信息科学与技术丛书

张长亮　编著

wxWidgets
跨平台程序开发

- wxWidgets程序分析
- wxWidgets事件处理
- wxWidgets图形用户界面
- wxWidgets数据结构
- wxWidgets多领域跨平台开发应用

机械工业出版社
CHINA MACHINE PRESS

信息科学与技术丛书

wxWidgets 跨平台程序开发

张长亮　编著

机械工业出版社

本书系统地介绍了 wxWidgets 及其跨平台程序开发。全书共有 11 章，第 1 章简要介绍了 wxWidgets 的发展历史和框架结构，第 2 章介绍了 wxWidgets 的程序框架及其实现，第 3 章深入剖析了 wxWidgets 的事件处理机制，第 4～11 章详细讲述了 wxWidgets 的图形用户界面、数据结构及 wxWidgets 在图像与绘图、多媒体、打印处理、多线程、网络通信和数据库方面的编程。

本书的目的是帮助读者全面、深入地认识 wxWidgets，既适合 wxWidgets 程序员和编程爱好者阅读，也可作为高校计算机及相关专业的教材。

书中主要程序的完整代码可在 http://www.cmpbook.com 免费下载。

图书在版编目（CIP）数据

wxWidgets 跨平台程序开发 / 张长亮编著. —北京：机械工业出版社，2012.9
（信息科学与技术丛书）

ISBN 978-7-111-39655-0

Ⅰ. ① w… Ⅱ. ① 张… Ⅲ. ① C 语言－程序设计 Ⅳ. ① TP312

中国版本图书馆 CIP 数据核字（2012）第 209632 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：车 忱
责任编辑：车 忱
责任印制：张 楠

北京双青印刷厂印刷
2012 年 11 月第 1 版 · 第 1 次印刷
184mm × 260mm · 23.25 印张 · 576 千字
0 001 － 3 000 册
标准书号：ISBN 978-7-111-39655-0
定价：59.80 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社 服 务 中 心：（010）88361066
销 售 一 部：（010）68326294
销 售 二 部：（010）88379649
读者购书热线：（010）88379203

网络服务
教 材 网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952

出 版 说 明

随着信息科学与技术的迅速发展，人类每时每刻都会面对层出不穷的新技术和新概念。毫无疑问，在节奏越来越快的工作和生活中，人们需要通过阅读和学习大量信息丰富、具备实践指导意义的图书来获取新知识和新技能，从而不断提高自身素质，紧跟信息化时代发展的步伐。

众所周知，在计算机硬件方面，高性价比的解决方案和新型技术的应用一直备受青睐；在软件技术方面，随着计算机软件的规模和复杂性与日俱增，软件技术不断地受到挑战，人们一直在为寻求更先进的软件技术而奋斗不止。目前，计算机和互联网在社会生活中日益普及，掌握计算机网络技术和理论已成为大众的文化需求。由于信息科学与技术在电工、电子、通信、工业控制、智能建筑、工业产品设计与制造等专业领域中已经得到充分、广泛的应用，所以这些专业领域中的研究人员和工程技术人员越来越迫切需要汲取自身领域信息化所带来的新理念和新方法。

针对人们了解和掌握新知识、新技能的热切期待，以及由此促成的人们对语言简洁、内容充实、融合实践经验的图书迫切需要的现状，机械工业出版社适时推出了“信息科学与技术丛书”。这套丛书涉及计算机软件、硬件、网络和工程应用等内容，注重理论与实践的结合，内容实用、层次分明、语言流畅，是信息科学与技术领域专业人员不可或缺的参考书。

目前，信息科学与技术的发展可谓一日千里，机械工业出版社欢迎从事信息技术方面工作的科研人员、工程技术人员积极参与我们的工作，为推进我国的信息化建设作出贡献。

机械工业出版社

前　言

本书是国内第一部系统地介绍 wxWidgets 及其跨平台程序开发的中文著作，旨在更好地促进 wxWidgets 在国内的推广应用。

以 Linux 为代表的开源操作系统正日益挑战 Windows 在操作系统领域的主导地位，形成了多种操作系统共存的局面。在这种情况下，软件开发者在开发前如果选择了单一的目标平台和使用对象，一旦在平台转变时所编写的代码就会很快变得无用而被淘汰，于是催生了一个重要的软件研发方向，即实现所编写的软件能够在多个平台即跨平台运行，这可以有效避免同一软件在不同平台上运行时需要全部重新编写而产生的人力、物力的浪费，必将给软件企业带来巨大利益。跨平台软件的研发也就成为软件企业的研发热点，而研发人员开发跨平台软件的重要工具便是跨平台开发库。wxWidgets 作为典型的跨平台开发库，在国外已经获得了广泛应用，国内很多人也在学习使用 wxWidgets。但国内还没有系统介绍 wxWidgets 的中文书籍，目前市场上能够看到的大多是英文资料，这对学习 wxWidgets 以及 wxWidgets 在国内的推广应用产生了很大的限制。正是由于这一点，本书对 wxWidgets 进行了系统介绍，以期能为感兴趣的读者学习应用 wxWidgets 贡献自己的一点绵薄之力。

