

<<C专家编程>>

图书基本信息

书名：<<C专家编程>>

13位ISBN编号：9787115308603

10位ISBN编号：7115308608

出版时间：2013-2

出版时间：人民邮电出版社

作者：范德林登

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<C专家编程>>

### 内容概要

《C专家编程》展示了最优秀的C程序员所使用的编码技巧，并专门开辟了一章对C++的基础知识进行了介绍。

书中C的历史、语言特性、声明、数组、指针、链接、运行时、内存以及如何进一步学习C++等问题进行了细致的讲解和深入的分析。

全书撷取几十个实例进行讲解，对C程序员具有非常高的实用价值。

## 作者简介

Peter van der Linden是一名软件开发工程师及作者。他曾在Sun公司和苹果公司工作多年，并撰写了一系列有关C语言、Java语言及Linux操作系统的技术图书，目前他是摩托罗拉公司的Android技术推广。

## 书籍目录

第1章 C：穿越时空的迷雾 1.1 C语言的史前阶段 1.2 C语言的早期体验 1.3 标准I/O库和C预处理器 1.4 K&R C 1.5 今日之ANSI C 1.6 它很棒，但它符合标准吗 1.7 编译限制 1.8 ANSI C标准的结构 1.9 阅读ANSI C标准，寻找乐趣和裨益 1.10 “安静的改变”究竟有多少安静 1.11 轻松一下——由编译器定义的Pragmas效果 第2章 这不是Bug，而是语言特性 2.1 这关语言特性何事，在Fortran里这就是Bug呀 2.2 多做之过 2.3 误做之过 2.4 少做之过 2.5 轻松一下——有些特性确实就是Bug 2.6 参考文献 第3章 分析C语言的声明 3.1 只有编译器才会喜欢的语法 3.2 声明是如何形成的 3.3 优先级规则 3.4 通过图表分析C语言的声明 3.5 typedef可以成为你的朋友 3.6 typedef int x(10)和#define x int(10)的区别 3.7 typedef struct foo{...foo;}的含义 3.8 理解所有分析过程的代码段 3.9 轻松一下——驱动物理实体的软件 第4章 令人震惊的事实：数组和指针并不相同 4.1 数组并非指针 4.2 我的代码为什么无法运行 4.3 什么是声明，什么是定义 4.4 使声明与定义相匹配 4.5 数组和指针的其他区别 4.6 轻松一下——回文的乐趣 第5章 对链接的思考 5.1 函数库、链接和载入 5.2 动态链接的优点 5.3 函数库链接的5个特殊秘密 5.4 警惕Interpositioning 5.5 产生链接器报告文件 5.6 轻松一下——看看谁在说话：挑战Turing测验 第6章 运动的诗章：运行时数据结构 6.1 a.out及其传说 6.2 段 6.3 操作系统在a.out文件里干了些什么 6.4 C语言运行时系统在a.out里干了些什么 6.5 当函数被调用时发生了什么：过程活动记录 6.6 auto和static关键字 6.7 控制线程 6.8 setjmp和longjmp 6.9 UNIX中的堆栈段 6.10 MS-DOS中的堆栈段 6.11 有用的C语言工具 6.12 轻松一下——卡耐基—梅隆大学的编程难题 6.13 只适用于高级学员阅读的材料 第7章 对内存的思考 7.1 Intel 80x86系列 7.2 Intel 80x86内存模型以及它的工作原理 7.3 虚拟内存 7.4 Cache存储器 7.5 数据段和堆 7.6 内存泄漏 7.7 总线错误 7.8 轻松一下——“Thing King”和“页面游戏” 第8章 为什么程序员无法分清万圣节和圣诞节 8.1 Portzbie度量衡系统 8.2 根据位模式构筑图形 8.3 在等待时类型发生了变化 8.4 原型之痛 8.5 原型在什么地方会失败 8.6 不需要按回车键就能得到一个字符 8.7 用C语言实现有限状态机 8.8 软件比硬件更困难 8.9 如何进行强制类型转换，为何要进行类型强制转换 8.10 轻松一下——国际C语言混乱代码大赛 第9章 再论数组 9.1 什么时候数组与指针相同 9.2 为什么会发生混淆 9.3 为什么C语言把数组形参当作指针 9.4 数组片段的下标 9.5 数组和指针可交换性的总结 9.6 C语言的多维数组 9.7 轻松一下——软件/硬件平衡 第10章 再论指针 10.1 多维数组的内存布局 10.2 指针数组就是liffe向量 10.3 在锯齿状数组上使用指针 10.4 向函数传递一个一维数组 10.5 使用指针向函数传递一个多维数组 10.6 使用指针从函数返回一个数组 10.7 使用指针创建和使用动态数组 10.8 轻松一下——程序检验的限制 第11章 你懂得C，所以C++不在话下 11.1 初识OOP 11.2 抽象——取事物的本质特性 11.3 封装——把相关的类型、数据和函数组合在一起 11.4 展示一些类——用户定义类型享有和预定义类型一样的权限 11.5 访问控制 11.6 声明 11.7 如何调用成员函数 11.8 继承——复用已经定义的操作 11.9 多重继承——从两个或更多的基类派生 11.10 重载——作用于不同类型的同一操作具有相同的名字 11.11 C++如何进行操作符重载 11.12 C++的输入/输出(I/O) 11.13 多态——运行时绑定 11.14 解释 11.15 C++如何表现多态 11.16 新奇玩意——多态 11.17 C++的其他要点 11.18 如果我的目标是那里，我不会从这里起步 11.19 它或许过于复杂，但却是惟一可行的方案 11.20 轻松一下——死亡计算机协会 11.21 更多阅读材料 附录 程序员工作面试的秘密

