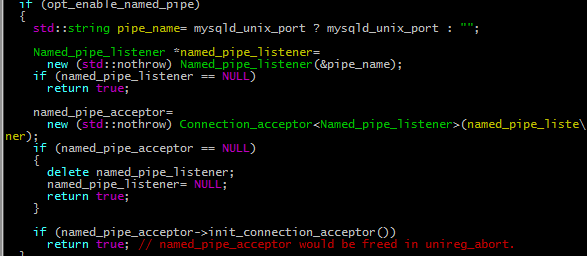
在进行通讯前需要进行通讯相关的初始化工作。如下是服务器启动时，进行初始化的函数，通过该函数，完成创建通讯端口等工作。



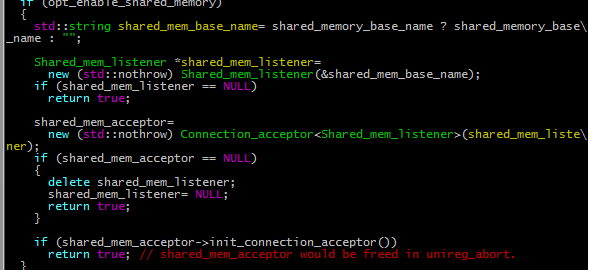
socket通讯的初始化



命名管道的初始化



共享内存的初始化



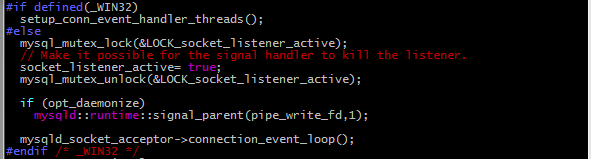
函数network\_init执行结束后，通讯相关的初始化已经完成，但此时服务器并不能对外提供服务，需要监听启动后，才能进行对外的服务。

## 监听/建立连接

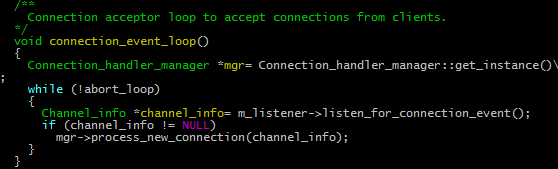
### 监听

通讯组件初始化完毕之后，服务器会调用通讯组件中启用通讯组件中的监听功能，对通讯端口进行监听，进行连接的建立。

在mysqld\_main这个主线程的主函数中，通过如下代码进行启动监听功能



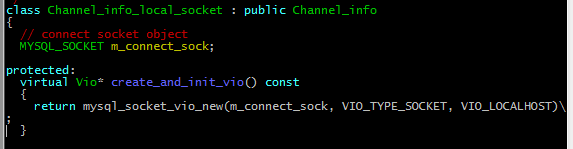
该函数不断监听通讯端口，当客户端进行连接后，创建Channel\_info类对象，并由该对象进行连接创建。



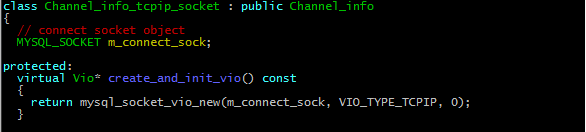
#### Channel\_info

从字面理解Channel是频道的意思，可以理解为通信的通道，该类型的对象记录通信通道的信息。根据《启动》一章，可以知道有3种不同类型的通讯方式。Channel\_info类只是一个基类，针对不同的通讯方式有具体对应的Channel\_info子类

