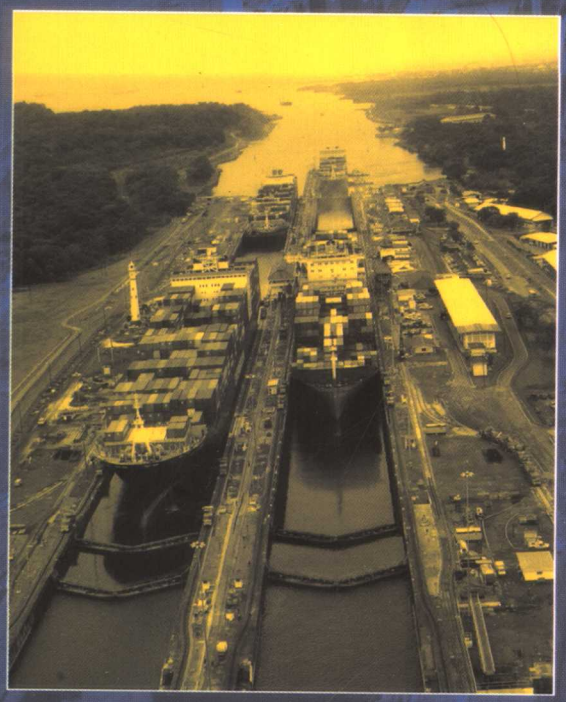
PRENTICE HALL

An Introduction to Design Patterns in C++ with Qt 4

C++ 设计模式

——基于 Qt 4 开源跨平台开发框架



- 学习如何重用库类来编写功能强大的程序
- 利用早已被证实的设计模式来最大化代码的复用
- 详细的代码示例可以教会您充分利用 Qt 4 的强大特性

PEARSON Education

(美) Alan Ezust　Paul Ezust　著　　李仁见　战晓明　译

清华大学出版社

C++设计模式

——基于Qt 4开源跨平台开发框架

(美) Alan Ezust
Paul Ezust 著

李仁见 战晓明 译

清华大学出版社

北 京

北京市版权局著作权合同登记号 图字：01-2006-4774

图书在版编目(CIP)数据

C++设计模式——基于 Qt 4 开源跨平台开发框架/(美)伊斯特(Ezust, A.)，(美)伊斯特(Ezust, P.)著；李仁见，战晓明译. —北京：清华大学出版社，2007.8

书名原文：An Introduction to Design Patterns in C++ with Qt 4
ISBN 978-7-302-15740-3

Ⅰ. C… Ⅱ.①伊… ②伊… ③李… ④战… Ⅲ. C 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2007)第 109474 号

责任编辑：王 军 徐燕萍
装帧设计：孔祥丰
责任校对：成凤进
责任印制：李红英
出版发行：清华大学出版社 **地 址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn **邮 编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175 **邮购热线**：010-62786544
投稿咨询：010-62772015 **客户服务**：010-62776969
印 刷 者：清华大学印刷厂
装 订 者：北京市密云县京文制本装订厂
经 销：全国新华书店
开 本：185×260 **印 张**：32.5 **字 数**：791 千字
版 次：2007 年 8 月第 1 版 **印 次**：2007 年 8 月第1 次印刷
印 数：1～4000
定 价：59.80 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103 产品编号：022891-01

译 者 序

提到C++编程，我们首先想到的自然是Microsoft的Visual C++，然而在开源软件发展得如火如荼的今天，在开发人员一直比较推崇的Linux平台上，我们还有另外一种选择——Qt！Qt的单一源程序的兼容性、它所有拥有的丰富特点、它能提供的C++性能、源代码的可用性、高质量的文档和技术支持使得Qt日益变得流行。

而在今天软件系统越来越复杂、开发周期和开发效率的要求越来越高的计算机软件业，有一种我们不得不提的技术，那就是设计模式！设计模式的出现在计算机软件业掀起了一场革命！其实设计模式并不是高不可攀的技术，它的很多思想都很简单，甚至有些早已在软件开发中普遍应用。当然，要学习开发复杂系统、加入一个庞大的开发团队，学习设计模式是必不可少的。