全书共有 11 章。第 1 章简要介绍了 wxWidgets 的发展历史和框架结构，第 2 章介绍了 wxWidgets 的程序框架及其实现，第 3 章深入地剖析了 wxWidgets 的事件处理机制，第 4～11 章详细地讲述了 wxWidgets 的图形用户界面、数据结构及 wxWidgets 在图像与绘图、多媒体、打印处理、多线程、网络通信和数据库方面的开发应用。本书在介绍这些内容时，还带领读者深入 wxWidgets 的内部实现过程，使读者既知其然也知其所以然，同时本书也鼓励读者多阅读 wxWidgets 的源代码，学习其中有益的编程技术。书中的代码仅供读者学习或者研究，它们有一部分是笔者编写的，有一部分是修改或者直接引用了 wxWidgets 自带的示例代码，还有一部分来自开源社区，这些代码均经过了笔者的正确性测试，并且书中主要程序的完整代码可在 http://www.cmpbook.com 免费下载。

本书并不否认其他优秀开发库的存在，也不鼓励读者放弃学习其他优秀开发库的机会，因为 wxWidgets 是开放的，任何对 wxWidgets 感兴趣的读者都可以把好的技术拿到 wxWidgets 中来，为 wxWidgets 的完善和发展作出贡献。本书编写过程中得到了很多人的热情帮助，感谢 wxWidgets 开源社区的 wxWidgets 爱好者们，感谢开源代码以及开源文档的作者们，特别是 Julian Smart 及其团队，为世界贡献了伟大的 wxWidgets。

由于时间关系及水平有限，书中错误在所难免，恳请读者指正和见谅。作者的邮箱是 freesoft_os@sohu.com，欢迎来信交流探讨。

作　者

目　录

第1章 概 述

1.1 跨平台开发

随着计算机软硬件技术特别是开源软件的蓬勃发展，操作系统领域出现了多种操作系统共存的局面，应用软件面临着平台转变后会很快变得无用而被淘汰的风险。而从另一个角度来看，开发具有跨平台特性的软件，使其能够运行在多种操作系统下，使得采用不同操作系统的用户都可以使用，则能使软件最大限度地占有市场。由于这些因素，跨平台开发成为软件业研究和发展的重要课题。

1.1.1 什么是跨平台开发

跨平台开发泛指为使软件可以在多种操作系统及不同硬件架构的计算机上运行，即能够做到跨操作系统和硬件平台运行而进行的开发活动。目前不但很多著名的开源软件项目提供跨平台的自由软件，而且不少软件厂商也将他们的产品向不同的平台移植。在进行跨平台开发时，由于不同平台各有特性，软件的功能、性能与移植性及开发工具也各不相同，因此很难得到一般性的、与平台及工具种类无关的开发方式。

1.1.2 跨平台开发方式

跨平台软件的开发方式大体有三种，第一种是先在一个基准平台上开发好，然后移植到其他操作系统。第二种是采用虚拟机技术，利用与平台无关的语言进行开发。第三种是采用跨平台模块组合。

1．移植

软件移植是比较传统的做法，也是一直广泛应用的方式，即把在一种操作系统下运行的软件通过改写部分代码，移植到另外一种或多种操作系统下运行。软件最初的设计、开发语言以及移植时使用的工具决定了移植的代价。如果软件在最初设计时没有考虑到以后的移植，大量采用了平台相关的技术和平台相关开发库进行设计，那么对这些代码的移植将会异常艰辛。比较极端的情况是最初使用的开发语言不能做到跨平台，对这样的软件进行移植将不得不更换开发语言来重新编写全部代码，还要进行全新的测试。下面是进行移植时常用的一些做法。

（1）预编译宏指令

采用跨平台语言编写的软件代码根据平台相关性可以分为平台相关的和平台无关的。如C/C++编写的软件，在进行移植时，平台无关的代码基本上不需要做大的改动，但平台相关的代码需要进行较大调整，这时使用 C/C++的预编译宏指令可以做到让编译器在编译时根据当前的平台来自动选择需要编译的代码，即进行条件编译。对下面的代码，编译器会在编译前根据当前平台来选择具体的窗口类：

```
#if defined(__WXMSW__)
#define wxWindowNative wxWindowMSW
#elif defined(__WXGTK__)
#define wxWindowNative wxWindowGTK
#elif defined(__WXMGL__)
#define wxWindowNative wxWindowMGL
#elif defined(__WXX11__)
#define wxWindowNative wxWindowX11
#elif defined(__WXMAC__)
#define wxWindowNative wxWindowMac
#endif
```

