使用Autoconf检测MySQL相关的软件包

在你的程序(或者工程)中，如果编译阶段需要检测当前环境中是否存在MySQL客户端相关的库文件时，你可以使用Autoconf来帮你完成这个工作，轻盈、优雅、无痛。阅读本文需要了解简单GNU Autoconf使用。

目录

1. 本文的目标

目的：编译时，根据configure参数(如果有--with-mysql)，选择性编译对应的MySQL相关的功能。

实现：使用已经写好的m4脚本：ax_lib_mysql.m4

2. 如何利用Autoconf实现

大部分你想到的事情都已经有人做过尝试了。这件事情也不例外，Autoconf中有很多脚本和指令帮你做事情。这里，需要使用ax_lib_mysql.m4来帮助我们。先把该文件放到程序/工程目录中，并在configure.ac中新增如下指令来检测MySQL库文件和版本：

```
m4_include(ax_lib_mysql.m4)
AX_LIB_MYSQL()
AM_CONDITIONAL(BUILD_MYSQL_SUPPORT, test x$MYSQL_VERSION != x)
```

说明：AX_LIB_MYSQL()设置了三个变量，可以在Makefile.am中直接使用，分别是MYSQL_CFLAGS、MYSQL_LDFLAGS、MYSQL_VERSION，另外还会在config.h中预定义宏HAVE_MYSQL；
AM_CONDITIONAL(...)则会根据是否需要开启MySQL支持，来设置变量BUILD_MYSQL_SUPPORT，这个变量可以在Makefile.am中使用。

在程序源代码中一般有两种方式可以获取HAVE_MYSQL宏的方式：一个是直接包含config.h；另一个是在你程序的CFLAGS中新增-DHAVE_MYSQL。(注意：有的变量是可以在Makefile.am中使用，有的则是可以在C源代码中使用)

方法一：直接include config.h

Autoconf工具会将所有的预定义宏存放在config.h(默认情况)中，并在编译器选项中新增-DHAVE_CONFIG_H(通过@DEFS@)。因为文件ax_lib_mysql.m4中，包含了如下代码(如果加上--with-mysql并且找到了对应mysql_config，那么如下代码生效)：

```
AC_DEFINE([HAVE_MYSQL], [1],
[Define to 1 if MySQL libraries are available])
```

所以，config.h中会有对应的宏定义：

/* Define to 1 if MySQL libraries are available */
#define HAVE_MYSQL 1

在你的源代码中(一般是头文件)，新增如下代码：

#ifdef HAVE_CONFIG_H
#include >config.h<
#endif

这之后，就可以在你的源代码中，使用#ifdef HAVE_MYSQL ... #endif这样的写法了

方法二：编译器选项新增-DHAVE_MYSQL

因为文件ax_lib_mysql.m4包含了变量定义MYSQL_CFLAGS/MYSQL_LDFLAGS/MYSQL_VERSION，所以，简单的可以在Makefile.am中，直接根据这些变量来新增gcc编译参数。类似如下写法：

if MYSQL_VERSION
XXX_CFLAGES= -DHAVE_MYSQL
endif

这之后，也可以在你的源代码中，使用#ifdef HAVE_MYSQL ... #endif这样的写法了

小结：上面两种方法一个需要修改Makefile.am、一个需要修改头文件，可以根据个人喜好来决定怎么做。

3. 更多关于ax_lib_mysql.m4的使用

常见的configure写法

有了上面的设置，程序就可以通过如下的方式来确定是否将MySQL客户端的支持编译到源代码中：

./configure --with-mysql
...
./configure --with-mysql[=no|yes]
...
./configure --with-mysql[=/YOUR_ENV_PATH/mysql_config] #如果mysql_config不在当前的$PATH中，则需要显示指定。
...

如果对最低版本有要求

另外，如果你对MySQL版本有要求，例如，你希望只有检测到5.5以上的MySQL客户端，才编译对MySQL的支持，则可以在configure.ac中这样使用AX_LIB_MYSQL：

AX_LIB_MYSQL(5.5.18)

修改--with-mysql的默认行为

这里意思是说，如果在configure中没有--with-mysql选项时，则编译时不加上对MySQL的支持(如果写了)，也就是说如下两种写法意思相同：

./configure --with-mysql=no
./configure

ax_lib_mysql.m4的默认行为并非如此，需要对其代码做小小的修改：

```
@@ -61,7 +61,7 @@
            MYSQL_CONFIG="$withval"
```

```
        fi
        ],
-       [want_mysql="yes"]
+       [want_mysql="no"]
     )
     AC_ARG_VAR([MYSQL_CONFIG], [Full path to mysql_config program])
```

这样就如愿了。

4. 更一般的DEBUG选项

其实使用Autoconf这种用法更一般的是开启或者关闭DEBUG选项。这个实现会比上面简单很多。

目标：编译时，根据configure参数(如果有--enable-debug)，则执行程序中#ifdef DEBUG ... #endif。(经常看到这样的写法吧)

相比上面的--with-mysql这个就简单多了(没有版本信息、不需要找mysql_config等)，所以实现也简单多了，只需在你的configure.ac中新增如下代码：

```
AC_ARG_ENABLE(debug,
AS_HELP_STRING([--enable-debug],
               [enable debugging, default: no]),
[case "${enableval}" in
             yes) debug=true ;;
             no)  debug=false ;;
             *)   AC_MSG_ERROR([bad value ${enableval} for --enable-debug]) ;;
esac],
[debug=false])
AM_CONDITIONAL(DEBUG, test x"$debug" = x"true")
```

如果configure时，带有参数--enable-debug，则设置调用AM_CONDITIONAL设置遍历DEBUG。这样就可以在Makefile中根据遍历DEBUG，来选择性的新增编译参数-DDEBUG，所以配套的还需要再Makefile.am中新增：

```
if DEBUG
XXX_CFALGS=... -DDEBUG
else
XXX_CFALGS=...
fi
```

这时，你的代码中就可以写#ifdef DEBUG ... #endif了。

另一种包含config.h的方法跟前面类似，只不过需要将AM_CONDITIONAL那里换成：

```
if test x"$debug" = x"true"
AC_DEFINE([HAVE_MYSQL], [1],
[Define to 1 if MySQL libraries are available])
fi
```

那么程序代码中include >config.h<就可以了。

参考链接

[Creating Your Own Configuration]

[Autotools 实例分析]

[Usage of AM_CONDITIONAL]