第一章 wxWidgets 简介

什么是 wxWidgets？一个简单的回答就是它是一个免费、源代码开放、跨平台的提供本地观感的图形用户界面（GUI）的类库（主要是 C++，同时也有其他各种语言的绑定版本）。

1.1 免费

免费这一点，对于用惯破解软件的我们而言，似乎并不那么令人动心。但当你成为一个公司的职员，开始做项目的时候，你就会发现 License 成为你很头痛的问题，因为公司的行为会受到中国日益完善的打击盗版的法律体系的制约。想想为什么在几乎家家用的都是盗版 Windows 的情况下，微软照样能在中国赚回大把大把的钱？靠的就是政府和公司这些大客户。而且身在其中的你不购买它的授权还不行，不用打官司，飞个律师函过来就够你老板头疼的了。在公司里，你用个 WinRAR 都会招致律师函的……所以，免费绝对是使用 wxWidgets 进行开发的最大好处——更妙的是，虽然免费，可它经过十余年的发展，已经非常成熟，趋于完善，而且还在继续被开源社区的程序员们热火朝天地开发着。

1.2 源代码开放

源代码开放又意味着什么呢？

1.2.1 可以最早尝试到新的特性和得到最快的 Bug 的修复

只需要从 SVN 上检出（check out）最近的源代码，然后编译，就能获得最新版的 wxWidgets。如果你是个高手，还可以修复你发现的 Bug 并提交补丁（Patch）。

1.2.2 读文档不够明白之处，可直接阅读源代码来理解

在国外，你在论坛上提问时，可能会有人叫你 RTFM 或 RTFS，它们是 Read the fantastic manual 和 Read the fantastic source 的缩写，意思是让你去读文档和源代码。这是一种良好的学习习惯，先自学和自行查找（搜 Google 和搜论坛），然后再向其他人求助。

wxWidgets 的文档相当完善，例程也非常丰富，基本能满足使用者的需要。但是对一些细微之处的理解，阅读拥有丰富注释的源代码会比阅读文档中更为抽象的描述要好理解的多。——唯一遗憾的是 wxWidgets 的文档目前没有一个翻译团队对其进行中译，英文不好的同好可能就比较郁闷了。

1.2.3 可供学习的优良编程风格

这里所说的编程风格，包括代码的书写风格，也包括设计模式（Design Patern）。作为开源软件，良好的代码书写风格是起码的要求。而就我目前对 wxWidgets 的构架的了解，它也是学习设计模式的

一个绝好的教材[1]。

1.3 跨平台的提供本地观感的图形用户界面（GUI）类库

1.3.1 跨平台

跨平台，意味着你用 wxWidgets 写的程序，只写一次，就能在各个操作系统上用各编译器编译运行。目前，wxWidgets 支持 Windows（包括最新的 Vista）、苹果的 Mac OS（Carbon 接口和 Cocoa 接口）、Linux（GTK+接口和 X11 接口）、 Unix（Motif 接口）、OS/2 甚至 WinCE 和 PalmOS。当然，这需要你在写程序的时候就对这一点多加注意，采用一些可以适用于各个平台的写法，比如不要指定一个 Button 的绝对大小和绝对位置，而是用 Sizer 进行布局，再比如程序的图标，最好使用 xpm 格式而非 Windows 专有的 ico 格式，等等。不过，如果你只需要在一个特定平台下编程，也可以使用一些平台专有的特性——wxWidgets 也提供了一些类封装了各平台的一些专有特性。

1.3.2 本地观感

同一个用 wxWidgets 写的程序，在 Windows 下会长得跟其他 Windows 程序一样，在苹果下则会长得跟其他苹果程序一样，这是因为在 wxWidgets 内部是用各操作系统本地的 API 实现的，而不是用同一套 wxWidgets 自己的实现（当然，如果你喜欢，也可以使用同一套 wxWidgets 自己的实现，即 wxUniversal）。本地观感是我们应该选择 wxWidgets 而非其它跨平台 GUI 库的最重要的一点。