本书提供了一种同时学习C++编程和设计模式的好方法。书中从简单的C++语法开始讲起，逐渐深入到高层的C++应用，在这个过程中把设计模式的思想贯彻给读者。如果读者能够完整地学习完本书，那么就可以初步掌握C++语言，同时也可以掌握设计模式的精髓。书中的例子简单、有趣而又有用，通过不断的实践，能够进一步加深读者对这些软件开发技术的理解。尤其是第25章给出的MP3播放器的例子，它综合使用了本书中的重要知识，一步一步地引导读者开发一个拥有一定使用价值的小插件，注意，仅仅通过一本书的学习和实践，读者就能够开发自己的软件了！

当然，软件开发不是一蹴而就的，它对实际动手能力的要求非常高，读者必须通过不断地学习和实践，逐步地提高编程能力和软件设计与开发能力。但是，好的开始是成功的一半，就从本书开始大家的软件开发之旅吧！

在翻译本书的过程中，我们注意到Qt中文论坛(http://qtcn.org/bbs)已经出现了本书中文版的需求帖子，感谢大家一直以来的关注！经过2006年那个枯燥和干燥的冬天，现在本书终于全部翻译完毕，很快就可以与大家见面了。不过由于时间仓促，纰漏在所难免，望大家能够理解。如果在阅读本书的过程中发现了错误，望与我们联系(wkservice@tup.tsinghua.edu.cn)，谢谢！

译者 于湖南长沙

2007年3月

前　言

C++在 1989 年被标准化之前，就已经被广泛使用了很多年，这使得 C++相对于当今流行的其他编程语言来说要相对成熟一些。C++是一种能够用于创建快速、高效和关键任务系统的重要语言，同时也是最灵活的编程语言之一，能够给开发者提供多种编程风格，其中囊括高层的 GUI 代码和底层的设备驱动。

在 20 世纪 90 年代的最初几年，C++是最流行和使用最广泛的面向对象(OO)编程语言。许多计算机科学(CS)专业的学生都是通过 C++来学习面向对象编程的，这是因为 C++允许 C 程序员较容易地转换到 OOP，而在此之前许多 CS 教授则一直在教授 C 语言编程。

从 1996 年左右开始，Java 超越 C++成为学生们开始学习的一种面向对象语言。我们认为 Java 之所以会如此流行，有如下几个原因：

- Java 语言本身比 C++语言要更简单。
- Java 有内建的垃圾回收机制，因此程序员无需关注恼人的内存释放工作。
- 开发工具箱中包含了一个标准 GUI 类集合。
- 内建的 String 类支持 Unicode。
- Java 语言本身支持多线程。
- 创建和“插入”Java 档案(JAR)要比重编译和重链接库容易得多。
- 许多 Web 服务器提供 Java API，能够容易地进行集成。
- Java 程序是平台独立的(Wintel、Solaris、Mac OS、Linux、*nix 等)。

如果把 C++与 Qt 4 结合起来使用，则可以获得许多上面所述的 Java 语言的优点。

- Qt 提供了一组更容易理解的 GUI 类，而且相对于 Java 的 Swing 类来说，它们运行更快，看起来也更加舒服，使用起来更加灵活。
- 信号与槽要比 Java 中的(Action|Event|Key)Listener 接口更易使用。
- Qt 4 拥有插入体系结构，这使得我们可以将代码加载到一个应用中而无需进行重编译或重链接。
- Qt 4 提供了 foreach 机制，可以对一个集合进行迭代，进行简单的读写操作。

尽管 Qt 没有提供垃圾回收机制，我们仍然可以采用各种替代方法来避免在代码中直接进行堆对象删除操作。

- 容器(参见 10.2 节)
- 父对象与子对象(参见 9.2 节)
- auto_ptr(参见 16.3.2 节)
- QPointer(参见 19.9 节)
- 子对象(参见 2.8 节)
- 栈对象(参见 20.3 节)

使用 Qt 来开发 C++程序能够与 Java 一样易用、易懂、方便，而在速度、效率方面则远超后者，同时这也使得我们可能将程序从强调处理的服务器应用转换为强调图形界面的

