

前言

概述

本文详尽阐述了在Windows平台上使用MySQL源码安装的过程，并深入探讨了基于aarch64-none-linux-gnu工具链的MySQL移植技术。此外，文章还提供了对常见的基于C语言的MySQL API的详细使用指南。通过本文的学习，您将能够更高效地安装、移植和使用MySQL，为您的开发工作带来极大的便利。

版本

V1.0.0

修改说明

无

Mysql安装(Windos 源码)

前言

概述

不知道写什么

环境

- Windows 10 专业版

主要参考资料

- [MySQL 5.7详细下载安装配置教程-CSDN博客](#)

安装流程

- 下载Mysql5.7源码
- 配置环境变量
- 创建my.ini文件
- 安装Mysql5.7
- 设置Mysql root用户密码

安装Mysql

下载源码

- 下载Mysql

进入[Mysql官网](#)选择DOWNLOADS中的MySQL Comm server

进入下载界面选择Archives选项选择Product Version为对应版本5.7.36，选择Operating System为Windos平台，选择对应平台对应位数版本，选择Windos(X86,64-bit).ZIP Archive。

- 解压源码

在D盘创建文件夹Mysql5.7，并将之前下载的压缩包移动到Mysql5.7文件夹中，并解压到对应文件夹中。

设置环境变量

- 创建MYSQL_HOME环境变量

右击此电脑->选择属性->点击高级系统设置->点击环境变量->在系统环境变量中点击新建。变量名输入"**MYSQL_HOME**",变量值为之前下载Mysql源码解压的文件目录，点击确定。

- 编辑Path

选中系统变量中的Path->点击编辑选项->点击新建->"**%MYSQL_HOME%\bin**"->点击确定。

配置my.ini文件

在之前解压目录中创建my.ini文件，并输入如下配置信息：

```
1 [mysqld]
2 #端口号
3 port = 3306
4 #mysql-5.7.27-winx64的路径
5 basedir=D:\Mysql5.7\mysql-5.7.36-winx64
6 #mysql-5.7.27-winx64的路径+\data
7 datadir=D:\Mysql5.7\mysql-5.7.36-winx64\data
8 #最大连接数
9 max_connections=200
10 #编码
11 character-set-server=utf8
12
13 default-storage-engine=INNODB
14
15 sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES
16
17 [mysql]
18 #编码
19 default-character-set=utf8
```

安装Mysql

打开CMD，运行Mysql源码中的安装命令。

- 以管理员权限打开CMD
- cd进入Mysql解压目录的bin文件夹中
- 执行mysqld -install
- 执行mysqld --initialize
- 执行net start mysql

设置用户密码

- 执行命令关闭Mysql服务

```
1 net stop mysql
```

- 修改my.ini文件

mysqld中任意位置添加skip-grant-tables

```
1 [mysqld]
2 #端口号
3 port = 3306
4 #mysql-5.7.27-winx64的路径
5 basedir=D:\Mysql5.7\mysql-5.7.36-winx64
6 #mysql-5.7.27-winx64的路径+\data
7 datadir=D:\Mysql5.7\mysql-5.7.36-winx64\data
8 #最大连接数
9 max_connections=200
10 #编码
11 character-set-server=utf8
12
13 default-storage-engine=INNODB
14
15 sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES
16
17 skip-grant-tables
18
19 [mysql]
20 #编码
21 default-character-set=utf8
```

- Mysql服务

```
1 net start mysql
```

- 连接数据库

```
1 mysql -u root -p
```

- 进入数据库

```
1      use mysql
```

- 修改密码

```
1      update user set authentication_string=password("xxxxxx") where  
      user="root";      xxxxxx为设置的密码
```

修改成功会提示“Query OK”

