wget  https://dev.mysql.com/get/Downloads/MySQL-5.5/mysql-5.5.59.tar.gz

tar xf mysql-5.5.59.tar.gz

cd mysql-5.5.59

groupadd mysql

useradd -g mysql  mysql -M  -s /sbin/nologin

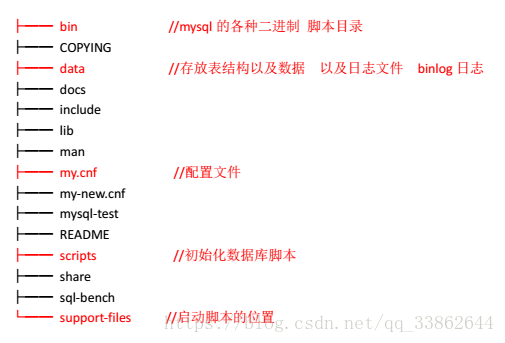
yum install cmake ncurses-devel  autoconf

yum install gcc gcc-c++

*#configure.cmake编译的（他的功能相当于./configure）*

cmake -DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql -DMYSQL\_UNIX\_ADDR=/tmp/mysql.sock -DMYSQL\_USER=mysql -DDEFAULT\_CHARSET=utf8 -DDEFAULT\_COLLATION=utf8\_general\_ci -DWITH\_MYISAM\_STORAGE\_ENGINE=1 -DWITH\_INNOBASE\_STORAGE\_ENGINE=1 -DWITH\_DEBUG=0 -DWITH\_READLINE=1 -DWITH\_EMBEDDED\_SERVER=1 -DENABLED\_LOCAL\_INFILE=1

make && make isntall



初始化数据库后才能启动

初始化数据库

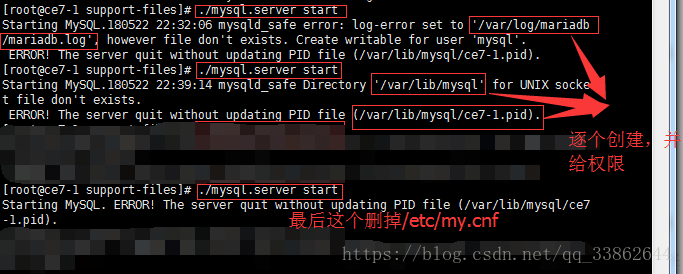
cd /usr/local/mysql/scripts

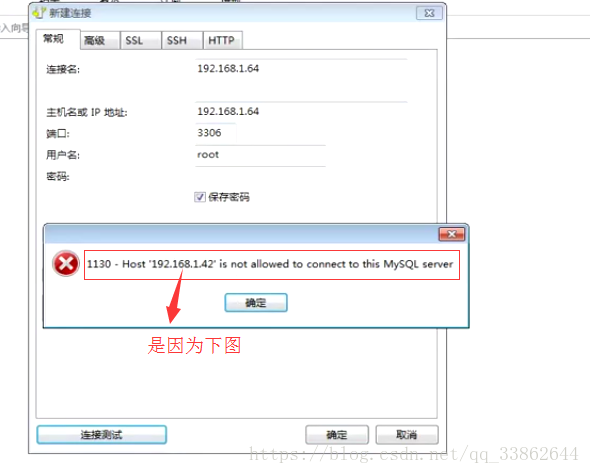
./mysql\_install\_db --user=mysql --datadir=/usr/local/mysql/data --basedir=/usr/local/mysql

