第一图书网, tushu007.com

# <<Linux开发工具箱>>

## 图书基本信息

书名：<<Linux开发工具箱>>

13位ISBN编号：9787302177869

10位ISBN编号：7302177864

出版时间：2008-9

出版时间：清华大学出版社

作者：（美）法斯克（Fusco，J）

页数：476

字数：670000

译者：贾严磊，董西广，王在奇

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

## 前言

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！

## 内容概要

本书将系统性地介绍GNU/Linux的开源工具，并通过可以被读者容易修改的简明示例说明如何使用这些工具，以满足读者的实际需求。
本书首先介绍如何下载、构建和安装开源项目，如何发布开源工具，以及如何避免将时间浪费到还未准备好的项目中，并引导读者构建自己的项目。
然后通过示例演示了如何在文本编辑器中进行查找操作，展示了几个常用文本编辑器中会用到的小技巧。
书中接下来对Linux内核的基础知识和操作系统的内部原理进行了详细且通俗易懂的阐述，并示范了如何将这些知识应用到更高级的工具中。
还重点讲解sar、vmstat、valpind和strace等工具的输出，以及如何将它们运用到应用程序中；如何利用各种编程API来开发适用于自己的工具。
最后，作者介绍了可以提高软件性能的工具；解释了如今市面上的多芯CPU的内在原理，并演示了如何从这些系统中获取最佳性能；以及介绍了在各种环境下调试代码的工具和技术。
对于程序员来说，如果希望自己开发出来的程序更为简洁有效，如果想对Linux编程环境有更深刻的理解，那么，本书提供的资料相信能给您带来惊喜！

第一图书网, tushu007.com


## 作者简介

John Fusco是GE Healthcare的一名软件开发人员，专门编写Linux应用程序和设备驱动程序。
他在Unix软件行业有十多年的工作经验，从内核2.0版本就开始开发Linux应用程序。
他曾为Embedded Systems Programming和Linux Journal撰写文章。