高速“游戏”。

使用 Qt 学习 C++的另外一个益处是 Qt 在开源项目中已经被广泛采用，因此就可以学习大量有价值的开源代码，对其进行重用，甚至有可能帮助修改和改进这些代码。

如何使用本书

第 I 部分包括对 C++语言、UML 和 Qt 核心的简单介绍。本部分的目的是尽量避免读者在后面阅读本书时不断地翻看前面的内容，编排时按照内容的简单程度和详细程度进行了排序，因此，即使是 C++初学者也不会在一开始就感到内容多到无法接受。

在第 II 部分，读者将会学习更高层次的编程思想、Qt 模块和设计模式。在讲解具体内容的过程中，我们巧妙地利用示例代码来介绍编写代码和使用这些知识的方法。

为了保证知识的完整性和便于读者进行深入的学习，我们在第III部分给出了一些第 I 部分所介绍内容的深层次扩展内容，它们虽然较为枯燥，但却是非常重要的 C++特性。当然，在阅读到此部分的时候，会发现理解这些难点其实也并不难。

在每一章的结尾，我们给出了一些练习和要点回顾问题。其中大部分的编程练习都可以在我们的 Web 网站上找到答案。对于那些要点回顾问题，如果答案在本章中无法找到，那么我们一般会指出其答案所在的位置。如果采用本书作为教程，那么这些问题可以由学生或老师在课堂上或者考试中进行提问。

本书中所有示例的源代码都包含在文件 src.tar.gz 中，可以从 http://oop.mcs.suffolk.edu/dist 和 www.tupwk.com.cn/downpage 处下载。

目　录

第Ⅰ部分　C++和 Qt 4 简介

第 1 章　C++简介 …… 3
1.1　C++概述 …… 4
1.2　C++简史 …… 4
1.3　在开源平台上安装 …… 4
1.3.1　*nix …… 5
1.3.2　从源代码安装 …… 6
1.4　在 Win32 平台上安装 …… 8
1.5　第一个 C++示例 …… 9
1.6　输入与输出 …… 12
1.7　标识符、类型与常量 …… 15
1.8　C++的基本数据类型 …… 17
1.8.1　main 函数与命令行参数 …… 19
1.8.2　代数运算 …… 20
1.9　C++标准库字符串 …… 23
1.10　流 …… 24
1.11　关键字 const …… 27
1.12　指针与内存访问 …… 28
1.12.1　一元运算符&与* …… 28
1.12.2　运算符 new 和 delete …… 30
1.13　const*与*const …… 32
1.14　引用变量 …… 34

第 2 章　类 …… 37
2.1　结构 …… 38
2.2　类定义 …… 39
2.3　成员访问限定符 …… 40
2.4　封装 …… 42
2.5　UML 简介 …… 43
2.6　类的友元 …… 44
2.7　构造函数 …… 44
2.8　子对象 …… 46
2.9　析构函数 …… 47
2.10　关键字 static …… 48
2.11　复制构造函数与赋值运算符 … 51
2.12　转换 …… 53
2.13　const 成员函数 …… 55

第 3 章　Qt 简介 …… 65
3.1　示例工程：使用 Qapplication 与 QLabel …… 66
3.2　Makefile、qmake 以及工程文件 …… 67
3.2.1　#include：搜索头文件 …… 69
3.2.2　make 命令 …… 70
3.2.3　清除文件 …… 71
3.3　获得在线帮助 …… 72
3.4　风格指南与命名约定 …… 73
3.5　Qt 核心模块 …… 74
3.6　流与日期 …… 74

第 4 章　列表 …… 77
4.1　容器简介 …… 78
4.2　迭代器 …… 78
4.3　关系 …… 80

第 5 章　函数 …… 85
5.1　函数声明 …… 86
5.2　重载函数 …… 86
5.3　可选参数 …… 89
5.4　运算符重载 …… 91
5.5　通过值传递参数 …… 94
5.6　通过引用传递参数 …… 96
5.7　const 引用 …… 99
5.8　函数返回值 …… 100
5.9　从函数中返回引用 …… 100
5.10　根据 const 属性进行重载 …… 101
5.11　内联函数 …… 103

