



The Linux Programmer's Toolbox

Linux

开发工具箱

——项目开发的最有效途径



(美) John Fusco 著
贾严磊 董西广 王在奇 译



清华大学出版社

Linux 开发工具箱

——项目开发的最有效途径

(美) John Fusco

著

贾严磊 董西广

译

王在奇

清华大学出版社

北 京

Authorized translation from the English language edition, entitled The Linux Programmer's Toolbox, 978-0-13-219857-8 by John Fusco, published by Pearson Education, Inc, publishing as Prentice-Hall, Copyright © 2007.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and TSINGHUA UNIVERSITY PRESS Copyright © 2008.

北京市版权局著作权合同登记号 图字: 01-2007-2720

本书封面贴有 Pearson Education(培生教育出版集团)防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

Linux 开发工具箱——项目开发的最有效途径/(美) 法斯克(Fusco, J.) 著; 贾严磊, 董西广, 王在奇 译.

—北京: 清华大学出版社, 2008.9

书名原文: The Linux Programmer's Toolbox

ISBN 978-7-302-17786-9

I. L… II. ①法…②贾… ③董… ④王… III. Linux 操作系统—程序设计 IV. TP316.89

中国版本图书馆 CIP 数据核字(2007)第 078423 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

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

印 刷 者: 北京四季青印刷厂

装 订 者: 三河市新茂装订有限公司

经 销: 全国新华书店

开 本: 185×230 印 张: 30.75 字 数: 670 千字

版 次: 2008 年 9 月第 1 版 印 次: 2008 年 9 月第 1 次印刷

印 数: 1~4000

定 价: 58.00 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: 025482-01

序

如果您已经掌握了使用 Linux 的基本操作，比如会使用 `ls`、`grep`、`find` 和 `sort` 这样的命令，而且还是一名 C 或 C++ 程序员，知道如何使用 Linux 系统调用，就应该明白除了能使用鼠标“单击”之外，Linux 还能给自己的生活带来更多的方便——只是现在还不知道怎样做到而已，所以您会问自己：“我该如何去做呢？”

本书会帮您回答这个问题。作者以其广博的知识，向那些非 Linux 初学者展示了如何攀越下一阶段并走向精通的求知之路。

从用于调试和性能分析的命令行工具，到 `/proc` 目录下的文件列表，本书将会向您介绍这些 Linux 行家里手所熟练使用的各种工具，从而使您在日常生活中更方便、更有效地使用 Linux。

本书除了告诉您很多的“是什么”（什么工具，什么选项，什么文件），还会告诉您“为什么”。本书会解释它们的运行原理，使您能知其然并知其所以然，最终将 Linux（还有 UNIX）的精华了然于胸。

本书是献给 Linux 程序员的一席饕餮大餐，希望您能用心品味，至少我是这么做的。
开始享受吧！

丛书主编
Arnold Robbins

前 言

Linux 从不缺乏工具，但其中的许多工具都是从 UNIX 那里继承而来的，而这些以两个字符命名的工具总令人匪夷所思，并勾起人们对最早通过纸带打孔来编写程序的那段艰辛岁月的回忆：那时，程序员总是试图用最少的字符给变量命名以节省纸带上的存储空间。值得高兴的是，那些日子已经一去不复返了，只留下了这些历史的见证。

许多这些早期遗留下来的工具现在依然非常有用，而且大部分还相当专业。每个工具虽然可能只执行一个任务，但却能完成得非常漂亮。非常专业的工具往往都有很多选项，以满足用户更具体的要求。假设您是第一次使用 `grep`，而且知道什么是正则表达式，但或许您还不能熟练掌握正则表达式的句法(别担心，其他人也没有掌握)，这没关系，因为很好地使用 `grep` 工具不一定非要成为精通正则表达式的高手。

从这本书中我希望您知道，有很多工具只要能够使用就可以了，没有必要精通它们。不必在有效地使用它们之前花费大量的时间去阅读帮助手册。我希望您能发现一些自己也许并不熟悉的新工具。本书中提到的一些工具有些是较早版本的，有些是最新版本的。所有这些工具都非常有用。随着对每个工具的深入学习，您会发现它的更多用途。

