WILEY

（第3版）

Assembly Language Step-By-Step: Programming with Linux 3rd Edition

汇编语言：基于Linux环境

Jeff Duntemann 著 梁晓晖 译

清华大学出版社

汇编语言：基于 Linux 环境(第 3 版)

(美) Jeff Duntemann 著

梁晓晖 译

清华大学出版社

北 京

内 容 简 介

本书是风靡美国的经典汇编语言畅销书籍的最新版，美国计算机领域著名作者Jeff Duntemann的力作。作者以其渊博的专业知识，丰富的实战经验，结合生动详尽的实例，全面系统地介绍了Linux环境下如何使用汇编语言进行程序设计以及与之有关的背景知识和相关工具的使用。本书写作风格独特，全书采用作者最有特色的对话式风格，结合大量源于生活的暗喻，将晦涩难懂的知识点条分缕析地呈现出来，以便读者能以轻松愉快的心情学习。

本书适合刚涉足Linux环境下汇编语言的读者，也可作为相关技术人员的参考书。

北京市版权局著作权合同登记　　图字：01-2010-1116

图书在版编目(CIP)数据

汇编语言：基于Linux环境(第3版)/(美)达特曼(Duntemann, J.)著；梁晓晖译. --北京：清华大学出版社，2014
书名原文：Assembly Language Step-by-Step: Programming with Linux
ISBN 978-7-302-34592-3

Ⅰ. ①汇… Ⅱ. ①达… ②梁… Ⅲ. ①汇编语言—程序设计 ②Linux操作系统 Ⅳ. ①TP313 ②TP316.89

中国版本图书馆CIP数据核字(2013)第284546号

责任编辑：文开琪
装帧设计：杨玉兰
责任校对：王　晖
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62791865

印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×230mm　　印　　张：36.75　　字　　数：675千字
版　　次：2014年2月第1版　　印　　次：2014年2月第1次印刷
印　　数：1～3000
定　　价：99.00元

产品编号：034967-01

序

汇编网技术专家组

信息已经充满我们这个时代的每个角落，先放下分辨有效信息还是无效信息这件事，我们马上就能看到我们获得信息的手段、数量和速度都不断迈入新的量级。在这背后提供支撑的是数以亿万计的软件和硬件，实现这些软件和硬件的技术已经形成一个庞大的领域，这个领域就是计算机科学与技术领域。成千上万的喜欢、热爱这个领域的工作者奉献着自己的绵薄之力，对这个领域进行不断的创新和改变。我们看到，这份绵薄之力，已经形成移动互联新时代里声势浩荡的华彩乐章。

有一门语言，是任何一名想在计算机科学与技术领域发展的人必须掌握的，这门语言就是汇编语言；有一门技术，是任何一名想在计算机科学与技术领域发展的人所必须熟悉的，这门技术就是应用汇编语言的技术。我们汇编网致力于帮助学习者更好地学习与应用汇编语言，并将其作为一件重要的事去实现。

这本书是美国计算机科学与技术领域中的一本经典技术著作，很高兴清华大学出版社能够出版这本书的中文版。这一举措，为我国相关领域的读者提供了更丰富的学习方式和技术内容。同时，我们汇编网也决定以我们所持有的教学理念为前提，通过汇编网(www.asmedu.net)这个公益空间与平台，向这本书的读者提供相关的学习支持。让我们与出版社、汇编网、广大读者一起，共同为我国相关领域的发展贡献我们的力量。

以此为序。

译者序

梁晓晖

Linux 自问世以来，受到广大计算机爱好者的喜爱，主要原因有两个。一是它属于自由软件，用户不用支付任何费用就可以获得它和它的源代码，并且可以根据自己的需要对它进行必要的修改，无偿使用，无约束地继续传播。另一个原因是，它具有 Unix 的全部功能，任何使用 Unix 操作系统或想要学习 Unix 操作系统的人都可以从 Linux 获益。

当前，介绍 Linux 下汇编编程的书籍相对较少，而能够堪称经典且在美国畅销不衰的书籍更是屈指可数，本书即为其中最引人瞩目的一本。本书的作者 Jeff Duntemann 曾经使用汇编语言开发过大量的软件，具有丰富的实践经验，并撰写过多本关于汇编语言编码和优化的书籍。我希望这本书也能为我国计算机领域的读者的学习和工作贡献力量。

