# linux编程

我可以选择在linux上直接编程然后生成软件，也可以在windows上采用跨平台编程。显然后一种方式才是不逃避问题的方式，直面代码编译平台与运行平台不同的问题。

这样就需要考虑到一些平台兼容性的代码编写方式。

lib文件是编译过程中生成的中间文件，在Windows下的格式是“xxx.lib”，而我在pro文件中指定lib文件时使用的如下方式：

LIBS    += dian\_matrix\_libd.lib

后来发现，这样的写法是非常不具有跨平台特性的。可以说是硬编码，因为Linux平台下的lib文件是xxx.o的形式，这样看来，直接指定名称及后缀是非常不可取的。

避免方法：使用如下形式：

LIBS    += -ldian\_matrix\_libd

QT是个跨平台框架，自然会提供全套的跨平台服务，-l前缀是QT工程文件中表征lib文件的标记，加上这个前缀，QT工具会自动寻找对应lib文件，完成跨平台的任务。

严重依赖字符类型的的编码方式使得只能使用ascii编码才能正确访问，wstring使用 unicode编码，ubuntu下使用utf-8编码，造成了显示乱码的现象。

解决方案：

使用QT库提供的QTextCodec类完成各类编码的转换工作，使其统一转换成“GB18030”编码方式，就可以正常的访问汉字字模信息了。

* 文件路径问题

在Linux下运行编译结果时，发现找不到HZK16文件，但是明明已经放在目录下了。经分析，最终确定是文件路径的分隔符“/”和“/”的问题。

使用QDir::toNativeSeparators函数可以解决此问题。

注：这个问题是经常出现的，非常让人烦恼，最近的编程中，发现QT含有QDir, QFile, QFileInfo 等多种类支持很多的文件操作，几乎将所有的文件操作（包括文件名、路径、目录的获取）封装起来，非常容易实现跨平台编程。

* 串口端口问题

在Windows下使用的是COM1~COMn的形式，而Linux下使用的ttySn的形式，如何编写跨平台的程序？

目前为止，我还没有找到一个非常好的方式，只是使用了条件编译的方式来限定不同的字符，如下：

#ifdef \_WINDOWS

"COM";

#else

"/dev/ttyS";

#endif

for (int i= 0; i < 4; i++)

{

QString fullName= commonName +

#ifdef \_WINDOWS

QString::number(i+1);

#else

QString::number(i);

#endif

serialDevice->addItem(fullName);

}

这一点我还没懂

* 平台各异代码

#ifdef Q\_OS\_WIN是一个方法

# 内核编译

关于fedora的内核编译

<https://fedoraproject.org/wiki/Building_a_custom_kernel/zh-cn>