在本书中使用的“工具”一词是广义的。对我而言，创造工具与使用工具同等重要，所以本书包含了各种 API 用法的介绍，这在其他书籍的细节中通常没有涉及过。另外，该书还提供了 Linux 内核工作机制的一些背景知识，这将有助于您对一些工具的理解。本书从一个独特角度——用户角度来介绍内核，您将会找到足够的信息来理解内核为每个进程设置的底层规则，不必为此而去读内核源代码。

本书中没有拼凑的 `man` 页面或拼接成文本的其他文档。GNU 和 Linux 开发人员做了大量工作将他们做的工作文档化，但对于不熟练的用户来说，很难找到那些文档。与其花时间看那些过时的再版文档，不如了解一些查找最新文档的新方法。

虽然 GUN/Linux 文档非常丰富，但往往其可读性不强。如果为了熟悉一种工具而去读 10 000 字的帮助文档，却仍然对该工具的功能和用法毫无头绪，这显然是徒劳的。这就是本书要解决的问题。本书将解释怎样使用一种工具，以及为什么要使用它，并尽可能提供简明扼要的例子，您可以亲自输入并修改，以加强对这些工具和 Linux 本身的理解。

本书提到的所有工具都是免费使用的。大多数是由标准 Linux 版本所提供的，对于那些未提供的工具，书中已经给出了它们的 URL，您可以自行下载。

IV Linux 开发工具箱——项目开发的最有效途径

这些资料会被尽可能的设置得读起来更有趣味性。

本书读者对象

本书写给那些希望成为更有效率的，并且对 Linux 编程环境有更深入理解的中高级 Linux 程序员。如果您是一位有经验的 Windows 程序员，且在 Linux 环境下游刃有余，那么这本书也同样适合您。

非程序员读者也可以从中受益，因为这里涉及到的许多工具和主题不只局限于编程应用。如果您是一位系统管理员，或者只是一位 Linux 爱好者，那么这本书同样也有您需要的信息。

本书目的

我编写本书的目的是将它作为我曾为 *Linux Journal* 所撰写的一篇名为“Ten Command Every Linux Developer Should Know”的后续篇章。文章的灵感来源于我本人作为一名 Linux 程序员的经验。在我的日常工作中，我努力做到了每天花一些时间学习新知识，即使这意味着当前工作的短暂停滞，但我却从中受益颇多。让我感到惊讶的是，有很多次我学习了一种工具但最终却证明这种工具是没有用的，这促使我后来试图寻找立刻使用它的方法。这也成为我不断学习的强大动力。希望您通过学习本书，能够以我为范例来加强对基本规则技巧的掌握。

学习这些知识纯粹是一种乐趣。如果您跟我一样，以 Linux 学习为乐，那么促使自己学习更多的知识是没有问题的。因为 Linux 是开源码，您可以有机会了解到它所有的内部工作机制，而这对于像 Windows 一样的封闭源代码环境是不可能的。本书会介绍一些可以免费使用的资源来帮助您获取更多的知识。

如何阅读本书

每章的内容都可独立成章。后面章节的学习可能会用到前面章节的一些背景知识。只要可能，书中会交叉引用一些可以帮助您查找必要背景信息的资料。

示例是最好的老师，所以本书尽可能地提供简单的例子。建议您能够尝试实践这些例子。

本书结构

第 1 章 开源工具的下载和安装：包括发行开源码使用的机制，讨论了不同版本使用的多种软件包格式以及它们各自的优缺点，并介绍了一些维护软件包的工具以及使用方法。

第 2 章 从源代码构建：涉及到构建开源项目的基本知识，介绍了一些用来创建软件的工具以及合并工具，本章还提供了一些熟练使用 `make` 的方法和技巧。还演示了如何配置同 GNU 的 `autoconf` 工具一同发行的项目，以使用户可以定制它们来满足要求。最后，讲解了许多程序员经常误解的构建阶段，并说明了一些很可能碰到的错误和警告，以及如何解释它们。

