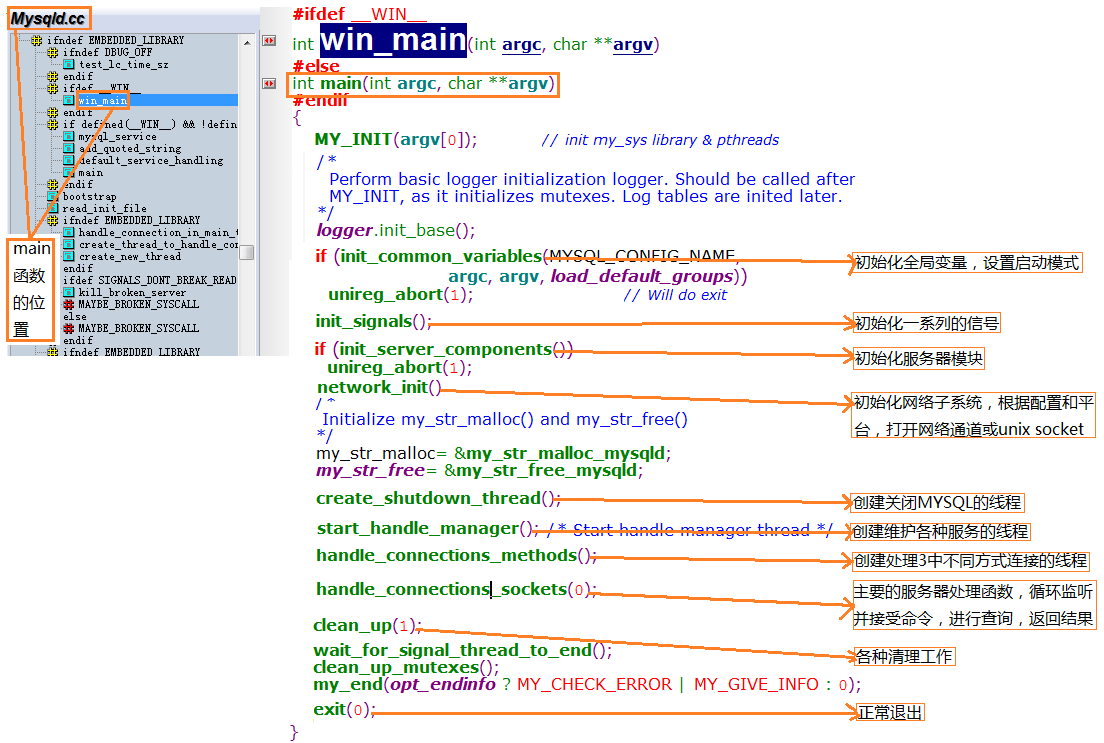
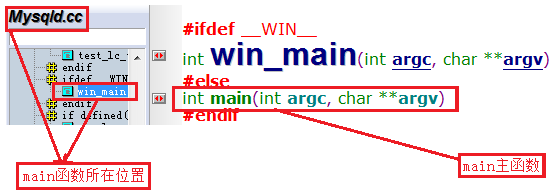
MYSQL\_C/S\_处理流程

-----------寇晓锋

-------望发现错误或有改进之处请发至1035316071@qq.com

一切都是从熟悉的main()函数开始的，这些代码都在mysqld.cc。Main()函数是Mysql服务器开始执行的地方，是服务器的可执行代码被加载到内存里以后执行的第一个函数。这个函数里的几百行代码是用来完成启动Mysql服务器和进行系统初始化工作的。下面截图 给出了main()函数经过高度浓缩的框架。



进入main（）函数之后，我想介绍给大家的第一个子函数是init\_common\_variables()，这个函数使用命令行参数来控制MySQL服务器将如何工作，服务器就是在这里解读命令行参数的，并根据那些参数而启动各种模式的。这个函数负责设置系统变量和把服务器启动为指定的模式。

init\_server\_components()，函数负责对MySQL各个子系统的一系列数据日志进行初始化，这些日志记录记载着在MySQL系统内发生的各种事件、语句执行情况等。

