<http://blog.51cto.com/devbean/308358>

所谓跨平台，其实有两种含义，一是跨硬件平台，一是跨软件平台。对于硬件平台，很多时候我们都会不自觉的忽略掉，因为硬件差异虽然很大，但是我们能够接触到的却很少。目前 PC 系列基本都是兼容的，并且编译器可能会帮助我们完成这个问题，因此如果你的程序没有用到汇编语言，基本很难考虑到这种跨平台的支持。但是，如果你的程序需要接触到硬件，不管是因为功能的需要还是因为性能的需要，就不得不考虑这个问题。比如，Photoshop 的颜色处理直接使用汇编语言编写，以达到最好性能，这就不得不考虑硬件相关的问题。通常我们说跨平台，更多的只软件的跨平台，也就是跨操作系统。现在主流操作系统 Windows，Linux/Unix，基本算作是两大阵营吧，它们之间的软件都是不兼容的，所以这种跨平台是我们接触比较多的。

说起跨平台，就不得不提 Java。这是 Java 的卖点之一：“一次编写，到处运行”。Java 之所以能够实现跨平台，是因为 Java 源代码编译成一种中间代码，运行 Java 程序，实际上是在 JVM 中。你编写出的 Java 程序是跨平台的，但是 JVM 不是跨平台的，必须根据你的操作系统选择 JVM。这是适配器模式的典型应用 :-)

选择一个本身就是跨平台的库，你的工作量就会小很多，比如 Qt。Qt 已经帮我们封装好很多平台相关的代码。如果你打开一个 Qt 的源代码，就可能找到很多关于操作系统的判断。简单来说，我们使用 QMainWindow::show() 就可以显示一个窗口。在 Windows 上，Qt 必须调用 Win32 API 完成；在 Linux 上，你就可能需要调用 GNOME 或者 KDE 的 API。但是，无论如何，这部分代码都不是我们关心的，因为 Qt 已经替我们完成了。所以，如果你的程序没有与平台相关的代码，那么只需要在 Windows 上编译成功，然后拿到 Linux 上重新编译一下，通过一些简单测试，或者还需要调整一下 UI 比例等等，就可以拿去发行了。

但是，如果有部分代码不得不依赖操作系统，比如我们调用列出目录的命令，Windows 下是 dir，而 Linux 下是 ls，这就不得不根据平台进行编译了。这里我们就拿这个例子尝试一下。

mainwindow.h

1. #ifndef MAINWINDOW\_H
2. #define MAINWINDOW\_H
4. #include <QMainWindow>
6. **class** QProcess;
8. **class** MainWindow : **public** QMainWindow
9. {
10. Q\_OBJECT
12. **public**:
13. MainWindow(QWidget \*parent = 0);
14. ~MainWindow();
16. **private** slots:
17. **void** openProcess();
18. **void** readResult(**int** exitCode);
20. **private**:
21. QProcess \*p;
22. };
24. #endif // MAINWINDOW\_H

mainwindow.cpp

1. #include <QProcess>
2. #include <QPushButton>
3. #include <QMessageBox>
4. #include <QTextCodec>
6. #include "mainwindow.h"
8. MainWindow::MainWindow(QWidget \*parent)
9. : QMainWindow(parent)
10. {
11. p = **new** QProcess(**this**);
12. QPushButton \*bt = **new** QPushButton("display", **this**);
13. connect(bt, SIGNAL(clicked()), **this**, SLOT(openProcess()));
14. }
16. MainWindow::~MainWindow()
17. {
19. }
21. **void** MainWindow::openProcess()
22. {
23. #if defined(Q\_OS\_WIN32)
24. p->start("cmd.exe", QStringList() << "/c" << "dir");
25. #elif defined(Q\_OS\_LINUX)
26. p->start("ls", QStringList() << "/home/usr\_name");
27. #endif
28. connect(p, SIGNAL(finished(**int**)), **this**, SLOT(readResult(**int**)));
29. }
31. **void** MainWindow::readResult(**int** exitCode)
32. {
33. **if**(exitCode == 0) {
34. #if defined(Q\_OS\_WIN32)
35. QTextCodec\* gbkCodec = QTextCodec::codecForName("GBK");
36. QString result = gbkCodec->toUnicode(p->readAll());
37. #elif defined(Q\_OS\_LINUX)
38. QTextCodec\* utfCodec = QTextCodec::codecForName("UTF-8");
39. QString result = utfCodec->toUnicode(p->readAll());
40. #endif
41. QMessageBox::information(**this**, "dir", result);
42. }
43. }

例子和前面是一样的，我们只是在 cpp 文件中添加了一些代码。我们使用 #if 这些预处理指令通过判断 Q\_OS\_WIN32 这样的宏是否存在，来生成相应的平台相关的代码。这些宏是在 Qt 中定义好的，我们只需根据它们是否存在进行判断。这样，我们的程序就可以在 Windows 和 Linux 下都可以编译运行。

进行跨平台编程，你不得不需要接触到一些平台相关的东西，比如文字符编码等，都是需要好哈学习的。所以建议是尽可能使用 Qt 提供的标准函数进行编程，这样就不必写一大堆 if 判断，代码也更加清晰。