HZ BOOKS
华章IT

腾讯公司资深研发工程师多年后台开发经验总结，获腾讯、Facebook、微软、阿里、百度多位资深技术专家高度认可。

完整勾勒后台开发技术能力体系，多维度讲解了成为一名后台开发工程师所需掌握的核心技术、开发工具和实践方法，后台工程师修炼必读！

徐晓鑫 著

Server-side Development: Technology and Practices

后台开发
核心技术与应用实践

机械工业出版社
China Machine Press

Server-side Development: Technology and Practices

后台开发

核心技术与应用实践

徐晓鑫 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

后台开发：核心技术与应用实践 / 徐晓鑫著 . —北京：机械工业出版社，2016.8

ISBN 978-7-111-54339-8

I. 后… II. 徐… III. 网络－开发 IV. TP393.092

中国版本图书馆 CIP 数据核字（2016）第 167884 号

后台开发：核心技术与应用实践

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：李 艺　　责任校对：董纪丽

印　　刷：三河市宏图印务有限公司　　版　　次：2016 年 8 月第 1 版第 1 次印刷

开　　本：186mm × 240mm 1/16　　印　　张：26.5

书　　号：ISBN 978-7-111-54339-8　　定　　价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88379426 88361066　　投稿热线：（010）88379604

购书热线：（010）68326294 88379649 68995259　　读者信箱：hzit@hzbook.com

Preface 序

C++ 可能是计算机历史上最早被发明的高级程序语言，同时也是当今最活跃的程序设计语言之一。C++ 很强大，强大到你可以使用它做任何层面的开发；C++ 也很脆弱，脆弱到需要程序员自己去控制内存回收，一个不小心就会使整个程序 Core Dump。C++ 语言的创始人 Bjarne Stroustrup 曾私下承认，为了提高 C++ 程序员的薪水和地位，在设计 C++ 编译器版本过程中有意地增加了 C++ 语言的难度，使 C++ 更偏向于资深程序员的使用习惯，提高学习门槛，从而增加 C++ 程序员的身价。学习曲线的增加并不是没有任何回报的，在服务端后台开发、处理多并发的海量网络请求方面，C++ 语言有天然的优势。因此，当应用的用户量、并发量迅速增长，达到一定量级之后，后端服务的技术架构都会转变为 Linux C++。

要做一名优秀的使用 C++ 进行后台开发的程序员，只掌握 C++ 语言是远远不够的，还需要掌握如何进行编译、链接、调试，如何使用网络协议、IO 模型和一些常用的类库，等等。我曾经面试过不少后台开发程序员，他们往往很重视语言本身，但是对一些语言之外的东西理解不够透彻，影响了他们的技术发展。我也读过不少相关方面的技术书籍，往往都过多地停留在语言层面，忽略了实际开发工作中需要用到的知识。

晓鑫在腾讯从事开发工作多年，有丰富的后台开发经验，她从实际的后台开发经验出发，讲解了后台开发中需要用到的方方面面的知识。从 C++ 语言出发，又不止于 C++ 语言，本书可以说是一本 Linux C++ 后台开发的实战典范。当知道晓鑫在写这么一本书的时候，我真心为国内的众多开发者感到高兴。如果读者有意愿成为一名从事 Linux 后台开发的程序员，本书无疑是一本最佳的参考书籍。

研发是一项讲究实战的工作，一切不从实际工作出发的技术书籍都是纸上谈兵，没有实际意义。一本优秀的技术书籍应该是这样的：当读者按照书中的内容进行实操的时候，读者写的每一行代码都是有价值的，能够在实际工作中派上用场。本书恰好做到了这一点。这是

一位技术书籍作者对读者的起码诚意。

软件工程师是一种需要坚定、踏实、精益求精的“工匠精神”的职业，心浮气躁、得过且过的态度不可能把代码写好。老一辈的人说“字如其人”，在软件领域，我们同样可以说“代码如其人”，一个人的行事风格和为人态度都会体现到他所写的代码上面。按照晓鑫的书去学习，读者可以潜移默化地学习到她多年后台开发所炼就的“工匠精神”。我想，相对于所学习到的知识，这于一个工程师来说更为重要。