（2）定义通用的数据类型

不同的平台上，相同数据类型的字节长度并不一致，比如 C 语言中的 long int 类型在典型的 32 位机器中占 4B，而在一台 Alpha 机器上则占 8B。因此提高程序可移植性的一个方面就是使程序对不同数据类型的确切大小不敏感，为了达到这个目的就需要定义通用的数据类型，如下面的代码：

```
typedef int Int32;
typedef unsigned int Uint32;
#define SIZEOF_INT 4
#define SIZEOF_LONG 4
#define SIZEOF_WCHAR_T 2
#define SIZEOF_SIZE_T 4
#define SIZE_T_IS_UINT
#define SIZEOF_VOID_P 4
#define SIZEOF_SIZE_T 4
```

（3）利用平台移植工具

目前 PC 的处理能力已经不比采用 UNIX 操作系统的工作站差，将原来 UNIX 下的软件移到 PC 运行可以有效降低对该软件所需硬件平台的投入；另一方面，将 PC 下的软件移到 UNIX 运行，可以充分发挥 UNIX 的处理能力，同样意义重大。在这样的背景下，一些专业研究跨平台开发技术的公司和社会团体针对特定的应用推出了各种软件移植工具，运用这些工具可以大大减少移植的代价。

1）从 UNIX/Linux 到 Windows。Cygwin 是 Cygnus Solutions 公司的产品，它在 Windows 下为用户模拟了一个 UNIX 运行环境，可以帮助程序开发人员方便地把应用程序从 UNIX/Linux 移植到 Windows，是一个功能强大的移植工具。Cygwin 由仿真库和一套 GNU 开发工具集（如 GCC、GDB）组成，仿真库在 Windows 下实现了 UNIX 的应用程序接口（API），工具则用来进行应用程序开发。Cygwin 可以在 Windows CE 以外，Windows 95 以上的所有非 Beta 版本的 Windows 操作系统下工作，如 Windows 98、Windows 2000、Windows XP 等。在 Cygwin 提供的仿真环境下，通过开发工具集中的 GCC 编译器，很多原来在 UNIX/Linux 下用 GCC 编译的软件可以不加修改地在 Windows 下编译并运行，这极大地方便了软件从 UNIX 到 Windows 的移植，特别是一些开源项目。

NuTCRACKER 是 MKS 公司的产品，它在 Windows 下为用户提供了 UNIX 工具、UNIX 库以及一个 X Server，使用户可以方便地将 UNIX/Linux 下编写的程序移植到 Windows 下运行，甚至可以直接使用 Visual C++集成开发环境编译来自 UNIX/Linux 的代码，这大大节省了移植的成本。MKS 公司的 MKS Toolkit 企业开发版不仅可以开发、移植和部署非图形化的 UNIX 应用程序和脚本程序，而且还可以移植原来使用 X Window、OpenGL 的图形化的 UNIX 应用程序。

2）从 Windows 到 UNIX/Linux。Wine 是 Windows API 的开源实现，在 UNIX/Linux 下为用户提供了一个运行 Windows 程序的简捷平台，通过它提供的移植开发工具包以及程序加载器，Windows 下的二进制文件甚至可以不加修改地运行在 UNIX/Linux 下，大大节省了移植所需要付出的劳动量。

Visual MainWin 是 MainSoft 公司的产品，这是一个企业级的移植和跨平台开发软件，提供了从 Windows 程序代码直接生成 UNIX/Linux 程序的方法。用户可以利用 Visual C++集成开发环境来编写程序代码，然后将它们拿到 UNIX 下，使用 Visual MainWin 生成本地执行程序。它由两部分组成，分别为 Visual MainWin SDK 和 Visual MainWin 运行库, SDK 是一个安装在 Visual Studio 中的跨平台插件，这个插件可以配合运行库来生成高可靠性的 UNIX 平台代码。最新的 Visual MainWin 版本已经可以移植.NET 框架和 MFC 7 的应用程序到 UNIX 平台。

2．虚拟机

虚拟机是一种特殊的软件，它可以在计算机平台和终端用户之间创建一种环境，而终端用户能够基于虚拟机所创建的环境来操作软件。根据用途和与直接机器的相关性，虚拟机又可以分为程序虚拟机和系统虚拟机，程序虚拟机为运行单个计算机程序而设计，而系统虚拟机则为用户提供了一个可以运行完整操作系统的系统平台。

采用程序虚拟机技术的典型代表就是 Java 语言，这是由 Sun 公司（已被 Oracle 公司收购）提出的一种跨平台开发语言，其跨平台性主要表现在由它编写的字节码文件可以在任何具有 Java 虚拟机即 JVM 的计算机或者电子设备上运行。而在系统虚拟机方面做得比较突出的是 OS on OS 技术，典型代表是 VMware 公司的 VMware 和 Oracle 公司的 VirtualBox，它们可以在一台 Linux 或 Windows 计算机上同时模拟出多台运行不同操作系统的虚拟机，其中包括 Linux、Solaris、OS/2 等。这些虚拟出来的计算机就像物理上的独立计算机一样工作，用户不需要分区或重新开机就能在同一台计算机上使用两种以上的操作系统；并且不同操作系统的操作环境被完全隔离，安装在不同操作系统下的软件和资料也能得到有效保护；用户还可以在不同的操作系统之间共享文件和应用程序。这些都给用户带来了很大的便利。