与本地观感相并列的一点，是本地实现。我们在后面会看到，wxWidgets 不仅仅是 GUI，它在许多功能中都采用相应操作系统的本地实现，这样，你的程序可以更高效，也能和本地的其他程序更协调地工作。

1.3.3 类库的封装

在 Windows 下，有不少人使用 wxWidgets 来作为 MFC[2]的一个更为优雅的替代者。MFC 由于微软的强势，被广泛使用，但任何一个使用过它的人，都会郁闷于它出于历史原因而残留下来的很多丑陋的地方。何况在微软钻进.net 的胡同里之后，它的官方支持已经日渐式微。wxWidgets 的类继承结构与 MFC 很相似，你几乎找不到 MFC 中有而 wxWidgets 中没有的东西（想想你同时在多少个平台上获得了同样的东西），而且你会看见它们被以更优雅的形式封装起来，并使用了清晰整齐美观的命名（类名通常是加上 wx 前缀的完整英文单词，接口函数名也通常是完整的英文谓宾结构，每个单词的首字母都大写，每个参数的类型和名字也都有很强的指示作用——哪像 MFC，动辄 LPLVHITTESTINFO 这样的参数类型，谁看着能舒服？用 wxWidgets 写的程序，不用注释都能看懂大半。）更加惊喜的是，有太多在你用 MFC 编程时需要求助于第三方类库的功能，内建在 wxWidgets 里，并被封装得很顺手。

wxWidgets 使用 C++写成。我个人非常热爱 C++，虽然现在 Java 很火，但我不喜欢它的许多莫名其妙的语法和功能简化带来的对编程的限制，不喜欢它强迫我们写被托管的代码，不喜欢自己的程序被放进一个砂箱里（只要一个功能 Java 虚拟机或 Java API 不支持，我们就永远做不了），不喜欢它表象混乱的指针模型和事实上也相当混乱的类库结构。Java 对 C++的许多“改进”，不应当由语言来做，

---

[1] 最近看了阎宏编著的《Java 与模式》，突然想写本《wxWidgets 与模式》，可惜，时间与精力……

[2] Microsoft Foundation Class，微软基础类库

而应该靠程序员自律（良好的编程风格）、编译器特别支持（如垃圾回收）和类库（如智能指针）来完成。

C++的发展历史中，相当遗憾的一笔，是它没有及时发展出自己的类库，标准库迟迟才建起，而跨平台 GUI 类库这一块，更是被 Java 先声夺人了。C++有了 wxWidgets，是真正的如虎添翼，不仅不再在这方面被 Java 压一头，而且反过来彰显了 C++的优越性——本地观感与本地实现。

然而 wxWidgets 不只是 C++语言的类库，它还有多种语言的绑定版本，包括 Perl、Python、Ruby、Lua、Basic、C#、Euphoria 甚至 JavaScript，真是琳琅满目。

1.3.4 不仅仅是 GUI

作为 GUI 类库，wxWidgets 提供了

- 你所能想到的各类窗体（Window、Frame 或 Panel，包括 MDI[3]，甚至启动画面、每日提示、向导等）
- 各类对话框、各类控件及相关的数据交换与验证
- 可浮动可停靠的窗体、可由用户定义的布局
- 菜单、工具栏、状态栏以及对系统托盘栏的访问
- 各种 GDI[4]类和 DC[5]类，乃至于对 OpenGL 的支持
- 用 Sizer 进行的布局框架
- 良好设计的事件处理（静态事件表和动态事件挂钩）、快捷键设置
- 剪贴板和拖放（Drag&Drop）操作支持
- 文档/视图框架
- 在线帮助框架
- 完整的打印框架，包括打印预览、页面设置以及对 PostScript 的支持
- ……

这些我不打算在这里详细介绍，你可以在官方文档[6]、Julian Smart 等著的《wxWidgets 跨平台 GUI 开发》[7]或后续的教程中找到它们的详细介绍。

然而，wxWidgets 库远远超出一个单纯的 GUI 类库，而是一个多才多艺的类库，提供了：

- 完善的 Unicode 支持、各种编码的相互转换
- 对编写国际化（多语言）程序的支持
- 各类数据结构，包括 GUI 需要用到的数据结构（如点、一个不规则区域、光标）、实用的数据结构（如日期时间、64 位整数、可变类型）、基本的容器（如链表、Hash 族容器）、支持 Unicode 的字符串等。