第3章 查找帮助：介绍了您可能不太了解的 Linux 版本中的各种文件格式，还介绍了用来获取这些格式的工具，并讨论了使用它们的有效方法。

第4章 编辑和保存源文件：讨论了程序员使用的各种文本编辑器以及它们各自的优缺点，并介绍了每个程序员都应注重的并用来衡量编辑器优劣的一些特性。本章也涉及到版本控制的基本知识，这对于软件项目管理是非常关键的技术。

第5章 开发者必备内核知识：从用户角度来介绍内核。本章中提供了理解 Linux 系统的工作机制所需的必备背景知识，并介绍了几个允许用户查看代码如何影响内核的工具。

第6章 进程：重点讲解进程、进程的特性以及如何管理进程。本章介绍了引入工具以及理解它们为什么有用所需的大量背景知识。另外，本章还介绍了几个编程 API，可以用来创建自己的工具。

第7章 进程通信：介绍了进程间通信(IPC)的一些概念。本章包含了第8章所需的大部分背景知识。对于每个 IPC 机制，介绍了需要在运行实例环境下使用它的 API。

第8章 使用 shell 命令调试 IPC：介绍了一些用来调试使用 IPC 的应用程序的工具。基于第7章中的内容，有助于解释这些工具的输出信息，这些信息是难以理解的。

第9章 性能优化：介绍了衡量系统性能及个人应用程序性能的工具。本章通过一些示例说明编程设计是如何影响性能的，还讨论了多核处理器所特有的几个性能问题。

第10章 调试：介绍了一些可以用来调试应用程序的工具和技术。并介绍了一些包括 Valgrind 和 Electric Fence 在内的开源内存调试工具，而且深入讲解了 gdb 的性能以及如何有效地使用它。

目 录

第 1 章 开源工具的下载和安装	1
1.1 简介	1
1.2 什么是开放源码	2
1.3 开放源码的意义	2
1.3.1 搜索工具	2
1.3.2 版本格式	3
1.4 存档文件	4
1.4.1 识别存档文件	5
1.4.2 查询存档文件	6
1.4.3 提取存档文件	9
1.5 认识软件包管理器	10
1.5.1 源代码或二进制格式的选择	11
1.5.2 使用软件包	12
1.6 关于安全性和软件包	13
1.6.1 验证的必要性	14
1.6.2 软件包的基本认证	14
1.6.3 数字签名的软件包验证机制	15
1.6.4 RPM 格式的 GPG 签名	16
1.6.5 何时不能验证软件包	19
1.7 检查软件包目录	20
1.7.1 查看软件包	20
1.7.2 深入理解 RPM 软件包	22
1.7.3 深入理解 Debian 软件包	23
1.8 软件包更新	25
1.8.1 Apt: 高级软件包工具	26
1.8.2 Yum: Yellowdog 修订版更新	26
1.8.3 Synaptic: The GUI Front	

End For Apt	27
1.8.4 up2date: Red Hat 软件包更新 ..	28
1.9 小结	29
1.9.1 本章用到的工具	29
1.9.2 网络资源	29
第 2 章 从源代码构建	31
2.1 简介	31
2.2 构建工具	32
2.2.1 背景知识	32
2.2.2 make 工具	34
2.2.3 程序的链接	52
2.2.4 深入理解库	53
2.3 创建过程	57
2.3.1 GNU 构建工具	57
2.3.2 配置阶段	57
2.3.3 构建阶段: make	59
2.3.4 安装阶段: make install	60
2.4 理解错误和警告	60
2.4.1 常见的 Makefile 错误	61
2.4.2 配置阶段的错误	63
2.4.3 创建阶段的错误	64
2.4.4 理解编译器错误	66
2.4.5 理解编译器警告信息	68
2.4.6 理解链接器错误	76
2.5 小结	77
2.5.1 本章用到的工具	77
2.5.2 网络资源	78