- 手动停止Mysql服务

打开服务

选中MySQL并停止此服务

- 修改my.ini **删除或屏蔽skip-grant-tables**

```
1 [mysqld]
2 #端口号
3 port = 3306
4 #mysql-5.7.27-winx64的路径
5 basedir=D:\Mysql5.7\mysql-5.7.36-winx64
6 #mysql-5.7.27-winx64的路径+\data
7 datadir=D:\Mysql5.7\mysql-5.7.36-winx64\data
8 #最大连接数
9 max_connections=200
10 #编码
11 character-set-server=utf8
12
13 default-storage-engine=INNODB
14
15 sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES
16
17 #skip-grant-tables
18
19 [mysql]
20 #编码
21 default-character-set=utf8
```

- 再次打开一个cmd
- 启动Mysql服务

```
1 net start mysql
```

- 连接数据库

密码为之前自己设置的密码

```
1 mysql -u root -p
```

- 进入数据库

```
1 use mysql
```

通过密码连接数据库，成功会提示"Database changed"并使前缀变为"mysql>"

安装如果遇到错误常考常见安装错误章节。

Mysql移植

前言

概述

项目需要使用数据库完成数据存储且要求具有数据库同步功能。由于Mysql是最大的开源数据库故数据库选择位Mysql。数据库需要部署到本地(ARM)上，Mysql没有对应的安装包，需要自己移植。且Mysql官方最低版本只有5.7才有官方手册，Mysql8以上的版本需要gcc10、cmake3.75版本太高交叉编译器不支持，于是选用Mysql5.7版本。

Mysql编译依赖与boost和openssl库，openssl库依赖与ncurses库，Mysql交叉编译生成的编译脚本无法在主机上运行，所以还需要主机编译的脚本文件。所以Mysql移植一共需要boost库、ncurses库、openssl库以及Mysql Linux版本脚本，所以移植Mysql前应该先移植boost、ncurses、openssl以及编译主机Mysql(linux)。移植步骤均是下载源码、配置对应设置、编译、安装、查看编译库文件格式。移植方案根据mysql源码一共有两种一种是下载带boost源码的mysql源码，另一种是下载不带boost源码自己下载boost源码编译后通过mysql源码调用。本次采用方案一移植(第二种不会配置boost)。

- 注意

- 由于历史原因问题导致Mysql交叉编译常见问题没有参考文章(但是编译完下班第二天找不到了)
- Mysql5.7.36版本依赖的openssl版本为openssl1.1.1、boost版本为boost1.59.0(其他版本看对应Mysql脚本文件)
- Mysql主机编译和交叉编译用的Mysql源码是同一份

环境

- 虚拟机版本 VMware® Workstation 16 Pro 16.1.2 build-17966106
- Ubuntu版本 Ubuntu 18.04.6 LTS 64位
- Ubuntu编译工具链版本 gcc version 7.5.0
- 交叉编译工具链 aarch64-none-linux-gnu 64位
- 交叉编译工具链版本 gcc version 9.2.1 20191025 (GNU Toolchain for the A-profile Architecture 9.2-2019.12 (arm-9.10))

- vim版本 vim version 8.0.3741
- cmake版本 cmake version 3.10.2

主要参考文献

- 网友资料
 - [MySQL 5.7详细下载安装配置教程-CSDN博客](#)
 - [mysql交叉编译方式](#)
 - [mysql交叉编译到arm平台 \(hasi\)](#)
- 官方手册
 - [MySql5.7参考手册](#)
 - [C API参考手册](#)
 - [boost1.59参考手册](#)
- 源码配置文件
 - ncurses配置文件 命令 `./configure --help`
 - boost配置文件 命令 `./bootstrap.sh --help` `./b2 --help`
 - openssl配置文件 命令 `./config --help`

移植流程

- 移植ncurses
- 移植boost
- 移植openssl
- 编译主机mysql
- 交叉编译mysql

Mysql移植

移植ncurses

下载源码

ncurses官方下载[ncurses6.1](#)版本源码，复制到虚拟机中解压到ncurses-6.1文件夹，并在同级目录下创建用于存放交叉编译文件的ncurses-6.1-arm文件夹。

- 下载官方源码
- 复制压缩包到虚拟机中
- 解压到ncurses-6.1文件夹中


```
1 tar -zxvf ncurses-6.1.tar.gz
```