---

[3] Multiple Document Interface，多文档界面

[4] Graphics Device Interface，图形设备接口，例如笔刷、颜色、字体

[5] Device Context，设备上下文，你可以把它想象为一块画布

[6] http://www.wxwidgets.org/manuals/stable/

[7] Wesley 大侠翻译的中文版可在网上搜索到

- 多进程、多线程程序开发支持
- 对载入动态库（Window 下是动态链接库 dll、*nix 平台则是共享对象库 so）的支持
- 完整的 Socket 网络编程支持，包括 IP、TCP、UDP、HTTP 和 FTP 等。
- 对 Windows 的 DDE[8]、ActiveX（OLE）的支持
- ODBC 数据库访问支持，SQL 支持。

- 文件、文件夹、路径相关的各种操作
- 虚拟文件系统，可用于访问压缩文档、HTTP 或 FTP 等因特网上的文件以及把内存当作一个虚拟文件来访问。
- 各类流的支持

- 对 zip 和 tar 压缩文档的读写支持
- 对 bmp、jpg、gif、png 等图片格式的读写支持，乃至可以播放 gif 和 ani 动画
- 以本地多媒体支持为后端的多媒体播放功能

- 富文本编辑控件
- 可进行多种编程语言语法高亮、代码折叠等功能的代码编辑控件
- XRC（XML-based resource，基于 XML 的资源文件）系统，使用它可将 GUI 设计存储进文本文件，并在运行时读取，从而实现用户界面与程序功能得更好的分离。

- 正则表达式解析、命令行参数解析（Parser）、字符串标记化（Tokenizer）
- 一整套 HTML 的解析、显示、打印支持
- XML 解析支持

- wxWidgets 自己的运行时类型信息机制（RTTI）
- 各式智能指针
- 内建的内存管理、检测系统、大量的调试（debug）宏支持，甚至包括可在程序崩溃时输出可视化调试报告的控件

- 一整套日志记录功能（包括可视化的和基于文件或标准流的）
- 对将用户设置保存为配置文件（Windows 下可用注册表）的支持
- ……

[8] Dynamic Data Exchange，动态数据交换

第二章 轻装上阵跨平台

本章的目的是讲述，如何实现在不依赖任何 IDE[9]的情况下，使用 gcc 编译器在 Windows、Linux、Mac OS 三大平台上编译 wxWidgets、wxWidgets 的例子，以及利用 wxWidgets 提供的 Bakefile 支持来编译自己的 wxWidgets 程序。本章还涉及到了一些上下游的细节，一些初次尝试者会卡壳的地方。

2.1 为什么要自己编译 wxWidgets

为什么要自己编译 wxWidgets 这么麻烦？wxWidgets 为什么不提供一个只需要点“下一步”的安装程序？为什么不提供一个 IDE 直接就支持 wxWidgets，像安装完了 VC++直接就能在里面写 wxWidgets 程序，就能运行，效果直接?

关于第一个和第二个问题，一个很大的原因是 wxWidgets 是跨平台的，在它作为一个类库发布的时候，你说它提供哪些平台上的安装程序好呢？事实上，wxWidgets 主要是以源代码的形式发布的，伴随它的，还有各主要平台上的 Makefile，甚至一些主要 IDE 的工程文件（比如 VC 的工程，从 6.0 到 2005 都有支持），这已经为我们编译它提供了相当的便利了。

倒也有人在做 wxWidgets 的打包工作，就我注意到的而言，其中一个是 wxPack[10]，另外一个是 wxWidgets installation wizard[11]，而且现在官网上也有了 Windows 下的 Installer。我没有选择它们的原因是：我们不应当等待和依赖他人，而应该“自己动手，丰衣足食”，这是一个人起码的志气。而且，今天我们学会了（并习惯了）编译 wxWidgets 库，明天就能够编译 Boost 库、Linux 内核等等等等，只要有源码，我们就能编出最新的库，编出最新的软件，还可以边改边编，这就是 DIY 的自在与乐趣。

