

debian

Debian 参考手册

Osamu Aoki (青木修)

版权 © 2013-2026 青木修

Debian 参考手册（版本 2.153）(2026-11-29 05:38:00 UTC) 旨在作为一份 Debian 系统安装后的用户指南，为 Debian 系统的使用与管理提供广泛的概览。它通过为非开发者编写的 shell 命令示例来涵盖系统管理的方方面面。

Contents

1 GNU/Linux 教程 **1**

1.1 控制台基础 . . . 1

1.1.1 shell 提示符 . . . 1

1.1.2 GUI 下的 shell 提示符 . . . 2

1.1.3 root 账户 . . . 2

1.1.4 root shell 提示符 . . . 3

1.1.5 GUI 系统管理工具 . . . 3

1.1.6 虚拟控制台 . . . 3

1.1.7 怎样退出命令行提示符 . . . 3

1.1.8 怎样关闭系统 . . . 3

1.1.9 恢复一个正常的控制台 . . . 4

1.1.10 建议新手安装的额外软件包 . . . 4

1.1.11 额外用户账号 . . . 5

1.1.12 sudo 配置 . . . 5

1.1.13 动手时间 . . . 6

1.2 类 Unix 文件系统 . . . 6

1.2.1 Unix 文件基础 . . . 6

1.2.2 文件系统深入解析 . . . 7

1.2.3 文件系统权限 . . . 7

1.2.4 控制新建文件的权限：umask . . . 10

1.2.5 一组用户的权限（组） . . . 10

1.2.6 时间戳 . . . 11

1.2.7 链接 . . . 12

1.2.8 命名管道（先进先出） . . . 13

1.2.9 套接字 . . . 14

1.2.10 设备文件 . . . 14

1.2.11 特殊设备文件 . . . 15

1.2.12 procfs 和 sysfs . . . 15

1.2.13 tmpfs . . . 16

1.3 Midnight Commander (MC) . . . 16

1.3.1 自定义 MC . . . 16
1.3.2 启动 MC . . . 16
1.3.3 MC 文件管理 . . . 17
1.3.4 MC 命令行技巧 . . . 17
1.3.5 MC 内部编辑器 . . . 17
1.3.6 MC 内部查看器 . . . 18
1.3.7 自动启动 MC . . . 18
1.3.8 MC 中的虚拟文件系统 . . . 18
1.4 类 Unix 工作环境基础 . . . 18
1.4.1 登录 shell . . . 18
1.4.2 定制 bash . . . 19
1.4.3 特殊按键 . . . 20
1.4.4 鼠标操作 . . . 20
1.4.5 分页程序 . . . 21
1.4.6 文本编辑器 . . . 21
1.4.7 设置默认文本编辑器 . . . 21
1.4.8 使用 vim . . . 22
1.4.9 记录 shell 活动 . . . 23
1.4.10 基本的 Unix 命令 . . . 23
1.5 简单 shell 命令 . . . 25
1.5.1 命令执行和环境变量 . . . 25
1.5.2 “$LANG”变量 . . . 25
1.5.3 ”$PATH” 变量 . . . 26
1.5.4 ”$HOME” 变量 . . . 27
1.5.5 命令行选项 . . . 27
1.5.6 Shell 通配符 . . . 27
1.5.7 命令的返回值 . . . 28
1.5.8 典型的顺序命令和 shell 重定向 . . . 29
1.5.9 命令别名 . . . 30
1.6 类 Unix 的文本处理 . . . 30
1.6.1 Unix 文本工具 . . . 30
1.6.2 正则表达式 . . . 31
1.6.3 替换表达式 . . . 32
1.6.4 正则表达式的全局替换 . . . 33
1.6.5 从文本文件的表格中提取数据 . . . 34
1.6.6 用于管道命令的小片段脚本 . . . 35

