qt5.9编码探索

qt的中文乱码发生很平凡，比如，源码格式不是gbk就是utf-8，你的qtCreator设置的编码有可能是utf-8，uft8bom,gb2312。都有可能导致乱码，再加上各种平台，比如win、ubuntu、mac不同平台编码。

乱码产生原因是编码格式，只要编码格式按照qt解析正确了，运行时就保证了不乱码。1是源代码文件编码和qt设置编码，2.QString字符串编码，3。打开文件编码。

我采用的是qt5.9.1，在ubuntu、mac、win下来进行展开编码实验。

<http://doc.qt.io/qt-5/qtcore-index.html>

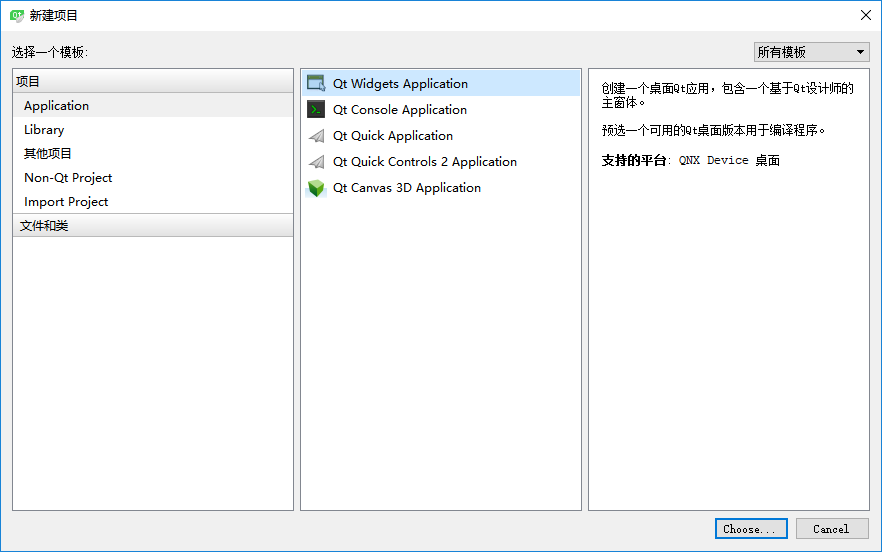
<http://doc.qt.io/qt-5/qtextcodec.html>

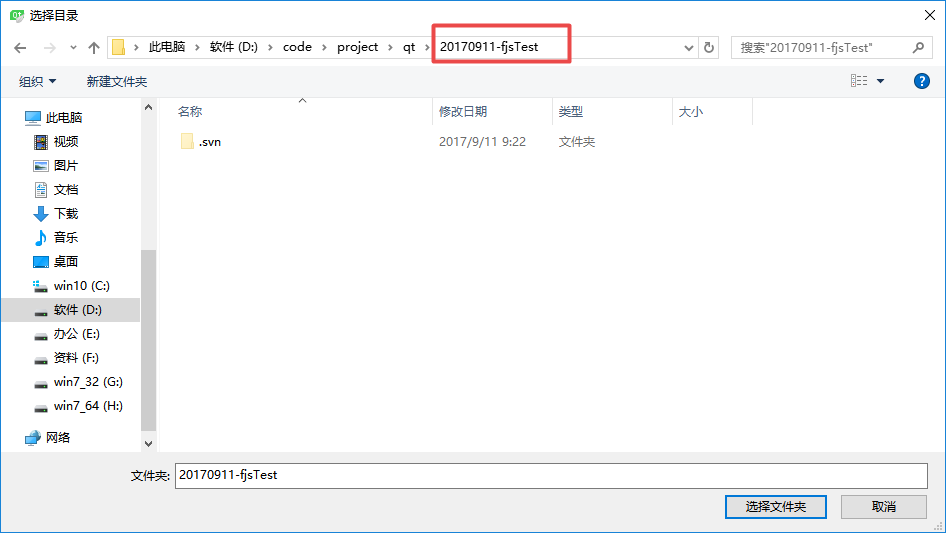
总结如下：

* 源文件（代码文件）编码：设置为utf-8
* qtQreator设置为utf-8
* 在msvc编辑器下：字符串 中文符号：用QObject::tr()
  + 在minGW编辑器中则没有问题，所以为了通用还是包含起来吧。
* 屏蔽C4819错误： warning: C4819: 该文件包含不能在当前代码页(936)中表示的字符。请将该文件保存为 Unicode 格式以防止数据丢失
  + win32-msvc\*:QMAKE\_CXXFLAGS += /wd"4819"