使用虚拟机的好处虽然很多，但也有很多突出问题，比如程序执行效率的降低，这是由于凡是需要硬件执行的指令都需要在虚拟机上重新分析和安排，而且运行在虚拟机上的软件被局限在虚拟机提供的资源里，这自然降低了效率。所以在一般情况下，虚拟机技术比较适合快速地做出软件原型，快速地占领市场。当软件的效率受到用户关注时，还是要转向移植和跨平台模块技术。

3．跨平台模块组合

跨平台模块组合目前正广泛应用于跨平台软件开发，特别适用于开发具有跨平台需求的新产品，它实际上是套用了组件的思想来进行跨平台软件的开发。在新的软件产品开始进行设计时，如果有跨平台运行的需求，可以将软件按功能划分成多个可组合的模块，定义好模块间的接口，然后由多个模块小组独立完成单个模块的设计、实现和测试，最后项目整合小组利用开发好的模块逐步组合成完整的产品，并完成在目标平台上的测试。由于各个模块都进行了跨平台设计，所以整合后产品的跨平台问题将会大大减少，这样可以在短时间内推出一个产品在多个平台上的运行版本。

模块思想最大的好处是有利于软件的复用。一般情况下，规模化经营的软件公司的产品

大都形成了系列化，这使得软件之间会有很多功能通用的部分。采用模块的思想，将这些部分独立出来，形成标准化功能模块，交给专门的小组负责维护和升级，而各产品组则把更多的精力放在产品功能的更新上，这将有利于公司整体效率的提高。如果一个公司的产品大多需要跨平台运行，那么组织专门的跨平台模块组是非常合适的。

跨平台模块对目标平台的选择也比较复杂，一般都要包括三个典型的基准平台：Windows、Linux 和 Solaris，如果公司的产品还要支持 Apple 公司的平台，就还得加上 Mac OS X。由于平台自身实现的多样性，在设计和编码时，考虑兼容性就要让效率做出让步，而当软件整体功能比较关注执行效率时，就要在模块的兼容性和效率上做出适当的平衡，这就使得跨平台模块的开发并不轻松，但由于模块功能相对比较独立，模块间的耦合度不大，所以开发跨平台模块的难度还是在一定范围之内的。

目前，不仅大型的专业软件公司将传统的以产品为团队的开发模式转移到了以模块为团队的开发模式，很多自由和共享软件组织也向外界提供开放源码的跨平台开发库来支持基于模块的开发模式，这些开源跨平台开发库包括跨平台网络开发库 ACE，跨平台多媒体开发库 SDL（Simple DirectMedia Layer），跨平台线程开发库 ZThread，跨平台图形用户界面（GUI）开发库 wxWidgets，FLTK 等，这些开发库不仅成熟，而且在不断优化。本书主要介绍 wxWidgets。

1.2 wxWidgets 简介

wxWidgets 是一个跨平台的开源 GUI 框架库，已经在世界范围内广泛应用于各领域，其用户有个体软件开发者、科研院校，以及 AOL、AMD 等大公司。实际上它并不仅限于进行 GUI 设计，它可以实现现代应用程序能够实现的所有功能，包括进程间通信、网络应用、数据库访问等，并且能够很好地支持功能扩展，同第三方开发的库代码具有良好的兼容性，这使得基于 wxWidgets 开发的应用程序能够轻松地移植到不同的平台，而不仅仅是移植界面。

1.2.1 wxWidgets 历史

1992 年，Julian Smart 在 Edinburgh 大学制作一个叫做 Hardy 的图表工具，希望该图表工具既能在 Windows 下运行又能在以 X Window 为图形显示基础的 UNIX 下运行。而当时可用的商业跨平台开发框架非常昂贵。在这种情况下，他决定自己创建一个跨平台的编程框架，wxWidgets（w 代表 Windows，x 代表 X Window）就这样诞生了。1992 年 9 月，wxWidgets 1.0 被上传到学校的 FTP 服务器以供更多的开发者使用。wxWidgets 最初是建立在 XView 和 MFC 1.0 基础上的，由于 Borland C++的使用者抱怨其对 MFC 的依赖，所以后来 Julian Smart 用纯 Win32 SDK 重写了 wxWidgets。XView 被 Motif 取代后，wxWidgets 很快提供了对 Motif 的支持。

随着 wxWidgets 用户社区的成立以及邮件列表的建立，大量的新代码和补丁开始融入到 wxWidgets 中，其中包括 Markus Holzem 提供的对 Xt 的支持。wxWidgets 渐渐拥有了越来越多的来自世界各地的使用者，它提供的产品质量和产品支持甚至被认为要好过商业产品。

1997 年，在 Markus Holzem 的帮助下，新版的 wxWidgets 2 API 发布。在 Wolfram Gloger 的建议下，Robert Roebling 开始领导支持 GTK+的 wxWidgets 即 wxGTK 的开发，现在 wxGTK 已经成为 wxWidgets 在 UNIX/Linux 下最主要的开发库。到了 1998 年，支持 Windows 和 GTK+的 wxWidgets 版本被合入并处于版本控制工具 CVS 的控制之下。Vadim