## 书籍目录

第1章 开源工具的下载和安装 1.1 简介 1.2 什么是开放源码 1.3 开放源码的意义 1.3.1 搜索工具 1.3.2 版本格式 1.4 存档文件 1.4.1 识别存档文件 1.4.2 查询存档文件 1.4.3 提取存档文件 1.5 认识软件包管理器 1.5.1 源代码或二进制格式的选择 1.5.2 使用软件包 1.6 关于安全性和软件包 1.6.1 验证的必要性 1.6.2 软件包的基本认证 1.6.3 数字签名的软件包验证机制 1.6.4 RPM格式的GPG签名 1.6.5 何时不能验证软件包 1.7 检查软件包目录 1.7.1 查看软件包 1.7.2 深入理解RPM软件包 1.7.3 深入理解Debian软件包 1.8 软件包更新 1.8.1 Apt：高级软件包工具 1.8.2 Yum：Yellowdog修订版更新 1.8.3 Synaptic：The GUI Front End For Apt 1.8.4 up2date：Red Hat软件包更新 1.9 小结 1.9.1 本章用到的工具 1.9.2 网络资源第2章 从源代码构建 2.1 简介 2.2 构建工具 2.2.1 背景知识 2.2.2 make工具 2.2.3 程序的链接 2.2.4 深入理解库 2.3 创建过程 2.3.1 GNU构建工具 2.3.2 配置阶段 2.3.3 构建阶段：make 2.3.4 安装阶段：make install 2.4 理解错误和警告 2.4.1 常见的Makefile错误 2.4.2 配置阶段的错误 2.4.3 创建阶段的错误 2.4.4 理解编译器错误 2.4.5 理解编译器警告信息 2.4.6 理解链接器错误 2.5 小结 2.5.1 本章用到的工具 2.5.2 网络资源第3章 查找帮助 3.1 简介 3.2 在线帮助工具 3.2.1 man页面 3.2.2 man结构 3.2.3 查找man页面：apropos 3.2.4 查找正确的man页面：whatis 3.2.5 在man页面中查找 3.2.6 一些推荐的man页面 3.2.7 GNUinfo 3.2.8 浏览info页面 3.2.9 查找info页面 3.2.10 推荐info页面 3.2.11 桌面帮助工具 3.3 其他 3.3.1 /usr/share/doc 3.3.2 交叉引用和索引 3.3.3 查询软件包 3.4 文件格式 3.4.1 TeX/LaTtex/DVI 3.4.2 Texinfo 3.4.3 DoCbook 3.4.4 HTML 3.4.5 PostScript 3.4.6 便携式文件格式（PDF） 3.4.7 troff 3.5 来自互联网的信息 3.5.1 www.gnu.org 3.5.2 sourceforge.net 3.5.3 Linux文件项目 3.5.4 LIsenet（世界性的新闻组网络系统） 3.5.5 邮件列表 3.5.6 其他论坛 3.6 查找Linux内核信息 3.6.1 内核构建 3.6.2 内核模块 3.6.3 混杂（miscellaneous）文件 3.7 小结 3.7.1 本章用到的工具 3.7.2 在线资源第4章 编辑和保存源文件 4.1 简介 4.2 文本编辑器 4.2.1 默认编辑器 4.2.2 在文本编辑器中查找 4.2.3 vi和Emacs 4.2.4 Vim：vi扩展 4.2.5 Emacs 4.2.6 反对复制品 4.2.7 GUI文本编辑器 4.2.8 内存使用率 4.2.9 编辑器概述 4.3 版本控制 4.3.1 版本控制基础 4.3.2 定义版本控制的术语 4.3.3 支持工具 4.3.4 diff和patch简介 4.3.5 检查和合并更改 4.4 源代码的优化器和浏览器 4.4.1 缩进代码优化器 4.4.2 Astyle风格 4.4.3 用cflow分析代码 4.4.4 用ctags分析代码 4.4.5 用cscope浏览代码 4.4.6 用Doxygen浏览和记录代码 4.4.7 使用编译器分析代码 4.5 小结 4.5.1 本章用到的工具 4.5.2 参考资料 4.5.3 在线资源第5章 开发者必备内核知识 5.1 简介 5.2 用户模式与内核模式 5.2.1 系统调用 5.2.2 用户空间与内核空间的数据传送 5.3 进程调度程序 5.3.1 初识调度 5.3.2 阻塞，抢先占有和放弃 5.3.3 调度的优先与公平 5.3.4 优先权和Nice值 5.3.5 实时优先权 5.3.6 创建实时进程 5.3.7 进程状态 5.3.8 时间度量 5.4 设备和设备驱动程序 5.4.1 设备驱动程序的类型 5.4.2 内核模块 5.4.3 设备节点 5.4.4 设备和输入/输出 5.5 I/O调度程序 5.5.1 Ljlnus电梯式调度（aka noop） 5.5.2 I/O调度程序的截止时间 5.5.3 先占I/O调度程序 5.5.4 完整的公平地排队I/O调度程序 5.5.5 选择一个I/O调度程序 5.6 用户空间的内存管理 5.6.1 虚拟内存的解释 5.6.2 内存耗尽 5.7 小结 5.7.1 本章用到的工具 5.7，2 本章讨论的APIs 5.7.3 在线资源 5.7.4 参考资料第6章 进程 6.1 简介 6.2 进程的产生 6.2.l fork和vfork 6.2.2 写拷贝 6.2.3 clone 6.3 exec函数 6.3.1 可执行脚本 6.3.2 可执行目标文件 6.3.3 二进制文件 6.4 wait实现进程同步 6.5 进程的内存占用 6.5.1 文件描述符 6.5.2 堆栈 6.5.3 常驻内存和固定内存 6.6 设定进程限制 6.7 进程和Drocfs 6.8 进程管理工具 6.8.1 通过ps命令显示进程信息 6.8.2 使用formats增加进程信息 6.8.3 查找名中带有ps和pgrep的进程 6.8.4 利用pmap查看进程使用的空间 6.8.5 通过名字发送信号给进程 6.9 小结 6.9.1 系统调用和本章用到的API 6.9.2 本章用到的工具 6.9.3 在线资源第7章 进程通信 7.1 简介 7.2 使用纯文本文件的IPC 7.2.1 文件加锁 7.2.2 使用文件进行IPC的缺点 7.3 共享内存 7.3.1 POSIX共享内存API 7.3.2 SystemV共享内存API 7.4 信号 7.4.1 向进程发送信号 7.4.2 信号处理 7.4.3 信号掩码和信号处理 7.4.4 实时信号 7.4.5 具有sigqueue和sigaction的高级信号 7.5 管道 7.6 套接字 7.6.1 创建套接字 7.6.2 使用socketpair的本地套接字示例 7.6.3 使用本地套接字的客户端/服务器端示例 7.6.4 使用网络套接字的客户端/服务器端示例 7.7 消息队列 7.7.1 System V消息队列 7.7.2 POSIX消息队列 7.7.3 POSIX与System V的消息队列的区别 7.8 信号量 7.8.1 POSIX信号量使用的API 7.8.2 System V信号量使用