黄世飞

腾讯云平台技术总监

Introduction 绪　论

0.1　什么是后台开发

听到“后台开发”这个词，估计读者心中都或多或少会有一些自己的感性认识，这种认识可能有一些差别，但估计大部分人都有这么一种看法：“后台开发”是编写一些用户看不见的程序，也就是非界面的程序，既不是网页，也不是 App，更不是桌面程序，因为这些都是用户看得见的（被称为“前台开发”）。这种感性认识在一定程度上是正确的，但是它不够具体，也不够全面。

我们这里所说的“后台开发”的确是用户“看不见”的部分，但是还有很多界面性的程序是给企业内部人员使用的，这些虽然是界面程序，但是对于最终用户来说也是“看不见”的。举个例子，开发一个电子商务网站，提供给客户进行商品购买的网页是用户看得见的，不属于“后台”，但是电商网站内部员工使用的“用户管理系统”，“订单管理系统”等，也是用户看不见的，但它们不属于本书中所指的“后台”。在某些场合，或者某些人的习惯中，这些内部使用的系统也叫“后台”，这几种说法都没有错，希望读者在听到的时候，知道说话人具体指的是什么。

本书介绍的“后台开发”指的是“服务端的网络程序开发”，从功能上可以具体描述为：服务器收到客户端发来的请求数据，解析请求数据后处理，最后返回结果，如图 0-1 所示。

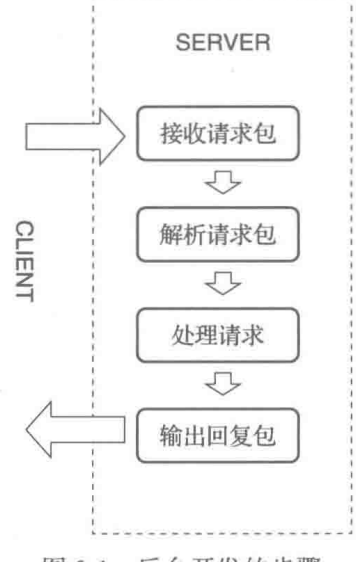


图 0-1　后台开发的步骤

这里的 SERVER 就是本书所指的“后台程序”，或者“服务器”。SERVER 接收请求的方式既可以是通过 TCP 请求包，也可以是 HTTP 请求包（其实也是 TCP 连接）。如果是 TCP 请

求，二进制的格式会常见一些；如果是 HTTP 方式的请求，请求包的格式一般是 JSON 或者 XML，或者自定义的 ASCII 文本。解析请求包的方式自然是与请求包的格式相对应的，接收到的是什么格式的包，就用对应的格式解析（如果是自定义的格式，就按照自定义的方式去解析）。“处理请求”这一步是后台程序的具体业务逻辑的体现。很多封装好的后台框架会把其他三步都做好，但是这一步还是需要开发者自己去实现，因为只有开发者自己清楚，程序是要去做“登录”还是去做“注册”的事情。“输出回复包”和“接收请求包”是对应的，一般来说，收到的是 JSON，那么回复的也是 JSON，收到的是 XML，那么发送的也是 XML，其他格式也是一样的。这四个步骤是所有后台程序都会有的，无论使用什么语言去实现，都可以看到这四个步骤的影子。

CLIENT 指的是向 SERVER 发起请求，并接收 SERVER 回复的一方，通常称为“客户端”。既然后台程序是通过 TCP 或者 HTTP 接收和回复消息的，那么只要是能够发起 TCP 或者 HTTP 连接的都可以作为客户端，可以是浏览器、PC 端的程序、安卓应用、IOS 应用，等等。

0.2 时间就是金钱，效率就是生命

上世纪 80 年代，在改革开放初期的深圳蛇口，为了加快蛇口港的建设，一块“时间就是金钱，效率就是生命”的巨幅标语矗立在蛇口工业区的马路边（如图 0-2 所示），拉开了特区建设的序幕。

图 0-2　1981 年矗立在深圳蛇口工业区的巨幅标语

其实这个口号后面还有两句：“安全就是法律，顾客就是皇帝”，这四句加一起，简直就是当今互联网时代科技公司安身立命的根本。互联网最讲究效率并且一切都从用户体验出

发。在腾讯、百度和阿里这样的互联网公司里，每一次的版本发布都是和时间在赛跑。各种以效率为核心的开发团队合作模式被创造：敏捷开发、极限编程、SCRUM、结对编程，双周迭代，等等。

