安装完Mysql 后，使用VS打开源码开开眼，我嘞个去，这代码和想象中怎么差别这么大呢？感觉代码有些凌乱，注释代码都写的比较随意，好像没有什么统一的规范，不同的文 件中代码风格也有差异，可能Mysql经过了很多牛人的手之后，集众牛人之长吧。也可能是我见识比较浅薄，适应了自己的代码风格，井底之蛙了，总之还是怀 着敬畏的心情开始咱的源码之旅吧。本人菜鸟，大神轻拍。

Mysql可以启动起来了，应该怎么学习呢？总不能从main开始一步一步的 看吧，Mysql作为比较底层的大型软件，涉及到数据库实现的方方面面，没有厚实的数据库理论基础和对Mysql各个模块相当的熟悉，从main开始势必 会把自己引入某个死胡同啊，什么都看，最后啥也不会，咱伤不起。

经过思考后，我想还是通过客户端来调试服务器，从而学习服务器代码比较现实。 也就是通过客户端的动作，看服务器的反应。比如从客户端的登录动作来看SERVER如何进行通信、用户识别、鉴定以及任务分配的，通过CREATE TABLE，来看SERVER如何解析DDL语句以及针对不同的存储引擎采取的不同的物理存储方式，通过INSERT语句，来看SERVER如何进行 Btree的操作。通过SELECT语句来看如何进行SQL语句语法树的创建和优化的，通过ROLL BACK，来看SERVER事务是如何实现的。这里主要是通过跟踪代码学习Mysql数据库实现的思想，对于具体的代码不去做过多的追究(主要是我对 C++不是很熟悉)，好读书，不求甚解，呵呵。

由此，暂时准备了以下几条SQL语句，来有针对的进行SERVER的分析

1、 LOGIN(登录)
mysql.exe –uroot –p
2、 DDL(建表语句)
create table tb_myisam(c1 int, c2 varchar(256)) engine = myisam;
create table tb_innodb(c1 int, c2 varchar(256)) engine = innodb;
3、 INSERT
Insert into tb_myisam values(1 , ’寂寞的肥肉’);
Insert into tb_innodb values(1 , ’寂寞的肥肉’);
4、 SELECT
Select c1 from tb_myisam;
Select * from tb_innodb;
5、 ROLLBACK
复制代码
大家都知道，mysql可以通过多个客户端，进行并发操作，当然也包括登录 了。在别人登录的时候，其他的用户可能正在进行一些其它的操作，因此对于登录我们猜测应该有专门的线程负责客户端和服务器的连接的创建，以保证登录的及时 性，对于每个连接的用户，应该用一个独立的线程进行任务的执行。

首先介绍下mysql中创建线程的函数，创建线程的函数貌似就是 _begin_thread，CreateThread，我们通过VS在整个解决方案中进行查找，bingo！在my_winthread.c中找到了调 用_begin_thread的函数pthread_create，在os0thread.c中找到了调用CreateThread的函数 os_thread_create，一个系统怎么封装两个系统函数呢？？再仔细看下，发现my_winthread.c是在项目mysys下，而 os0thread.c是在项目innobase下。innobase！！这不就是innodb的插件式存储引擎么，原来这是存储引擎自己的封装的底层函数，哥心中豁然开朗了。我想Mysql应用范围如此之广，除了开源之外，插件式的存储引擎功不可没啊，用户可以根据自己的实际应用采取不同的存储引擎，对于大公司，估计会开发自己的存储引擎。

下面分析下pthread_create是如何调用_begin_thread的，先粗略看下源码。

```
int pthread_create(pthread_t *thread_id, pthread_attr_t *attr,
pthread_handler func, void *param)
{
HANDLE hThread;
struct pthread_map *map;
DBUG_ENTER(“pthread_create”);
if (!(map=malloc(sizeof(*map))))
DBUG_RETURN(-1);
map->func=func;
map->param=param;
pthread_mutex_lock(&THR_LOCK_thread);
#ifdef __BORLANDC__
hThread=(HANDLE)_beginthread((void(_USERENTRY *)(void *)) pthread_start,
attr->dwStackSize ? attr->dwStackSize :
65535, (void*) map);
#else
hThread=(HANDLE)_beginthread((void( __cdecl *)(void *)) pthread_start,
attr->dwStackSize ? attr->dwStackSize :
65535, (void*) map);
#endif
```