- 创建ncurses-6.1-arm文件夹作为ncurses-6.1交叉编译安装目录

```
1 mkdir ncurses-6.1-arm
```

修改配置

- 编译选项说明

名称	作用
host	交叉编译器
prefix	编译完成的绝对路径
CC	指定交叉编译器
with-shared	生成共享库
without-progs	禁止使用程序如(tic)构建/安装

- 进入ncurses-6.1目录，设置交叉编译器和安装目录

```
1 cd ncurses-6.1
2 ./configure --host=aarch64-none-linux-gnu --prefix=/home/xiang/Mysql/ncurses-6.1-arm
  CC=aarch64-none-linux-gnu-gcc --with-shared --without-progs
```

- **注意：ncurses安装路径需要使用绝对路径且路径末尾没有"/"。**

编译

- 执行make编译文件

```
1 make

2 make install //将编译文件安装到configure指定的文件夹

3 cd ncurses_6.1-arm

4 ls
```

- 查看安装文件格式是否和交叉编译器一致

```
1 cd lib

2 file libcurses.so.1
```

移植boost

下载源码

在boost官方下载[boost_1_59_0](#)版本源码，复制到虚拟机中解压到boost_1_59_0文件夹，并在同级目录下创建用于存放交叉编译文件的boost_1_59_0-arm文件夹。

- 官网下载[boost_1_59_0](#)
- 复制到虚拟机
- 解压到boost_1_59_0到boost_1_59_0文件夹

```
1 tar -vxf boost_1_59_0.tar.gz
```

- 创建boost_1_59_0-arm文件夹作为boost交叉编译安装目录

```
1 mkdir boost_1_59_0

2 ls
```

修改配置

- 配置选项说明

名称	作用
prefix	安装路径

- 进入解压的boost_1_59_0文件夹输入配置命令

```
1 cd boost_1_59_0
2 ./bootstrap.sh --prefix=/home/xiang/Mysql/boost_1_59_0-arm
```

编译

- 用vim打开project-config.jam配置文件，修改文件中的编译器选项
- 将using gcc 修改为 交叉编译器路径
- 路径应该有空格
- 编译

```
1 ./b2 //编译
2 ./b2 install //安装
```

- 进入编译文件路径并用file命令查看编译文件是否和交叉编译器格式相匹配。

```
1 cd ../boost_1_59_0-arm/lib
2 file libboost_atomic.so.1.59.0
```

移植openssl

下载源码

在openssl官方下载[openssl3.2.0](#)版本源码，复制到虚拟机中解压到openssl3.2.0文件夹，并在同级目录下创建用于存放交叉编译文件的openssl3.2.0-arm文件夹。

- 官方下载[openssl3.2.0](#)(Mysql5.7需要openssl最低版本为1.01)

- 将下载文件复制到虚拟机
- 将openssl-3.2.tar.gz解压到openssl-3.2文件夹

```
1 tar -zxvf openssl-3.2.tar.gz
2 ls
```

- 创建openssl-3.2.0-arm文件夹作为交叉编译安装

```
1 mkdir open-3.2.0-arm
2 ls
```

修改配置

- 配置说明

名称	说明
prefix	安装路径(绝对路径))
cross-compile-prefix	编译器前缀
no-asm	不需要汇编
no-shared	静态编译
shared	共享库编译

- 进入openssl-3.2.0后，通过config配置Makefile选项

```
1 cd openssl-3.2.0
2 ./config --prefix=/home/xiang/Mysql/openssl-3.2.0-arm --cross-compile-prefix=aarch64-  
none-linux-gnu- no-asm no-shared
```

编译

- 执行make编译和安装命令

```
1 make
2 make install
```

- 查看安装文件格式是否和交叉编译器一致

```
1 cd ../openssl-arm/bin
2 file openssl
```

编译主机mysql

安装依赖

主机编译也需要 boost 、 openssl 和 ncurses 库，在Ubuntu系统中openssl和ncurses库可以通过apt-get 命令下载，具体命令如下：

```
1 apt-get install openssl
2 apt-get install libssl-dev
3
4 openssl version    //查看安装的openssl版本
5
6 /* 通过dpkg安装工具查看版本号 */
7 dpkg -l libssl-dev openssl
```