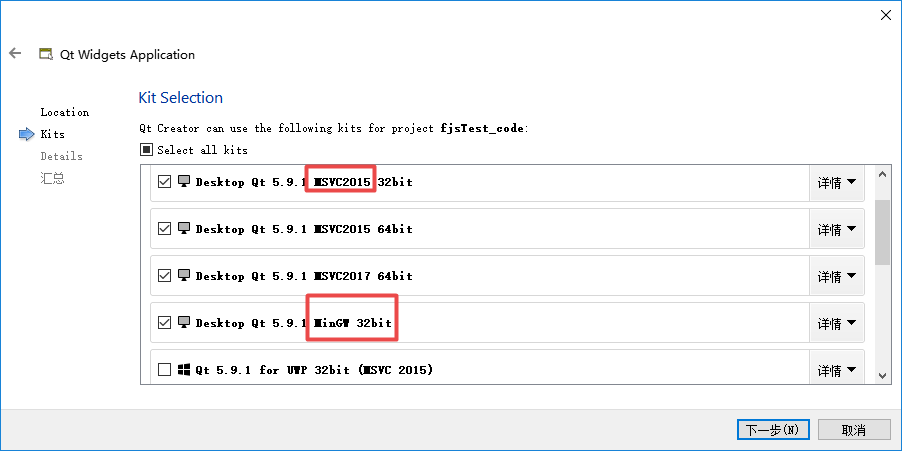
# 1建立项目：

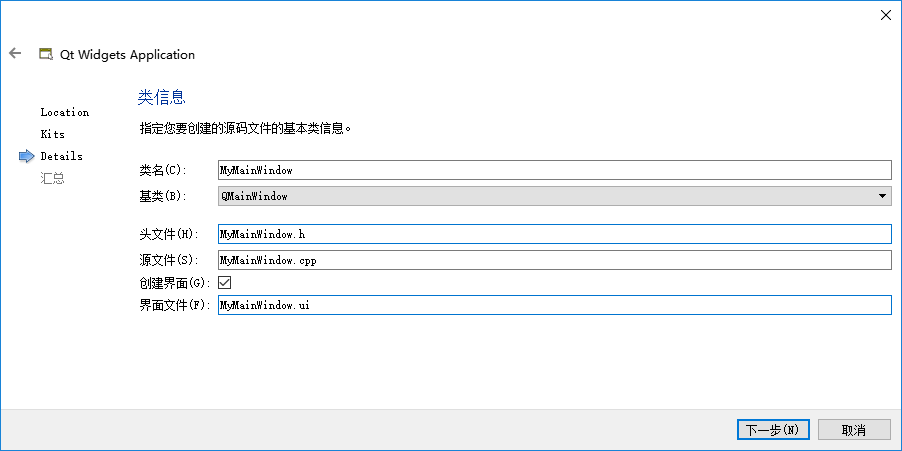
先建立一个qt窗体项目：

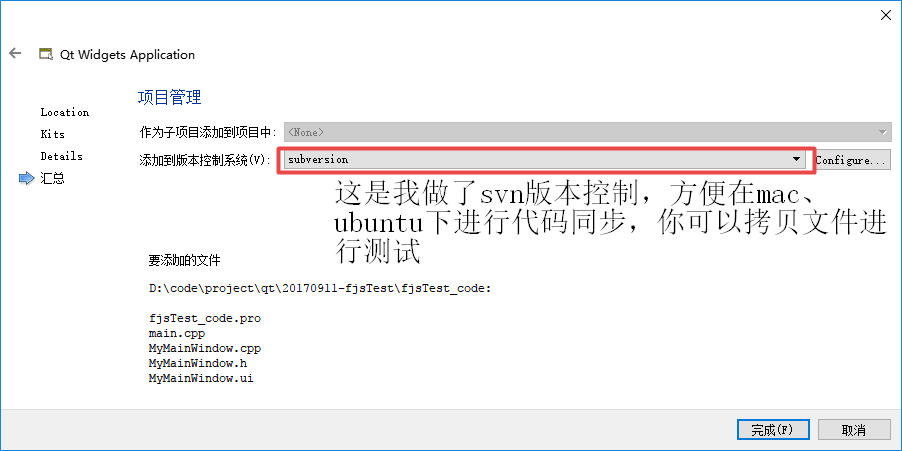






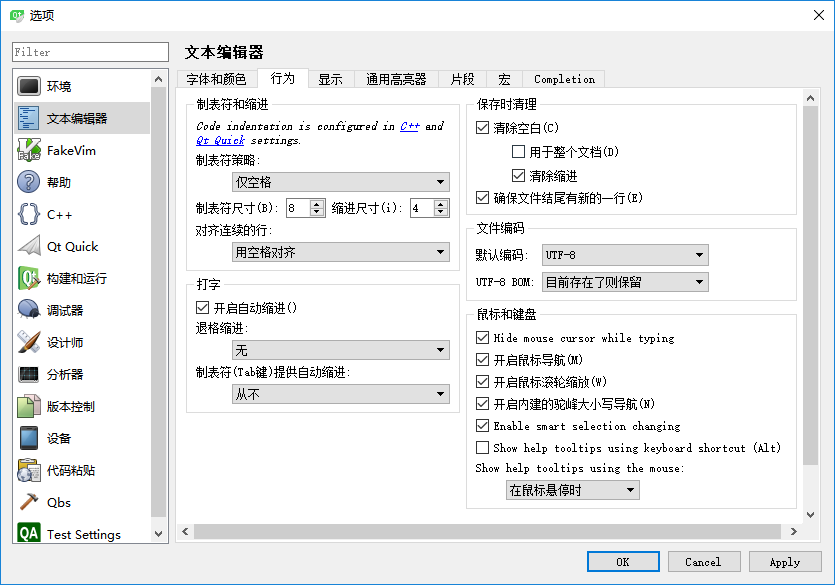






# 2.字符串在各平台测试

设置qt编码格式为UTF-8+BOM的编码方式

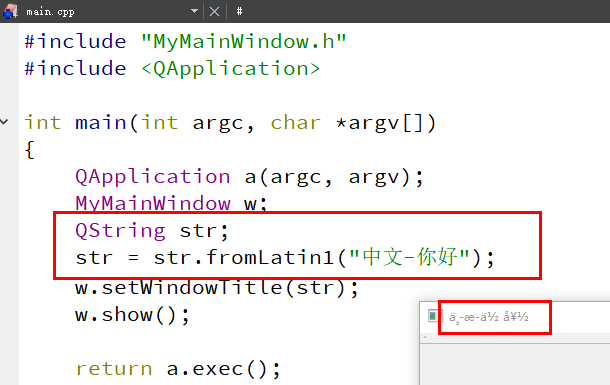


用notepad++打开编码是utf8



QString 内部采用的是Unicode，因为qt5是默认就是utf8。

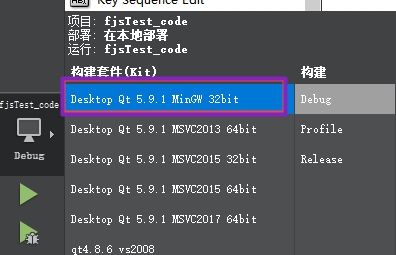
在main.cpp里添加内容



在界面添加一个Text Edit进行测试字符串。

文件编码是utf8的，这样在win、ubuntu、mac下跑是没有问题的。

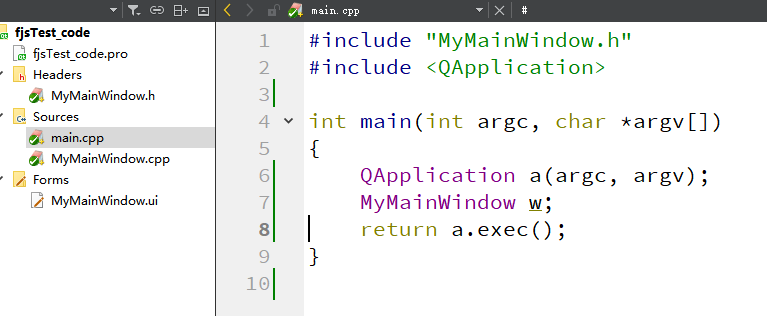
qt是minGW的这样跑起来则是问题不大，但若是qt是msvc+linux的则并不好处理的。