在此,我想特别提醒大家注意两个非常重要的my\_库函数：my\_str\_malloc() and my\_str\_free()。这两个函数指针是在服务器启动代码是（差不多刚刚开始执行）被设置的。这两个MySQL函数增强了出错处理方面的功能，安全性更有保障，所以你应该始终使用它们来代替C/C++语言里的malloc（）函数。

acl\_init(...)函数的任务就是启动MySQL的身份验证和访问控制子系统，这个关系着整个系统的安全，所以才会这么早就出现在MySQL服务器的启动代码里。

接下来的两个函数将创建两个非常重要的辅助线程。create\_shutdown\_thread()函数创建的线程负责在收到有关信号时关闭MySQL， start\_handle\_manger()函数创建的线程负责处理各种服务器范围内的维护功能。

初始化工作完成之后，MySQL已经做好准备接受连接了。然后Handle\_connections\_methods()函数来了，这个函数的主要工作是新建3个子线程，他们分别接受TCP/IP、命名管道以及共享内存这三种方式的连接。一般情况下客户都是用TCP/IP（socket）来连接MySQL服务器的，这是最有弹性的通信方式。但是在嵌入式软件的应用环境中，需要采用后两种通信方式。



当启动代码执行到这个地方的时候，系统已经为接收来自客户端的连接做好了准备工作。接下来，handle\_connections\_sockets(0)函数实现了一个监听器，它将循环等待来自客户端的连接。后面我们在详细讨论这个函数。

上面我们已经看到了MySQL系统是如何启动，以及控制权如何转到等待用户连接的监听器循环上。

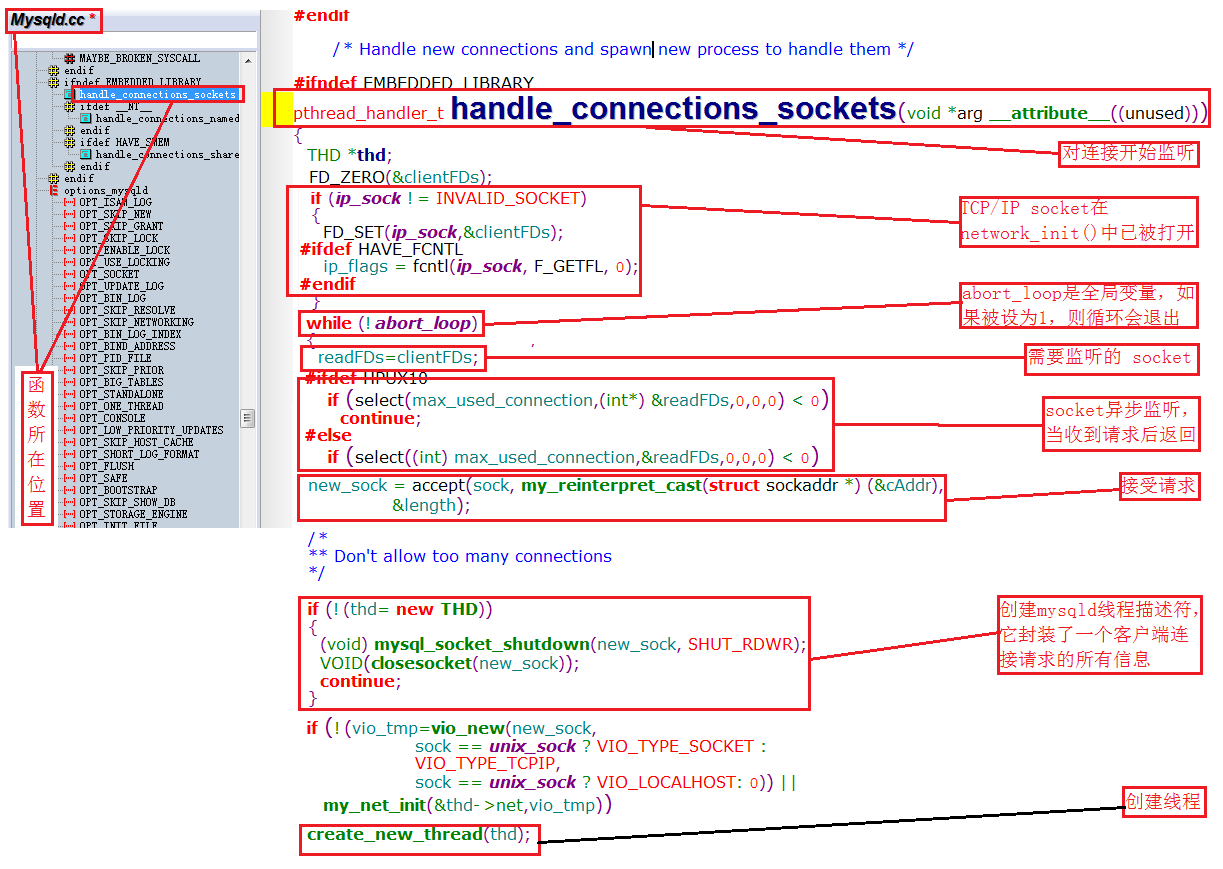
连接是由客户端（程序）创建的，客户端软件将把它们拆分并封装为一组数据包，然后把那些数据包放到网络上，服务器的网络子系统将从网络通信线路上捕获那些数据包，并且在服务器上把他们重组为数据。