的API 7.9 小结 7.9.1 本章中用到的系统调用和API 7.9.2 参考资料 7.9.3 在线资源第8章 使用sheLL命令调试IPC 8.1 简介 8.2 打开文件时用到的工具 8.2.1 lsof 8.2.2 fuser 8.2.3 ls 8.2.4 file 8.2.5 stat 8.3 查看文件中的数据 8.3.1 字符串命令 8.3.2 xxd命令 8.3.3 hexdump命令 8.3.4 od命令 8.4 用于VIPC系统的内核工具 8.4.1 V系统共享内存 8.4.2 V系统消息队列 8.4.3 V系统的信号量 8.5 POSIX IPC用到的工具 8.5.1 POSIX共享内存 8.5.2 POS[X消息队列 8.5.3 POSIX信号量 8.6 信号用到的工具 8.7 管道和套接字用到的工具 8.7.1 管道和FIFO 8.7.2 套接字 8.8 使用索引识别文件和IPC对象 8.9 小结 8.9.1 本章用到的工具 8.9.2 在线资源第9章 性能优化 9.1 简介 9.2 系统性能 9.2.1 内存问题 9.2.2 CPU利用率和总线冲突 9.2.3 设备和中断 9.2.4 查找系统性能问题的工具 9.3 应用程序性能 9.3.1 计时命令的第一步 9.3.2 x86info处理器结构 9.3.3 使用Valgrind检查指令效率 9.3.4 ltrace简介 9.3.5 使用strace监视程序性能 9.3.6 传统性能优化工具：gcov和gprof 9.3.7 OProfile简介 9.4 多处理器平台 9.4.1 SMP硬件的类型 9.4.2 SMP机上的编程实现 9.5 小结 9.5.1 本章介绍的基本性能 9.5.2 本章介绍的专业术语 9.5.3 本章用到的工具 9.5.4 在线资源 9.5.5 参考资料第10章 调试 10.1 简介 10.2 最基本的调试工具：Drintf 10.2.1 使用pdntf存在的问题 10.2.2 有效地使用printf 10.2.3 关于printf调试工具的结束语 10.3 GNU下最好用的调试器：gdb 10.3.1 使用gdb运行代码 10.3.2 停止和重新执行 10.3.3 检查和管理数据 10.3.4 使用gdb连接正在运行的进程 10.3.5 调试内核文件 10.3.6 使用gdb进行多线程调试 10.3.7 调试优化的代码 10.4 调试共享对象 10.4.1 使用共享对象的时间和原因 10.4.2 创建共享对象 10.4.3 定位共享对象 10.4.4 覆盖默认共享目标的位置 10.4.5 共享对象的安全问题 10.4.6 共享对象使用的工具 10.5 查找内存问题 10.5.1 两次释放 10.5.2 内存泄露 10.5.3 缓冲区溢出 10.5.4 glibc工具 10.5.5 使用valgrind调试内存 10.5.6 使用E1ectric Fence检测内存泄漏 10.6 非常规技术 10.6.1 创建自己的黑匣子 10.6.2 获取运行时的堆栈轨迹 10.6.3 强制内核转储 10.6.4 使用信号 10.6.5 使用procfs调试 10.7 小结 10.7.1 本章用到的工具 10.7.2 在线资源 10.7.3 参考资料