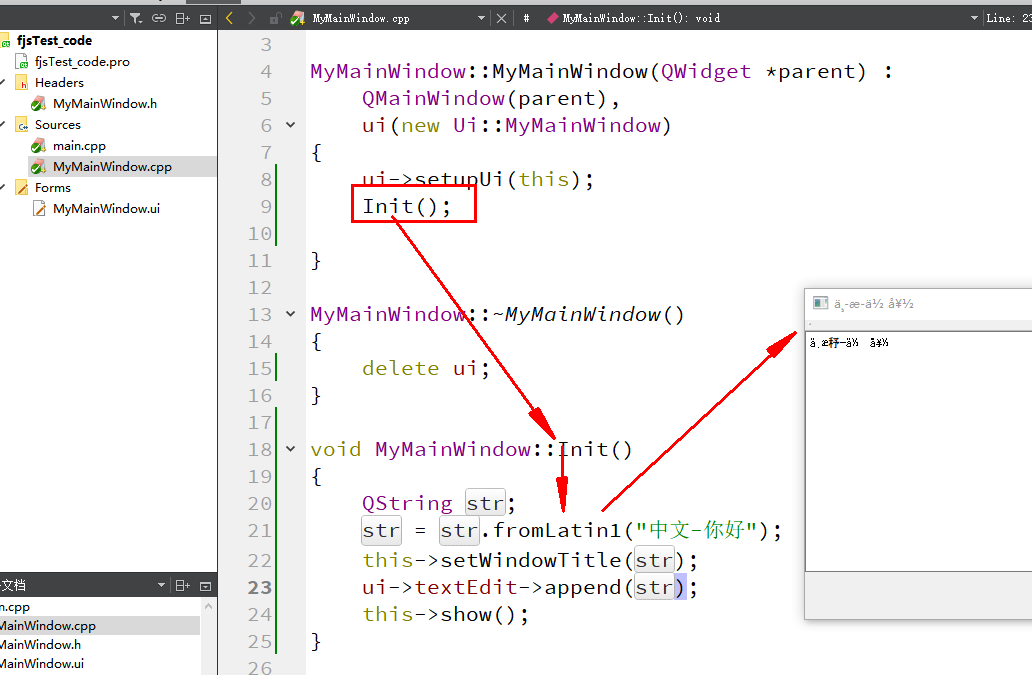


咱们就先按照win:minGW+ubuntu+mac，进行测试：

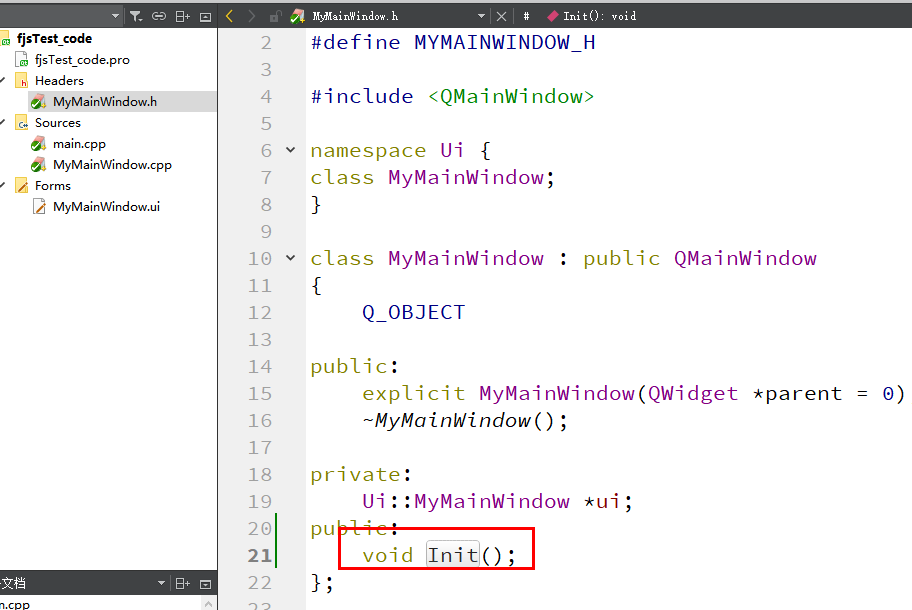
修改源代码：

main.cpp



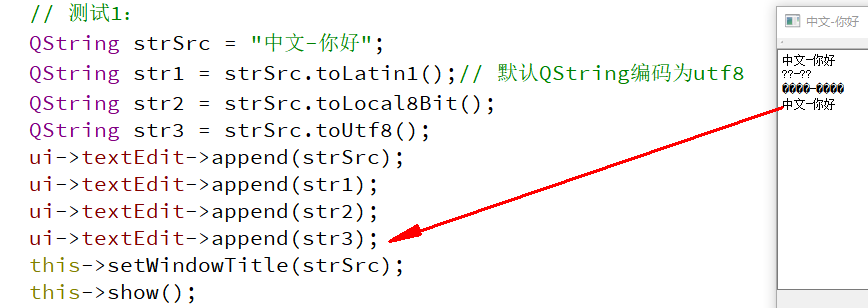


稍微改变了一些，莫要在意这样是不是合理，重点是测试。

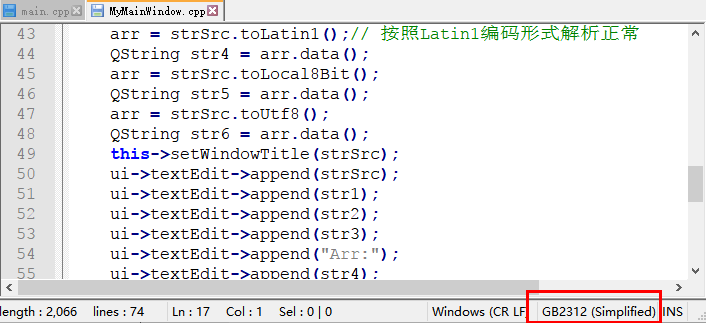


## 测试1：

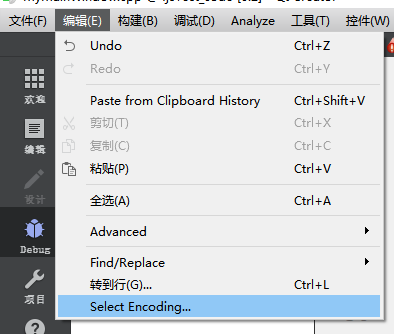
|  |
| --- |
| // 测试1：  QString strSrc = "中文-你好";  QString str1 = strSrc.toLatin1();// 默认QString编码为utf8  QString str2 = strSrc.toLocal8Bit();  QString str3 = strSrc.toUtf8();  ui->textEdit->append(strSrc);  ui->textEdit->append(str1);  ui->textEdit->append(str2);  ui->textEdit->append(str3);  this->setWindowTitle(strSrc);  this->show(); |