话虽这么说，初次尝试时因为缺少指引而频频遭遇莫名其妙的错误实在让人气馁，我也曾经历过这个阶段，所幸坚持下来了，没有放弃对 wxWidgets 的学习，这也是为什么我希望能够为初学者们写一份清晰的指南。

至于第三个问题，就 Windows 下而言，wxDev-C++是一个不错的选择，它自带 MinGW 编译器，也帮你把 wxWidgets 编译好了，甚至在你新建一个 wxWidgets 工程时，帮你写好了该链接的库。你只需要看着 wxWidgets 的教程，开始编程，并点菜单编译，一切都为你打点好了。但我个人并不推荐 wxDev-C++作为 wxWidgets 深入开发的 IDE——它是用 Delphi 写的，只能在 Windows 下运行，即它不是跨平台的，而且它的 RAD 部份也过于像 VB 的思维，而不是更利于跨平台开发的 Sizer 思维，甚至它可能娇惯你写出不能跨平台的代码，还有，它没有提供太多我需要 IDE 提供的功能。下一章我将

---

[9] Integrated Development Evironment，综合开发环境，例如 Visual C++

[10] http://wxpack.sourceforge.net/Main/HomePage

[11] http://www.upcase.de/wxinstaller.html，参见 http://wxforum.shadonet.com/viewtopic.php?t=4755

介绍我推荐的 IDE 和 RAD 工具——Code::Blocks 和 wxFormBuilder，它们都是用 wxWidgets 写成，好用、开源、跨平台、免费。

2.2 获得 wxWidgets

wxWidgets 是以源代码的形式发布的。通常我们使用它的稳定版。本文写作的时候的稳定版是 2.8.7，而 2.9.0 即将推出。如果想试用最新的功能，又不怕它的 Bug，可以玩玩最新的 wxWidgets。

2.2.1 下载最近的稳定版 wxWidgets 源代码

http://www.wxwidgets.org/downloads/是 wxWidgets 的下载页面。看到 Current Stable Release 一节，里面有三个小节，第一个小节 Source Archives 中的 wxAll 就是我们要下载的，里面包含了所有平台的源代码，还可以选择不同的压缩格式，Windows 下常见的 zip 格式，Linux 下常见的 tar.gz。下载下来后，解压到某个地方，比如我在 Windows 下就是 C:\，解压出来会有一个 wxWidgets-版本号的文件夹。

2.2.2 获得最新的 wxWidgets

★ 下载 wxWidgets 的 Daily Snapshot

要获得最新的 wxWidgets，一个简单点的办法是下载 wxWidgets 的 Daily Snapshot，它是对每天的 SVN 版本的 wxWidgets 的一个打包。在 http://www.wxwidgets.org/downloads/最下面有一个 Daily Snapshot 的链接，它会链接到一个文件列表，里面的文件的最后修改日期都是今天。可以下载里面的 wxWidgets.zip 或 wxWidgets.tar.gz，相当于最新的 wxAll。而 wxWidgets-snapshot.tar.gz 这个大家伙，则还包含 wxWidgets 网站内容、开发者用到的一些脚本，甚至 SVN 的版本信息——这样一来，你可以在下面这种方法中，省去检出（check out）的一步，直接 svn update 就可以了。

★ 从 SVN trunk 中检出（checkout）最新的 wxWidgets

上面这种方法中，如果下载的是 wxWidgets.zip 或 wxWidgets.tar.gz，虽然第一次简单，可每次都要重新下载，相当麻烦。另外一个第一次复杂些的办法，是用 Subversion 从 wxWidgets 的开发砂箱（trunk）中检出（check out）wxWidgets，以后每次只需要更新那些又被开发团队修改过的文件，这就方便多了。

先要下载 Subversion，它是一个版本控制工具。

可在 http://subversion.tigris.org/project_packages.html#binary-packages 下载到它在各平台上的可执行文件。其中 Ubuntu 只需要在终端键入命令

```
apt-get install subversion
```