2 Debian 软件包管理 36
2.1 Debian 软件包管理的前提 . . . 36
2.1.1 Debian 软件包管理 . . . 36
2.1.2 软件包配置 . . . 36
2.1.3 基本的注意事项 . . . 37
2.1.4 持续升级的生活 . . . 38
2.1.5 Debian 档案库基础 . . . 39
2.1.6 Debian 是 100% 的自由软件 . . . 42
2.1.7 软件包依赖关系 . . . 43
2.1.8 包管理的事件流 . . . 44
2.1.9 对包管理问题的第一个回应 . . . 45
2.1.10 如何挑选 Debian 软件包 . . . 45
2.1.11 怎样和不一致的要求协作 . . . 46
2.2 基础软件包管理操作 . . . 46
2.2.1 `apt` vs. `apt-get` / `apt-cache` vs. `aptitude` . . . 46
2.2.2 命令行中的基础软件包管理操作 . . . 47
2.2.3 aptitude 的交互式使用 . . . 49
2.2.4 aptitude 的按键绑定 . . . 49
2.2.5 aptitude 软件包视图 . . . 50
2.2.6 aptitude 搜索方式选项 . . . 51
2.2.7 aptitude 正则表达式 . . . 51
2.2.8 aptitude 的依赖解决 . . . 53
2.2.9 软件包活动日志 . . . 53
2.3 aptitude 操作范例 . . . 53
2.3.1 查找感兴趣的软件包 . . . 53
2.3.2 通过正则表达式匹配软件包名称来列出软件包 . . . 53
2.3.3 使用正则表达式匹配浏览 . . . 53
2.3.4 完整地清理已删除软件包 . . . 54
2.3.5 调整自动/手动安装状态 . . . 54
2.3.6 全面的系统升级 . . . 55
2.4 高级软件包管理操作 . . . 56
2.4.1 命令行中的高级软件包管理操作 . . . 56
2.4.2 验证安装的软件包文件 . . . 58
2.4.3 预防软件包故障 . . . 58
2.4.4 搜索软件包元数据 . . . 58
2.5 Debian 软件包内部管理 . . . 58
2.5.1 档案库元数据 . . . 58
2.5.2 顶层“Release”文件及真实性 . . . 58
2.5.3 档案库层的“Release”文件 . . . 60

2.5.4 获取用于软件包的元数据 60
2.5.5 APT 的软件包状态 61
2.5.6 aptitude 的软件包状态 61
2.5.7 获取的软件包的本地副本 61
2.5.8 Debian 软件包文件名称 61
2.5.9 dpkg 命令 62
2.5.10 update-alternatives 命令 63
2.5.11 dpkg-statoverride 命令 63
2.5.12 dpkg-divert 命令 63
2.6 从损坏的系统中恢复 63
2.6.1 不兼容旧的用户配置 64
2.6.2 软件包数据缓存错误 64
2.6.3 使用 dpkg 命令进行救援 64
2.6.4 缺少依赖导致的安装失败 64
2.6.5 具有相同文件的不同软件包 65
2.6.6 修复损坏的软件包脚本 65
2.6.7 恢复软件包选择数据 65
2.7 软件包管理技巧 66
2.7.1 上传软件包的是谁？ 66
2.7.2 限制 APT 的下载带宽 66
2.7.3 自动下载和升级软件包 66
2.7.4 更新和向后移植 67
2.7.5 外部软件包档案库 67
2.7.6 不使用 **apt-pinning** 的混合源档案库软件包 67
2.7.7 使用 **apt-pinning** 调整获选版本 68
2.7.8 阻止推荐的软件包的安装 70
2.7.9 使用带有 `unstable` 软件包的 `testing` 版本 70
2.7.10 使用带有 `experimental` 软件包的 `unstable` 版本 71
2.7.11 紧急降级 72
2.7.12 equivs 软件包 72
2.7.13 移植一个软件包到 stable 系统 73
2.7.14 用于 APT 的代理服务器 73
2.7.15 更多关于软件包管理的文档 74

3 系统初始化 75
3.1 启动过程概述 . . . 75
3.1.1 第一阶段：UEFI . . . 75
3.1.2 第二阶段：引载加载程序 . . . 76
3.1.3 第三阶段：迷你 Debian 系统 . . . 77
3.1.4 第四阶段：常规 Debian 系统 . . . 78
3.2 应急救援系统 . . . 78
3.2.1 USB 上的 GRUB UEFI 应急救援系统 . . . 79
3.2.2 USB 上的 Linux 即用救援系统 . . . 79
3.2.3 从 GRUB 启动的 Linux 即用应急救援系统 . . . 80
3.3 Systemd . . . 81
3.3.1 Systemd 初始化 . . . 81
3.3.2 Systemd 登录 . . . 82
3.4 内核消息 . . . 82
3.5 系统消息 . . . 82
3.6 系统管理 . . . 83
3.7 其它系统监控 . . . 83
3.8 系统配置 . . . 83
3.8.1 主机名 . . . 83
3.8.2 文件系统 . . . 83
3.8.3 网络接口初始化 . . . 85
3.8.4 云系统初始化 . . . 85
3.8.5 调整 sshd 服务的个性化例子 . . . 85
3.9 udev 系统 . . . 86
3.10 内核模块初始化 . . . 86