把cpp文件改成gb2312

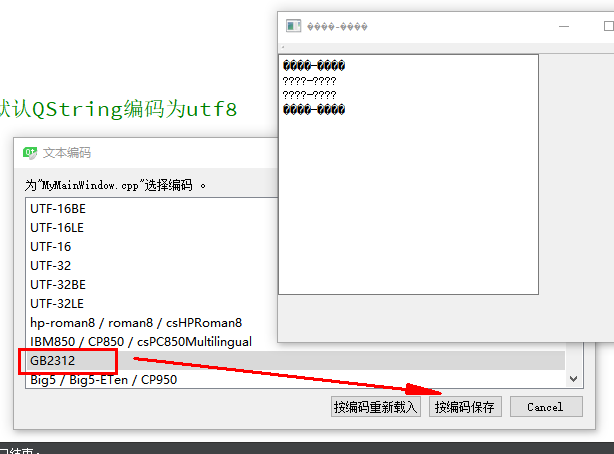


或者qt里面



qtCreator界面上显示的则是“按编码载入”来显示完成的。如果“按编码保存”，则是文件以此编码保存，和notepad++显示编码是一样的。

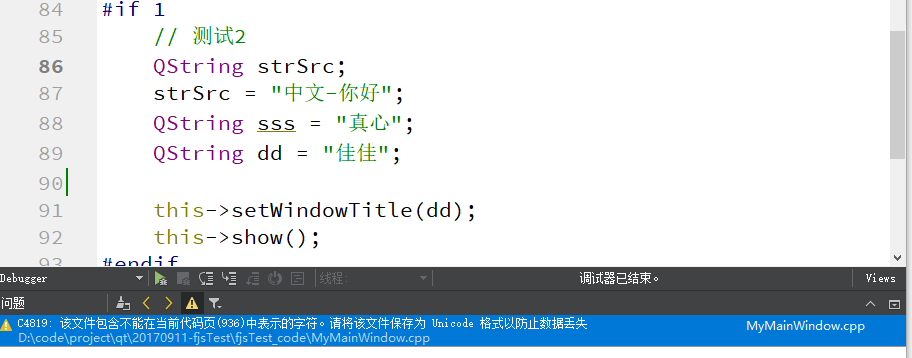
由此见 是乱码。



## 测试2：

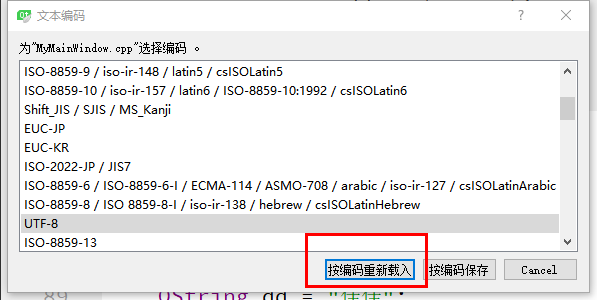
QtCreator的编码依旧是utf-8

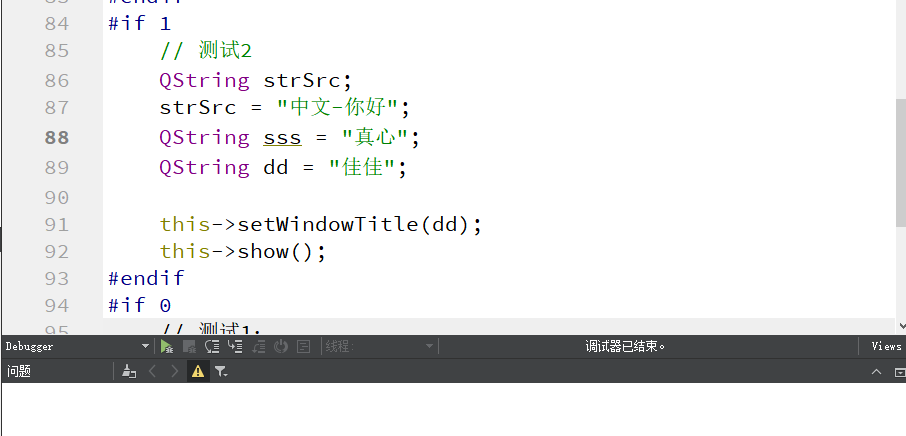
文件编码也依旧是uft8默认设置的。



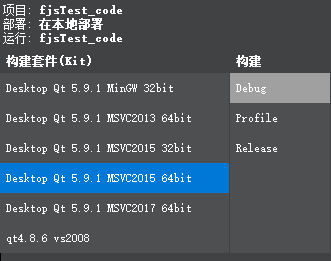
每当添加一行QString ff = "真心"; 一保存就会出现警告

但是如果按照uft8编码载入，再次编译这个警告就会消失，很奇怪的。





这个肯定是msvc编辑器咯，并不是mingW编辑器



一般来说这个警告确实没有什么的，去掉的方法有：

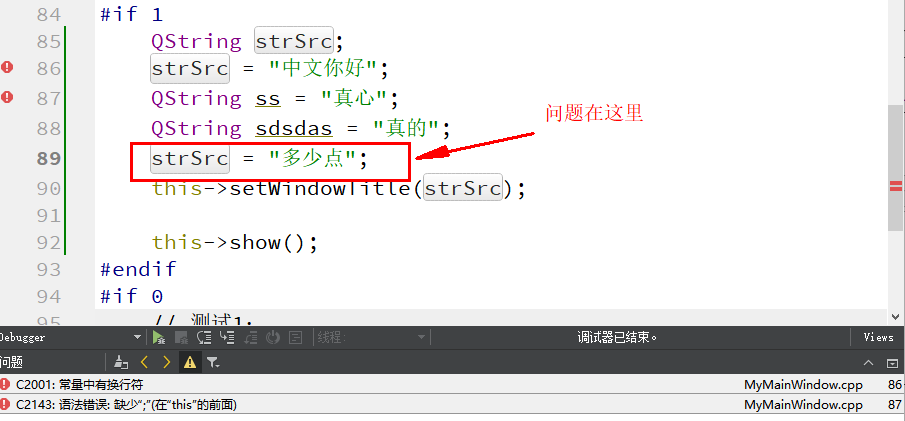
1.转换code文件为其他编码，gb2312或者Unicode，但是不建议这样做哦，因为跨平台编码缘故，导致中文乱码，不同编码会造成QString定义的字符串编码不一样哦。

2.屏蔽错误

在.pro文件中添加下面一行就可以了。

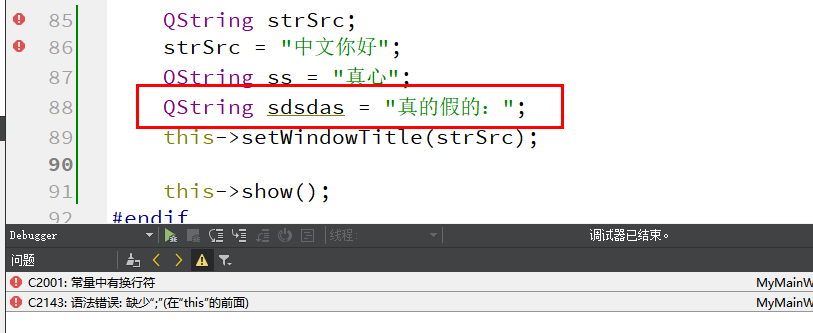
|  |
| --- |
| win32-msvc\*:QMAKE\_CXXFLAGS += /wd"4819" |

换行符错误：

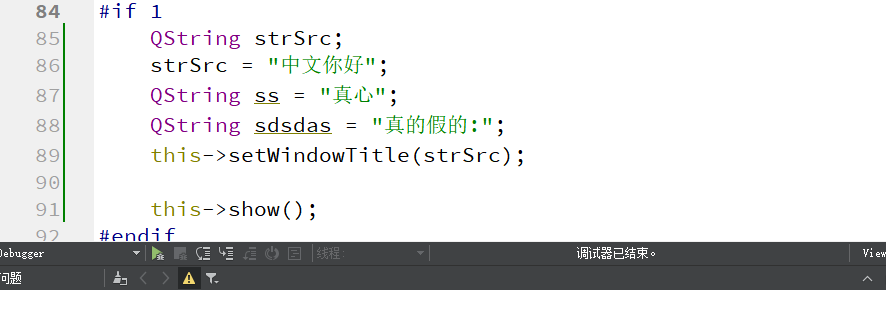




但是删除了，再重新输入，也就正常了，不过输入中文标点符号就出错了



改为英文标点：



解决方案：

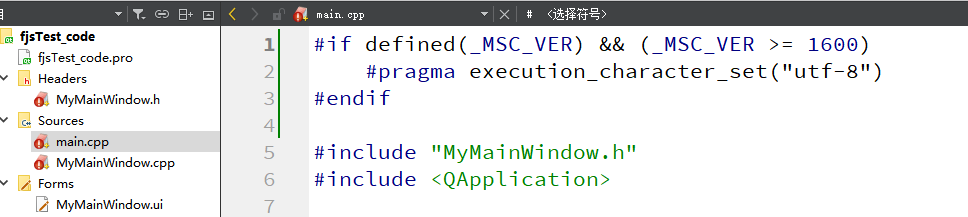
1.去除中文中的标点符号，用英文代替

2.设置编码为utf-8无bom

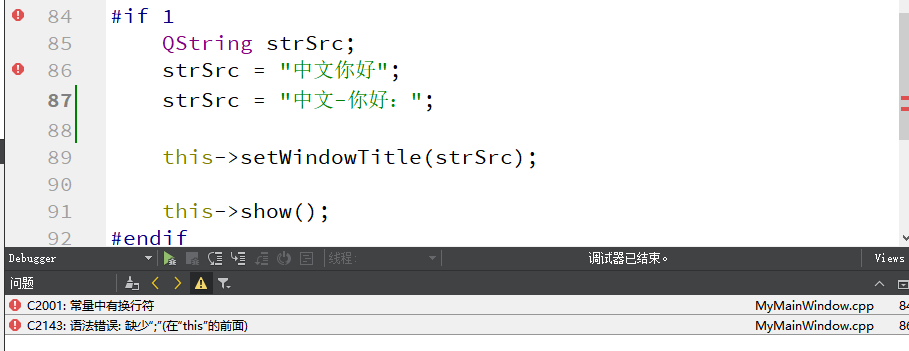
在main.cpp添加

#pragma execution\_character\_set("utf-8")

|  |
| --- |
| #if defined(\_MSC\_VER) && (\_MSC\_VER >= 1600)  #pragma execution\_character\_set("utf-8")  #endif |



