《电脑报》实用技术丛书

[美] Peter van der Linden 著
张自力 黄益民 等译

云南科技出版社

让你迅速达到专家级水平的

C 程序设计奥秘

[美]Peter van der Linden 著
张自力 黄益民 王旭仁 陈义红译

云南科技出版社

内容提要

本书是 PvdL 及其在 Sun 公司编译程序与操作系统小组的同事们多年来在 C 程序设计方面的经验总结。作者以生动风趣的笔墨，热情洋溢地向读者介绍了许多鲜为人知的 C 程序设计的技术与技巧，并对令程序设计人员感到困惑的诸多问题进行了解答。

本书避免了一般 C 语言方面的书籍中冗长乏味的讲解，读来让人兴趣盎然。书中提供了一个程序设计人员通常要经过多年的实践方能获得的技巧、线索与捷径，是 C 程序设计人员不可多得的宝贵财富。

本书是 C 程序设计语言的高级教程，可帮助读者快速掌握 C 语言的内在本质及专家级程序设计技巧。

版 权 声 明

Original English Language edition published by Prentice Hall, Inc.

©1994 Sun Microsystems, INC. All Rights Reserved.

本书世界范围内中文版版权，已由美国 SunSoft, Inc. 独家授予西南师范大学电脑应用科技开发公司。未经许可，不得以任何形式和手段复制或抄袭本书内容。

C 程序设计奥秘

[美]Peter van der Linden 著

张自力 黄益民 王旭仁 陈义红译

责任编辑：刘正荣

封面设计：李光宇

出　　版：云南科技出版社出版

排　　版：电脑报社照排部

印　　刷：重庆电力印刷厂印刷

开　　本：787 × 1092 毫米　1/16

印　　张：15.5　　字数：300 千字

版　　次：1998 年 6 月第 1 版第 1 次印刷

书　　号：**ISBN** 7 - 5416 - 1135 - 2/**TP**·31

定　　价：18.00 元

献　辞

我特将本书献给比萨饼、有黑白斑点的短毛狗、吊床中的星期日下午和喜剧作品。如果这些东西多一些，这世界会更好一点，我打算在本书完成之后重新认识他们。

实际上，我想下周日的下午就躺在吊床上摇拽，并嘲笑我的短毛狗试图吃比萨饼。

我还要感谢英国约克郡的 Theakston Brewing 公司的上乘之作。

译者前言

在美国，有两本倍受读者青睐的C语言书籍，一本是Kernighan和Ritchie所著的《C Programming Language》，一本便是本书——《C程序设计奥秘》。前者能让你迅速、全面地掌握C语言的基本知识，而后者则可以使你在C语言方面的程序设计能力迅速提高，达到专家级水平。

的确，只要你上书店看看，便可知C语言方面的书种类不少，但许多都单调乏味。程序设计是一项奇妙的、充满活力、富有挑战性的活动，有关程序设计的书籍也就应该热情洋溢！本书作者通过将乐趣(fun)置于功能(function)中的方式，使得本书不仅富有教育意义，同时也生动有趣。

本书介绍了许多鲜为人知的C程序设计技巧与技术，以及一些有趣的小故事。译者将此书奉献给广大程序设计人员，衷心希望能与大家一起分享程序设计的乐趣！

本书第1－4章由黄益民翻译，前言、引言及第7、8章由张自力翻译，第5、6章由陈义红翻译，第9、10、11章由王旭仁翻译，鞠林翻译了第6章的部分内容。全书由张自力、黄益民负责统稿与审定。

本书原文生动、诙谐，译者虽竭尽全力以保持原来的风格，但限于水平，不妥之处肯定存在，诚请读者批评指正。

译 者

1998年6月于重庆

此为试读，需要完整PDF请访问：www.ertongbook.com

前 言

最近到书店里逛逛,看到许多枯燥无味的 C 和 C++教材,真令我失望。很少有作者表达这样的思想,即任何人都应享受程序设计的乐趣。所有奇妙之处都被冗长且单调乏味的段落所挤掉。如果你能长时间保持清醒的头脑以阅读的话,这样的书籍或许是有用的,但程序设计本身并非如此。

程序设计是一项奇妙的、充满活力、富有挑战性的活动,有关程序设计的书籍也就应该热情洋溢!通过将乐趣(fun)置于功能(function)中的方式,使得本书不仅富有教育意义,同时也生动有趣,如果这并非你所喜欢的,则请将此书放回书架,只是拜托放到一个更显眼的位置。

好的,既然我们已有一打一打的关于 C 程序设计的书籍,那么这一本又有什么独特之处呢?