对于作者渊博的知识和极度认真的态度，以及为了以最容易让人们接受的方式来阐明一个知识点所花费的良苦用心，我深表敬意。

而我感触颇深的是全书独特的写作风格。Jeff Duntemann 完全采用生活化的语言，以轻松的对话方式对 Linux 汇编编程的相关知识以及学习方法进行了生动细致的讲解。全书中有大量的基于生活的暗喻，为读者更好地理解相关知识提供了多个视角以及更低的门槛。以生活化的语言来撰写一度令人觉得枯燥乏味的底层计算机语言类技术书籍，本书的确另辟蹊径。

作为畅销书的新版本，在保持上一版精华内容的同时，本书进行了大量修订。全书从相关软件基础(如计算机数制系统)，到软件编码思想(如利用过程和宏解决程序的复杂性)，以及后续扩展(如 C 库函数调用)等方面，对 Linux 下汇编语言编程等进行了全方位的介绍，为学习者尤其是初学者奠定了坚实、广泛的基础。

在翻译过程中，感谢我的家人对我的诸多关怀和帮助，在此尤其要感谢我的婆婆，要不是她老人家为我腾出宝贵的时间，本书的翻译工作恐难如期完成！

还要感谢清华大学出版社的各位编辑，是她们为本书的完成提供了大量宝贵的指导和建议，保证了本书的顺利出版！

由于水平有限，翻译中有不妥或错误之处在所难免，敬请广大读者批评指正。

前言

“你为什么要做那件事情？”

那是 1985 年，我和其他几个焦躁不安的媒体大腕一起，乘坐一辆出租车去纽约参加记者招待会。那时我刚刚开始媒体生涯(作为 *PC Tech* 杂志的技术编辑)，我的第一本书还有几个月就要出版。我碰巧坐在一个已经成名的编程栏目的作者/大师旁边。我们谈论了很多这样那样的事情，他给我留下了深刻的印象。我不会说出他的名字，因为他已经在这个领域做了很多事情，如果他不把戒烟当成第一要务的话，肯定会作出更多的贡献。

但是，碰巧我是一个 Turbo Pascal 的狂热爱好者，我真正想做的事情是要学习如何编写能够使用全新的 Microsoft Windows 用户界面的 Turbo Pascal 程序。在谈到下面这个臭名昭著的问题之前，他皱了皱鼻子，做了个鬼脸苦笑道：

“你为什么想做那件事情呢？”

我以前从来没有听到过这个问题(尽管后来不止一次地听到它)，所以有点猝不及防。为什么？因为，嗯，因为……我想知道它的工作原理。

“嗨！那是 C 语言要做的事情。”

更深一层的讨论使我彻底对 Pascal 迷失了方向。但是通过一番寻根究底之后，我了解到，不能用 Turbo Pascal 编写 Windows 应用程序。这样做是不可能的。或者这个编程栏目作者/大师根本不知道怎么做，或者二者兼而有之。我不知道真相属于哪一种，但是我的确知道了那个臭名昭著的问题的含义。

各位看官，注意：当某些人问你“为什么你想做那件事情？”，它的真实含义是：“你在问我如何做一件或者通过使用我喜欢的工具无法实现，或者彻底位于我的经验之外的事情，但是我又不想承认这一点，以免丢脸。所以……如何和那些黑鹰队较量？”

多年以来，我一遍又一遍地听到这句话：

问：如何创建一个不用扫描就能读出它的长度的 C 字符串？

答：你为什么想做那件事情？

问：如何用汇编语言编写一个能够在 Turbo Pascal 中调用的子程序？

答：你为什么想做那件事情？

问：如何用汇编语言编写 Windows 应用程序？

答：你为什么想做那件事情？

你明白了吧。对于这个臭名昭著的问题的答案通常是一样的，如果哪个狡猾的人这样问你，你应该立即以最快的速度反驳道：*因为我想知道它的工作原理。*

这是一个很不充分的回答。我每次回答这个问题都用这个答案，但是有一次例外，那是很多年以前，我宣布要写一本教那些初次接触编程的人们如何用汇编语言来编写程序的书。

问：天哪，你为什么想做那件事情？

答：因为要想具备理解所有其他编程体系工作原理的本事，这是现有的最佳方法。

对于程序员而言，有一件事情高于一切，那就是理解事情的工作原理。而且，学习如何成为一名程序员的过程几乎就是一个掌握事情工作原理的过程。这可以在各个级别上完成，取决于你所使用的工具。如果你正在用 Visual Basic 编程，你必须理解某些事情的工作原理，但是这些事情总体来说局限于 Visual Basic 本身。Visual Basic 在程序员和计算机之间放置了一个“层”，这个层隐藏了很多机制(同样的情况适用于 Delphi、Java、Python 以及很多其他的高级语言编程环境)。如果你在使用 C 编译器，那么你距离计算机更近了一些，你将看到更多的机制——而且必须看到，只有了解了这些机制的工作原理才能够正确地使用它。然而，即便如此，还有一些内容是隐藏的，哪怕是对于那些老练的 C 程序员而言。