第 3 章 查找帮助	79
3.1 简介	79
3.2 在线帮助工具	80
3.2.1 man 页面	80
3.2.2 man 结构	81
3.2.3 查找 man 页面: apropos	82
3.2.4 查找正确的 man 页面: whatis	84
3.2.5 在 man 页面中查找	85
3.2.6 一些推荐的 man 页面	86
3.2.7 GNU info	88
3.2.8 浏览 info 页面	88
3.2.9 查找 info 页面	90
3.2.10 推荐 info 页面	91
3.2.11 桌面帮助工具	91
3.3 其他	92
3.3.1 /usr/share/doc	92
3.3.2 交叉引用和索引	93
3.3.3 查询软件包	94
3.4 文件格式	95
3.4.1 TeX/LaTeX/DVI	95
3.4.2 Texinfo	96
3.4.3 DoCbook	96
3.4.4 HTML	97
3.4.5 PostScript	98
3.4.6 便携式文件格式(PDF)	99
3.4.7 troff	100
3.5 来自互联网的信息	100
3.5.1 www.gnu.org	100
3.5.2 sourceforge.net	101
3.5.3 Linux 文件项目	101
3.5.4 Usenet(世界性的新闻组 网络系统)	102
3.5.5 邮件列表	102
3.5.6 其他论坛	102

3.6 查找 Linux 内核信息	103
3.6.1 内核构建	103
3.6.2 内核模块	105
3.6.3 混杂(miscellaneous)文件	106
3.7 小结	106
3.7.1 本章用到的工具	106
3.7.2 在线资源	106

第 4 章 编辑和保存源文件	109
4.1 简介	109
4.2 文本编辑器	110
4.2.1 默认编辑器	111
4.2.2 在文本编辑器中查找	111
4.2.3 vi 和 Emacs	113
4.2.4 Vim: vi 扩展	113
4.2.5 Emacs	130
4.2.6 反对复制品	137
4.2.7 GUI 文本编辑器	139
4.2.8 内存使用率	144
4.2.9 编辑器概述	145
4.3 版本控制	145
4.3.1 版本控制基础	145
4.3.2 定义版本控制的术语	147
4.3.3 支持工具	148
4.3.4 diff 和 patch 简介	149
4.3.5 检查和合并更改	152
4.4 源代码的优化器和浏览器	157
4.4.1 缩进代码优化器	158
4.4.2 Astyle 风格	160
4.4.3 用 cflow 分析代码	160
4.4.4 用 ctags 分析代码	163
4.4.5 用 cscope 浏览代码	163
4.4.6 用 Doxygen 浏览和记录代码	164
4.4.7 使用编译器分析代码	165
4.5 小结	167

4.5.1	本章用到的工具	168	5.7.1	本章用到的工具	241
4.5.2	参考资料	168	5.7.2	本章讨论的 APIs	242
4.5.3	在线资源	168	5.7.3	在线资源	242
第 5 章	开发者必备内核知识	171	5.7.4	参考资料	242
5.1	简介	171	第 6 章	进程	243
5.2	用户模式与内核模式	172	6.1	简介	243
5.2.1	系统调用	173	6.2	进程的产生	243
5.2.2	用户空间与内核空间的 数据传送	175	6.2.1	fork 和 vfork	244
5.3	进程调度程序	175	6.2.2	写拷贝	244
5.3.1	初识调度	176	6.2.3	clone	245
5.3.2	阻塞, 抢先占有和放弃	177	6.3	exec 函数	245
5.3.3	调度的优先与公平	178	6.3.1	可执行脚本	246
5.3.4	优先权和 Nice 值	182	6.3.2	可执行目标文件	248
5.3.5	实时优先权	183	6.3.3	二进制文件	248
5.3.6	创建实时进程	185	6.4	wait 实现进程同步	250
5.3.7	进程状态	186	6.5	进程的内存占用	252
5.3.8	时间度量	190	6.5.1	文件描述符	254
5.4	设备和设备驱动程序	198	6.5.2	堆栈	259
5.4.1	设备驱动程序的类型	199	6.5.3	常驻内存和固定内存	260
5.4.2	内核模块	200	6.6	设定进程限制	260
5.4.3	设备节点	201	6.7	进程和 procfs	263
5.4.4	设备和输入/输出	210	6.8	进程管理工具	265
5.5	I/O 调度程序	217	6.8.1	通过 ps 命令显示进程信息	265
5.5.1	Linux 电梯式调度(aka noop)	218	6.8.2	使用 formats 增加进程信息	267
5.5.2	I/O 调度程序的截止时间	219	6.8.3	查找名中带有 ps 和 pgrep 的进程	269
5.5.3	先占 I/O 调度程序	219	6.8.4	利用 pmap 查看进程使用 的空间	269
5.5.4	完整的公平地排队 I/O 调度程序	219	6.8.5	通过名字发送信号给进程	271
5.5.5	选择一个 I/O 调度程序	219	6.9	小结	271
5.6	用户空间的内存管理	220	6.9.1	系统调用和本章用到的 API	272
5.6.1	虚拟内存的解释	220	6.9.2	本章用到的工具	272
5.6.2	内存耗尽	232	6.9.3	在线资源	272
5.7	小结	241			