## 章节摘录

5.1 简介本章假定你已掌握了一定的编写Linux应用程序的经验并且对内核已有了初步的了解。
本章将涵盖一些与内核相关的主题。
这些主题在论述内核实质的书中经常出现。
与这些书不同的是，本章更侧重从应用角度进行分析。
本章所涵盖的内容有Linux的调度程序，近来这些调度程序经历了许多变革。
此外，本章还将阐述进程优先权和抢占方式调度策略，以及它们的角色和实时应用。
32位的地址空间足够满足需要，因此多数应用系统在实际操作中从未受阻。
但现在，32位的系统已能支持超过4GB的RAM，许多程序员往往在操作受阻时还不知道他们遭遇的是怎样的问题。
学完本章后，你将对这些问题有个更清晰的认识，并且学会如何在操作中规避它们。
本章还将介绍输入输出系统以及它与进程之间的关系。
也许你已经被现代处理器的高速时钟速度搞得眼花缭乱，但不料却失望地发现低速的启动设备程序正严重扼杀处理器的操作性能。
本章将介绍一些Linux程序设计模型中的低效构件，以及如何在操作中绕开它们。
本章还将详细讨论Linux 2.6版本中I/O调度程序的改进，以及如何更好地发挥其性能。
5.2 用户模式与内核模式执行进程的模式有两种：用户模式和内核模式。
你编写的代码和执行所链接的库是在用户模式下。
当进程需要内核的服务时，就必须执行内核代码，而且这只能在内核模式下运行。
这听起来很简单，但是真正操作时存在许多困难。
首先来讨论为什么需要两种操作模式。
原因之一就是安全性。
当一个进程在用户模式下执行时，它所占的内存空间对它来说是唯一的。
Linux是一个多用户操作系统，因此一个进程不能访问另一进程的内存空间，因为另一进程可能包含密码或一些敏感信息。
用户模式可以确保一个进程只能访问它自己的内存空间。
此外，如果一个进程破坏了它自己的内部结构，它也只能影响它自己，而不会波及到其他任何进程，当然更不会影响到整个系统。
在用户模式下，进程所能访问的内存称为用户空间(user space)。
因为该系统是作为一个整体运作的，所以内核需要维护数据结构以控制系统中的每个进程，因此内核提供了一个所有的进程共享的内存区域。
因为系统中的所有进程都在执行内核，所以每个进程都需要访问同一个共享的内存区域。
然而，为了保证安全性，内核代码和数据结构必须严格独立于用户代码和数据。
这就是我们需要内核模式的原因。
只有内存代码可以在内核模式下运行，访问共享的内核数据以及执行特权指令。
把进程在内核模式下访问的内存区域叫做内核空间(kernel space)。
内核空间只有一个，在内核模式下的每一个进程都可以访问内核空间，但与用户空间不同的是，内核空间是每一个进程所特有的。
图5-1显示了所有进程中内核的虚拟地址分配。
在这个例子中， 1G的虚拟地址高地址分配给了内核，剩下部分则分配给了进程。
这样的分配方式可以在构建内核时确立。
但这种所谓的3G/1G分配方式在很多现存的内核中(也)是常见的。
在这种配置下，内核的所有地址都大于0XCJ0000000。
当进程要访问这些地址时，就必须在内核模式下执行。

## 编辑推荐

《Linux开发工具箱:项目开发的最有效途径》共分10个章节，系统性地介绍GNU/Linux的开源工具，并通过可以被读者容易修改的简明示例说明如何使用这些工具，具体内容包括开源工具的下载和安装、从源代码构建、查找帮助、编辑和保存源文件、开发者必备内核知识等。
《Linux开发工具箱:项目开发的最有效途径》可供各大专院校作为教材使用，也可供从事相关工作的人员作为参考用书使用。

## 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