5.12 内联还是宏扩展……104

第 6 章 继承与多态……109
6.1 简单派生……110
6.2 多态派生……116
6.3 从抽象基类中派生……121
6.4 继承设计……124
6.5 重载、隐藏与覆盖……126
6.6 构造函数、析构函数与拷贝赋值运算符……127
6.7 处理命令行参数……129

第II部分 高级编程

第 7 章 库……139
7.1 代码容器……140
7.2 重用其他库……141
7.3 组织库：依赖管理……142
7.4 安装库：实验室练习……145
7.5 框架与组件……146

第 8 章 设计模式简介……149
8.1 目录与文件：QDir 与 QFileInfo……150
8.2 Visitor 模式……151
8.3 使用继承来定制 Visitor……153

第 9 章 QObject……159
9.1 QObject 的子对象管理……161
9.2 组合模式：父对象与子对象……163
9.3 QApplication 与事件循环……166
9.3.1 布局：初观……168
9.3.2 连接到槽……169
9.3.3 信号与槽……170
9.4 Q_OBJECT 与 moc：一览表……174
9.5 值与对象……175
9.6 tr()与国际化……176

第 10 章 范型与容器……179
10.1 范型与模板……180
10.1.1 函数模板……180
10.1.2 类模板……181
10.2 容器……184
10.3 管理容器、组合与聚合……185
10.4 隐式共享类……188
10.5 范型、算法和运算符……189
10.6 serializer 模式……191
10.7 分类映射示例……193

第 11 章 Qt GUI 部件……199
11.1 部件分类……200
11.2 QMainWindow 和 QSettings……201
11.3 对话框……204
11.4 图像与资源……208
11.5 部件的布局……211
11.5.1 Spacing、Stretching 和 Struts……214
11.5.2 在布局之间移动部件……215
11.6 QActions、QMenus 和 QMenuBars……219
11.7 QActions、QToolbars 和 QActionGroups……221
11.8 区域与 QDockWidgets……228
11.9 QStringList 的视图……229

第 12 章 并行……233
12.1 QProcess 与进程控制……234
12.1.1 进程与环境……236
12.1.2 Qonsole:在 Qt 中编写一个 Xterm……239
12.1.3 带有键盘事件的 Qonsole……241
12.2 线程与 QThread……244
12.2.1 QPixmap 和 Qthread 直观演示例子：电影播放器……245
12.2.2 带有 QTimer 的电影播放器……248
12.2.3 多线程、队列和 Loggers 的例子：Giant……250
12.2.4 线程安全与 QObjects……256

12.3 总结：QProcess 和 QThread …257

第 13 章 验证与正则表达式 ……259
13.1 验证器 ……260
13.2 正则表达式 ……261
13.2.1 正则表达式语法 ……262
13.2.2 正则表达式：电话号码识别 ……264
13.3 正则表达式验证 ……267

第 14 章 解析 XML ……271
14.1 Qt XML 模块 ……274
14.2 事件驱动解析 ……275
14.3 XML、树型结构和 DOM ……278
14.3.1 Visitor 模式：DOM 树遍历 ……280
14.3.2 使用 DOM 生成 XML …… 283

第 15 章 元对象、性质和反射编程 … 289
15.1 反模式 ……290
15.2 QMetaObject：MetaObject 模式 ……291
15.3 类型识别与 qobject_cast ……292
15.4 Q_PROPERTY 宏：描述 QObject 性质 ……294
15.5 QVariant 类：访问性质 ……297
15.6 DataObject：QObject 的一个扩展 ……299
15.7 性质容器：PropsMap ……301

第 16 章 更多设计模式 …… 303
16.1 创建型模式 ……304
16.1.1 抽象工厂 …… 305
16.1.2 抽象工厂和库 …… 306
16.1.3 qApp 和 Singleton 模式… 308
16.1.4 创建规则和友元函数 (友元函数的真正用处) … 309
16.1.5 使用工厂的好处 …… 312
16.2 Serializer 模式回顾 ……315
16.2.1 导出到 XML …… 317
16.2.2 通过抽象工厂导入对象…318
16.3 Façade 模式 …… 322
16.3.1 实用 Façade ……325
16.3.2 智能指针：auto_ptr ……325
16.3.3 FileTagger：Façade 示例 ……326