启动mysql

cd /usr/local/mysql/support-files

./mysql.server start|stop|restart





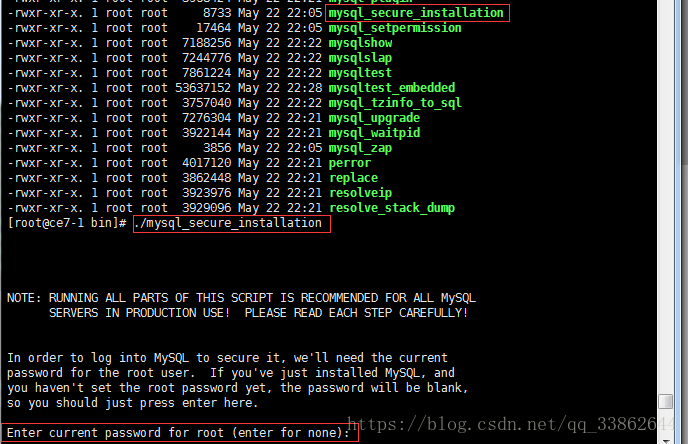
连接数据库（默认密码为空，要关防火墙）

## ****删除匿名用户 配置 mysql 快捷访问****

cd /usr/local/mysql/bin

会报一个找不到mysql....这一系列命令的错

PATH="$PATH:/usr/local/mysql/bin";    //设置成永久生效的全局变量，将它添加到vim /etc/profile的最后一行



1、Enter current password for root (enter for none)    //输入当前root用户的密码，是''，直接回车

2、Set root password?    //设置root的新密码，输入y

    New password    //输入新密码

    Re-enter new password    //在输入一遍新密码

3、Remove anonymous users?    //删除匿名用户，输入y

4、Disallow root login remotely?    禁止root用户的远程登录（也就是说，root用户只能在服务器登录。用普通用户连接navicat），输入y

5、Remove test database and access to it?    //删除测试数据库，输入y

6、Reload privilege tables now?    //刷新权限，输入y

## **创建并给普通用户分配权限(**供其连接远程工具navicat等**)**

grant select,delete,update,insert on blog.\* to 'blog'@'192.168.2.101' identified by '123456';

flush privileges;

设置全局变量

vim  /etc/profile

PATH="$PATH:/usr/local/mysql/bin"   //永久生效

命令行执行

export PATH="$PATH:/usr/local/mysql/bin"  //临时执行

或者 cp  /usr/local/mysql/bin/mysql  到  /usr/bin

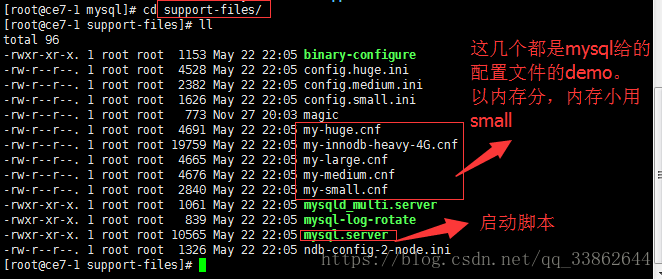
### 配置文件详解

让mysql还是用/etc/init.d启动

cd /usr/local/mysql/support-files/  #切换进mysql安装目录下的启动脚本

cp mysql.server /etc/init.d/mysql    #mysql的启动脚本，复制到/etc/init.d并下面改个名字

/etc/init.d/mysql start                     #使用新指令启动mysql



注意：这里的配置文件，随便复制出去一个，改成my.cnf就能用了

## ****配置文件my.cnf****

[client]

port = 3306

socket = /tmp /mysql.sock

[mysqld]

user = mysql             #数据库初始化时指定的，初始化没指定，直接在配置文件中改

port = 3306

socket = /tmp/mysql.sock

basedir = /usr/local/mysql  #数据库初始化时指定的，初始化没指定，直接在配置文件中改

datadir = /data/mysql/data/  #数据库初始化时指定的，初始化没指定，直接在配置文件中改

max\_connections = 3000  #MySQL允许最大的进程连接数，如果经常出现Too Many Connections的错误提示，则需要增大此值。默认100（此处修改就是永久修改）

查看最大连接数  show variables like ‘%connections%’

查看当前最大连接数  show processlist

set global  max\_connections=300;//临时修改

永久修改：是改配置文件  然后重启数据库

注意：这些全局变量都是在my.cnf中配置的，如果没有就是默认

解决方法：

现在线上正用着呢，不能停并发大，连接数不够用了。

就用set global先设置，配置文件的也改咯。有时间咯在重启mysql

#sort\_buffer\_size = 2M  //connection级参数，在每个connection（session）第一次需要使用这个buffer的时候，一次性分配设置的内存。

Sort\_Buffer\_Size 并不是越大越好，由于是connection级的参数，过大的设置+高并发可能会耗尽系统内存资源。例如：500个连接将会消耗 500\*sort\_buffer\_size(8M)=4G内存

Sort\_Buffer\_Size 超过2KB的时候，就会使用mmap() 而不是 malloc() 来进行内存分配，导致效率降低。 系统默认2M，使用默认值即可

#join\_buffer\_size = 2M  #用于表间关联缓存的大小，和sort\_buffer\_size一样，该参数对应的分配内存也是每个连接独享。系统默认2M，使用默认值即可

thread\_cache\_size = 300  #默认38

服务器线程缓存这个值表示可以重新利用保存在缓存中线程的数量,当断开连接时如果缓存中还有空间,那么客户端的线程将被放到缓存中,如果线程重新被请求，那么请求将从缓存中读取,如果缓存中是空的或者是新的请求，那么这个线程将被重新创建,如果有很多新的线程，增加这个值可以改善系统性能.通过比较 Connections 和 Threads\_created 状态的变量，可以看到这个变量的作用。设置规则如下：