4 认证和访问控制 88
4.1 一般的 Unix 认证 . . . 88
4.2 管理账号和密码信息 . . . 90
4.3 好密码 . . . 90
4.4 设立加密的密码 . . . 90
4.5 PAM 和 NSS . . . 91
4.5.1 PAM 和 NSS 访问的配置文件 . . . 92
4.5.2 现代的集中式系统管理 . . . 92
4.5.3 “为什么 GNU su 不支持 wheel 组” . . . 93
4.5.4 严格的密码规则 . . . 93
4.6 安全认证 . . . 93
4.6.1 确保互联网上的的密码安全 . . . 93
4.6.2 安全 Shell . . . 94

4.6.3 互联网额外的安全方式 . . . 94
4.6.4 root 密码安全 . . . 94
4.7 其它的访问控制 . . . 95
4.7.1 访问控制列表 (ACLs) . . . 95
4.7.2 sudo . . . 96
4.7.3 PolicyKit . . . 96
4.7.4 限制访问某些服务端的服务 . . . 96
4.7.5 Linux 安全特性 . . . 97

5 网络设置 98
5.1 基本网络架构 . . . 98
5.1.1 主机名解析 . . . 98
5.1.2 网络接口名称 . . . 100
5.1.3 局域网网络地址范围 . . . 100
5.1.4 网络设备支持 . . . 101
5.2 现代的桌面网络配置 . . . 101
5.2.1 图形界面的网络配置工具 . . . 101
5.3 没有图像界面的现代网络配置 . . . 102
5.4 现代云网络配置 . . . 102
5.4.1 使用 DHCP 的现代云网络配置 . . . 103
5.4.2 使用静态 IP 的现代云网络配置 . . . 103
5.4.3 使用 Network Manager 的现代云网络配置 . . . 103
5.5 底层网络配置 . . . 103
5.5.1 Iproute2 命令 . . . 103
5.5.2 安全的底层网络操作 . . . 104
5.6 网络优化 . . . 104
5.6.1 找出最佳 MTU . . . 104
5.6.2 WAN TCP 优化 . . . 106
5.7 Netfilter 网络过滤框架 . . . 106

6 网络应用 108
6.1 网页浏览器 . . . 108
6.1.1 伪装用户代理字符串 . . . 109
6.1.2 浏览器扩展 . . . 109
6.2 邮件系统 . . . 109
6.2.1 电子邮件基础 . . . 109
6.2.2 现代邮件服务限制 . . . 110
6.2.3 历史邮件服务端期望 . . . 110
6.2.4 邮件传输代理 (MTA) . . . 111

6.2.4.1 exim4 的配置 . . . 111
6.2.4.2 带有 SASL 的 postfix 配置 . . . 113
6.2.4.3 邮件地址配置 . . . 113
6.2.4.4 基础 MTA 操作 . . . 114
6.3 服务器远程访问和工具 (SSH) . . . 114
6.3.1 SSH 基础 . . . 116
6.3.2 远程主机上的用户名 . . . 116
6.3.3 免密码远程连接 . . . 117
6.3.4 处理其它 SSH 客户端 . . . 117
6.3.5 建立 ssh 代理 . . . 117
6.3.6 从远程主机发送邮件 . . . 118
6.3.7 SMTP/POP3 隧道的端口转发 . . . 118
6.3.8 怎样通过 SSH 关闭远程系统 . . . 118
6.3.9 SSH 故障排查 . . . 118
6.4 打印服务和工具 . . . 118
6.5 其它网络应用服务 . . . 119
6.6 其它网络应用客户端 . . . 120
6.7 系统后台守护进程（daemon）诊断 . . . 120

7 GUI（图形用户界面）系统 122
7.1 GUI（图形用户界面）桌面环境 . . . 122
7.2 GUI（图形用户界面）通信协议 . . . 123
7.3 GUI（图形用户界面）架构 . . . 124
7.4 GUI（图形用户界面）应用 . . . 124
7.5 用户目录 . . . 126
7.6 字体 . . . 126
7.6.1 基础字体 . . . 126
7.6.2 字体栅格化 . . . 128
7.7 沙盒 . . . 128
7.8 远程桌面 . . . 130
7.9 X 服务端连接 . . . 130
7.9.1 X 服务端本地连接 . . . 130
7.9.2 X 服务端远程连接 . . . 130
7.9.3 X 服务端 chroot 连接 . . . 131
7.10 剪贴板 . . . 131