下载源码

官网下载[Mysql5.7.36](#)源码，并复制到虚拟机中的Ubuntu系统中解压到mysql-5.7.36，在同级目录创建mysql-5.7.36-pc文件夹。

将压缩包复制到Ubuntu系统中

解压两份mysql-5.7.36，一份用于主机编译，一份用于交叉编译

```
1 #Ubuntu> tar -zxvf mysql-5.7.36.tar.gz
2 #Ubuntu> cp mysql-5.7.36 mysql-5.7.36-pc
3 #Ubuntu> mv mysql-5.7.36 mysql-5.7.36-arm
```

修改配置

DDOWNLOAD_BOOST	是否下载 Boost 库
DWITH_BOOST	boost路径
DCMAKE_INSTALL_PREFIX	Mysql安装路径

```
1 cd mysql-5.7.36-pc
2 sudo cmake . -DDOWNLOAD_BOOST=1 -DWITH_BOOST=/home/xiang/Mysql/mysql-5.7.36-pc/boost -
  DCMAKE_INSTALL_PREFIX=/home/xiang/Mysql/mysql-5.7.36-pc/lintall
3 ls //查看boost是否安装完成
```

编译

执行make安装命令

```
1 sudo make
2 sudo make install
```

通过file查看安装文件属性

```
1 cd ../mysql-5.7.36-pc/
2 ls
3 cd bin/
4 ls
5 file mysql
```

交叉编译mysql

下载源码

使用之前主机编译的源码，或者重新下载参考主机编译篇。

修改配置

进入Mysql解压的文件夹，修改CMakeLists.txt。直接通过配置参数设置也是一样的，主要是为了设置ncurses、boost、openssl等依赖路径和编译器选择。

- 修改CMakeLists.txt配置文件

```
1 vim CMakeLists.txt
```

```

1 # this is required
2 SET(CMAKE_SYSTEM_NAME Linux)
3 SET(CMAKE_CROSSCOMPILING TRUE)
4
5 # specify the cross compiler
6 SET(CMAKE_C_COMPILER /home/xiang/ti-processor-sdk-linux-rt-am62xx-evm-
08.06.00.42/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/aarch64-none-linux-gnu-
gcc)
7 SET(CMAKE_CXX_COMPILER /home/xiang/ti-processor-sdk-linux-rt-am62xx-evm-
08.06.00.42/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/aarch64-none-linux-gnu-
g++)
8
9 # where is the target environment
10 SET(CMAKE_FIND_ROOT_PATH /home/xiang/ti-processor-sdk-linux-rt-am62xx-evm-
08.06.00.42/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/aarch64-none-linux-gnu)
11
12 # search for programs in the build host directories (not necessary)
13 SET(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)
14 # for libraries and headers in the target directories
15 SET(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
16 SET(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)
17 SET(CMAKE_FIND_ROOT_PATH_MODE_PACKAGE ONLY)
18
19 # configure Boost
20 SET(BOOST_ROOT /home/xiang/Mysql/boost_1_59_0-arm)
21 SET(BOOST_INCLUDE_DIR /home/xiang/Mysql/boost_1_59_0-arm/include)
22 SET(BOOST_LIBRARY_DIR /home/xiang/Mysql/boost_1_59_0-arm/lib)
23
24 # openssl configuration
25 SET(OPENSSL_INCLUDE_DIR /home/xiang/Mysql/openssl-3.2.0-arm/include)
26 SET(OPENSSL_LIBRARY /home/xiang/Mysql/openssl-3.2.0-arm/lib/libssl.so)
27 SET(CRYPTO_LIBRARY /home/xiang/Mysql/openssl-3.2.0-arm/lib/libcrypto.so)
28
29 SET(CMAKE_CXX_LINK_FLAGS "-L/home/xiang/Mysql/openssl-3.2.0-arm/lib -lssl -lcrypto")
30