写下这些文字的时候是我在腾讯工作的第五个年头，这五年让我对效率有了更深刻的认识。还是一个学生的时候，和大家一样，我也曾一字不落地读过《UNIX 环境高级编程》，《UNIX 环境网络编程》一二三卷，《TCP/IP 详解》一二三卷，《C++ Primer》等书籍，这些都是非常经典的开发书籍。它们的共同特点是大而全，不漏掉任何一个知识点，并且每个知识点都讲得非常详细。但在实际的开发工作中，可能用到的知识点只有 20%，其他的 80% 则很少用到。这也是我写这本书的初衷：用最短的篇幅，讲解实际后台开发中用到的核心知识点，让读者可以快速进入到实际的开发工作中。

也许有读者会觉得这很急功近利，不利于组建完整的知识体系。其实，软件开发是一门讲究实操的技术，知道多少并不重要，重要的是能够用好多少。如果把一本经典书籍读 3 遍，但是没有写过一行代码，那可以认为是没有读。边写代码边读书才是最好的学习方式，在读一本技术书籍的时候，最好让自己快速进入写代码的状态，一边写代码，一边通读书籍，在具体需要用到书上某个技术点的时候，再回头仔细阅读相关的章节。在这个循环往复的过程中，才能把书上的知识点转化为自己的知识点。完成多个这样的循环后，再回过头来审视自己已经掌握的知识点，把一些没有掌握的知识点搞清楚。这样的学习过程实际上更有利于完善自己的知识体系。

0.3 后台开发的知识体系

接下来简单介绍一下后台开发的知识体系，也就是说，要完整掌握后台开发，需要掌握哪些知识点，这些知识点也是本书会讲解到的知识点。图 0-3 展示了对后台开发知识体系一个比较全面的梳理。

以上这六部分知识点会是本书将会覆盖的内容。与专门介绍某一类知识的书籍不同，比如《C++ Primer》介绍 C++ 的方方面面，《UNIX 环境编程》介绍 UNIX 环境编程的方方面面，本书从“实战”的角度出发，介绍“后台开发”需要用到的知识和工具。读完本书，读者可能不会对 C++ 精通，不会对 Linux 精通，也不会对 TCP/IP 精通，但是却可以学会如何进行“后台开发”，这些“精通”可以一个个慢慢补全。