8 国际化和本地化 132
8.1 语言环境 . . . 132
8.1.1 UTF-8 语言环境的基本原理 . . . 132
8.1.2 语言环境的重新配置 . . . 133
8.1.3 文件名编码 . . . 134
8.1.4 本地化信息和翻译文档 . . . 134
8.1.5 语言环境的影响 . . . 134
8.2 键盘输入 . . . 135
8.2.1 Linux 控制台和 X 窗口的键盘输入 . . . 135
8.2.2 Wayland 键盘输入 . . . 136
8.2.3 IBus 支持的输入法 . . . 136
8.2.4 Fcitx 支持的输入法 . . . 136
8.2.5 一个日语的例子 . . . 136
8.3 显示输出 . . . 139
8.3.1 终端配置 . . . 139
8.3.2 东亚环境下宽度有歧义的字符 . . . 140

9 系统技巧 141
9.1 控制台技巧 . . . 141
9.1.1 清晰的记录 shell 活动 . . . 141
9.1.2 screen 程序 . . . 142
9.1.3 在目录间游走 . . . 143
9.1.4 Readline 封装 . . . 143
9.1.5 扫描源代码树 . . . 143
9.2 定制 vim . . . 144
9.2.1 用内部特性定制 vim . . . 144
9.2.2 用外部软件包定制 vim . . . 146
9.3 数据记录和展示 . . . 147
9.3.1 日志后台守护进程（daemon） . . . 147
9.3.2 日志分析 . . . 147
9.3.3 定制文本数据的显示 . . . 148
9.3.4 定制时间和日期的显示 . . . 148
9.3.5 shell 中 echo 的颜色 . . . 148
9.3.6 有颜色输出的命令 . . . 149
9.3.7 记录编辑器复杂的重复操作动作 . . . 149
9.3.8 记录 X 应用程序的图像 . . . 150
9.3.9 记录配置文件的变更 . . . 150
9.4 监控、控制和启动程序活动 . . . 151
9.4.1 进程耗时 . . . 151

9.4.2 调度优先级 . . . 151
9.4.3 ps 命令 . . . 152
9.4.4 top 命令 . . . 152
9.4.5 列出被一个进程打开的文件 . . . 152
9.4.6 跟踪程序活动 . . . 152
9.4.7 识别使用文件和套接字的进程 . . . 153
9.4.8 使用固定间隔重复一个命令 . . . 153
9.4.9 使用文件循环来重复一个命令 . . . 153
9.4.10 从 GUI 启动一个程序 . . . 154
9.4.11 自定义被启动的程序 . . . 155
9.4.12 杀死一个进程 . . . 156
9.4.13 单次任务时间安排 . . . 156
9.4.14 定时任务安排 . . . 156
9.4.15 基于事件的计划任务 . . . 157
9.4.16 Alt-SysRq 键 . . . 157
9.5 系统维护技巧 . . . 158
9.5.1 谁在系统里？ . . . 158
9.5.2 警告所有人 . . . 158
9.5.3 硬件识别 . . . 158
9.5.4 硬件配置 . . . 158
9.5.5 系统时间和硬件时间 . . . 160
9.5.6 声音基础设施 . . . 160
9.5.7 关闭屏幕保护 . . . 161
9.5.8 关闭蜂鸣声 . . . 161
9.5.9 内存使用 . . . 161
9.5.10 系统安全性和完整性检查 . . . 162
9.6 数据存储技巧 . . . 163
9.6.1 硬盘空间使用情况 . . . 163
9.6.2 硬盘分区配置 . . . 163
9.6.3 使用 UUID 访问分区 . . . 164
9.6.4 LVM2 . . . 164
9.6.5 文件系统配置 . . . 165
9.6.6 文件系统创建和完整性检查 . . . 165
9.6.7 通过挂载选项优化文件系统 . . . 166
9.6.8 通过超级块（superblock）优化文件系统 . . . 166
9.6.9 硬盘优化 . . . 166
9.6.10 固态硬盘优化 . . . 167
9.6.11 使用 SMART 预测硬盘故障 . . . 167
9.6.12 通过 $TMPDIR 指定临时存储目录 . . . 167