第 7 章 进程通信	273
7.1 简介	273
7.2 使用纯文本文件的 IPC	274
7.2.1 文件加锁	279
7.2.2 使用文件进行 IPC 的缺点	279
7.3 共享内存	279
7.3.1 POSIX 共享内存 API	280
7.3.2 System V 共享内存 API	283
7.4 信号	286
7.4.1 向进程发送信号	286
7.4.2 信号处理	287
7.4.3 信号掩码和信号处理	288
7.4.4 实时信号	291
7.4.5 具有 sigqueue 和 sigaction 的高级信号	293
7.5 管道	295
7.6 套接字	296
7.6.1 创建套接字	296
7.6.2 使用 socketpair 的本地套 接字示例	298
7.6.3 使用本地套接字的客户端/ 服务器端示例	300
7.6.4 使用网络套接字的客户端/ 服务器端示例	305
7.7 消息队列	305
7.7.1 System V 消息队列	306
7.7.2 POSIX 消息队列	309
7.7.3 POSIX 与 System V 的消息 队列的区别	314
7.8 信号量	314
7.8.1 POSIX 信号量使用的 API	318
7.8.2 System V 信号量使用的 API	321
7.9 小结	323
7.9.1 本章中用到的系统 调用和 API	323

7.9.2 参考资料	325
7.9.3 在线资源	325
第 8 章 使用 shell 命令调试 IPC	327
8.1 简介	327
8.2 打开文件时用到的工具	327
8.2.1 lsof	328
8.2.2 fuser	329
8.2.3 ls	330
8.2.4 file	330
8.2.5 stat	330
8.3 查看文件中的数据	331
8.3.1 字符串命令	334
8.3.2 xxd 命令	334
8.3.3 hexdump 命令	335
8.3.4 od 命令	336
8.4 用于 V IPC 系统的内核工具	337
8.4.1 V 系统共享内存	337
8.4.2 V 系统消息队列	340
8.4.3 V 系统的信号量	340
8.5 POSIX IPC 用到的工具	341
8.5.1 POSIX 共享内存	342
8.5.2 POSIX 消息队列	342
8.5.3 POSIX 信号量	343
8.6 信号用到的工具	344
8.7 管道和套接字用到的工具	346
8.7.1 管道和 FIFO	346
8.7.2 套接字	347
8.8 使用索引识别文件和 IPC 对象	349
8.9 小结	351
8.9.1 本章用到的工具	351
8.9.2 在线资源	351