《C 程序设计奥秘》应该是每一个程序员在 C 语言方面的第二本书。书中绝大部分内容、技巧和技术,在任何其它书中都找不到。如果这些内容有记载的话,通常也是在有经验的程序员的手册中的边缘空白处,或老的打印结果的背面用铅笔标注的。这些知识是作者及在 Sun 公司编译程序与操作系统小组的同僚们通过多年的 C 程序设计而积累起来的,有许多 C 方面的有趣故事和民间传说,如与 Internet 相连的自动售货机,在外层空间的软件问题,以及 C 程序中的小错误如何使 AT&T 的整个长途电话网络处于瘫痪状态等。最后一章是有关 C++的简明教程,以帮助你掌握这种日渐流行的 C 语言分支。

本书适用于 ANSI 标准 C,这在许多 PC 和 UNIX 系统中均可找到。对只在典型的 UNIX 平台中方能找到的、与复杂硬件相关的 C 语言的独特方面(如虚拟存储等),本书也有详尽阐述。对 PC 机内存模式和 Intel 8086 系列对 C 代码的影响,书中也进行了全面的描述。已掌握了 C 语言基本知识的人将会发现,本书提供了一个程序设计人员通常要经过多年的实践才能获得的技巧、线索与捷径,包含了令许多程序设计人员感到困惑的论题:

- typedef struct bar {int bar} bar;的实际意义是什么?
- 可以向同一函数传递不同大小的多维数组吗?
- extern char * p;与另一文件中的 char p[100]为什么能或为什么不能匹配?
- 什么是总线错误?什么是分段冲突?
- char * foo[ ]和 char(* foo)[ ]之间的区别是什么?

如果你对上面的某些问题尚不肯定,而又想知道 C 程序设计的专家是如何处理的,则请继续阅读!如果你已经知道上述所有的事情及有关 C 语言的其它每件事,则无论如何也应拥有本书以增强你的知识,并告诉书店职员:你是帮朋友买的。

PvdL 于加州硅谷

致　　谢

这不是你在大多数其它书中所看到的没有多少用处的致谢部分：对作者曾经向他借过钱的人，从小学开始的伙伴，一直到配偶的亲戚等的一串无力的赞扬之词，最后以卑躬屈膝但又明显的向其导师拍马屁的话语结束（“最后感谢伟大、权威的某某教授，他关于卫生纸应挂于卷轴的前面还是后面的问题的研究已经完成，从而解决了这一极其困难的问题”）……本书的致谢绝非如此！这里列出的名单是名符其实的，这些都是在我写作本书过程中确实真诚帮助过我的人。实际上，列于此的每个人都已赢得了社会的承认。并且你也可以断定，当我在塔西提岛的海滨消闲，花费这丰厚的稿酬时，我会想起他们，我真的会想他们！

首先，我要特别感谢 Phil Gustafson 和 Brian Scearce 所给予的帮助，他们阅读了整个手稿并提出了许多改正和改进意见。他们是如此的认真、努力，以致于他们现在都立志献身科学。