Zeitlin 加入到项目中来，做了大量的设计和编码工作。同年，Stefan Csomor 开始着手增加支持 Mac OS X 开发的 wxWidgets 即 wxMAC。

1999 年，Vaclav Slavik 的 wxHTML 类和以 HTML 为基础的帮助文件显示控件被加入进来。2000 年，SciTech 公司开始开发 wxUniversal，它提供了属于 wxWidgets 自己的不依赖于其他图形库的窗口控件，以便支持没有窗口控件库的操作系统。wxUniversal 最初被用于为 SciTech 公司的 MGL 提供支持。

到了 2002 年，Julian Smart 和 Robert Roebling 在 wxUniversal 的基础上开发出了 wxX11，这个库仅依赖于 UNIX 和 X11，因此它几乎可以在任何类 UNIX 的环境下使用并且可以用在底层的系统开发中。

2003 年，wxWidgets 开始提供对 Windows CE 的支持，同年 Robert Roebling 在 GPE 嵌入式 Linux 平台上演示了使用 wxGTK 编写的程序。

2004 年，由于微软公司在商标方面提出了异议，wxWidgets 被迫从它原来的名字“wxWindows”改为现在的名字。

同样是在 2004 年，Stefan Csomor 和很多热心的参与者彻底修改了支持 Mac OS X 的开发库 wxMAC，改善了外观显示并提高了程序性能。在 David Elliot 的领导下基于 Cocoa 的开发库 wxCocoa 被稳步设计并改进，William Osborne 也着手开发能够使 wxWidgets 的程序在 Palm OS 上运行的开发库 wxPalmOS。

2005 年 4 月，2.6 版的 wxWidgets 发布了，几乎针对所有平台的开发库在这个版本都有了大幅的改进和提高。

2006 年 8 月，2.7 版的 wxWidgets 发布，在这个版本中，高级用户界面类 wxAUI 增加了对可停靠窗口的支持，并提供了很多新的控件。很快，同年 12 月，2.8 版的 wxWidgets 发布，相比之前的版本，控件类型更加丰富，开始支持 Mac OS X 核心绘图，对 UNIX 下的 MIME 类型的支持也有所改进。

2007 年 6 月，针对 Borland C++ Builder 的窗口设计插件 wxForms 发布，使开发者可以应用 C++ Builder 窗体设计器/编辑器的全部强大功能，在单一平台下能够更加方便地进行代码编辑和调试，创建适用于 Windows、Mac OS X 和 Linux 下的跨平台应用。

2007 年 11 月，wxWidgets 发布了可视组件库 wxVCL，包含 8 个基础类库以及超过 600 个功能函数。

2009 年 5 月，访问工具包 AxTk 发布，使开发者可以在 Windows、Mac OS X 和 Linux/UNIX 下创建使用语音导航的高效程序。

2009 年 7 月，网络浏览器控件库 wxWebConnect 发布，它将 Mozilla 基金会的 Gecko 引擎即 XULRunner 封装成用户友好的功能类，包括嵌入式浏览器控件、Web 内容搜索、Web 页面打印、同 DOM 交互等。

2009 年 9 月，2.9 版的 wxWidgets 发布，相对于 2.8 版本，该版本进行了大量的改进，包括对 Unicode 更好的支持、新的 wxOSX/Cocoa 开发库以及许多其他新的特性和 bug 的修复。

关于 wxWidgets 更多更新的消息，读者可参阅 wxWidgets 官方网站 http://www.wxWidgets.org。

1.2.2　wxWidgets 评述

跨平台 GUI 库有很多，比起其他的跨平台 GUI 库，wxWidgets 有下述优势。

1）wxWidgets 的许可协议属于 LGPL。LGPL 要求使用它开发的软件并不一定要公开源码，无论是开源软件的开发者还是纯粹商业软件开发者都可以免费使用 LGPL 授权的软件开发包，这无疑能大大增加对用户的吸引力。

2）wxWidgets 支持的平台很全面，这些平台包括 Windows 3.1/95/98/NT/2000/XP、带有 Motif 1.2 版本以上的 UNIX、带有 GTK+的 UNIX/Linux/*BSD 及 Mac OS X 等。wxWidgets 为其所支持的各种平台提供了几乎一致的 API，不再需要程序员去专门学习特定平台的 API，并且相比于特定平台的 API 而言，wxWidgets 提供的 API 显得更加清晰和简洁，即使程序员只在一种平台上开发应用程序，也会觉得使用 wxWidgets 是值得的。由于消除了不同平台底层接口的复杂性，应用程序可以在相同的设计规范下开发，在不同的平台实现几乎一样的用户交互界面及操作，这样不仅使程序员从操作系统之间的差异中解脱出来，把更多的精力用于程序的设计，而且使用户在熟悉一种平台上的软件操作后，转移到不同的平台也能熟练操作。