第 9 章 性能优化	353	10.3.1 使用 gdb 运行代码	423
9.1 简介	353	10.3.2 停止和重新执行	424
9.2 系统性能	353	10.3.3 检查和管理数据	432
9.2.1 内存问题	354	10.3.4 使用 gdb 连接正在 运行的进程	441
9.2.2 CPU 利用率和总线冲突	363	10.3.5 调试内核文件	442
9.2.3 设备和中断	365	10.3.6 使用 gdb 进行多线程调试	445
9.2.4 查找系统性能问题的工具	371	10.3.7 调试优化的代码	446
9.3 应用程序性能	377	10.4 调试共享对象	449
9.3.1 计时命令的第一步	378	10.4.1 使用共享对象的 时间和原因	449
9.3.2 x86info 处理器结构	379	10.4.2 创建共享对象	449
9.3.3 使用 Valgrind 检查指令效率	382	10.4.3 定位共享对象	450
9.3.4 ltrace 简介	385	10.4.4 覆盖默认共享目标的位置	451
9.3.5 使用 strace 监视程序性能	387	10.4.5 共享对象的安全问题	451
9.3.6 传统性能优化工具: gcov 和 gprof	388	10.4.6 共享对象使用的工具	452
9.3.7 OProfile 简介	394	10.5 查找内存问题	454
9.4 多处理器平台	400	10.5.1 两次释放	454
9.4.1 SMP 硬件的类型	400	10.5.2 内存泄露	455
9.4.2 SMP 机上的编程实现	404	10.5.3 缓冲区溢出	455
9.5 小结	406	10.5.4 glibc 工具	457
9.5.1 本章介绍的基本性能	406	10.5.5 使用 valgrind 调试内存	460
9.5.2 本章介绍的专业术语	406	10.5.6 使用 Electric Fence 检测 内存泄漏	465
9.5.3 本章用到的工具	406	10.6 非常规技术	467
9.5.4 在线资源	407	10.6.1 创建自己的黑匣子	467
9.5.5 参考资料	407	10.6.2 获取运行时的堆栈轨迹	470
第 10 章 调试	409	10.6.3 强制内核转储	471
10.1 简介	409	10.6.4 使用信号	473
10.2 最基本的调试工具: printf	410	10.6.5 使用 procfs 调试	473
10.2.1 使用 printf 存在的问题	410	10.7 小结	475
10.2.2 有效地使用 printf	414	10.7.1 本章用到的工具	476
10.2.3 关于 printf 调试工具的 结束语	421	10.7.2 在线资源	476
10.3 GNU 下最好用的 调试器: gdb	422	10.7.3 参考资料	476

Chapter

1



开源工具的下载和安装

1.1 简介

本章介绍了自由软件不同的发行版本，以及怎样熟练掌握它们和找到它们的途径，并详细地说明了存档文件和包文件，以及用于管理它们的最常见的工具命令。

不明来源的软件可能存在风险，这里也涉及到一些安全方面的问题，您应该有意识地采取一些措施来保护自己。本章介绍了验证和信任机制的概念，说明了如何将它们应用到安全机制中，并介绍了当无法通过验证时，如何查看软件包和存档文件。

最后，介绍了管理软件包的工具以及有效使用它们的方法。

1.2 什么是开放源码

开源是指自由软件的行业术语，由 Open Source Initiative¹ (OSI, 开放源码促进会、开放源码组织)首创。这个组织由 Richard Stallman 创建，旨在推动以 GNU 项目为源的自由软件原则。它的目标之一是反对那些消极的墨守成规的自由软件，促进源代码的免费共享。

起先，很多商家不敢使用开源软件。毋庸置疑，一些大型软件公司的市场部门曾对它做出改动。俗语说：“天下没有免费的午餐”，一些人担心这种许可证(像 GNU 公共许可证)如同一个病毒，如果他们在项目中使用了开源软件，那么他们的项目也要将源代码公之于众。

值得庆幸的是，大多数这样的担心慢慢减少。许多大型开发商在自己的项目中自由使用并改进开放源代码，有些甚至全部基于开源软件。精灵现身于世了。

1.3 开放源码的意义

对于大多数人来说，开源软件只不过是许多自由使用的高质量软件罢了。遗憾的是，现在使用的许多软件并不都是高质量的，但却是必不可少的一部分。好的项目理念日益繁茂，欣欣向荣；差的却日渐枯萎，郁郁而终。获取开源软件有些像摘水果：要靠经验才能知道它是否成熟。

优胜劣汰的自然选择规律无处不在。在源码水平上，选择具有不同特征的代码(基于补丁程序)以插入最优的代码。作为一个消费者，会选择若干项目来下载，以增强项目的生命力。没人会为无人使用的项目开发代码，下载的越少越不能吸引开发人员，下载的越多意味着有更多的开发人员使用它，进而意味着能可以从中选择更多的代码，从而得到更好的代码。有时选择一个开发项目就像一场赌博，是拿时间和为之付出的努力作赌注，不可避免会做出让人后悔的选择，但要记住：“这就是过程”。

对于一些人来说，不清楚自己获得的是什么或许也是一种乐趣，就和打开生日礼物时一样。对其他人来说，这非常麻烦，而且浪费时间。假如要寻找能够即装即用的现成软件，这也有用户所需要的开源软件——只是数量有限而已。幸运的是，因特网上有很多资源可以帮助用户做出不错的选择。