1GB 内存配置为8，

2GB配置为16，

3GB配置为32，

4GB或更高内存，可配置更大。

#thread\_concurrency = 8  #系统默认为10，使用10先观察

设置thread\_concurrency的值的正确与否, 对mysql的性能影响很大, 在多个cpu(或多核)的情况下，错误设置了thread\_concurrency的值, 会导致mysql不能充分利用多cpu(或多核), 出现同一时刻只能一个cpu(或核)在工作的情况。

thread\_concurrency应设为CPU核数的2倍. 比如有一个双核的CPU, 那么thread\_concurrency的应该为4；2个双核的cpu, thread\_concurrency的值应为8

query\_cache\_size = 64M  #在MyISAM引擎优化中，这个参数也是一个重要的优化参数。但也爆露出来一些问题。

机器的内存越来越大，习惯性把参数分配的值越来越大。这个参数加大后也引发了一系列问题。

我们首先分析一下 query\_cache\_size的工作原理：一个SELECT查询在DB中工作后，DB会把该语句缓存下来，当同样的一个SQL再次来到DB里调用时，DB在该（查询的）表没发生变化的情况下把结果从缓存中返回给Client。

这里有一个关建点，就是DB在利用Query\_cache工作时，要求该语句涉及的表在这段时间内没有发生变更。

那如果该表在发生变更时，Query\_cache里的数据又怎么处理呢？

首先要把Query\_cache和该表相关的语句全部置为失效，然后在写入更新。那么如果Query\_cache非常大，该表的查询结构又比较多，查询语句时效也慢，一个更新或是Insert就会很慢，这样看到的就是Update或是Insert怎么这么慢了。

所以在数据库写入量或是更新量也比较大的系统，该参数不适合分配过大。而且在高并发，写入量大的系统，建议把该功能禁掉。

query\_cache\_limit = 4M  #指定单个查询能够使用的缓冲区大小，缺省为1M

query\_cache\_min\_res\_unit = 2k  #默认是4KB，设置值大，对大数据查询有好处，但如果你的查询都是小数据查询，就容易造成内存碎片和浪费

查询缓存碎片率 = qcache\_free\_blocks / qcache\_total\_blocks \* 100%

#如果查询缓存碎片率超过20%，可以用FLUSH QUERY CACHE整理缓存碎片，或者试试减小query\_cache\_min\_res\_unit，如果你的查询都是小数据量的话。

#查询缓存利用率 = (query\_cache\_size – qcache\_free\_memory) / query\_cache\_size \* 100%

#查询缓存利用率在25%以下的话说明query\_cache\_size设置的过大，可适当减小;查询缓存利用率在80%以上而且qcache\_lowmem\_prunes > 50的话说明query\_cache\_size可能有点小，要不就是碎片太多。

#查询缓存命中率 = (qcache\_hits – qcache\_inserts) / qcache\_hits \* 100%

#default-storage-engine = MyISAM

#default\_table\_type = InnoDB      #开启失败，因为你上面设置的表引擎不是innodb

myisam\_recover  #自动检查和修复没有适当关闭的 MyISAM 表

skip-name-resolve

lower\_case\_table\_names = 1

server-id = 1 #主从配置，1为master

innodb\_buffer\_pool\_size = 2048M       //默认是物理内存的50%

这对Innodb表来说非常重要。Innodb相比MyISAM表对缓冲更为敏感。MyISAM可以在默认的 key\_buffer\_size 设置下运行的可以，然而Innodb在默认的 innodb\_buffer\_pool\_size 设置下却跟蜗牛似的。由于Innodb把数据和索引都缓存起来，无需留给操作系统太多的内存，因此如果只需要用Innodb的话则可以设置它高达 70-80% 的可用内存（相对于物理内存计算的）。一些应用于 key\_buffer 的规则有 — 如果你的数据量不大，并且不会暴增，那么无需把 innodb\_buffer\_pool\_size 设置的太大了

log-bin = mysql-bin  #binlog日志文件，记录【增删改】的sql语句，这样就是开启了，如果数据库出问题了，可以恢复。该文件默认保存在/usr/local/mysql/data/mysql-bin.00001...(存的是二进制看不懂)

expire\_logs\_days     =7   #binlog过期清理时间（单位天），清理之前要确定自己备份过数据库