```

- 设置配置参数


```
1 cmake . -LH -DCMAKE_INSTALL_PREFIX=/usr/xiang/mysql -  
  DMYSQL_DATADIR=/usr/local/arm/mysql/data -DDEFAULT_CHARSET=utf8 -  
  DDEFAULT_COLLATION=utf8_general_ci -DEXTRA_CHARSETS=all -DENABLED_LOCAL_INFILE=1 -  
  DCMKE_CXX_COMPILER=arm-linux-gnueabi-hf-g++ -DCMAKE_C_COMPILER=arm-linux-gnueabi-hf-gcc  
  -DWITH_BOOST=/home/velarn/work/boost/boost_1_59_0 -  
  DCURSES_INCLUDE_PATH=/home/velarn/work/ncurses/build/include/ncurses -  
  DCURSES_LIBRARY=/home/velarn/work/ncurses/build/lib/libncurses.a
```

编译

执行make并根据问题提示和常见问题解决

```
1 make
```

编译完成后执行安装命令，将编译文件安装到对应位置

```
1 make install
```

Mysql API

概述

Mysql使用可以通过sql命令也可以通过Mysql官方提供的Mysql API接口使用，以C语言API为例(其它的不会)介绍API下载和常用的API。在官网就就可以下载对应的系统的API压缩包

API下载

- Linux C API下载

在[官方下载](#)页面中，选择需要的版本和Linux - Generic操作系统，选择对应位数的压缩包。

- Windows C API下载

在[官方下载](#)页面中，选择需要的版本和Microsoft Windows操作系统，选择对应位数的压缩包。

- [其他语言API下载](#)

- 注意 Mysql没有提供arm版本的库，需要自己交叉编译，arm库在交叉编译完成的文件夹lib目录中。

常见API介绍

名称	介绍
my_init()	初始化客户端
mysql_close()	关闭客户端连接
mysql_query()	执行sql语句

常见错误

Mysql安装(Windos 源码)

问题1

问题描述

执行命令:**mysqld -install**, 出现Install of the Service Denied。

解决办法

用管理员权限执行CMD

参考文献

[MySQL 5.7详细下载安装配置教程-CSDN博客](#)

问题2

问题描述

执行命令: **net start mysql**时出现服务启动失败。

解决办法

我的问题是之前安装了Mysql8, 由于Mysql安装时默认端口为3306, Mysql启动时端口被Mysql8占用了, 使用taskkill关闭对应端口。

- 执行命令: **netstat -ano**寻找对应3306端口 (Mysql服务器启动默认端口为3306)

```
1 netstat -ano
```

- 执行命令: **tasklist|findstr 端口号对应的PID号**查询对对应端口程序

```
1 tasklist|findstr 18192
```

- 执行命令：**taskkill /pid 端口号对应的PID号 /f**关闭对应端口

```
1 tasklist|findstr 18192
```

关闭占用端口程序后，重新启动就可以了。

参考文献

最详细的解决方法：[MySQL服务无法启动。服务没有报告任何错误。端口问题，配置问题。_mysql服务无法启动没有任何错误-CSDN博客](#)

问题3

问题描述

cmd提示无法识别mysql命令

解决办法

使用管理员身份打开cmd,前缀为C:\Windos\system32>

参考文献

无

移植Mysql

移植ncurses

问题1

问题描述

找不到编译器

解决办法

由于版本兼容问题引起的缺少环境。

```
1 sudo apt-get install libtool
2 cp /usr/share/libtool/build-aux/config.sub .
3 cp /usr/share/libtool/build-aux/config.guess .
```

参考文献

[configure: error: cannot run /bin/sh ./config.sub解决办法](#)