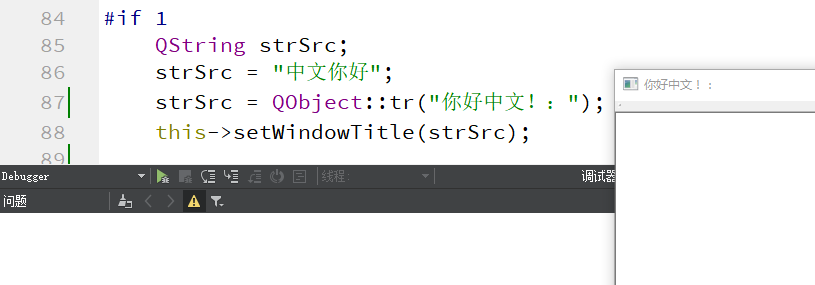
但是编译后发现还是有错误



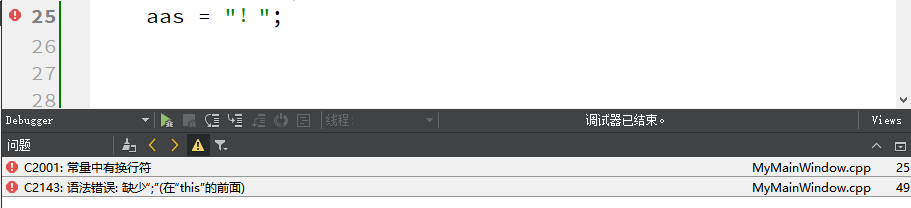
这还是有问题的啊，解决方案如下:

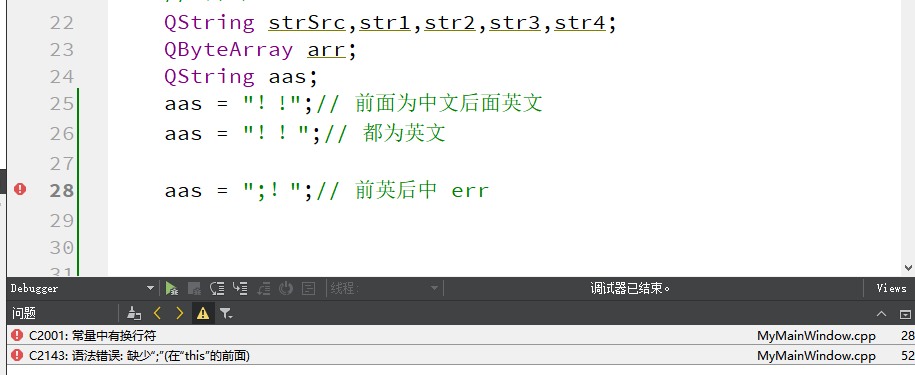
|  |
| --- |
| QObject::tr("你好中文！："); |

这样去解决中文符号问题。已经ok了吧



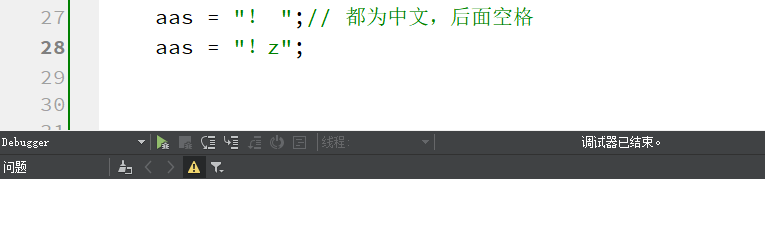
其实咋一看是可以的，其实经不起折腾，其实问题不在这里

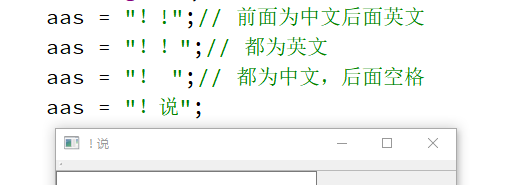


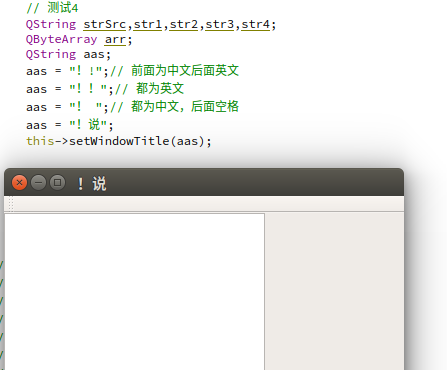


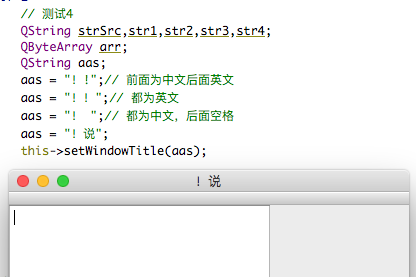
所以：现象是两个符号，前面必须为中文，才ok，空格来解决是可以的。两个中文符号包括汉字，或者前中后英文。