1.3.1 搜索工具

在网上开始搜索之前，首先要看一下 CD 的发行版本。假如是从套装 CD 或 DVD 安装的 Linux，那么一定有许多未安装的工具。大多数发行的 CD 版本比默认安装的版本多一些软件，举例来说，当安装操作系统时，会提示选择安装种类，这可能会导致用户随意地安装软件包，

¹ www.opensource.org。

系统主要是基于“工作站”或“服务器”的角度。

可以手动地从安装光盘寻找原始软件包，然后将它们添加到安装软件中。缺点是软件包并没有按特定的顺序进行分类，所以必须知道所要找到的软件名称。一些发行版本采用图形界面将软件包分类，以帮助用户选择要安装的软件。

假如不知道要查找的具体名称，可以使用因特网。一些网站为开源软件提供了类似票据交换场所的服务，www.freshmeat.net 就是其中一员。在那里会发现软件是分门别类的，所以很容易就可以找到所需的软件。例如，我在著书过程中，从 Freshmeat 上搜索“word processor”这个词，结果找到了 71 个相关的内容。可想而知，从 71 个不同的结果中选择是非常困难的。

Freshmeat 允许对结果进行筛选以缩小选择范围。我的搜索结果中包含除了 Linux 和处于开发阶段的项目外的多种操作系统。所以我选择将搜寻范围限定在由 Linux 支持的成熟的使用开放式系统互连参考模型(OSI)的开源认证的那些项目中(Freshmeat 的搜索结果中也包括商业软件)，这样就减少到只有 12 项了，更容易搜索。给定的 word processor 这个术语具有宽泛的解释，更进一步地查找表明仍有一些项目不是我们要找的。再次过滤之后，我才找到一些著名的高质量的项目，如 AbiWord，还有一些我以前从来没听说过的。但仍有一些没有出现的，如 OpenOffice，我也将其写入这本书中。后来证明，我找不到 OpenOffice 是因为它是属于 Office/Business::Office Suites，而不是 word processor。这件事使我明白，如果没找到想要的信息，不是因为不存在，而是因为没找到，所以继续寻找肯定会找到。

1.3.2 版本格式

既然已经找到了所感兴趣的软件，那么可能会有更多的选择。成熟的项目通常提供一种或多种版本的准备安装的软件包，只有较少的仅在存档文件中提供源代码和二进制文件。通常这对用户来说是个很好的提示。下载软件包可能就像是买新车，不必知道它是怎样工作的，只需要转动钥匙，然后起动即可。相反，下载源码或二进制文件的存档文件就像买旧车，如果对车有所了解，那么会好一些；否则，就无所适从了。

通常，当一个项目提供一个安装包时，标志着这个项目已经成熟，也标志着该项目的发行周期日趋稳定。如果这个项目每周都发布新的版本，那它很可能不用担心打包的问题。一个软件包文件，如果运气好的话，也许能安装并运行它，但即使是一辆新车，也可能偶尔有瑕疵。

存档文件是软件包文件的一类，它适用于基于 Linux 的项目，通常是存档的 tar 文件。存档文件是存档成单个文件的文件集合，它使用像 tar 命令一样的存档工具。这些文件经常使用 gzip 程序命令进行存档以节省空间，通常作为 tar 文件或 tarballs 被提及。

tar 文件是项目发布源码的首选格式，它们容易创建和使用，而且每个程序员也都熟悉 tar 程序。有时会发现 tar 文件中包含二进制可执行代码，这对打包来说是个快捷但不理想的选

择，应该避免使用这种方法，除非清楚地知道自己在做什么。总的来说，tar 文件适用于对编程和系统管理有所了解的人群。

1.4 存档文件

在下载和安装开源软件的过程中，可能会遇到各种各样的存档文件。存档文件是包含其他文件集合的文件。如果您是一个 Windows 用户，肯定非常熟悉占主导地位的 Windows 的存档文件 PKZip。Linux 的存档效用函数与之非常类似，而与 PKZip 不同的是，它们不包括压缩。Linux 的存档工具专注于存档，而将压缩交给另一种工具(典型的是 gzip 或 bzip2)，这就是 UNIX 的思想。当然，因为在 Linux 下可以选择多种存档文件，但作为开源的使用者，必须取之所用。所以即使您可能只遇到 tar 文件，但至少也应知道其他工具也是可用的。