Fedora 只需要在终端键入命令：

```
yum install subversion
```

安装之后，可在命令行中输入 svn 并回车，如果出来信息叫你输 svn help，就说明你安装成功了。这时，可将目录更换到一个你打算放 wxWidgets 的地方（比如我在 Windows 下就是 D:\SVN），然后输入命令

```
svn checkout http://svn.wxwidgets.org/svn/wx/wxWidgets/trunk wxWidgets
```

就可以了。

下一次你想要更新的 wxWidgets，只需要进入 wxWidgets 所在目录（对于我是 D:\SVN\wxWidgets），并键入命令：

```
svn update
```

更多的信息，可参见 http://www.wxwidgets.org/develop/svn.htm。

2.3 Windows (XP/Vista)

2.3.1 安装 MinGW 环境

★ 什么是 MinGW

MinGW 是一个编译工具集，对于第一次听说它的人来说，这个名字可比较吓人，究竟什么意思啊？其实，它是 **Min**imal **G**NU[12] for **W**indow 的缩写，可以理解为在 Windows 下用 GNU 自由软件最小集合，我们在这里只使用它的 Gcc 部份。Gcc 全称是 GNU Compiler Collection，它其实不只是 C/C++编译器，还有 Objective-C，Ada，Java 等语言的编译器，更重要的是它构建了一整套程序的“生产流水线”，由各个相对小的程序分工协作，完成程序的最终生成过程。VC 在底层所用的工具集，也具有相类似的分工，在 MinGW 里学到的方法，很容易移植到 VC 的工具集上。

为什么选择 MinGW 而不是 VC 的编译器？主要是觉得从在 Windows 下就开始用 gcc 编译，可以更大程度地保证程序可以移植到 Linux 和 Mac 下。另外，众所周知，仍被广泛使用的 VC6.0 的编译器很不支持 C++标准。不过，VC2005/2008 的 Express 版是免费下载的，对标准的支持也比较好，如果要选用 VC 的编译器，选用它们比较好，它们在编译速度和生成的可执行文件大小方面也较 MinGW 有一定优势。

又为什么不将之与 MSYS 合在一起使用 configure？主要是觉得在 Windows 下 Make 就够用了，configure 就太 Linux 化了，而且纯属浪费。而且从 GNU make 中学到的，可以运用到 VC 的 nmake 上。

★ 下载和安装 MinGW

在 http://www.mingw.org/点左边的 Download，出现一个好长的英文文章，再在 Contents 中点 Downloads，就到了一段非常短的英文，里面有 1 个链接 Sourceforge File Release，点它就到了 MinGW 在 Sourceforge 上的下载页面。如果你喜欢开源软件，以后少不了跟 Sourceforge 打交道，熟悉一下它的界面吧。

上面好长的一个文件列表，下哪个呢？一个最为简单的方法是下载个向导，它帮你下你需要的，这个向导是 Automated MinGW Installer，在第一行。我们这里不采用这个方法，这一方面因为我个人用它时下得好慢，另外一方面是我喜欢自己动手。

我们主要要下载的东西的关键字有：

---

[12] GNU 工程创始于一九八四年，旨在开发一个类似 Unix ，且为自由软件的完整的操作系统：GNU 系统。GNU 的内核尚未完成，所以 GNU 使用 Linux 作为其内核。GNU 和 Linux 以这样的方式组合成为 GNU/Linux 操作系统，目前有数百万用户。GNU 是 Linux 世界大多数基本自由软件的来源。

1. gcc-core　　GCC 的核心
2. gcc-g++　　GNU C++编译器
3. gdb　　GNU 调试器
4. mingw32-make　　GNU Make，我们需要它来自动化编译多文件的程序
5. mingw-runtime　　MinGW 运行时环境，包括 C 头文件、一些静态库、平台专有起始代码等。
6. binutils　　编译、链接等必需的一些工具。
7. w32api　　Windows API 的头文件和静态库,没有它们根本无法编译出 GUI 程序。
8. mingw-utils　　小工具。