解决方案：便是加空格进行处理。







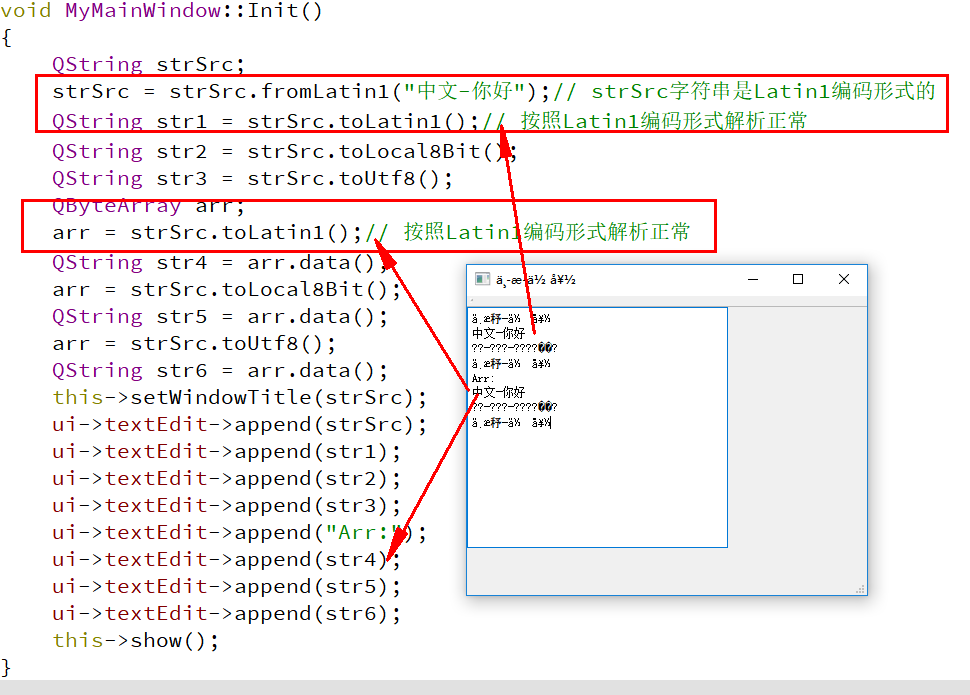


其实mac和ubuntu下用的编码都是utf-8，文件也都是，编辑器是gcc的MinGW。所以差别并不大。

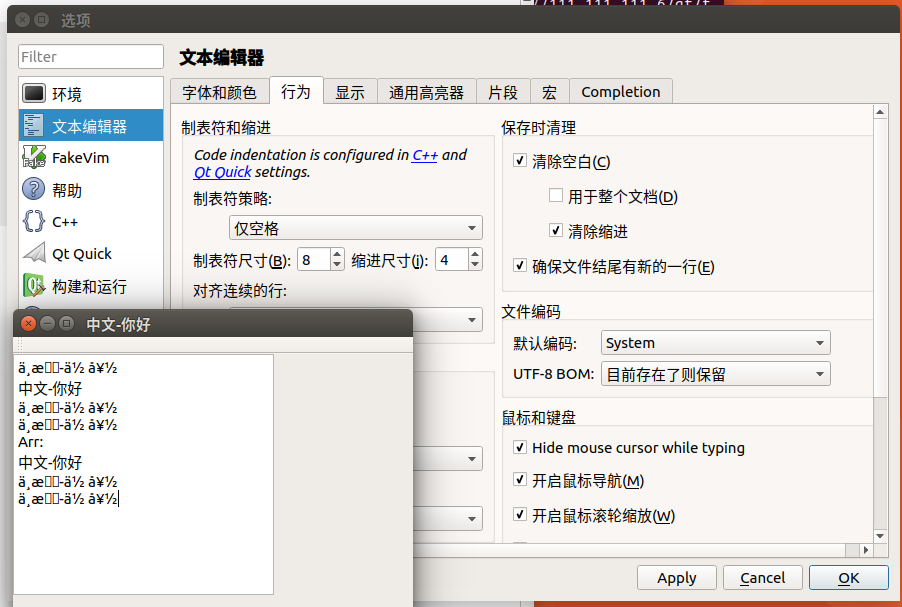
## 测试3：

Latin1转成其他编码

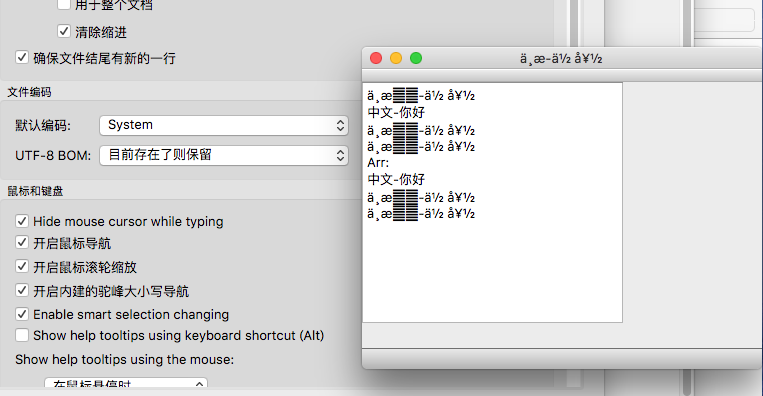
|  |
| --- |
| QString strSrc;  strSrc = strSrc.fromLatin1("中文-你好");// strSrc字符串是Latin1编码形式的  QString str1 = strSrc.toLatin1();// 按照Latin1编码形式解析正常  QString str2 = strSrc.toLocal8Bit();  QString str3 = strSrc.toUtf8();  QByteArray arr;  arr = strSrc.toLatin1();// 按照Latin1编码形式解析正常  QString str4 = arr.data();  arr = strSrc.toLocal8Bit();  QString str5 = arr.data();  arr = strSrc.toUtf8();  QString str6 = arr.data();  this->setWindowTitle(strSrc);  ui->textEdit->append(strSrc);  ui->textEdit->append(str1);  ui->textEdit->append(str2);  ui->textEdit->append(str3);  ui->textEdit->append("Arr:");  ui->textEdit->append(str4);  ui->textEdit->append(str5);  ui->textEdit->append(str6);  this->show(); |



ubuntu下测试



mac:

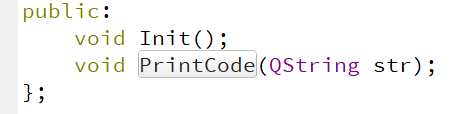


字符串是什么类型的就需要转成什么样式的来解析，

## 测试4：

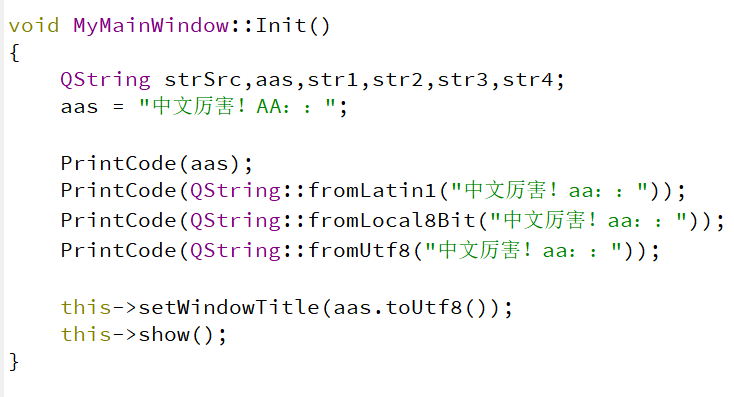
1.源文件编码为：gb2312（ASCII）

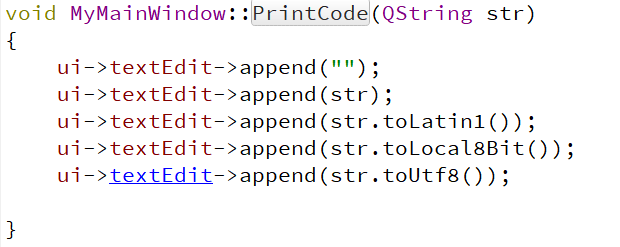
在.h中定义void PrintCode(QString str);



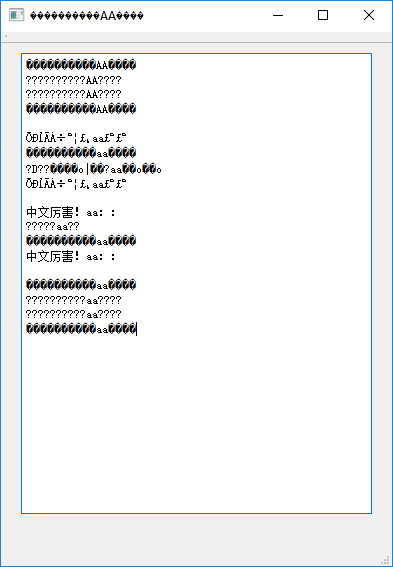
在.cpp中实现

|  |
| --- |
| void MyMainWindow::Init()  {  QString strSrc,aas,str1,str2,str3,str4;  aas = "中文厉害！AA：：";  PrintCode(aas);  PrintCode(QString::fromLatin1("中文厉害！aa：："));  PrintCode(QString::fromLocal8Bit("中文厉害！aa：："));  PrintCode(QString::fromUtf8("中文厉害！aa：："));  this->setWindowTitle(aas.toUtf8());  this->show();  }  void MyMainWindow::PrintCode(QString str)  {  ui->textEdit->append("");  ui->textEdit->append(str);  ui->textEdit->append(str.toLatin1());  ui->textEdit->append(str.toLocal8Bit());  ui->textEdit->append(str.toUtf8());  } |