第 17 章 模型与视图 ……331
17.1 M-V-C：控制器 …… 332
17.2 动态表单模型 …… 333
17.2.1 表单模型 ……336
17.2.2 表单视图 ……338
17.2.3 未预见的类型 ……340
17.2.4 控制 Actions ……341
17.2.5 DateObject 表单模型 ……343
17.3 Qt 4 模型和视图 …… 347
17.4 表模型 …… 348
17.5 树模型 …… 354

第 18 章 Qt SQL 类 ……359
18.1 MySQL 简介 …… 360
18.2 查询与结果集合 …… 363
18.3 数据库模型 …… 364

第III部分 C++语言参考

第 19 章 类型与表达式 ……369
19.1 运算符 …… 370
19.2 逻辑表达式的估值 …… 373
19.3 枚举 …… 373
19.4 有符号与无符号整数类型…… 375
19.5 标准表达式转换 …… 377
19.6 显式类型转换 …… 378
19.7 使用 ANSI C++类型转换进行安全类型转换 …… 379
19.7.1 static_cast 与 const_cast …379
19.7.2 reinterpret_cast ……382
19.7.3 不使用 C 语言风格的类型转换的原因 ……383

19.8 运行时类型识别(RTTI)……383
19.9 成员选择运算符……385

第 20 章 作用域类与存储类……391
20.1 声明与定义……392
20.2 标识符作用域……393
20.2.1 标识符的默认作用域小结……394
20.2.2 文件作用域与块作用域及操作符::……395
20.3 存储类……397
20.4 名字空间……400
20.4.1 匿名名字空间……402
20.4.2 开放的名字空间……402
20.4.3 名字空间、静态对象与 extern……403

第 21 章 语句与控制结构……405
21.1 语句……406
21.2 选择语句……406
21.3 循环……409
21.4 异常……411
21.4.1 异常处理……411
21.4.2 异常类型……411
21.4.3 抛出事件……412
21.4.4 try 与 catch……415
21.4.5 再谈 throw……419
21.4.6 重新抛出的异常……420
21.4.7 异常表达式……422

第 22 章 内存访问……427
22.1 指针的“病理学”……428
22.2 与堆内存有关的指针“病理学”进阶……430
22.3 内存访问概要……432
22.4 数组……432
22.5 指针算术操作……433
22.6 数组、函数和返回值……434
22.7 几种不同的数组……436
22.8 有效指针操作……436
22.9 new 操作失败的处理方法……438
22.9.1 set_new_handler(): 解决 new 失败的另一种方法…439
22.9.2 使用 set_new_handler 和 bad_alloc……440
22.9.3 检测 null：测试 new 失败的更新方法……441
22.10 本章小结……442

第 23 章 继承详述……445
23.1 虚函数表指针和虚函数表…446
23.2 多态和虚析构函数……448
23.3 多重继承……450
23.3.1 多重继承的语法……450
23.3.2 带抽象接口的多重继承…452
23.3.3 解决多重继承冲突……453
23.4 public、protected 和 private 派生……456

第 24 章 其他相关话题……459
24.1 带有变长参数列表的函数…460
24.2 资源共享……461

第Ⅳ部分 编 程 作 业

第 25 章 MP3 点唱机作业……467
25.1 数据模型：Mp3File……469
25.2 Visitor：生成播放列表……470
25.3 Preference：一个枚举类型…471
25.4 重用 id3lib……473
25.5 PlayListModel 序列化……475
25.6 测试 Mp3File 相关类……475
25.7 简单查询和过滤器……476
25.8 Mp3PlayerView……478
25.9 模型和视图：PlayList……479
25.10 源选择器……479
25.11 持久设置……481
25.12 给 FileTagger 编辑表格视图……481