3）wxWidgets 提供本地外观。有些框架为不同平台提供相同的窗口效果，或者使用某些预先设定的主题来模拟实现与平台相仿的外观。相对而言，wxWidgets 使用了本地的窗口系统，所以程序外观不仅看起来与主平台一致，而且事实上它就是本地外观。

4）wxWidgets 的设计使它获得了广泛的编译支持，在其所支持的平台上可被多种编译器编译，如在 Windows 上有 MS Visual C++、Borland C++、Watcom C++、Cygwin、MinGW、Metrowerks CodeWarrior、Digital Mars C++，在 Linux 上几乎所有的编译器包括 GCC 都可以编译 wxWidgets，而且像 Code::Blocks，wxDsigner 这类开发工具本身就是用 wxWidgets 开发的跨平台集成开发环境。

5）wxWidgets 的主体是由 C++构建的，但并不是必须通过 C++才能使用 wxWidgets。wxWidgets 拥有许多其他语言的绑定，如 wxPython、wxRuby 和 wxPerl，使开发者在用其他语言编写程序的时候也可以使用 wxWidgets。

当然，目前 wxWidgets 也有一些不足：

1）文档、资源相对缺乏。

2）缺乏很好的商业化支持。

相比于它的好处，上述不足就显得微不足道了，在进行复杂的跨平台应用特别是桌面应用开发时，wxWidgets 应该是首选开发库。

1.2.3 wxWidgets 框架结构

在开始学习编程之前，应先从总体上认识一下 wxWidgets。下面分别从 wxWidgets 的体系结构、库和源代码的组织三个部分介绍 wxWidgets 的框架结构。

1．体系结构

wxWidgets 的体系结构可以分为四层：最上层是 wxWidgets 为不同平台提供的一致的 API，其次是封装了各自平台 API 的 Port，第三层是各个平台的 API，最后是平台操作系统，如图 1-1 所示。

2．类库

wxWidgets 通常提供的是类库的源代码，需要程序员来编译生成本地库。从版本 2.5.0 以后，wxWidgets 既能以 monolithic build 方式编译成单一的大库，也能以 multilib build 方式

编译成几个小库，这些库包括：

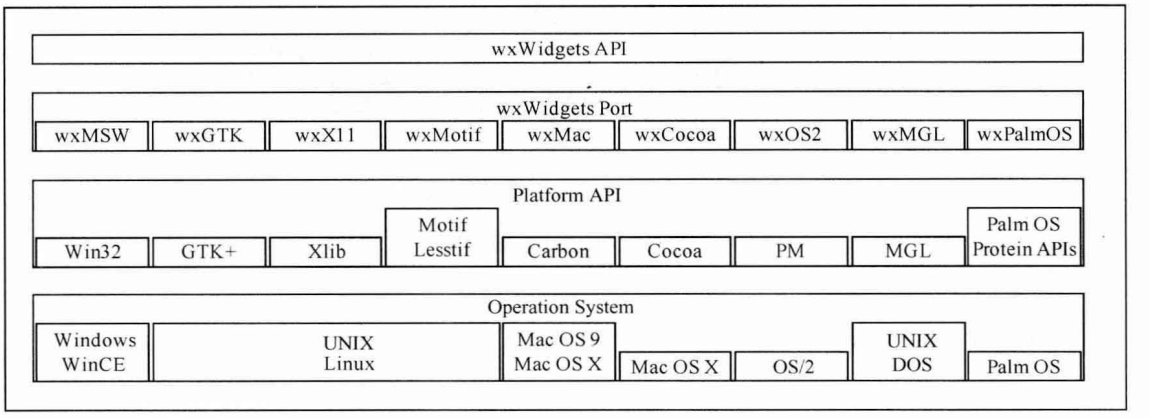

图 1-1　wxWidgets 体系结构

1）wxAui：高级用户界面库。由该库编写的应用程序界面可以实现由用户进行修改和定义。

2）wxBase：基础库。所有 wxWidgets 程序都必须链接该库，它包含了 wxWidgets 强制依赖的类（如 wxString）以及其他用来移植的抽象基类（如 wxWindowbase）。wxBase 可以用来开发控制台程序，而不需要任何 GUI 库或者在 UNIX 下运行 X Window。

3）wxNet：网络开发库。包含了 wxSocket、以 socket 为基础的 IPC 类和依赖于 wxBase 的 wxInternetFSHandler。

4）wxRichText：通用的 rich text 控件库，目前还不能读写 RTF 文件，使用的文件格式为 XML。

5）wxXML：XML 文档解析库。不建议使用，将来的版本会包含新的 XML 处理类，使用类似于 DOM 的 API，从而给使用当前版本 API 的应用程序带来兼容性问题。

6）wxCore：核心 GUI 库。包含了如 GDI 类和控件类，所有 wxWidgets GUI 应用程序都必须链接该库，除了控制台程序。

7）wxAdvanced：高级 GUI 库，依赖于 wxCore 和 wxBase。

8）wxMedia：多媒体库。目前仅包括 wxMediaCtrl，将来会加入更多的类。

9）wxGL：OpenGL 应用库，用来实现 OpenGL 和 wxWidgets 的集成。与其他库不同的是，该库不是 monolithic 库的一部分，而是总被作为独立的库来编译。