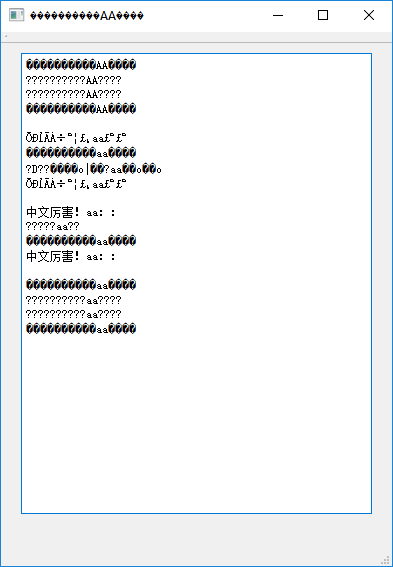


ASCII码的源文件：MSVC编辑器编译



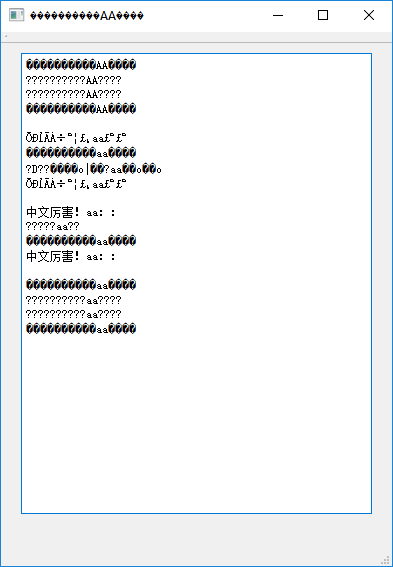


进行MinGW编译

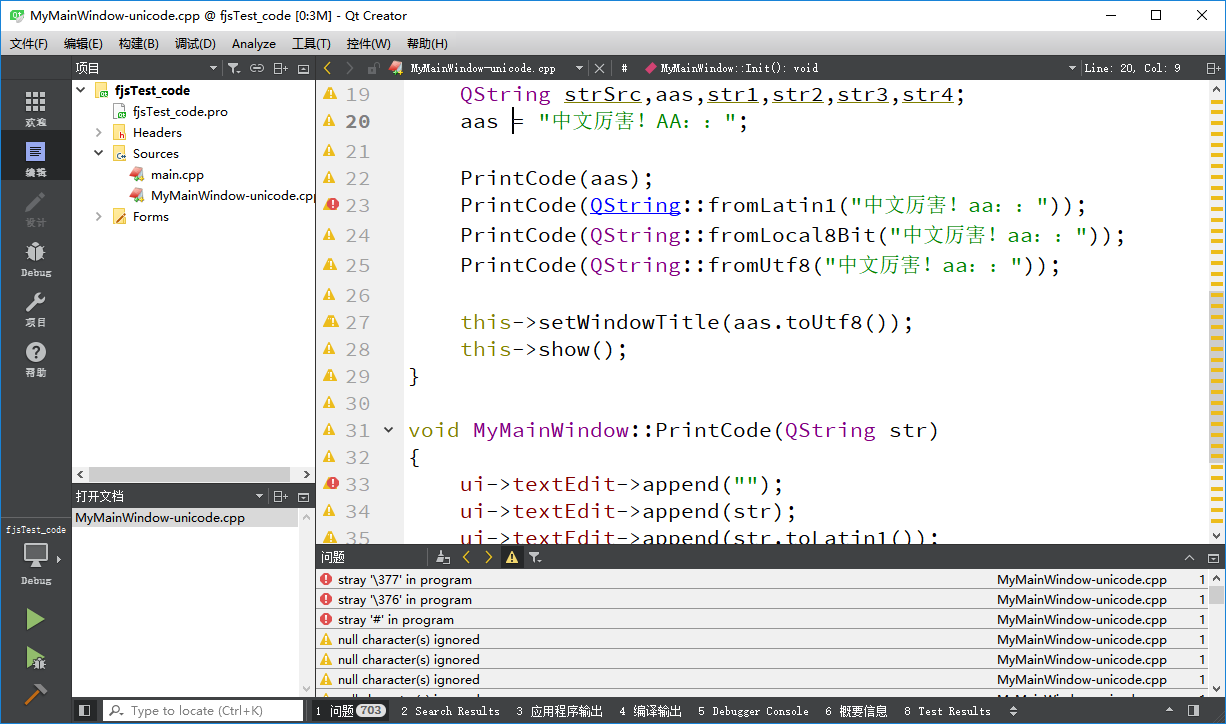


效果是一样的

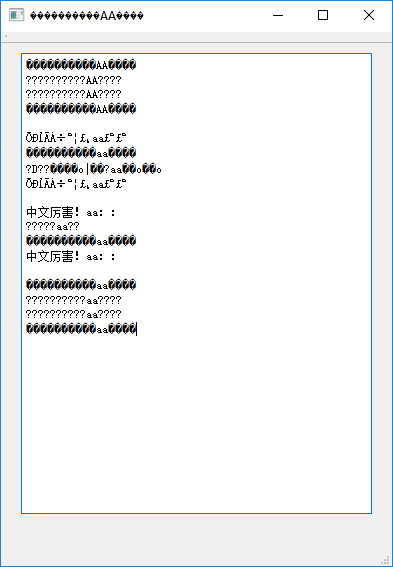
Unicode码的源文件：MSVC编辑器编译



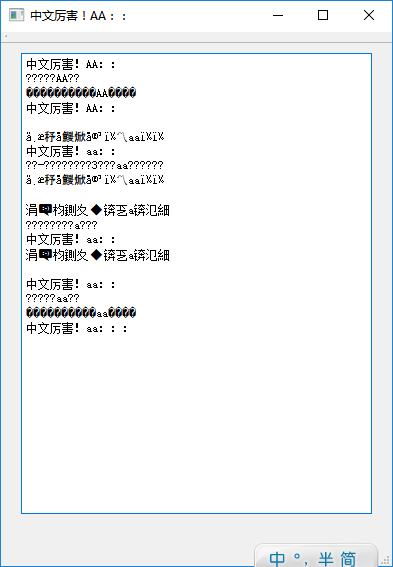
MinGW：



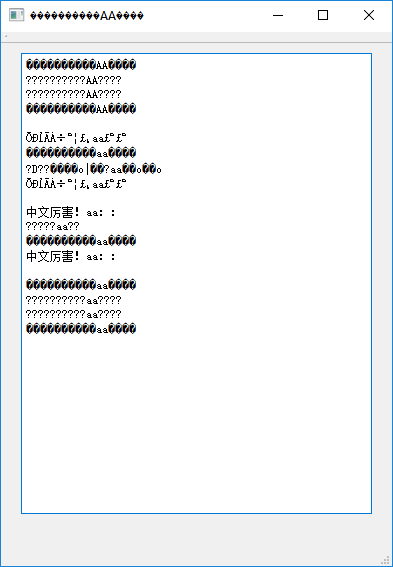
utf8:msvc



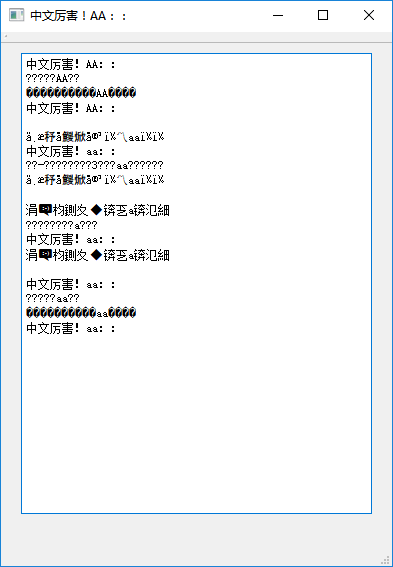
mingw:



utf8+：msvc



mingw



但是对比之后就会发现会有规律，其实网上也有通用的解决方案

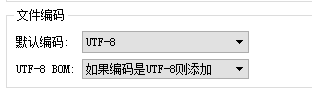
对于此类的问题，就是uft8编码在msvc上的问题咯，若想使用中文，也想跨平台。

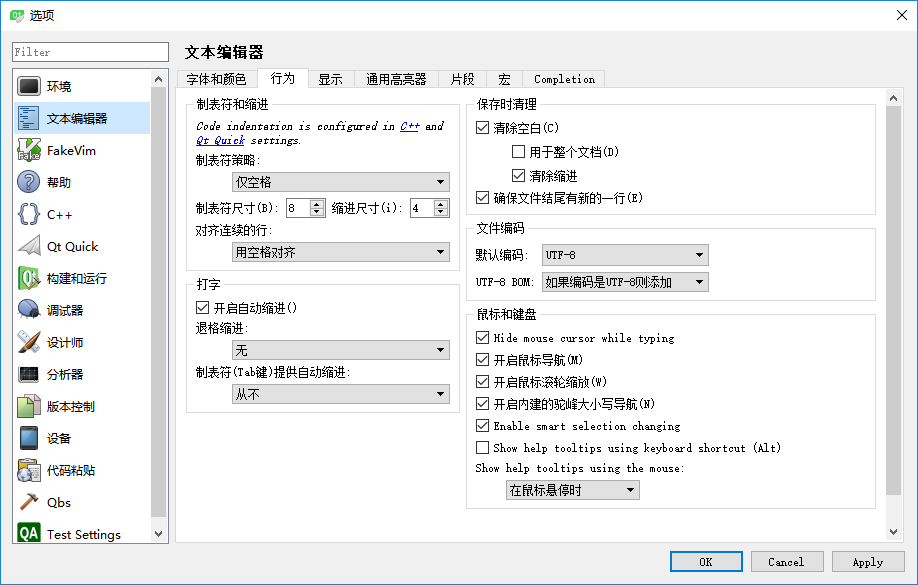
## 解决方案如下：

win环境，Qt Creator+微软VS编译器（msvc）:

* 只在win下，选择gbk编码就行
* win+linux：utf8+bom

1.qtCreator环境设置为utf8+bom



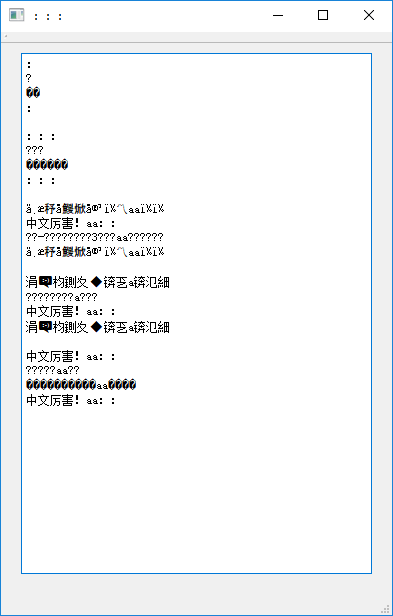


2.所有的文件转成utf8+BOM编码，而不是uft8无bom编码。

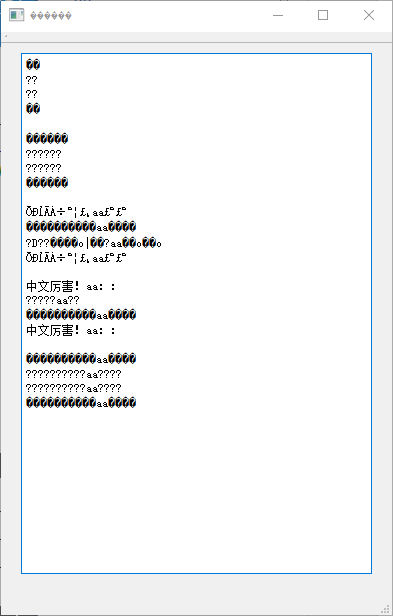
3.在.h（相应.cpp不用添加）头文件中添加

|  |
| --- |
| #if defined(\_MSC\_VER) && (\_MSC\_VER >= 1600)  #pragma execution\_character\_set("utf-8")  #endif  或者  #if \_MSC\_VER >= 1600  #pragma execution\_character\_set("utf-8")  #endif |

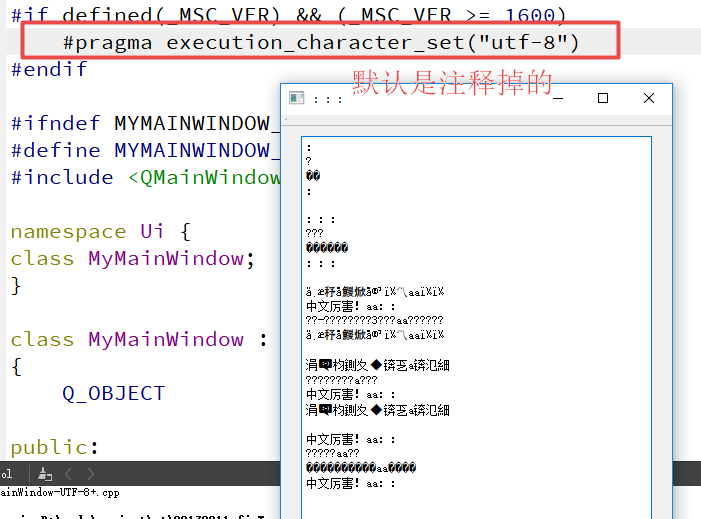
### 测试：在msvc下



注释掉：#pragma execution\_character\_set("utf-8")



在MinGW下，这行#pragma execution\_character\_set("utf-8")是不起作用的。



所以编码问题到此就完成了，不过我采用的较高版本的qt，对于qt4没有进行测试，也没有打算回到过去，所以研究新技术 ，成长吧。

|  |
| --- |
| //qt版本低于qt5  #if QT\_VERSION < QT\_VERSION\_CHECK(5,0,0)  //VS版本低于VS2010  #if defined(\_MSC\_VER) && (\_MSC\_VER < 1600)      QTextCodec::setCodecForTr(QTextCodec::codecForName("GBK"));  #else  QTextCodec::setCodecForTr(QTextCodec::codecForName("UTF-8"));    #endif  #endif  #if \_MSC\_VER >= 1600  #pragma execution\_character\_set("utf-8")  #endif |