问题2

问题描述

_32053.c:835:15: error: expected ')' before 'int'

```
1 /* 错误描述 */
2 arm-linux-gnueabi-gcc -DHAVE_CONFIG_H -I../ncurses -I. -
  I/home/charles/code/build_systemd/_install/include -D_GNU_SOURCE -
  D_FILE_OFFSET_BITS=64 -DNDDEBUG -I. -I../include -
  I/home/charles/code/build_systemd/_install/include/ncurses -
  I/home/charles/code/build_systemd/_install/include --param max-inline-insns-
  single=1200 -c ../ncurses/lib_gen.c -o ../objects/lib_gen.o
3 In file included from ../ncurses/curses.priv.h:283:0,
4             from ../ncurses/lib_gen.c:19:
5 _32053.c:835:15: error: expected ')' before 'int'
6 ../include/curses.h:1594:56: note: in definition of macro 'mouse_trafo'
7 #define mouse_trafo(y,x,to_screen) wmouse_trafo(stdscr,y,x,to_screen)
8                                     ^
9 Makefile:785: recipe for target '../objects/lib_gen.o' failed
10
```

解决办法

- 清空配置

```
1 export CPPFLAGS="-P"
2 ./configure --host=aarch64-none-linux-gnu --prefix=/home/xiang/MySQL/ncu --without-cxx-
  binding
3 make
```

- 导出配置

```
1 export CPPFLAGS="-P" //导出配置
2 ./configure --host=aarch64-none-linux-gnu --prefix=/home/xiang/MySQL/ncu --without-cxx-binding
3 make
```

参考文献

[Ncurses 5.9 Compilation Error - error: expected '\)' before 'int'](#)

移植boost

问题1

问题描述

编译器路径错误

解决办法

重新在project-config.jam配置文件中写正确路径和编译器名

参考文献

无

移植openssl

问题1

问题描述

aarch64-linux-gnu-gcc: error: unrecognized command line option '-m64'

解决办法

删除(屏蔽)Makefile中第所有-m64选项,本版本共有两处使用了-m64.分别是3509行和3510行.

参考文献

[openwrt 交叉编译 unrecognized command line option -m64 错误](#)

问题2

问题描述

没有权限创建文件夹

解决办法

- 修改安装目录到当前用户组当前用户目录下
- 使用**sudo**提升权限

```
1 sudo make install
```

参考文献

无

问题3

问题描述

找不到编译器

解决办法

- 改变root环境变量

```
1 su root
2 vim etc/profile    /* 打开 /etc/profile 文件 需要打开一次改变时间戳 */
3 source /etc/profile
4 make install
```

- 修改到当前用户组当前用户目录

```
1 make clean
2 --prefix=xxx/xxx/xxx
3 make
4 make install
```

参考文献

../libtool:line XXXX:arm-linux-ranlib command not found

问题4

问题描述

ossl_prov_digest_reset 未定义

解决办法

无

参考文献

无

编译Mysql(主机Linux)

问题1

问题描述

编译时没有文件读写权限

```
1 /*Cmake 打开文件的读写权限 */
2 CMake Error: Cannot open file for write: /home/xiang/MySql/mysql-5.7.36/CMakeFiles/abi_check.dir/depend.make.tmp
3 CMake Error: : System Error: Permission denied
4 CMakeFiles/abi_check.dir/build.make:74: recipe for target 'CMakeFiles/abi_check.dir/depend' failed
5 make[2]: *** [CMakeFiles/abi_check.dir/depend] Error 2
6 CMakeFiles/Makefile2:99: recipe for target 'CMakeFiles/abi_check.dir/all' failed
7 make[1]: *** [CMakeFiles/abi_check.dir/all] Error 2
8 Makefile:162: recipe for target 'all' failed
9 make: *** [all] Error 2
```

解决办法

安装Mysql时会下载相关依赖,安装时应该使用超级用户权限

参考文献

无

问题2

问题描述

Cmake已存在

解决办法

编译Mysql时会自己生成Cmake且无法自动覆盖,所以每次配置前应该应该删除CMakeCache.txt.

```
1 rm CMakeCache.txt
2 make
```

参考文献

无

编译Mysql(arm)