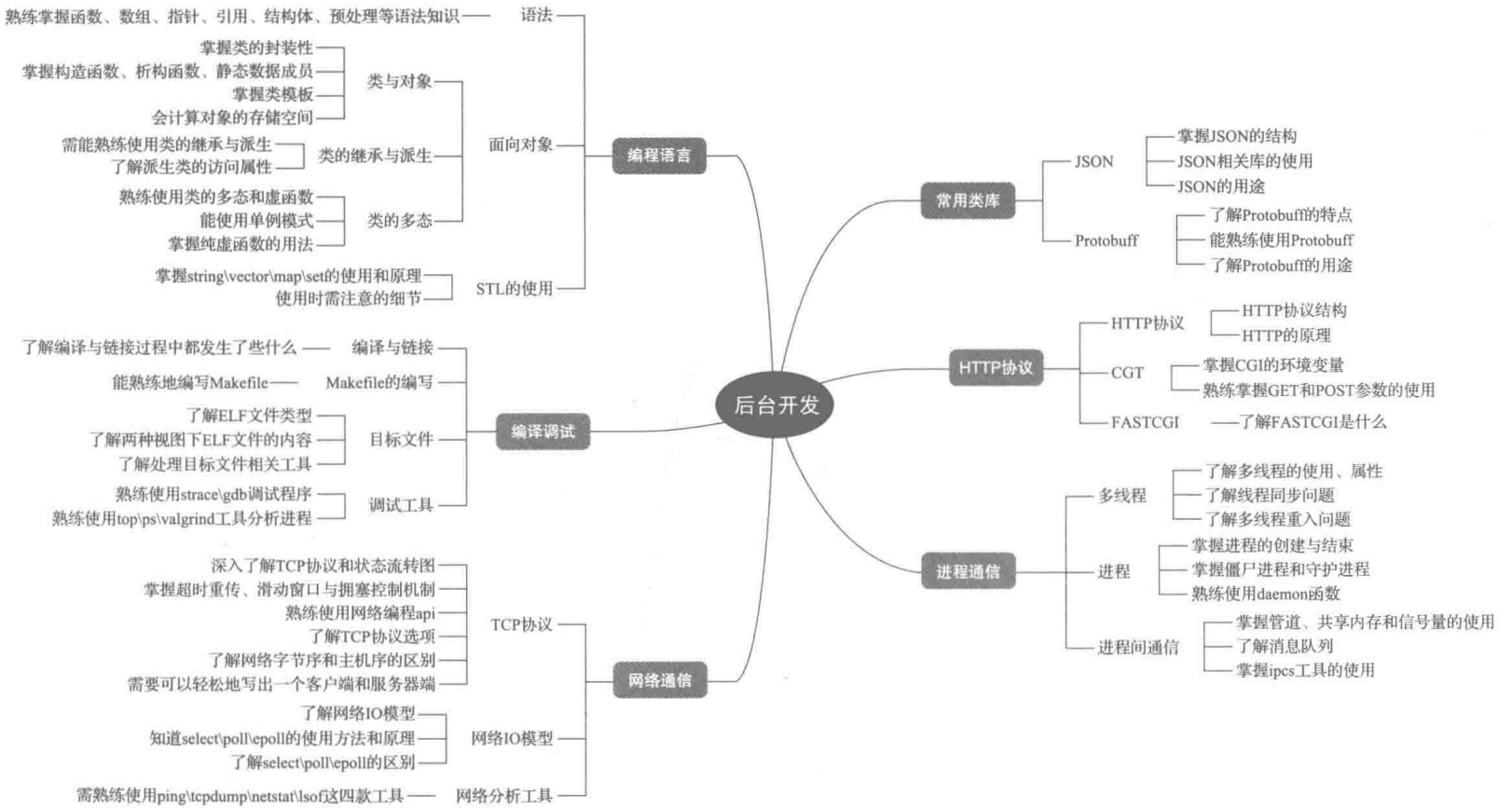


图 0-3　后台开发工程师技术能力体系图

细心的读者可能会发现，编程语言和编辑器只是其中的一小部分，要把后台开发做好，需要掌握的东西比想象中的要多。同时，这些知识点既有一些纯理解性的内容，也有一些工具性的东西。这也是程序开发的最大特点，既要掌握理论的东西，也要掌握相应的工具，这样才能把理论的知识用起来。在实际开发中，这些工具性的知识可能更重要，因为理论是脑子里想的，但工具产生的东西才是真正的产出，才是开发者最终需要的。

0.4 如何阅读本书

本书以 C++ 为编程语言，讲述后台开发的核心技术与应用实践。全书共 13 章，在逻辑上分为以下六部分：

第一部分为第 1 ~ 3 章，主要是编程语言方面的知识，包括函数、函数重载、函数模板、数组、指针、引用、结构体和预处理的使用；面向对象的介绍，包括类的使用、继承与派生和类的多态；常用 STL 的介绍，包括 string、vector、map 和 set 的使用方法与原理。如果读者已经对 C++ 非常了解，可以跳过这部分，也可以配合《C++ Primer》一起阅读。

第二部分为第 4 ~ 5 章，主要是编译原理和调试方法相关的知识。编译原理相关知识包含编译与链接的具体过程，makefile 的编写、目标文件的内容与处理目标文件相关工具的使用；调试方法相关内容主要介绍了用 strace 分析系统调用、用 gdb 调试进程与分析 coredump 文件、用 top 命令分析系统负载情况、用 ps 命令查看系统进程和用 valgrind 工具分析进程的内存使用情况等。

第三部分为第 6 ~ 8 章，主要是网络相关的知识，包括 TCP 协议的关键知识点和 TCP server 的实现，网络 IO 模型和 select、poll 与 epoll 三个重要函数的使用，还有 ping、tcpdump、netstat 和 lsof 这四个网络分析工具的使用。掌握这部分知识，读者可以自己独立实现能处理海量请求的 TCP server。

第四部分为第 9 ~ 11 章，主要是多线程、进程和进程间通信相关的知识，包括多线程的使用、多线程的同步和重入问题，父子进程、僵尸进程、守护进程和进程间通信的方式。读者可以配合《UNIX 环境高级编程》一起阅读。

第五部分是第 12 章，主要是 HTTP 协议的介绍与使用、CGI 的设计原理与实现和 FASTCGI 的简单介绍。掌握这部分知识，读者可以轻松实现 Web 应用的后台交互部分。