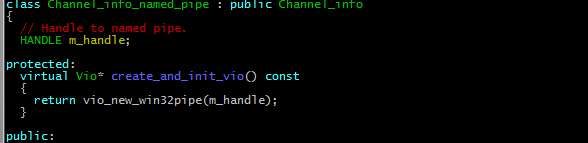
Linux下通过socket文件连接的Channel类



通过网卡进行socket连接的Channel类



通过命名管道连接的Channel类

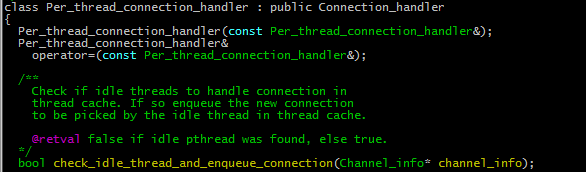


### 建立连接

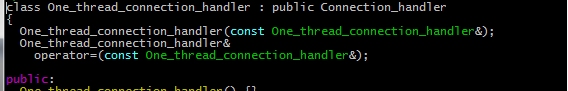
Channel对象创建后，说明有客户端进行连接。需要根据创建的Channel对象，进行连接创建。由此引出MySQL的连接关键。

MySQL源码自带两种连接管理，connection per thread连接管理和 connection one thread。前者是MySQL的默认管理方式，也就是每成功创建一个连接，都会有一个线程专门为该连接的客户端提供服务。后者是只有一个线程，这个线程为所有的客户端连接提供服务。Oracle针对MySQL还提供了企业版的线程池插件，该插件为收费内容，避免了connection per thread连接管理中连接过多情况下大量CPU资源消耗在线程切换上。

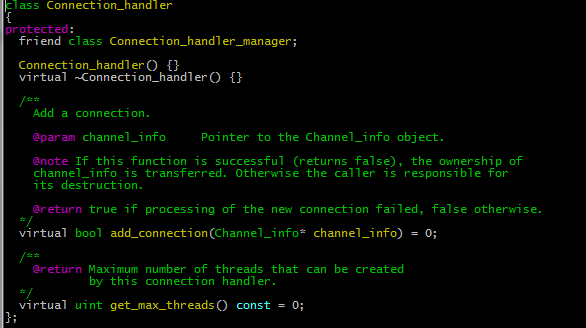
connection per thread连接管理的类



connection one thread连接管理的类



通过这两个类可以看到，实现MySQL管理，必须继承自Connection\_handler类，该类的定义如下



Connection\_handler类提供两个接口函数add\_connection和get\_max\_threads

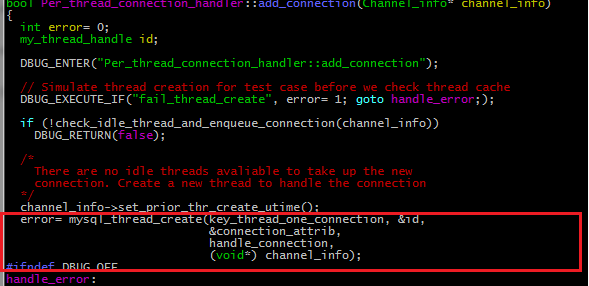
函数add\_connection实现将创建THD类对象，并对该类对象进行管理

函数get\_max\_threads获取该实现下最大线程数

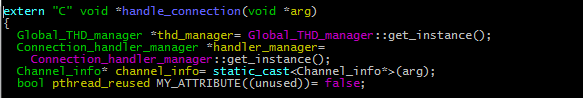
#### connection per thread的实现

该部分将介绍MySQL的connection per thread的实现，该实现为社区版所提供。

如下是connection per thread的add\_connection实现，红框部分是该实现为连接创建服务线程的部分



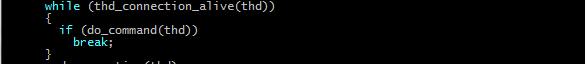
在如下函数handler\_conenction中，完成连接的创建，以及客户端请求的处理