25.13　数据库视图 ……482

第Ⅴ部分　附　录

附录 A　C++保留的关键字 ……487

附录 B　标准头文件 ……489

附录 C　开发环境 ……491

C.1　用于#including 文件的预处理器 ……491

C.2　链接器 ……493

C.3　调试 ……496

C.3.1　建立一个可调试的目标 ……497

C.3.2　gdb 快速入门 ……497

C.3.3　查找内在错误 ……499

C.4　Qt 助手和设计器 ……501

C.5　开源 IDE 和开发工具 ……502

C.5.1　UML 建模工具 ……504

C.5.2　jEdit ……504

参考文献 ……507

第 I 部分

C++和 Qt 4 简介

第 1 章　C++简介

第 2 章　类

第 3 章　Qt 简介

第 4 章　列表

第 5 章　函数

第 6 章　继承与多态

第1章 C++ 简 介

本章将简单介绍C++语言，定义一些基本概念，主要包括关键字、常量、标识符、声明、基本类型、类型转换等；同时还简要说明C++的历史、演化过程以及C++语言与C语言的关系。

1.1　C++概述

C++语言最初是 C 语言的一个超集，也称为“面向对象的 C”。它增加了几个高层功能，诸如强制类型转换、数据抽象、引用、操作符与函数重载，以及面向对象编程的若干支持，增强了 C 语言的功能。

C++保留了促进 C 语言流行和成功的特性：运行速度快、效率高，以及广泛的表达能力。这种广泛的表达能力使得程序员能够在从最底层(例如直接的操作系统调用和按位操作)到最高层(例如操作大而复杂的对象或对象图)的多个层次上进行编程。

C++设计之初的基本原则是：添加到 C++中的任何功能都不能引起不使用此功能的 C 语言代码的运行时开销[1]。当然，编译器的负担是肯定会增加的；另外，使用有些功能时也确实会引入运行时开销。但是使用 C++编译器编译的 C 程序应该与使用 C 编译器编译时运行一样快。

1.2　C++简史

C++由 Bjarne Stroustrup 在 Bell 实验室工作时所设计，最终由 Bell 实验室打包发行，1981 年 AT&T 公司内部开始出现最初的 C++版本，其后 C++根据用户的反馈逐步演化发展。

1986 年上半年 Stroustrup 撰写的著名 *C++ Programming Language* 出版。1989 年 C++2.0 版本发布，C++迅速成为一种严谨、实用的编程语言。同年，人们开始致力于制定 C++国际标准。

由美国国家标准化协会(American National Standards Institute，ANSI)X3J16 委员会负责的 C++标准化工作从 1989 年开始，至 1997 年结束。最后 X3 秘书处以 *Draft Standard—The C++ Language, X3J16/97-14882, Information Technology Council(NSTIC)Washington, DC* 为题公布了他们制定的 C++建议标准。1998 年 6 月，参加过历时 9 年的 ANSI/ISO(International Standards Organization，国际标准化组织)工作的来自 20 个主要国家的代表一致接受了该建议标准。

Stroustrup 撰写的 *C++ Programming Language* 于 1997 年出版第三版[Stroustrup97]，该书被广泛作为 C++参考手册。

ANSI/ISO 建议标准的 HTML 版本参见 http://oop.mcs.suffolk.edu/draftstandard。

1.3　在开源平台上安装

UNIX 及 UNIX 相关平台都自带了 C++的开源开发工具(ssh、bash、gcc)，基于 BSD 的 Mac OS/X 也是如此[2]。

1 不幸的是，异常处理打破了这个规则，如果打开异常处理，就会引入一定的运行时开销，这也是为什么许多库仍然不采用异常机制的原因。

2 BSD 代表 Berkeley Software Distribution，由加州大学伯克利分校的计算机系统研究小组(Computer Systems Research Group，CSRG)发布。自 20 世纪 80 年代以来，该组织一直是 UNIX 及其各种基本协议(例如 TCP/IP)的一个重要研究中心。

1.3.1　*nix

在讨论类 UNIX 操作系统(Linux、BSD、Solaris 等)独有的一些概念时，我们以*nix 简称此类操作系统。

另外一个重要的缩略语是 POSIX(Portable Operating System Interface for UNIX，UNIX 可移植操作系统接口)，该标准由 IEEE(Institute of Electrical and Electronics Engineers，美国电气与电子工程师协会)倡导。IEEE 是一个面向工程师、科学家以及学生的组织，以制定计算机和电子工业的开发标准而著称[3]。