反过来，如果你在使用汇编语言，那么你与计算机的距离就最大限度地缩短了。汇编语言什么都没有隐藏，而且也没扣留任何权利。当然，从另一方面来讲，由于在你和机器之间没有任何神奇的“层”来替你“照顾”某些事情，所以再无法为自己的无知找到任何托词。如果不知道事情的工作原理，那么你死定了——除非你拥有的知识够多，能自己把它弄明白。

我创作本书的目的不完全是为了教你汇编语言本身，这是关键所在。如果说本书有一个根本的指导思想，那就是：引入一种受过训练的对于机器的好奇心，以及一些基本的上下文，根据这些上下文，你能够从最底层开始探索计算机，并具备竭尽所能掌握它的信心。这是一件困难的事情，但是只要专注些，耐心些，没有什么你掌握不了的，而且只要有时间(我得提醒一下)，可能相当长。

事实上，这里我真正想教给你的是如何学习。

你需要准备些什么

要想按照我教的方式编写程序，你需要一台运行 Linux 的基于 Intel x86 的计算机。本书的正文和示例都假设至少运行在 386 上，但是因为 Linux 本身至少需要 386 环境，所以你已经满足硬件要求了。

你需要熟悉使用 Linux。我不能在本书中教你如何安装和运行 Linux，不过在某些不太直观易懂的情况下，我会提供一些暗示。如果你对 Linux 环境还没有熟悉，我建议你找本教程，借助它来进行工作。这样的书很多，但是我最喜欢的是 William von Hagen 撰写的 *Ubuntu 8.10 Linux Bible*(*Linux for Dummies* 虽然不错，但是内容不够全面)。

到底使用哪个 Linux 版本并不是很重要，只要至少基于 2.4 版的内核即可，当然，2.6 版更好一些。我用来编写本书示例程序的版本是 Ubuntu 8.10。哪一种图形用户界面并不重要，因为所有的程序都被编写为运行在纯文本 Linux 控制台下。汇编程序本身，即 NASM，也是纯文本。

我在讲解编程逻辑的时候以 Kate 编辑器为模型，它需要用到 GUI。实际上你可以使用任何你想用的编辑器。并不是程序本身需要 Kate 编辑器，而是如果你刚刚开始接触编程，或者经常使用高级语言的特定编辑环境的话，Kate 编辑器是一个很好的选择。

我在正文中引用的调试器是 Gdb，但是大多数情况下是通过它的内置 GUI 前端——Insight 来使用它。Insight 需要一个有效的 X Window 子系统，但是并不绑定到某一特定的 GUI 系统上，如 GNOME 或 KDE。

你不必事先知道如何安装和配置这些工具，因为我会在合适的时候，在相应的章节里涵盖所有必要工具的安装及配置。

注意，对于其他不是基于 Linux 内核的 Unix 实现而言，其功能的执行方式可能与基于 Linux 内核的不完全相同。例如在调用内核时，BSD Unix 就使用不同的规则。其他一些 Unix 版本(如 Solaris)我并不擅长。

学习计划

本书从基础开始讲起，我的意思是“零起点”。也许你已经入门，或者已略有心得也没关系。我对待起点问题非常慎重。我仍然相信，依次阅读本书各章并非浪费时间。复习很管用，嘿！你会发现，你了解的知识并不像想象中的那么多

(我经常这样)。

但是，如果你的时间很不充裕，下面这个学习进度安排表可以让你偷点儿懒。

1. 如果你已经理解了计算机编程的基本概念，可跳过第 1 章。
2. 如果你已经理解了除了十进制之外的其他数制(尤其是十六进制和二进制)背后的思想，可跳过第 2 章。
3. 如果你已经对计算机内部原理(内存、CPU 结构等)有了一些了解，可跳过第 3 章。
4. 如果你已经理解了 x86 内存寻址机制，可跳过第 4 章。
5. 不。停！勾掉它(第 4 点)。即使你已经理解了 x86 内存寻址机制，也请阅读第 4 章。

强调一下，这里的第 5 点是出于这样的原因：汇编语言编程是关于内存寻址的编程。如果不理解内存寻址机制，学习所有其他关于汇编的知识将毫无用处。所以，不要跳过第 4 章，无论你了解了或者认为了解了其他知识与否。从这一章开始，一直通读到本书的结尾。加载每一个示例程序，汇编每一个示例程序，运行每一个示例程序。力求理解每一个程序中的每一行代码。对什么内容都不要无条件地相信。