清楚了这些关键字之后，我们就可以在大量的文件中开始搜寻。下面我具体讲讲我使用的 GCC Version 4 中 2007 年 8 月 14 日 4.2.1-sjlj 版的下载过程,你可以对整个下载页面做个页面内搜索找到它。（本文写作时已经有更新的 Release，但是我还是先写着我熟悉的这个版本，讲解它的细节问题。）这个版本是可以工作在 Vista 下的，更早的版本则未必。

其中 gcc-core 和 gcc-g++是在 GCC Version 4 一栏里的，binutils 在 GNU Binutils 一栏里，mingw32-make 在 GNU Make 一栏里，mingw-runtime 在 GNU Source-Level Debugger 一栏里，w32api 在 MinGW API for MS-Windows 一栏里， mingw-runtime 在 MinGW Runtime 一栏里，很简单的一一对应，各自选最新的版本下载吧。

★ MinGW 和 wxWidgets 的环境配置

下载下来的这些 tar.gz 或 tar.bz2 包，用 WinRAR 或 7-zip 都是可以解压的。把它们都解压到一个文件夹，比如我的就是 C:\MinGW。注意，解压出来的结果，应该是在 C:\MinGW 下出现 bin、include、lib、libexec、doc、man、info 等文件夹，而不是长得像 gcc-core-4.2.1-sjlj-2 这样的文件夹。

解压之后，要建一些快捷方式。请进入 C:\MinGW\bin，为 mingw32-gcc-sjlj.exe 和 mingw32-g++-sjlj.exe 建立分别名为 gcc 和 g++的快捷方式,或者直接把它们复制一份并重命名也可以。这是因为 gcc 和 g++才是它们的标准名字（它们加上了 mingw 前缀和版本号后缀），如果不这样做，本文后面的 makefile 就找不到它们了。

接下来设置环境变量。由于我同时使用 wxWidgets 的 2.8.7 版本和 trunk 版本，所以我并不把环境变量设成全局的，而是每次在运行 MinGW 之前，都运行一个 Batch 脚本来设置一个暂时的环境。（不太严格地沿用 C++术语，这叫不想污染全局命名空间，呵呵）

下面是我的 Batch 脚本，命名为 evnset.bat，放置于 D:\wxBuildPack。其中我设了一些参数，你可以根据你的实际情况进行微调。这个脚本已经设置了 wxWidgets-2.8.7 的所在地。我还有另外一个 evnset9.bat，用来设置适用于 wxWidgets trunk 版本的环境变量。

```
set MinGW_PATH=C:\MinGW

set MinGW_SubBin=i686-pc-mingw32

set MinGW_Ver=4.2.1-sjlj

set wx_PATH=C:\wxWidgets-2.8.7
```

```
set PATH=%PATH%;%MinGW_PATH%\bin;%MinGW_PATH%\%MinGW_SubBin%\bin

set LIBRARY_PATH=%MinGW_PATH%\lib;%MinGW_PATH%\lib\gcc\mingw32\%MinGW_Ver%

set C_INCLUDE_PATH=%MinGW_PATH%\include

set
CPLUS_INCLUDE_PATH=%MinGW_PATH%\include;%MinGW_PATH%\lib\gcc\mingw32\%MinGW_
Ver%\include;%MinGW_PATH%\lib\gcc\mingw32\%MinGW_Ver%\include\c++;%MinGW_PAT
H%\lib\gcc\mingw32\%MinGW_Ver%\include\c++\backward;%wx_PATH%\include;%wx_PA
TH%\contrib\include;
```

这个脚本临时地设置了 PATH（系统到哪里去找可执行文件）、LIBRARY_PATH（系统到哪里去找需要的库文件），C_INCLUDE_PATH（系统到哪里去 C 头文件）、CPLUS_INCLUDE_PATH（系统到哪里去 C++头文件）。

如果你下载的是不同版本的 gcc，你需要自行检查 MinGW 内部的文件夹结构是否像上面这个脚本所勾勒的一样，你可能需要根据实际情况加以调整。同时注意一下 libexec\gcc\mingw32\版本号，这下面有 cc1 和 cc1plus，MinGW 在有些系统上可能会找不到它们。