本节主要面向使用*nix 操作系统的读者。

书中示例都在 Qt 4.1 下经过测试，因此我们建议读者安装 Qt 4.1 或更新的版本。针对本书设置计算机的第一步是确认系统完全安装了 Qt 4.1，这不仅包括源代码和编译库代码，还包括 Qt 帮助文档系统、编程示例以及 Qt Designer 程序。

如果系统中安装了 KDE 3.*(K 桌面环境)，那么就已经安装了 Qt 3.*的运行时版本。但我们给出的示例在 Qt 3 下无法编译通过，仍需要单独安装 Qt 4。

可以通过如下命令来查看系统内是否已经安装 Qt 以及 Qt 的版本：

```
which qmake
qmake -v
```

第一条命令的输出会显示可执行文件 qmake 在系统中的位置，如果输出类似于“bash: qmake: command not found”，那么可能的情况是：

- 没有完全安装 Qt(包括开发工具)
- 安装了 Qt，但 PATH 变量中没有包含路径“*/path/to/qt4/bin*”
- 由 qmake-qt 4 包管理器安装，以避免与 Qt 3 包发生冲突

如果通过第一条语句获知了 Qt 的安装位置，那么第二条语句“qmake –v”就可以输出 Qt 的版本信息。如果输出信息如下：

```
Using Qt version 4.x.y
```

那么可以通过下面的命令检查其他 Qt 工具是否可用：

```
which moc
which uic
which assistant
which designer
```

如果 qmake 所在的系统目录中存在上述可执行文件，则表明 Qt 4 已经安装成功并且可以使用了。

3 我们可以用 POSIX 正则表达式(参见 13.2)把*nix 表示成：(lin| mac-os| un| solar| ir| ultr| ai| hp)[iu]?[sx]

如果无法通过上面的测试，那么计算机中可能安装了 Qt 的早期版本，或者根本没有安装 Qt，也有可能在安装 Qt 4 时未安装某些组件，无论是哪种情况，都需要重新安装最新版本的 Qt 4。

从安装包安装 Qt 4

在一些*nix 系统中，可以从包含已编译代码的包安装 Qt 4。

使用*nix 系统的安装包管理器(比如 apt、urpmi、aptitude、kpackage、synaptic、rpmdrake 等)，可以轻松快捷地从包含 Qt 4 的安装包进行安装。下面是可用于 Debian 系统的安装包列表(操作系统不同，该列表也会有所不同)：

```
[ROOT@lazarus]# apt-cache search qt4
libqt4-coure - Qt 4 core non-GUI functionality run-time library
libqt4-debug - Qt 4 debugging run-time libraries
libqt4-dev - Qt 4 development files
libqt4-gui - Qt 4 core GUI functionality run-time library
libqut4-qt3support - Qt 3 compatibility library for Qt 4
libqt4-sql - Qt 4 SQL database module
qt4-designer - Qt 4 designer
qt4-dev-tools - Qt 4 development tools
qt4-doc - Qt 4 API documentation
libqt4-designer - Qt Designer libraryies
[ROOT@lazarus]#
```

正如上面所示，在 Debian 中 Qt 4 被划分成多个独立的安装包，这样开发者可以在配置时获得更好的灵活性。在开发时，需要全部的 Qt 4 安装包，包括开发工具、完整的头文件、文档、设计工具、辅助工具和示例。

1.3.2　从源代码安装

可以从 Trolltech[4]下载最新的源文件 tarball(一种文件打包格式)，然后进行解包和编译。通常用来解压该 tarball 的两个位置是/usr/local(需要是 root 用户)和$HOME/qt。

tarball 是利用 tar 命令生成的一个文件。为了便于存储和传送，一般使用 tar 命令把多个文件结合生成一个文件(扩展名通常是.tar)，此文件往往再次使用诸如 gzip 和 bzip2 等压缩工具进行压缩，形成一个扩展名为.gz 或者.bz 的文件。

tarball 的解压命令取决于其建立方式，一般可以通过文件的扩展名来进行判断：

```
tar -vxf whatever.tar      //使用"verbose"开关
tar -zxf whatever.tar.gz   //使用 gzip 压缩
tar -zxf whatever.tgz      //同样也是使用 gzip 压缩
```

4 http://www.trolltech.com/download/opensource.html