问题1

问题描述

无法找到SSL库

解决办法

Mysql5.7.36的Cmake匹配openssl版本号规则和openssl3.2的版本规则不一致。Mysql5.7Cmake匹配openssl版本号规则为匹配OPENSSL_VERSION_NUMBER宏而openssl3.2版本宏定义为主版本号宏OPENSSL_VERSION_MAJOR、副版本宏OPENSSL_VERSION_MINOR、补丁宏OPENSSL_VERSION_PATCH。将openssl版本替换为openssl-1.1.1在进行编译。

- openssl-3.2 版本宏定义(opnessl-3.2/include/openssl/opensslv.h)


```
1  /*
2   * Base version macros
3   *
4   * These macros express version number MAJOR.MINOR.PATCH exactly
5   */
6  # define OPENSSL_VERSION_MAJOR  3    //主版本号
7  # define OPENSSL_VERSION_MINOR  2    //副版本号
8  # define OPENSSL_VERSION_PATCH  0    //补丁
9  ... ..
```

- openssl-1.1.1版本宏定义(openssl-1.1.1/include/openssl/opensslv.h)

```

1  /*-
2  * Numeric release version identifier:
3  * MNNFFPPS: major minor fix patch status
4  * The status nibble has one of the values 0 for development, 1 to e for betas
5  * 1 to 14, and f for release. The patch level is exactly that.
6  * For example:
7  * 0.9.3-dev      0x00903000
8  * 0.9.3-beta1    0x00903001
9  * 0.9.3-beta2-dev 0x00903002
10 * 0.9.3-beta2     0x00903002 (same as ...beta2-dev)
11 * 0.9.3           0x0090300f
12 * 0.9.3a          0x0090301f
13 * 0.9.4           0x0090400f
14 * 1.2.3z          0x102031af
15 *
16 * For continuity reasons (because 0.9.5 is already out, and is coded
17 * 0x00905100), between 0.9.5 and 0.9.6 the coding of the patch level
18 * part is slightly different, by setting the highest bit. This means
19 * that 0.9.5a looks like this: 0x0090581f. At 0.9.6, we can start
20 * with 0x0090600S...
21 *
22 * (Prior to 0.9.3-dev a different scheme was used: 0.9.2b is 0x0922.)
23 * (Prior to 0.9.5a beta1, a different scheme was used: MMNNFFRBB for
24 * major minor fix final patch/beta)
25 */
26 # define OPENSSL_VERSION_NUMBER 0x1010100fL
27 # define OPENSSL_VERSION_TEXT   "OpenSSL 1.1.1 11 Sep 2018"
28 ...

```

- Mysql5.7.36匹配openssl版本规则(Mysql3.2/cmake/ssl.cmake)

```

1  IF(OPENSSSL_INCLUDE_DIR)
2  MESSAGE(STATUS "this is ${OPENSSSL_INCLUDE_DIR}")
3      # Verify version number. Version information looks like:
4      #   #define OPENSSSL_VERSION_NUMBER 0x1000103fL
5      # Encoded as MNFFPPS: major minor fix patch status
6  FILE(STRINGS "${OPENSSSL_INCLUDE_DIR}/openssl/opensslv.h"
7      OPENSSSL_VERSION_NUMBER
8      REGEX "^#[ ]*define[\\t ]+OPENSSSL_VERSION_NUMBER[\\t ]+0x[0-9].*"
9      )
10  STRING(REGEX REPLACE
11      "^.*OPENSSSL_VERSION_NUMBER[\\t ]+0x([0-9]).*$" "\\1"
12      OPENSSSL_MAJOR_VERSION "${OPENSSSL_VERSION_NUMBER}")
13  )
14  STRING(REGEX REPLACE
15      "^.*OPENSSSL_VERSION_NUMBER[\\t ]+0x[0-9]([0-9][0-9]).*$" "\\1"
16      OPENSSSL_MINOR_VERSION "${OPENSSSL_VERSION_NUMBER}")
17  )
18  STRING(REGEX REPLACE
19      "^.*OPENSSSL_VERSION_NUMBER[\\t ]+0x[0-9][0-9]([0-9][0-9]).*$" "\\1"
20      OPENSSSL_FIX_VERSION "${OPENSSSL_VERSION_NUMBER}")
21  )
22  ENDIF()