存档工具除了要保存文件名及数据外，还有一些特殊的要求。除了文件的路径名及数据之外，存档工具还需要保存文件的元数据。元数据(metadata)包括文件的所有者、组别，以及其他属性(如读/写/执行权限)。存档文件记录了所有诸如此类的信息，比如从文件系统中删除文件，然后从存档文件中无损地恢复。假如您存档了一个可执行文件，然后把它从文件系统中删除了，那么当要恢复时，它应该仍是可执行的。在 Windows 中，文件名通过它的扩展名(如.exe)表明它是否是可执行文件。而 Linux 使用元数据表明文件是否是可执行的，所以要保存用存档文件存储的这些数据。

在 Linux 中最常用的存档工具如表 1-1 所示。到目前为止，最流行的存档格式就是 tar。tar 是 tape archive 的缩写，它来源于磁带备份功能。现在，tar 已经成为用来存档批量文件的一种最常见的通用工具。使用 tar 工具，有时会遇到 cpio，它使用一种截然不同的语法来完成同样的功能。POSIX(可移植 UNIX 操作系统接口标准)标准存档效用函数 pax 可以识别 tar 文件、cpio 文件或者它自己的格式文件。我从未见过 pax 格式的发行版本，提到它只是为了论述的完整性。

最后值得提及的存档工具是 ar，它常用来创建软件开发所使用的目标代码库，但也可用来创建 Debian 版本的软件包文件。

表 1-1 常用的存档工具

工 具	说 明
tar	最流行的工具
cpio	RPM 格式内部使用的工具；应用不广泛
ar	Debian 软件包内部使用的工具，只有软件开发库使用。ar 文件没有路径信息

也可以找到一些用来处理由 PKZip 创建的 zip 文件的工具, 以及一些不常见的压缩存档工具, 如 lha。然而, Linux 开源程序不会以这些格式发行。如果见到 .zip 存档文件, 它也一定是为微软操作系统准备的。

在大多数情况下, 对于每种版本格式, 需要知道两点: 如何查询存档文件的内容以及如何从存档文件中提取文件。Linux 存档文件更注重基本功能, 它并不像 Windows 的存档文件, 有各种危险的附加功能。所以, 从存档文件中查询和提取文件大体上是安全的, 尤其当自己不是超级用户的时候。在提取文件之前查询存档文件是个明智的选择, 这样, 就不会不小心修改了系统中的同名文件。

1.4.1 识别存档文件

当从网上下载了一个存档文件时, 它很可能因节省带宽而被压缩了。有许多存档文件的命名规则, 其中的一部分如表 1-2 所示。

表 1-2 部分存档工具命名规则

扩展名	类型
.tar	tar 存档文件, 未压缩
.tar.gz.tgz	tar 存档文件, 使用 gzip 压缩
.tar.bz2	tar 存档文件, 使用 bzip2 压缩
.tar.Z.taz	tar 存档文件, 使用 UNIX 的 compress 命令压缩
.ar.a	ar 存档文件, 通常只用于软件开发
.cpio	cpio 存档文件, 未压缩

如果不确定文件的存档类型, 就记住使用 file 命令。当文件名不能提供什么信息时, 使用这个工具能够确定文件类型。当网络浏览器或其他工具使得文件名无法识别时, 它也是非常有用的, 例如, 假设有一个存档的 tar 文件, 文件名为 foo.x, 文件名不能反映文件的内容, 那么试一下下面的命令:

```
$ file foo.x
foo.x: gzip compressed data, from UNIX, max compression
```

现在可以知道它是用 gzip 存档的, 但仍然不确定它是否是 tar 文件, 可以用 gzip 将其解压缩, 然后再次使用 file 命令, 或者使用该命令的 -z 选项。

```
$ file -z foo.x
foo.x: tar archive (gzip compressed data, from UNIX, max compression)
```

现在可以确定它是 tar 文件了。