我也要感谢阅读了大部分清样的朋友和同事们：

Lee Bieber，

Keith Bierman（他名片上的头衔是“煽动者”，因此他应是这工作最合适的人选。），

Robert Corbett，Rod Evans，Doug Landauer，

Joseph McGuckin，Walter Nielsen，

Charlie Springer（他教我用手指进行二进制计数——这样可以数到 1023），

Nicholas Sterling，Panos Tsirigotis，

Richard Tuck，他们阅读了部分手稿，并坦率地阐明了他们的观点。

我非常感谢那些总是帮助我，通常是耐心地回答一连串无休止问题的人们：

Chris Aoki，Arindam Banerji，

Mark Brader，

Brent Callaghan，

David Chase，Jaseph T. Chew，

Adrian Cockcroft，Sam Cramer，

Steve Dever，Derek Dongray，

Joe Eykholt，Roger Faulkner，

Mike Federwisch，Dave Ford，

Sun 德国公司的 Burkhard Gerull，

Rob Gingell，

Cathy Harris（感谢提供丰富的常识），

Bruce Hildenbrand（和他那惊人的自行车飞车技艺），

Mike Kazar，Bob Jervis，Diane Kelly，

Charles Lasner，Bil Lewis，

Greg Limes, Tim Marsland,

Marianne Mueller,

Eugene N. Miya, Chuck Narad,

Bill Petro(感谢他令人鼓舞且不间断的历史课程),

Trelford Pinkerton, Alex Ramos,

Fred Sayward, Bill Shannon,

Mark D. Smith, Kathy Stark,

Dan Stein, Steve Summit,

Paul Tomblin,

Wendy van der Linden(他提出了用于 C++ 的咬住苹果(一种游戏——译者注)的继承实例,并改进了 Null 是有两个 L 的 Null 的说法。

Dock Williams,

Nigel"Gag Me" Witherspoon,

Brian Wong, Tom Wong.

我要感谢 Karin Ellison 编辑,她允许我使用前后不一致的隐喻,且好几次为了我而三更半夜去平息风波;感谢 Astrid Julienne,他回答了许多有关 Framemaker 软件的问题,感谢 Sun 公司图书馆的 Peter Van Coutren.

感谢 Prentice Hall 的 Mike Meehan、Camille Trentacoste、Susan Aumack、Eloise Starkweather 和 Nancy Boylan,感谢他们富有见地的帮助。

还要感谢 Dirk Wibble - O'Dooley 和 P.A.G. Embleton 等人,他们在我写作本书的过程中,未曾惹恼我,并没有把事情弄得太糟。我猜想他们属于不错的那种人。

书中的某些资料是通过交谈、e - mail、网络公告及业界友人的建议而产生的。对所知道的出处都作了说明,但如果疏忽忘了谁,还请原谅。

PvdL 于加州硅谷

引 言

C代码，C代码运行，请像C代码运行一样飞奔向前……

——Barbara Ling

所有的C语言程序做同样的事情：查找一个字符且不对其作任何处理。

——Peter Weinberger

不知你是否曾注意到，大量的C语言方面的书都具有诸如《C的陷井与圈套》或《C语言疑难问题集》或《令人困惑的C及其它难以理解的问题》等这样的书名，而其它的程序设计语言却没有像这种书名的书。这当然是有充分理由的。

C程序设计是一项需花多年时间才能熟练掌握的技巧。一个比较聪明的人能够很快学会C语言的基本知识，但得花较长的时间才能掌握C语言中的多种细微差别，才能写出足够多的程序——足够多的不同的程序，之后才能成为C程序设计的专家。用句通俗的话来说，这是能够在巴黎叫一杯咖啡与在地铁上能够告诉一个地道的巴黎人在何处下车之间的差别。这是一本关于ANSI C程序设计语言的高级教程，读者对象是那些已经会编写C语言程序，而又想尽快掌握其内在本质及专家级程序设计技巧的人们。

专家级程序员经过多年的实践，建立起了一个工具仓——一个含有习惯用法、代码段及熟练技巧的百宝囊。这些经验是随时间的推移慢慢积累起来的，有些是从更有经验的同事那里学来的，而有些是直接阅读或在维护他人所写的程序代码时获得的，其它的一些则是自己摸索出的。几乎每一个初学C语言的程序员都可独立地再现将

if(i = =3)写成

if(i =3)的错误。

这种令人痛苦的错误(试图进行比较却进行了赋值)一旦经历过一次便很少重犯。有些程序员已养成了先写常数的习惯，如if(3 = =i)。这样，如果偶然少写一个等号，编译程序就会报错："试图给常量赋值"。这虽不能解决在进行两个变量比较时的问题，但多少还是有点帮助。

价值2000万美元的程序错误

1993年春，在SunSoft公司的操作系统开发小组，我们得到一份描述异步I/O库中问题的

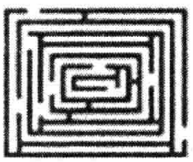
“第一优先权”的错误报告。这一错误使得向一客户销售价值2000万美元的硬件设备的合同执行受阻,因该客户明确指出需要此库程序能正常工作,所以我们强烈希望能够找出这一错误。在经过了一些专门的调试之后,终于跟踪到这一问题出自一条指令:

```
x = =2;
```