接下来我们把这个脚本所在的文件夹加到全局的环境变量中。右击我的电脑[13]，属性[14]，高级，环境变量。在用户变量或者系统变量中找到 PATH，然后编辑，在所有其他路径的最后面，先加上;作为分隔符，然后加上 D:\wxBuildPack（我们刚刚放 evnset.bat 的文件夹）。最后，一路确定，环境变量就设好了。

现在测试一下：开始，运行。输入 cmd，这样弹出命令行提示符窗口，输入 evnset，如果显示出了脚本中的那一大堆命令，就说明大环境搭好了。注意，evnset 所设置的路径，只作用于当前窗口，如果你另开一个窗口，你需要重新 evnset。你可以试着用它编译一个最简单的 HelloWorld 程序了。假设当前目录下你的 HelloWorld 源文件为 main.cpp，内容为：

```
#include <iostream>

using namespace std;

int main()

{

    cout << "Hello World!" << endl;

    cin.get();

    return 0;

}
```

[13] 在 Vista 下叫计算机

[14] 在 Vista 下还要在弹出来的控制面板窗口中点一下“改变设置”，这个需要管理员权限

在运行完 evnset 后，输入

```
g++ main.cpp –o main.exe
```

就应该能生成一个 main.exe 了。

2.3.2 编译 wxWidgets

★ 定制 wxWidgets

你可能已经注意到，在 wxWidgets 的根目录，散落着一些 readme-平台和 install-平台文本文件，其中平台包括 msw（Microsoft Windows 的缩写），gtk（Linux 下的 GTK+框架）和 mac（苹果 Mac OS）。install 就是该平台下编译 wxWidgets 的指引。在编译之前，最好先看一看。不过它们在记事本下看起来就好像从来不换行一样。这是因为它们是按照 linux 下的换行方式，比 Windows 少一个\r，Windows 就认不出来哪换行了。不过 Word 还是认得出来的，用 Word 来读吧。

INSTALL-MSW.txt 中 Basic options 和 Advanced options 两节是非常重要的，这里面的项可以让你自己根据自己的需要定制 wxWidgets。最基本的选项有：

- UNICODE　　如果你需要处理中文，或者你需要 unicode，就将它设为 1。
- DEBUG　　设为 1 为 debug 版（包含调试信息），0 为 release 版（体积小）
- SHARED　　设为 1 你将来写的 wxWidgets 程序运行会需要 dll（dll 10M 左右，你的程序自身大小就很小，可能只有几百 k），0 则可以独立运行，但自身体积起码是 M。
- MONOLITHIC 设为 1 则将 wxWidgets 编译为一个单一的大的库文件，0 则将其内部一个个小库分别编译，得到的是很多个小的库文件。建议设为 1，这样链接库的时候方便。如果你对最终发布的程序有体积要求，或者希望精细地选择链接什么不了链接什么，可以设为 0。

下面讲讲我自己的定制。

我的基本需求是，要能用 XRC，能用 OpenGL，能用 ODBC，能用中文，用 dll。在编译之前，我先写两个简单 Batch 脚本，分别为我编译的 debug 版本和 release 版本的 wxWidgets。这两个脚本以后编自己的 wxWidgets 程序依然有用。

第一个脚本名为 dbvdll.bat，意为 **d**e**b**ug　**v**ersion 的动态库，也放置于 D:\wxBuildPack。

```
   mingw32-make  -f  makefile.gcc  USE_XRC=1  USE_OPENGL=1  USE_ODBC=1  SHARED=1
MONOLITHIC=1 BUILD=debug UNICODE=1 clean

   mingw32-make  -f  makefile.gcc  USE_XRC=1  USE_OPENGL=1  USE_ODBC=1  SHARED=1
MONOLITHIC=1 BUILD=debug UNICODE=1 VENDOR=cb
```

第一行命令是清空以前的编译残留下来的.o 目标文件和.o.d 文件，它们的存在可能使编译结果不正确。

第二行命令是正式编译，注意 VENDOR=cb 这一句，这意味着编出来的 dll 会有一个_cb 的后缀。你可以把它改成自己的名字、自己的公司名什么的。我用的 cb 是指 Code::Blocks。