9.6.13 通过 LVM 扩展可用存储空间 . . . 167
9.6.14 通过挂载另一个分区来扩展可用存储空间 . . . 168
9.6.15 通过“mount --bind”挂载另一个目录来扩展可用存储空间 . . . 168
9.6.16 通过 overlay 挂载（overlay-mounting）另一个目录来扩展可用存储空间 . . . 168
9.6.17 使用符号链接扩展可用存储空间 . . . 168
9.7 磁盘映像 . . . 169
9.7.1 制作磁盘映像文件 . . . 169
9.7.2 直接写入硬盘 . . . 169
9.7.3 挂载磁盘映像文件 . . . 170
9.7.4 清理磁盘映像文件 . . . 171
9.7.5 制作空的磁盘映像文件 . . . 171
9.7.6 制作 ISO9660 镜像文件 . . . 172
9.7.7 直接写入文件到 CD/DVD-R/RW . . . 173
9.7.8 挂载 ISO9660 镜像文件 . . . 173
9.8 二进制数据 . . . 173
9.8.1 查看和编辑二进制数据 . . . 174
9.8.2 不挂载文件系统的情况下操作文件 . . . 174
9.8.3 数据冗余 . . . 174
9.8.4 数据文件恢复和诊断分析 . . . 175
9.8.5 把大文件分成多个小文件 . . . 175
9.8.6 清空文件内容 . . . 175
9.8.7 样子文件 . . . 176
9.8.8 擦除整块硬盘 . . . 176
9.8.9 擦除硬盘上的未使用的区域 . . . 176
9.8.10 恢复已经删除但仍然被打开的文件 . . . 177
9.8.11 查找所有硬链接 . . . 177
9.8.12 不可见磁盘空间消耗 . . . 178
9.9 数据加密提示 . . . 178
9.9.1 使用 dm-crypt/LUKS 加密移动磁盘 . . . 178
9.9.2 使用 dm-crypt/LUKS 挂载加密的磁盘 . . . 179
9.10 内核 . . . 179
9.10.1 内核参数 . . . 179
9.10.2 内核头文件 . . . 180
9.10.3 编译内核和相关模块 . . . 180
9.10.4 编译内核源代码：Debian 内核团队推荐 . . . 181
9.10.5 硬件驱动和固件 . . . 181
9.11 虚拟化系统 . . . 182
9.11.1 虚拟化和模拟器工具 . . . 182
9.11.2 虚拟化工作流 . . . 184
9.11.3 挂载虚拟磁盘映像文件 . . . 184
9.11.4 Chroot 系统 . . . 185
9.11.5 多桌面系统 . . . 186

10 数据管理 **187**
10.1 共享，拷贝和存档 . . . 187
10.1.1 存档和压缩工具 . . . 188
10.1.2 复制和同步工具 . . . 189
10.1.3 归档语法 . . . 189
10.1.4 复制语法 . . . 189
10.1.5 查找文件的语法 . . . 190
10.1.6 归档媒体 . . . 191
10.1.7 可移动存储设备 . . . 192
10.1.8 选择用于分享数据的文件系统 . . . 193
10.1.9 网络上的数据分享 . . . 195
10.2 备份和恢复 . . . 195
10.2.1 备份和恢复策略 . . . 195
10.2.2 实用备份套件 . . . 197
10.2.3 备份技巧 . . . 197
10.2.3.1 GUI（图形用户界面）备份 . . . 198
10.2.3.2 挂载事件触发的备份 . . . 198
10.2.3.3 时间事件触发的备份 . . . 199
10.3 数据安全基础 . . . 200
10.3.1 GnuPG 密钥管理 . . . 200
10.3.2 在文件上使用 GnuPG . . . 201
10.3.3 在 Mutt 中使用 GnuPG . . . 203
10.3.4 在 Vim 中使用 GnuPG . . . 203
10.3.5 MD5 校验和 . . . 203
10.3.6 密码密钥环 . . . 203
10.4 源代码合并工具 . . . 204
10.4.1 从源代码文件导出差异 . . . 204
10.4.2 源代码文件移植更新 . . . 205
10.4.3 交互式移植 . . . 205
10.5 Git . . . 205
10.5.1 配置 Git 客户端 . . . 206
10.5.2 基本的 Git 命令 . . . 206
10.5.3 Git 技巧 . . . 207
10.5.4 Git 参考 . . . 207
10.5.5 其它的版本控制系统 . . . 209