这是一个打印错误,程序员的手指压到”=”键上,偶然按了两次而不是一次,使得本为赋值语句变成了比较。这一语句写成了x与2比较,产生真或假值,并将结果弃掉。

C是这样的一种表述语言:编译程序对只对表达式求值,不产生副作用而又简单地将结果抛弃的语句不会作出任何反应。我们不知道是该庆贺我们运气好,找到了这一问题呢,还是灰心地哭述:如此常见的打印错误竟会引起如此代价昂贵的问题。某些版本的lint可能可以检查出这一问题,但因问题太简单而使人们不能自免这一基本工具。

本书收集了许多其它有益的故事,记录了众多经验丰富的程序员的智慧,从而使读者不必去重新独立发现每一问题。本书可用作在C语言程序设计这一块广漠的土地上纵横驰骋的指南,当然,尚有一些未探索过的角落没有包含在本书中。本书对一些主要的论题,如说明,数组/指针,及与之相关的许多线索与助记等进行了广泛的讨论。书中通篇使用ANSI C的名词术语,在必要的地方还翻译成了常规的英语(汉语)。



编程挑战

或

软件信条

示例框

在后面的讲述中,我们采用类似于此的标有”程序设计挑战”的方框。

框中编写程序的有用建议。

也有标有”手边的启发”的方框,框中内容自成体系,是一些在实际工作中有用的思想、经验规则或准则。你可在你自己的工作中采用之。如果你已有你自己更加喜欢的准则,则请忽略。

约定

一个约定是,我们使用水果和蔬菜名来命名变量(当然仅限在小的程序代码段,而非任何实际程序中):

```
char pear[40];
double peach;
int mango = 13;
```

long melon = 2001;

这样便于区分哪些是C语言的保留字,哪些是程序员提供的变量名。有人或许会说你不能将苹果和橙进行比较,但为什么不可以呢——它们两者都是长于树上,一只手可以握住,圆的可食用的东西。一旦你习惯了这种约定,水果方法确有帮助。另一约定是——有时重复一关键点以强调之。再说一遍,我们有时重复一关键点以强调之。

《C程序设计奥秘》象一本烹调书,收集了一些美味佳肴的菜谱,供读者一试身手。每一章均划分成既有联系又自成体系的几部分,这样使得从头至尾阅读本书,或随机挑选其中的一部分来阅读并仔细复习一单独的论题,都十分方便。所有技术细节均有许多C程序设计在实际中如何运用的真实故事点缀其间。幽默是掌握新东西的一项重要的技术,因此每章结束均附有"轻松一下"的一节,此节中含有令人惊讶的C语言故事或一段软件的民间传说,以便读者调节一下阅读的进程。

读者可将此书用作思想的源泉、C程序设计技巧与惯用法的手册,或用作从经验丰富的C语言编译程序编写者那里学习更多的有关ANSI C知识的资料。总之,本书收集了许多有用的思想,以帮助你掌握ANSI C这一程序设计艺术。本书将所有的信息、线索和准则收集整理在一起,呈现给你,依你的爱好自行排选。现在,抓一个信封将它翻过来,拿起你幸运的编程铅笔,坐到一个舒适的终端前,开始这快乐的游戏!

轻松一刻——调整文件系统

C和UNIX的某些方面偶尔也很随意,这些合理安排的奇怪想法本身并没什么不对。由IBM/Motorola/Apple公司联合开发的PowerPC芯片的系统结构中有一条E.I.E.I.O(Enforce In_order Execution of I/O——I/O的强制有序执行)指令。基于类似的精神,有一条UNIX命令tunefs,复杂系统的管理员可用这条命令改变一个文件系统的动态参数,以改善磁盘上块的布局。

与所有的伯克利UNIX命令类似,原始tunefs命令的联机手册上以一个"错误"部分结束。在这里,错误是:

Bugs:

This program should work on mounted and active file systems, but it doesn't. Because the superblock is not kept in the buffer cache, the program will only take effect it is run on dismounted file systems; if run on the root file system, the system must be rebooted. You can tune a file system, but you can't* tune a fish.