```

参考文献

无

问题2 [11%]

问题描述

无法运行extra/comp_err脚本无法生成comp_err.h文件

解决办法

复制之前编译的x86主机的mysql/extra/comp_err文件到现在的extra目录中。

```
1 cp comp_err ../../mysql-5.7.36-arm/extra/  
2 ls ../../mysql-5.7.36-arm/extra/comp_err
```

参考文献

忘记了 CSDN上某一篇文章

问题3 [11%]

问题描述

无法运行libmysql/libmysql_api_test脚本无法生成libmysql_api_test.h文件

解决办法

复制之前编译的x86主机的libmysql/libmysql_api_test文件到现在的libmysql目录中。

```
1 cp ./libmysql/libmysql_api_test ../mysql-5.7.36-arm/libmysql  
2 ls ../mysql-5.7.36-arm/libmysql/libmysql_api_test
```

参考文献

忘记了 CSDN上某一篇文章

问题4 [41%]

问题描述

无法运行scripts/commp_sql脚本无法生成commp_sql.h文件

解决办法

复制之前编译的x86主机的scripts/commp_sql文件到现在的scripts目录中。

```
1 cp scripts/commp_sql ../mysql-5.7.36-arm/scripts/  
2 ls -c ../mysql-5.7.36-arm/scripts/commp_sql
```

参考文献

忘记了 CSDN上某一篇文章

问题5 [41%]

问题描述

无法运行sql/gen_lex_hash脚本无法生成gen_lex_hash.h文件

解决办法

复制之前编译的x86主机的sql/gen_lex_hash文件到现在的sql目录中。

参考文献

忘记了 CSDN上某一篇文章

问题6 [41%]

问题描述

无法运行sql/gen_lex_token脚本无法生成gen_lex_token.h文件

解决办法

复制之前编译的x86主机的sql/gen_lex_token文件到现在的sql目录中。

```
1 cp ./sql/gen_lex_token ../mysql-5.7.36-arm/sql/  
2 ls -c ../mysql-5.7.36-arm/sql/gen_lex_token
```

参考文献

忘记了 CSDN上某一篇文章

问题7 [41%]

问题描述

arm编译 os0atomic.ic 、os0atomic.h 中没有 HAVE_IB_GCC_ATOMIC_COMPARE_EXCHANGE 与 IB_STRONG_MEMORY_MODEL 这两个宏定义。

解决办法

- 修改os0atomic.h 在#define IB_STRONG_MEMORY_MODEL后添加宏HAVE_ATOMIC_BUILTINS

```
1  ///home/xiang/Mysql/mysql-5.7.36-arm/storage/innobase/include/os0atomic.h
2  #if defined __i386__ || defined __x86_64__ || defined _M_IX86 \
3      || defined _M_X64 || defined __WIN__
4
5  #define IB_STRONG_MEMORY_MODEL
6
7  #else
8
9  #define HAVE_ATOMIC_BUILTINS
10
11 #endif /* __i386__ || __x86_64__ || _M_IX86 || _M_X64 || __WIN__ */
```

- 修改os0atomic.ic 将#elif defined(IB_STRONG_MEMORY_MODEL)改为#elif defined(HAVE_ATOMIC_BUILTINS)

```
1  ///home/xiang/Mysql/mysql-5.7.36-arm/storage/innobase/include/os0atomic.ic
2  #elif defined(HAVE_ATOMIC_BUILTINS)    //198行
```

参考文献

忘记了 某一论坛上的帖子

问题8 [69%]

问题描述

无法运行extra/protobuf/protoc脚本无法生成protoc.h文件

解决办法

复制之前编译的x86主机的extra/protobuf/protoc文件到现在的extra/protobuf目录中。

```
1  cp extra/protobuf/protoc ../mysql-5.7.36-arm/extra/protobuf
2  ls -c ../mysql-5.7.36-arm/extra/protobuf/protoc
```

参考文献

忘记了 CSDN上某一篇文章