11 数据转换 **210**
11.1 文本数据转换工具 . . . 210
11.1.1 用 iconv 命令来转换文本文件 . . . 210
11.1.2 用 iconv 检查文件是不是 UTF-8 编码 . . . 212
11.1.3 使用 iconv 转换文件名 . . . 212
11.1.4 换行符转换 . . . 212
11.1.5 TAB 转换 . . . 213
11.1.6 带有自动转换功能的编辑器 . . . 213
11.1.7 提取纯文本 . . . 213
11.1.8 高亮并格式化纯文本数据 . . . 215
11.2 XML 数据 . . . 215
11.2.1 XML 的基本提示 . . . 215
11.2.2 XML 处理 . . . 216
11.2.3 XML 数据提取 . . . 217
11.2.4 XML 数据检查 . . . 217
11.3 排版 . . . 217
11.3.1 roff 排版 . . . 217
11.3.2 TeX/LaTeX . . . 218
11.3.3 漂亮的打印手册页 . . . 219
11.3.4 创建手册页 . . . 219
11.4 可打印的数据 . . . 219
11.4.1 Ghostscript . . . 219
11.4.2 合并两个 PS 或 PDF 文件 . . . 220
11.4.3 处理可印刷数据的工具 . . . 220
11.4.4 用 CUPS 打印 . . . 220
11.5 邮件数据转换 . . . 220
11.5.1 邮件数据基础 . . . 222
11.6 图形数据工具 . . . 222
11.6.1 图形数据工具（元软件包） . . . 222
11.6.2 图形数据工具（GUI 图形用户界面） . . . 224
11.6.3 图形数据工具 (CLI 命令行) . . . 224
11.7 不同种类的数据转换工具 . . . 225

12 编程 **226**
12.1 Shell 脚本 . . . 226
12.1.1 POSIX shell 兼容性 . . . 226
12.1.2 Shell 参数 . . . 227
12.1.3 Shell 条件语句 . . . 228
12.1.4 shell 循环 . . . 229

12.1.5 Shell 环境变量 . . . 229
12.1.6 shell 命令行的处理顺序 . . . 230
12.1.7 用于 shell 脚本的应用程序 . . . 231
12.2 解释性语言中的脚本 . . . 231
12.2.1 调试解释性语言代码 . . . 231
12.2.2 使用 shell 脚本的 GUI 程序 . . . 232
12.2.3 定制 GUI（图形用户界面）文件管理器的行为 . . . 233
12.2.4 Perl 短脚本的疯狂 . . . 233
12.3 编译型语言代码 . . . 234
12.3.1 C . . . 234
12.3.2 简单的 C 程序（gcc） . . . 235
12.3.3 Flex —一个更好的 Lex . . . 235
12.3.4 Bison —一个更好的 Yacc . . . 235
12.4 静态代码分析工具 . . . 237
12.5 调试 . . . 237
12.5.1 基本的 gdb 使用命令 . . . 237
12.5.2 调试 Debian 软件包 . . . 239
12.5.3 获得栈帧 . . . 240
12.5.4 高级 gdb 命令 . . . 240
12.5.5 检查库依赖性 . . . 241
12.5.6 动态调用跟踪工具 . . . 241
12.5.7 调试与 X 相关的错误 . . . 241
12.5.8 内存泄漏检测工具 . . . 241
12.5.9 反汇编二进制程序 . . . 241
12.6 编译工具 . . . 242
12.6.1 make . . . 242
12.6.2 Autotools（自动化工具） . . . 243
12.6.2.1 编译并安装程序 . . . 243
12.6.2.2 卸载程序 . . . 243
12.6.3 Meson . . . 243
12.7 Web . . . 244
12.8 源代码转换 . . . 244
12.9 制作 Debian 包 . . . 244

A 附录 246

A.1 Debian 迷宫 . . . 246
A.2 版权历史 . . . 246
A.3 文档格式 . . . 247

List of Tables

1.1 有趣的文本模式程序包列表 4
1.2 软件包信息文档列表 4
1.3 重要目录的用途列表 7
1.4 “ls -l”输出的第一个字符列表 8
1.5 chmod(1) 命令文件权限的数字模式 9
1.6 **umask** 值举例 10
1.7 关于文件访问的由系统提供的著名组列表 11
1.8 著名的由系统提供用于特定命令运行的组列表 11
1.9 时间戳类型列表 12
1.10 特殊设备文件列表 15
1.11 MC 快捷键绑定 17
1.12 MC 中对回车键的响应 18
1.13 shell 程序列表 19
1.14 bash 的按键绑定列表 20
1.15 Debian 上的鼠标操作和相关按键操作列表 21
1.16 基本的 Vim 按键列表 22
1.17 基本的 Unix 命令列表 24
1.18 语言环境值的 3 个部分 25
1.19 语言环境推荐列表 26
1.20 ”$HOME” 变量值列表 27
1.21 Shell glob 模式 27
1.22 命令的退出代码 28
1.23 Shell 命令常见用法 29
1.24 预定义的文件描述符 30
1.25 BRE 和 ERE 中的元字符 32
1.26 替换表达式 32
1.27 管道命令的小片段脚本列表 35