而且，不仅限于此。当你理解了知识之后，修改那些示例程序。试着采用不同的方式完成它们。勇于尝试我没有提到的那些内容。大胆些。这样还不够，最好更疯狂一些——这些二进制位是没有感情的，可能发生的最糟糕的事情无非就是 Linux 抛出一个分段错误，它将会破坏你的程序(也许会伤害你的自尊)，但是并不会破坏 Linux(“保护模式”不是白叫的！)。唯一可能存在的困难是：当你在尝试某些知识的时候，要了解程序不工作的原因，而且要像你了解所有它做的其他事情一样的清晰。然后将其记录下来。

最后是我追求的目标：教给你一种方法，通过这个方法，你可以理解计算机的每一个组成部分正在做的事情，以及它们是如何协同工作的。这并不意味着我本人会讲解计算机的每一个组成部分——即使我活得再长，也做不到。计算过程已经不再像以前那样简单，但是如果具备了耐心研究和实验的品质，你很有可能自己把它完成。最后，学习的唯一方法是：一切要靠自己。不能否认，你能够找到一些指导性内容——来自朋友的，来自互联网的，来自像本书一样的书籍的，但那仅仅是指导，车轴上的润滑剂而已。你必须搞清楚谁才是真正的主人，是你还是机器，然后使其成为主人。只有汇编语言程序员才真正有资格声称是主人，

这是一个值得思考的事实。

留意大写习惯

汇编语言是所有编程语言中最为独特的一种，它对于大小写的区分没有统一的标准。在 C 语言中，所有的标识符都是区分大小写的，但是，在汇编语言中，存在根本不区分大小写的汇编编译器。我在本书中用到的汇编编译器 NASM，只对程序员定义的标识符区分大小写。但是，对于指令助记符以及寄存器名却不区分大小写。

在有关汇编语言的书籍中常有一些书写习惯，其中之一就是将 CPU 指令助记符和寄存器名在正文中大写，而在散布于正文之中的源代码文件和代码片段中小写。这里，我也遵循这个习惯。在讨论部分的正文里，我会说 MOV、EAX 寄存器、EFLAGS 等。在示例代码中，我会用 mov、eax 及 eflags 表示它们。

这样做出于两个原因。

- 在正文中，助记符和寄存器需要醒目。因为在众多的一般文字之中，非常容易失去它们的线索。
- 为了阅读和学习本书之外的其他已有文档和源代码，你需要能够轻松地阅读汇编语言，无论它是大写的，小写的，还是大小写混合的。适应同一内容的不同表达方式，这一点非常重要。

这可能会激怒某些 Unix 界的人们，因为他们盲目崇拜小写字符。我事先为激怒他们这件事情而道歉，但是我仍然坚定地认为，那样做就是一种盲目崇拜，而且相当孩子气。

为什么我又到这了

无论你选择从什么地方开始学习本书，现在都是时间开始行动了。你只需记住，无论你面对的是什么，管它是什么黄鼠狼、机器，还是你自己的经验不足，脑海中首先要记住一点：你学习的目的是为了搞清楚它的工作原理。

让我们出发吧。

致谢

首先，我要感谢 Wiley 的 Carol Long 和 Brain Herrmann，是他们允许这本书再版，并确保了再版的实现，而且比上次出版耗时更短。

对于所有三个版本，我都欠 Michael Abrash 太多人情债，是他始终如一地在诸多方面为我提供非常明智的建议，尤其是那些 Intel 微体系结构之间的晦涩难解的区别。

尽管 Randy Hyde、Frank Kotler、Beth 和 alt.lang.asm 上的其余所有人没有意识到，但是他们其实已经通过各种方式为我提供了帮助，正是由于他们听取和回答来自汇编语言初学者的要求，才更有助于我决定哪些内容必须涵盖在像本书一样的书籍中，哪些不应该包含在里面。

最后，一如既往，向 Carol 致敬！是她 40 年如一日给予我绝对的支持和神圣的友谊，使我能够从事诸如本书之类的项目，并且将其进行到底。

作者简介

Jeff Duntemann，技术作家、编辑和讲师，同时也是一个出版业分析师。在他涉足技术领域的 30 年中，他曾经担任过施乐公司的程序员和系统分析员，Ziff-Davis 出版公司的技术期刊编辑，Coriolis Group Books 及后来的 Paraglyph 杂志社的编辑部主任。他目前是一名技术出版顾问，同时拥有 Copperwood 出版社(lulu.com 的按需印刷出版商)。Jeff 与妻子 Carol 住在科罗拉多州斯普林斯市。

目录