新建了3个线程之后，handle\_connectins\_methods()函数进入一个长时间循环，直到3个连接线程全部退出后才退出。这里我主要看看socket的连接线程，我们的研究对象就是这个handle\_connections\_sockets()函数；

handle\_connections\_sockets()函数使用select()调用监听mysqld的端口，然后等待客户端的连接。等到一个客户端连接后，这个函数中会新建一个THD类型的变量，从连接建立开始，到SQL语法分析、查询执行、结果返回等等。这个变量一直都在，总之这是一个非常重要的变量。

还有struct st\_vio这个结构体，这个结构体是一个命令的中转站。在THD中也定义了一个vio类型的结构体。这个结构体的功能就是从储存套接字中读取通信内容，然后又把自己的值赋给THD的vio变量。VIO类型中详细的描述了一次请求，包括请求的内容、时间、请求的套接字地址等等。之后发生的事情就是把这个THD传递到服务线程，create\_new\_thread（）实现这个功能。

在MYSQL系统里，处理连接和创建线程的工作是由SQL接口子系统负责。换句话说，数据包（包含着查询命令）已经到达了服务器，并通过handle\_connections\_sockets(0)函数被探测到。这个函数其实是一个无限循环，该循环的退出条件是变量abort\_loop被设置为TRUE。

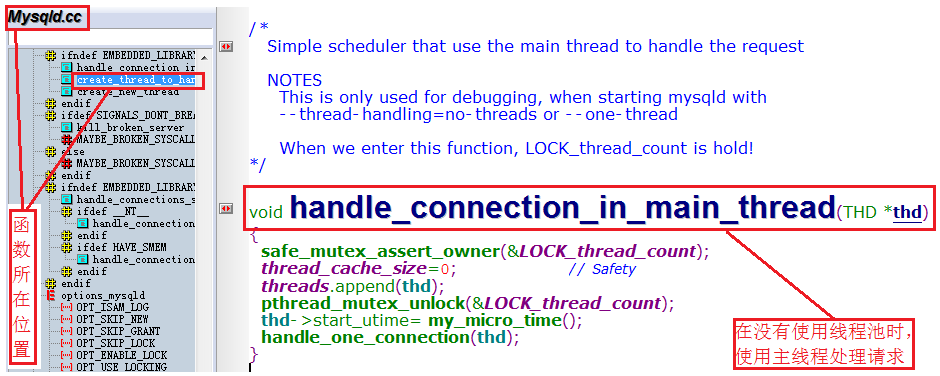


下面是连接处理的代码。在探测到一个连接的时候，该函数调用create\_new\_thread（）函数创建一个新的线程。最主要的MYSQL数据结构当中的第一个结构就是在这个函数里被创建出来的。THD类负责保存和管理与线程有关的所有信息。虽然THD类不是在一个独享的内存空间里被分配给线程的，但系统可以通过它对线程的执行情况进行全面的控制。