2.1 Debian 软件包管理工具列表 37
2.2 Debian 档案库站点列表 40

2.3 Debian 归档区域（area）列表 . . . 41
2.4 套件和代号的关系 . . . 41
2.5 解决特定软件包问题的主要网站 . . . 45
2.6 使用 apt(8), aptitude(8) 和 apt-get(8) / apt-cache(8) 的命令行基本软件包管理操作 . . . 48
2.7 aptitude(8) 中重要的命令选项 . . . 48
2.8 aptitude 的按键绑定 . . . 49
2.9 aptitude 视图 . . . 50
2.10 标准软件包视图的分类 . . . 50
2.11 aptitude 正则表达式 . . . 52
2.12 软件包活动日志文件 . . . 53
2.13 高级软件包管理操作 . . . 57
2.14 Debian 档案库元数据的内容 . . . 59
2.15 Debian 软件包的名称结构 . . . 61
2.16 Debian 软件包名称中每一个组件可以使用的字符 . . . 62
2.17 `dpkg` 创建的重要文件 . . . 62
2.18 用于 **apt-pinning** 技术的值得注意的 Pin-Priority 值列表。 . . . 69
2.19 Debian 档案库的专用代理工具 . . . 74

3.1 引导加载程序列表 . . . 76
3.2 `/boot/grub/grub.cfg` 文件上面部分菜单条目意义 . . . 77
3.3 Debian 系统启动工具列表 . . . 78
3.4 内核错误级别表 . . . 82
3.5 典型的 `journalctl` 命令片段列表 . . . 83
3.6 典型的 `systemctl` 命令片段列表 . . . 84
3.7 `systemd` 下其它零星监控命令列表 . . . 85

4.1 pam_unix(8) 使用的 3 个重要配置文件 . . . 88
4.2 “`/etc/passwd`”第二项的内容 . . . 89
4.3 管理账号信息的命令 . . . 90
4.4 生成密码的工具 . . . 91
4.5 PAM 和 NSS 系统中重要的软件包 . . . 91
4.6 PAM 和 NSS 访问的配置文件 . . . 92
4.7 安全和不安全的服务端口列表 . . . 94
4.8 提供额外安全方式的工具列表 . . . 94

5.1 网络配置工具一览表 . . . 99
5.2 网络地址范围列表 . . . 101
5.3 从旧的 `net-tools` 命令集到新的 `iproute2` 命令集转换表 . . . 104
5.4 底层网络命令列表 . . . 104
5.5 网络优化工具列表 . . . 105

5.6 最佳 MTU 值的基本指引方法 . . . 105
5.7 防火墙工具列表 . . . 106

6.1 网页浏览器列表 . . . 108
6.2 邮件用户代理列表 (MUA) . . . 110
6.3 基础的邮件传输代理相关的软件包列表 . . . 111
6.4 重要的 postfix 手册页列表 . . . 113
6.5 与邮件地址相关的配置文件列表 . . . 113
6.6 基础 MTA 操作列表 . . . 115
6.7 服务器远程访问和工具列表 . . . 115
6.8 SSH 配置文件列表 . . . 116
6.9 SSH 客户端启动例子列表 . . . 116
6.10 其它平台上免费 SSH 客户端列表 . . . 117
6.11 打印服务和工具列表 . . . 119
6.12 其它网络应用服务列表 . . . 119
6.13 网络应用客户端列表 . . . 120
6.14 常用 RFC 列表 . . . 121

7.1 桌面环境列表 . . . 122
7.2 著名的 GUI 架构软件包列表 . . . 124
7.3 著名的的 GUI（图形用户界面）应用列表 . . . 125
7.4 著名的 TrueType 和 OpenType 字体列表 . . . 127
7.5 著名的字体环境和相关软件包列表 . . . 127
7.6 著名的沙盒环境和相关软件包列表 . . . 129
7.7 著名的远程访问服务端列表 . . . 129
7.8 连接到 X 服务端的方式 . . . 130
7.9 操作字符剪贴板相关程序列表 . . . 131