## 章节摘录

版权页：插图： The SunOS development team is justly proud of our lint—clean kernel. We'd paid a lot of attention to getting the 4.x kernel to pass through lint with no errors, and we kept it that way. When we changed our source base from BSD UNIX to SVR4 in 1991, we inherited a new kernel whose lint status was unknown. We decided to lint the SVR4 kernel. This activity took place over several weeks and was known as the "lint party." It yielded about 12,000 unique lint warnings, each of which had to be investigated and corrected manually. By the end, changes had been made to about 750 source files, and the task had become known as "the lint merge from hell". Most of the lint messages just needed an explicit cast, or lint comment, but there were several real bugs shaken out by the process: Argument types transposed between function and call. A function that was passed one argument, but expected three, and took junk off the stack. Finding this cured an intermittent data corruption problem in the streams subsystem. Variables used before being set. The value is not just in removing existing bugs, but in preventing new bugs from contaminating the source base. We now keep the kernel lint—clean by requiring all source changes or additions to be run through lint and cstyle. In this way we have not only removed existing bugs, but are reducing the number of future bugs as well. Some programmers strenuously object to the idea of putting lint back into the compiler on the grounds that it slows the compiler down and produces too many spurious warnings. Unfortunately, experience has proven repeatedly that making lint a separate tool mostly results in lint not being used. The economics of software is such that the earlier in the development cycle a bug is found, the cheaper it is to fix. So it is a good investment to have lint (or preferably the compiler itself) do the extra work to find problems rather than the debugger; but better a debugger find the problems than an internal test group. The worst option of all is to leave the problems to be found by customers.

## <<C专家编程>>

### 编辑推荐

《C专家编程》是一本C语言经典著作，专家级的C编程指南，展示了最优秀C程序员的编码技巧。  
《C专家编程(英文版)》可以帮助有一定经验的C程序员成为C编程方面的专家，对于具备相当的C语言基础的程序员，本书可以帮助他们站在C的高度了解和学习C++。

### 名人推荐

即使你读过Andrew Koenig的《C陷阱与缺陷》，你还是应该看看Peter Van Der Linden的这本书。我想，他们两人的书你都应该千方百计地搞到，如获至宝地捧读。如果我是你的上司，这是必须的要求。  
——Francis Glassborow，ACCU主席

#### 版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>