10）wxHTML：HTML 库，用来解析和显示 HTML 文档。

11）wxODBC：数据库访问库。

12）wxQA：质量保证库。目前仅有 wxDebugReport 以及相关类。

13）wxDbGrid：数据表显示和编辑库。类 wxDbGridTableBase 可以看做是 wxGrid 和 wxDbTable 的组合，用来直观地访问数据表。

14）wxXRC：XML 资源处理库。包含的 wxXmlResource 类用于读取 XML 格式的 XRC 资源文件。wxWidgets 默认编译方式为 multilib build。图 1-2 列出了 wxWidgets 类库以及它们之间的依赖关系。

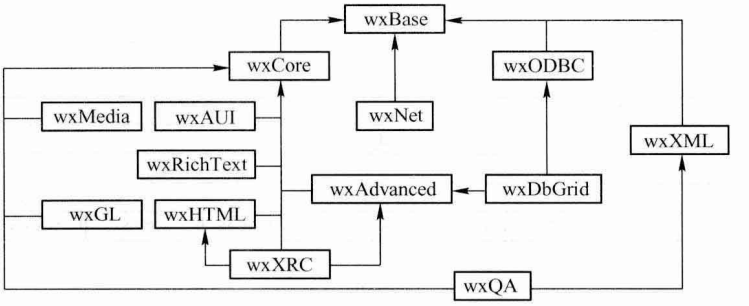

图 1-2　wxWidgets 类库

3．源代码组织结构

图 1-3 是 wxWidgets 的源代码组织结构。

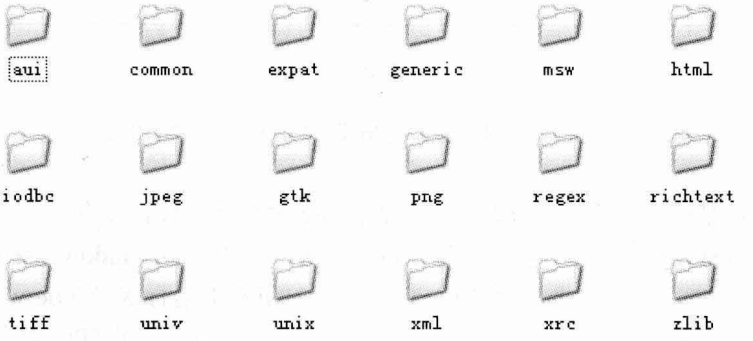

图 1-3　wxWidgets 源代码组织结构

wxWidgets 的源代码大体可以分为六部分：

1）通用代码部分，位于 common 目录内。被所有的平台使用，包括数据结构、运行时类型信息，还有一些被其他类继承的基类，如 wxWindowBase 等。

2）一般代码部分，位于 generic 目录内。包括 wxWidgets 实现的与平台无关的高级控件，如 wxWizard 和 wxCalendarCtrl 等。

3）通用控件部分，位于 univ 目录内。wxWidgets 实现了 wxUniversal，提供了属于自己的不依赖于任何其他图形库的窗口控件，以便支持没有原生窗口控件库的操作系统。

4）平台相关代码部分，位于 gtk、msw 等目录内。封装了平台本地 API 的类。

5）外来代码部分，位于 contrib 目录内。丰富了 wxWidgets 的应用功能，如音视频播放等。

6）第三方代码部分，位于 jpeg、png、regex、zlib 等目录内。都是独立于 wxWidgets 的项目，但是 wxWidgets 用它们来实现许多重要的功能。

此外，wxWidgets 库和内部代码组织具有交叉关系，如 wxNet 库里既包含通用代码又包括平台相关代码部分，其他的库又可能都包括一般代码部分。

4．继承关系

图 1-4 列举了 wxBase 和 wxCore 中部分类的继承关系。

其中，wxObject 是其他绝大部分类的基类。wxWindowGTK 和 wxTopLevelWindowGTK

是与 GTK+平台有关的窗口类，源码位于 gtk 目录下。Windows 下对应的类是 wxWindowMSW 和 wxTopLevelWindowMSW，位于 msw 目录下。针对不同的平台将这些类编译就得到了适用于本地的链接库。