第六部分是第 13 章，通过常用类库 JsonCPP 和 Protobuf 的使用，演示如何使用第三方库。

如果读者是后台开发的新手，建议从第 1 章开始阅读，如果读者已经有后台开发的经验，可以直接选择感兴趣的章节阅读。

0.5 勘误和资源

由于水平有限，加之编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。为了更好地与读者交流，我专门创建了一个微信公众号，读者可以通过以下二维码关注，与我进行交流。

书中的全部源代码可以从华章网站（www.hzbook.com）下载。

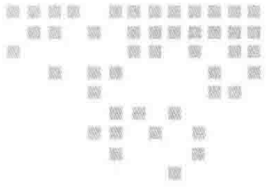
0.6 致谢

首先要感谢腾讯公司，让我可以在后台开发的领域里驰骋。

其次要感谢机械工业出版社华章公司的杨福川、高婧雅和李艺，感谢你们在我写作过程中提供的支持，因为有了你们的鼓励和帮助，我才能顺利完成全部书稿。

谨以此书献给我亲爱的家人，以及热爱软件开发的朋友们！

Contents 目　录



第 1 章 Chapter 1

C++ 编程常用技术

我们通过固定格式和固定词汇的“语言”来影响他人，让他人为我们做事情。语言有很多种，包括汉语、英语、法语、韩语等，虽然它们的词汇和格式都不一样，但是可以达到同样的目的，我们可以选择任意一种语言去与他人交流。同样，我们也可以通过“语言”来影响计算机，让计算机为我们做事情，这样的语言就叫作编程语言。

C 语言是 1972 年由美国贝尔实验室的 D.M.Ritchie 设计成功的，它是为计算机专业人员设计的，大多数系统软件和许多应用软件都是用 C 语言编写的。但是随着软件规模的增大，用 C 语言编写程序渐渐显得有些吃力了。C++ 也是由美国贝尔实验室的 Bjarne Stroustrup 博士及其同事于 20 世纪 80 年代初在 C 语言的基础上开发成功的。C++ 保留了 C 语言原有的所有优点，与 C 语言兼容，并且增加了面向对象的机制。用 C 语言写的程序基本上可以不加修改地用于 C++ 开发工具。从 C++ 的名字可以看出它是 C 的超集。C++ 既可用于面向过程的结构化程序设计，又可用于面向对象的程序设计，是一种功能强大的混合型程序设计语言。

本章主要讲述 C++ 中的常用技术，让读者可迅速地、由浅入深地熟悉这门语言。

1.1 第一个 C++ 程序

刚开始接触一门编程语言，一般会从写一个输出 Hello world 的程序开始。

【例 1.1】 用程序输出 Hello world。

```
#include<iostream>
using namespace std;
int main()
{
```

```
    cout<<"Hello world."<<endl;
    return 0;
}
```

把上述程序编写在一个叫 helloworld.cpp 的文件中，并将它放到 Linux 机器上的某个目录下，执行 g++ helloworld.cpp 命令，会在该目录下生成 a.out 文件。执行 ./a.out 命令，即可得到输出结果：Hello world.。

先看程序的第一行（#include<iostream>），这不是一个 C++ 语句，是一个预处理语句，编译器的预处理器把输入输出流的标准头文件包括在本程序中，所以不需要在句末加分号（；）。include 一个文件，就是把这个文件的所有内容都加进来。图 1-1 展示了包含文件的过程。

如图 1-1 所示，include 一个 .h 文件，就是等于把整个 .h 文件给复制到程序中，include 一个 .cpp 文件也是如此。

除了 #include< > 的方式来包含一个头文件，还会见到 #include" " 的方式来包含一个头文件。而 #include< > 与 #include" " 的区别是：#include< > 常用来包含系统提供的头文件，编译器会到保存系统标准头文件的位置查找头文件；而 #include" " 常用于包括程序员自己编号的头文件，用这种格式时，编译器先查找当前目录是否有指定名称的头文件，然后从标准头目录中进行查找。