该部分通过Channel对象创建THD类对象



该部分是不断接受客户端请求并进行处理的代码

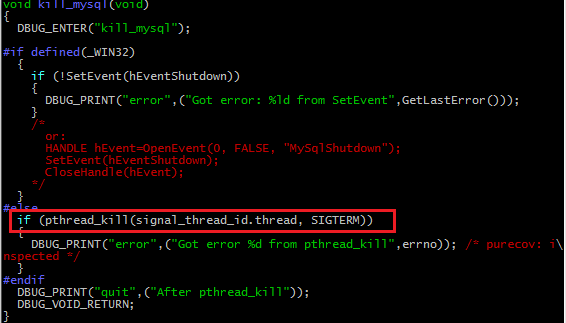


至此，是MySQL服务启动后到建立连接并提供服务的一个过程

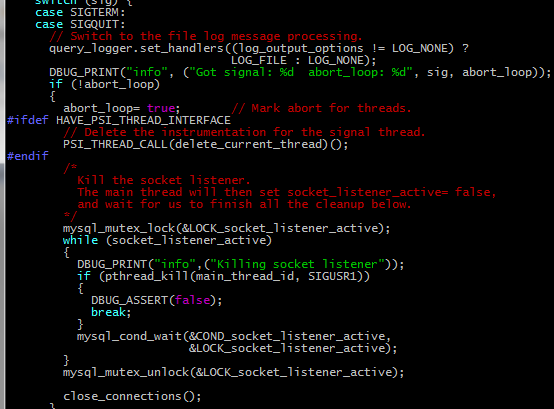
## 关闭

以connection per thread为例子，该部分讲解MySQL正常关闭过程中，连接是如何管理的。

执行shutdown语句或者调用命令mysqladmin shutdown后，服务器进入关闭流程。关闭服务器会调用如下函数，函数的红框部分将向函数signal\_hand所运行的线程发送信号SIGTERM，由signal\_hand所运行的线程进行关闭处理。



signal\_hand函数，在接收到SIGTERM信号后，设置abort\_loop为true，并向主线程发送SIGUSR1信号，目的是通知端口监听线程退出，只有在确保监听线程退出后，不会产生新的Channel对象，以及产生创建连接的行为后，才会进行下一步的操作。停止监听行为后，调用close\_connections关闭监听端口以及已建立的连接。



函数close\_connections进行连接相关的关闭处理

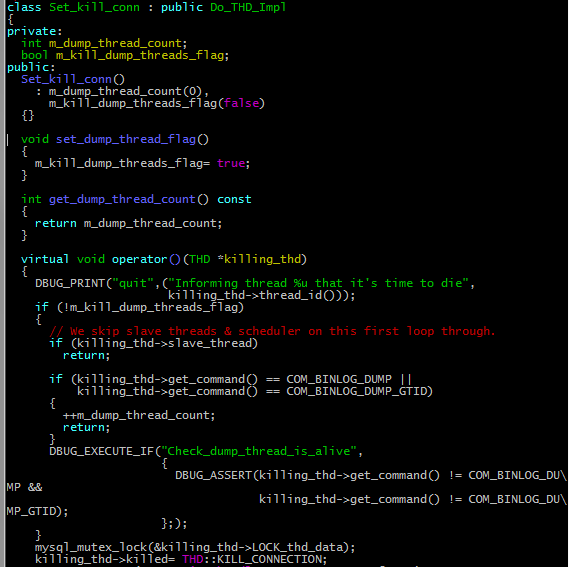
1、关闭监听端口



2、终止运行中的事务



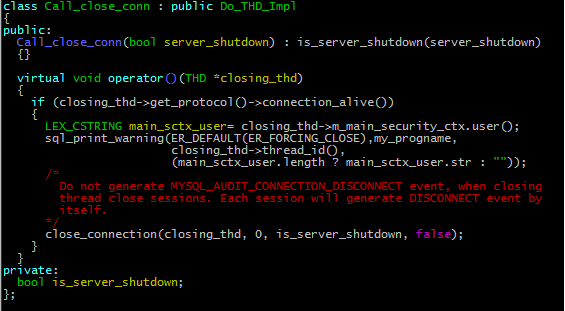
终止事务的实现



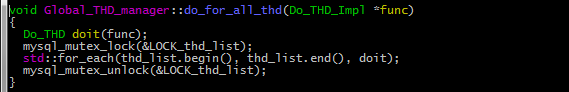
3、关闭端口



关闭连接的实现



do\_for\_all\_thd的实现，该部分代码对所有的连接进行传入函数的执行



当函数close\_connections执行完成后，通讯相关组件都被关闭，服务器继续进入关闭流程