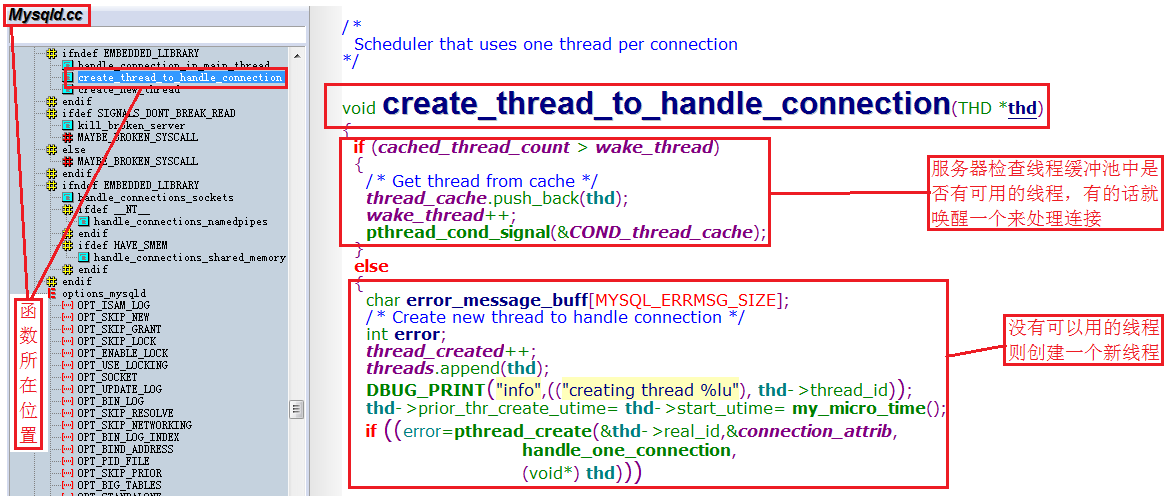
现在，客户端已经与服务器连接上了。接下来会发生些什么？一起去看看create\_new\_thread(thd)函数里会发生些什么。以下是create\_new\_thread(thd)函数的主要代码，你会首先看到一个用来锁定线程计数器的互斥调用。正如你在main()函数里看到的那样，这是避免其他线程竞争对该变量进行写操作所必需的。在新线程被创建出来之后，一个相应的解锁互斥调用将释放被锁定的资源。



对于没有开启线程池功能的连接处理**（只用于调试时用）**



对于启用线程池功能的连接处理



在这个函数开始的地方，程序试图在连接池里找一个现有的线程来重复使用。如果在连接池里没找到一个可重复使用的线程，系统调用pthread\_create()函数创建。

现在，用户的命令已经从客户端到达了服务器，服务器也为来自客户端的连接分配了一个线程来处理它，接下来，控制权转到了handle\_one\_connection（）函数

创建连接使用handle\_one\_connection函数:

handle\_one\_connection(void \*arg)调用do\_command()开关来处理来自客户端的连接（命令请求）。将执行结构返回给客户端。

之后，handle\_one\_connection()函数进入到线程结束和清理阶段。

但end\_thread()函数完成一些基本的清理后，服务器又要面临两个选择：缓存该线程或结束线程。

If(put\_in\_cache&&cached\_thread\_count<thread\_cached\_size&&!abort\_loop&&!kill\_cached\_threads)

若end\_thread()函数确定缓存该线程，就执行如下循环：

While(!abort\_loop&&wake\_thread&&!kill\_threads)

{(void)pthread\_cond\_wait(&COND\_thread\_cached,&LOCK\_thread\_count)};

Pthread\_exit();

到目前为止，MYSQL已经启动，而且客户端已经能连接上MYSQL，服务器也能处理连接了。