错误:

本程序应工作于已安装且活动的文件系统,但实际未如此。由于控制块未保留在高速缓冲存储器中,此程序只在其运行于已卸下的文件系统时有效;如果运行于根文件系统,则系统将重新引导。你可以调整一个文件系统,但不能调整一条鱼。

更有甚者,文档中有一条评论,吓坏了删掉其中最后一短语的人!此评论是:

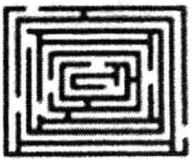
Take this out and a UNIX Demon will dog your steps from now until the time_t's wrap around.

删去这一短语则UNIX的守护神将从此跟踪你,直到UNIX中的计时程序time_t轮回。

*在英语中有"tuna fish"(金枪鱼肉),"tune a fish"取其谐音——译者注。

此为试读,需要完整PDF请访问: www.ertongbook.com

当 Sun 及整个世界改变到 SVR4 UNIX 后，便丢掉了这一奇怪想法。SVR4 的手册中不再有“错误”一节——已被更名为“注”(这是否在愚弄人?)。“金枪鱼肉”短语也消失了，且那些内疚的参与者也许被 UNIX 的守护神跟踪至今天。



编程挑战

计算机的日期计算

---

time _ t 何时会轮回?

请写一个程序求出它。

1.查看 time _ t 的定义，此定义在文件/usr/include/time.h 中。

2.编写一程序将最大值赋给 time _ t 类型的变量，然后将其传给 ctime( )转换成 ASCII 码串，打印此串。注意，ctime 与 C 语言无任何关系，只是“转换时间”之意。

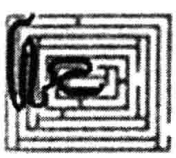
删掉前述评论的不知名的技术人员，将会担心被 UNIX 的守护神跟踪多少年? 修改你的程序求出之。

1.通过调用 time( )获得当前时间

2.调用 difftime( )以获得现在与 time _ t 的最大值之间的秒数。

3.将秒数转换成年、月、日、星期、天、小时、分的格式并输出。

比你预期的一生长吗?



编程解答

计算机的日期计算

此练习的结果将随 PC 机和 UNIX 系统的不同而变化，并依赖于 time _ t 的存储方式。在 Sun 系统中，此即用于长整型的一种类型定义(typedef)。第一个试探性解答如下：

```
# include <stdio.h>
# include <time.h>
int main( )      {
time _ t biggest = 0x7FFFFFFF;
printf("biggest = %s \ n",ctime(&biggest));
return 0;
}
```

此程序运行结果为：

```
biggest = Mon Jan 18 19:14:07 2038
```

但是，这并不是正确答案! 函数 ctime( )将参数转换成当地时间，而当地时间是依据你

在地球上的位置随统一协调时间(又称为格林威治平时)变化的。在本书写作的地方——加利福尼亚,比伦敦时间晚 8 小时,因而也就早几年。

我们实际应该用函数 gmtime( )来获取统一协调时间的最大值。此函数返回的不是一个可打印的字符串。因此再调用 asctime( )来获得可打印的串。将两者合在一起,可得如下修改后的程序:

```
# include < stdio.h >
# include < time.h >
int main(){
time _ t biggest = 0x7FFFFFFF;
printf("biggest = %s \ n",asctime(gmtime(&biggest)));
return 0;
}
```

此程序运行结果为:

```
Biggest = Tue Jan 19  03:14:07 2038
```

瞧! 又多挤出 8 个小时!

但我们还未完成这件事。如果地点是在新西兰,假定在 2038 年采用的是夏令时,则还会多 13 小时。由于新西兰处在南半球,因而在 1 月份开始夏令时。因为新西兰处在时区的最东边的位置,因此,它是在特殊日期会遇到程序问题的第一个国家,真是不幸。

看似简单的事情在软件中有时也会引起令人吃惊的混乱。认为对日期编程是一件容易的事情,一次就可以搞定的人,可能表明他(她)在这方面尚无多少经验。

目录

此为试读,需要完整PDF请访问: www.ertongbook.com