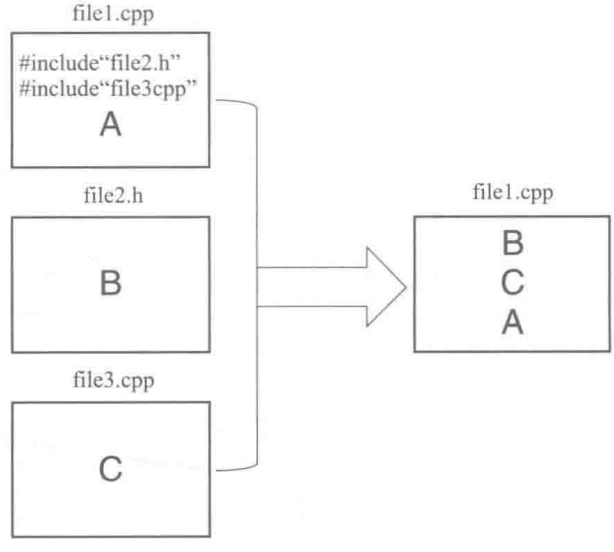


图 1-1　include 文件的原理

还经常会看到 #include<iostream> 和 #include<iostream.h> 的使用。事实上，#include<iostream> 和 #include<iostream.h> 是不一样的，因为 iostream 和 iostream.h 是两个不同的文件，前者没有后缀。实际上，在你的编译器 include 文件夹里面可以看到，两个文件打开后，里面的代码是不一样的。后缀为 .h 的头文件在 C++ 标准已经明确提出不再支持了，早些的 C 语言为了实现将标准库功能定义在全局空间里，声明放在在带 .h 后缀的头文件里。C++ 标准为了和 C 语言区别开，也为了正确使用命名空间，规定头文件不再使用后缀 .h。因此，当使用 <iostream.h> 时，相当于在 C 中调用库函数，使用的是全局命名空间，也就是早期的 C++ 实现方法。换句话说，iostream 是 iostream.h 的升级版，大部分的头文件都有一个不带 .h 扩展名的文件与之相对应。不过有个特例，<string> 并非 <string.h> 的升级版。

再看程序的第二行：“ using namespace std;”中使用了命名空间 std。命名空间是为了让大量类名共存而不至于引起冲突而设计的。C++ 标准函数库的所有元素都被声明在一个命名空间中，这就是 std 命名空间。为了能够访问它的功能，使用这条语句来表达将使用标准名空间中

定义的元素。这条语句在使用标准函数库的 C++ 程序中频繁出现，本书中大部分例子的代码中也将用到它，需要注意的是，最好不要在头文件中使用命名空间，否则容易造成命名冲突。

继续看程序的第三行：“ int main()”，这是主函数 (main function) 的起始声明。主函数是所有 C++ 程序的运行的起始点。不管它是在代码的开头、结尾还是中间，此函数中的代码总是在程序开始运行时第一个被执行。main 后面跟了一对圆括号 ()，表示它是一个函数。C++ 中所有函数都跟有一对圆括号 ()，括号中可以有一些输入参数。如例 1.1 中显示，主函数 (main function) 的内容紧跟在它的声明之后，由花括号 {} 括起来。

程序的第四行：“ cout<<"Hello world."<<endl;” 是本程序中最重要。cout 是 C++ 中的标准输出流（通常为控制台，即屏幕），这句话把一串字符串（本例中为 Hello World）插入到输出流中。cout 在的声明在头文件 iostream 中，所以要想使用 cout 必须将该头文件包括在程序开始处。注意这个句子以分号结尾。分号标示了一个语句的结束，C++ 的每一个语句都必须以分号结尾。C++ 程序员最常犯的错误之一就是忘记在语句末尾写上分号。

最后一行 (return 0;) 中返回语句 (return) 标志主函数 main() 执行结束，并将该语句后面所跟代码（在本例中为 0）返回。这是在程序执行没有出现任何错误的情况下最常见的程序结束方式。在后面的例子中会看到所有 C++ 程序都以类似的语句结束。

1.2　函数

1. 函数的定义

一个 C 程序是由若干个函数组成的，C 语言被认为是面向函数的语言，而 C++ 面向过程的程序设计也沿用了 C 语言使用函数的方法。在 C++ 面向对象的程序设计中，主函数以外的函数大多是被封装在类中的。主函数或其他函数可以通过类对象调用类中的函数。无论是 C 还是 C++，程序中的各项操作基本上都是由函数来实现的，程序编写者要根据需要编写一个个函数，每个函数用来实现某一功能。因此，读者必须掌握函数的概念以及学会使用和设计函数。

定义函数的一般格式是：

```
返回值类型 函数名 ([ 形参 ])
{
    函数体
}
```

在定义函数时函数名后面括号中的变量名称是形参。在主调中调用一个函数时，函数名后面括号中的参数是实参。

【例 1.2】 函数、形参、实参的使用举例。

```
#include<iostream>
using namespace std;
```