接下来是 rlvdll.bat，意为 **r**e**l**ease　**v**ersion 的动态库，大同小异：

```
   mingw32-make  -f  makefile.gcc  USE_XRC=1  USE_OPENGL=1  USE_ODBC=1  SHARED=1
```

```
MONOLITHIC=1 BUILD=release UNICODE=1 clean

    mingw32-make -f makefile.gcc USE_XRC=1 USE_OPENGL=1 USE_ODBC=1 SHARED=1
MONOLITHIC=1 BUILD=release UNICODE=1 VENDOR=cb
```

为了真的能用 OpenGL 和 ODBC，还需要修改 include/msw/setup.h。找到 wxUSE_GLCANVAS，修改该行为

```
    #define wxUSE_GLCANVAS 1
```

找到 wxUSE_ODBC，修改该行为

```
    #define wxUSE_ODBC 1
```

定制完成。

★ 编译 wxWidgets

wxWidgets 在 Windows 下用 MinGW 怎么编译，已经由 wxWidgets/build/msw 下的 makefile.gcc 指定清楚了。

确认下 C 盘有超过 1G 的硬盘空间（wxWidgets 编译需要的硬盘不会超过 1G，但是，你 C 盘空间少于 1G，你的系统也不会稳定吧），然后点开命令行提示符，进入 wxWidgets/build/msw，依次输入 evnset 和 dbvdll，离开电脑休息半个小时不到，就可以编译出你的 debug 版本的 wxWidgets 了，再输入 rlvdll，就可以把 release 版本的也编译出来。以我自己编译 2.8.7 为例：

```
    C:\Users\Utensil> cd C:\wxWidgets-2.8.7\build

    C:\wxWidgets-2.8.7\build> evnset

    [一长串输出略]

    C:\wxWidgets-2.8.7\build> dbvdll

    [编译时输出略]

    C:\wxWidgets-2.8.7\build> rlvlll

    [编译时输出略]
```

★ 编译成果

编了那么久，究竟编出了什么东西来啊?

编译的副产品在 wxWidgets/build/msw 的以 gcc_msw 开头的子文件夹里。如果编译的是 unicode 版本，后缀会以 u 开头；如果编译的是 debug 版本，接下来会有一个 d；如果编译的是动态库，会以 dll 结尾。例如非 unicode 的 release 版的静态库，这个子文件夹就是 gcc_msw 本身，没有任何后缀。而按我们刚刚的编译方法，会有 gcc_mswuddll 和 gcc_mswudll 两个子文件夹，里面是大量的.o 目标文件和.o.d 文件。

编译的真正成果在 wxWidgets/lib 的子文件 gcc_dll（动态库）和 gcc_lib（静态库）里。gcc_lib 里面会散落着一些.a 库文件。gcc_dll 里面则散落着一些.a 库文件和.dll 动态链接库文件。.a 的命名规则这里不详述，大家可自行观察。而.dll 文件，按我们刚刚的编译方法，会有 wxmsw28ud_gcc_cb.dll、

wxmsw28ud_gl_gcc_cb.dll、wxmsw28u_gcc_cb.dll、wxmsw28u_gl_gcc_cb.dll 四个。注意观察它们的后缀，意义是很明显的。需要说明一下，OpenGL 支持，是不会被编译到 wxWidgets 的那个单一库的 dll 里而会单独编译为一个 dll。

为了这些 dll 能被将来写的程序调用，我们将它们复制到 D:\wxBuildPack 中。

2.3.3 编译 wxWidgets 的例子

2.3.4 利用 wxWidgets 提供的 Bakefile 支持方便书写自己程序的 Bakefile

2.4 Linux(Fedora/Ubuntu)

2.4.1 安装 gcc 环境

2.4.2 使用 configure 生成 makefile

2.4.3 编译 wxWidgets、wxWidgets 的例子

2.4.4 如何巧妙借用 configure.status 生成自己程序的 makefile

2.5 Mac OS(Tiger)

2.5.1 安装终端

2.5.2 安装 XCode（内含 gcc）

2.6 wxWidgets 目录结构简介