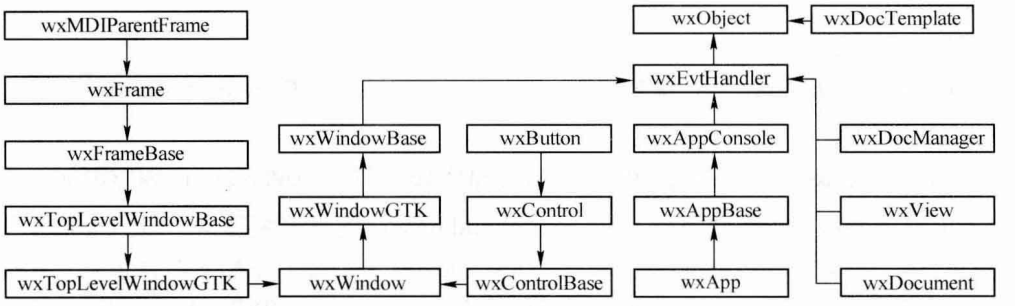

图 1-4　wxBase 和 wxCore 中部分类的继承关系

1.3　wxWidgets 开发环境的搭建

目前 wxWidgets 的开发工具很多，本书推荐使用 Code::Blocks 进行 wxWidgets 跨平台应用程序的开发。Code::Blocks 是一个基于 GPL2 的开源跨平台 C++集成开发环境，采用插件构架，它所提供的大量插件决定了这个集成开发环境具有丰富的功能和特色，用户可以使用它来开发多种程序，它提供了多达 20 种工程模板，包括常用的控制台程序、Win32 GUI 程序、动态/静态链接库、GTK+工程、OpenGL 工程、Qt4 工程、SDL 工程、Ogre 工程、wxWidgets 工程等，当然，用户要确保事先安装好相应的开发库。wxWidgets 作为一个优秀的跨平台 GUI 库，其功能早已不仅仅是用做 GUI 程序开发了，它完全能够胜任网络编程、数据库开发等诸多其他任务。所以，在使用 Code::Blocks 进行跨平台开发时，结合 wxWidgets 是最佳的选择，从某种程度上讲，Code::Blocks 就是专为 wxWidgets 量身打造的。下面将分别介绍在 Windows 和 Linux 下搭建 Code::Blocks+wxWidgets 开发环境的方法，其他平台上的搭建方法可依此类推。

1.3.1　安装

1．Windows

1）到 Code::Blocks 官方网站http://www.codeblocks.org/下载 Code::Blocks 的安装包，并用默认设置安装 Code::Blocks。

2）设置环境变量，在 PATH 环境变量中加上编译器所在的路径，如果下载的是带 MinGW 版本的 Code::Blocks，添加的路径一般为<Code::Blocks>\CodeBlocks\MinGW\bin，其中<Code::Blocks>代表的是 Code::Blocks 安装目录，使用其他编译器时的路径设置方法类似。

3）到 wxWidgets 官方网站 http://www.wxwidgets.org/下载 wxWidgets 的安装包，对于 Windows，下载 wxMSW，并解压安装。可以下载预编译安装包 wxPack，这样就省去了编译 wxWidgets 的麻烦。

4）编译 wxWidgets。wxWidgets 为不同的编译器提供了相应的 makefile 文件，如果编译器使用的是 GCC，用到的 makefile 文件将是 makefile.gcc，可以在命令行中使用下面的编译命令编译 wxWidgets 的库文件，也可以将用到的编译命令做成批处理文件，编译时双击批处理文件即可。

编译发行版本：

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

编译调试版本：

```
mingw32-make –f makefile.gcc BUILD=debug SHARED=1 USE_OPENGL=1 USE_ODBC=1
```

编译前要先修改当前目录到<wxWidgets>\build\msw，这里有编译 wxWidgets 需要用到的 makefile 文件，wxWidgets 为编译它的 samples、demos 和 utils 在相应的目录下也都提供了 makefile 文件。命令行中的编译选项 BUILD、SHARED、USE_OPENGL、USE_ODBC 的解释请参阅 docs/MSW/install.txt，编译选项也可在文件 config.gcc 中进行编辑，该文件在目录<wxWidgets>\build\msw 下。

2．Linux

1）到 wxWidgets 官方网站http://www.wxwidgets.org/下载 wxWidgets 的安装包，对于 Linux，下载 wxGTK，并解压安装。可以下载预编译安装包 wxPack，这样就省去了编译 wxWidgets 的麻烦。这里以 wxGTK-2.6.2.tar.gz 为例，下载后解压：

```
tar zxvf wxGTK-2.6.2.tar.gz
```

转到解压后的目录：

```
cd wxGTK-2.6.2
```

配置编译选项进行编译，这时得到的一般是动态链接库，如果想得到静态链接库，就需要添加编译选项--disable-shared。还可以通过编译选项 prefix 来指定开发库的安装位置，如果不指定则默认安装在/usr/local/lib 下。注意安装的时候需要以根用户身份安装：

```
./configure --with-gtk --with-odbc --with-opengl --enable-sound --enable-mediactrl
make
su <type root password>
make install
```

2）到 Code::Blocks 官方网站http://www.codeblocks.org/下载 Code::Blocks 的安装包，这里以 codeblocks-8.02-src.tar.bz2 为例，下载后解压：

```
tar jxf codeblocks-8.02-src.tar.bz2
```

转到解压后的目录：

```
cd codeblocks-8.02
```

配置编译选项进行编译然后安装，注意安装的时候需要在根用户下安装：

```
./configure    –prefix=/usr
make
su <type root password>
make install
```

Code::Blocks 的库文件同样默认安装在路径/usr/local/lib 下，这里最好通过编译选项 prefix 指定安装在路径/usr/lib 下。