8.1 IBus 和它的插件软件包列表 . . . 137
8.2 Fcitx5 和它的插件软件包列表 . . . 138

9.1 支持控制台活动的程序列表 . . . 141
9.2 screen 键绑定列表 . . . 143
9.3 `vim` 的初始化信息 . . . 147
9.4 系统日志分析软件列表 . . . 148
9.5 使用时间样式值的"`ls -l`" 命令的时间和日期的显示例子 . . . 149
9.6 图形图像处理工具列表 . . . 150
9.7 记录配置历史的软件包列表 . . . 150
9.8 监控和控制程序活动工具列表 . . . 151
9.9 调度优先级值列表 . . . 152

9.10 ps 命令样式列表 . . . 152
9.11 kill 命令常用信号列表 . . . 156
9.12 著名的 SAK 命令键列表 . . . 157
9.13 硬件识别工具列表 . . . 159
9.14 硬件配置工具列表 . . . 159
9.15 声音软件包 . . . 161
9.16 关闭屏幕保护命令列表 . . . 161
9.17 报告的内存大小 . . . 162
9.18 用于系统安全性和完整性检查的工具 . . . 162
9.19 硬盘分区管理软件包 . . . 164
9.20 文件系统管理包列表 . . . 165
9.21 查看和修改二进制数据的软件包列表 . . . 174
9.22 不挂载文件系统操作文件的软件包列表 . . . 174
9.23 向文件添加数据冗余的工具列表 . . . 174
9.24 数据文件恢复和诊断分析软件包列表 . . . 175
9.25 数据加密工具列表 . . . 178
9.26 Debian 系统内核编译需要安装的主要软件包列表 . . . 180
9.27 虚拟化工具列表 . . . 183

10.1 存档和压缩工具列表 . . . 188
10.2 复制和同步工具列表 . . . 189
10.3 典型使用场景下可移动存储设备可选择的文件系统列表 . . . 194
10.4 典型使用场景下可选择的网络服务列表 . . . 195
10.5 实用备份程序套件列表 . . . 197
10.6 数据安全基础工具列表 . . . 200
10.7 GNU 隐私卫士密钥管理命令的列表 . . . 200
10.8 信任码含义列表 . . . 201
10.9 在文件上使用的 GNU 隐私卫士的命令列表 . . . 202
10.10 源代码合并工具列表 . . . 204
10.11 git 相关包和命令列表 . . . 205
10.12 主要的 Git 命令 . . . 207
10.13 Git 技巧 . . . 208
10.14 其它版本控制系统工具列表 . . . 209

11.1 文本数据转化工具列表 . . . 210
11.2 编码值和用法的列表 . . . 211
11.3 不同平台的换行符样式列表 . . . 212
11.4 `bsdmainutils` 和 `coreutils` 包中的用于转换 TAB 的命令列表 . . . 213
11.5 用于提取纯文本数据的工具列表 . . . 214

11.6 高亮纯文本数据的工具列表 . . . 214
11.7 XML 预定义实体列表 . . . 215
11.8 XML 工具列表 . . . 216
11.9 DSSSL 工具列表 . . . 216
11.10XML 数据提取工具列表 . . . 217
11.11XML 美化打印工具列表 . . . 217
11.12排版工具的列表 . . . 218
11.13创建手册页的工具列表 . . . 219
11.14Ghostscript PostScript 解释器列表 . . . 220
11.15处理可印刷数据的工具列表 . . . 221
11.16有助于邮件数据转换的软件包列表 . . . 221
11.17图形数据工具列表（元软件包） . . . 223
11.18图形数据工具（GUI 图形用户界面）列表 . . . 223
11.19图形数据工具 (CLI 命令行) 列表 . . . 224
11.20不同种类的数据转换工具列表 . . . 225

12.1 典型 bashism 语法列表 . . . 227
12.2 shell 参数列表 . . . 227
12.3 shell 参数展开列表 . . . 228
12.4 重要的 shell 参数替换列表 . . . 228
12.5 在条件表达式中进行文件比较 . . . 229
12.6 在条件表达式中进行字符串比较 . . . 229
12.7 包含用于 shell 脚本的小型应用程序的软件包 . . . 231
12.8 解释器相关软件包列表 . . . 232
12.9 对话（dialog ）程序列表 . . . 232
12.10编译相关软件包列表 . . . 234
12.11兼容 Yacc 的 LALR 解析器生成器列表 . . . 235
12.12静态代码分析工具的列表 . . . 238
12.13调试软件包列表 . . . 238
12.14高级 gdb 命令列表 . . . 240
12.15内存泄漏检测工具的列表 . . . 241
12.16编译工具软件包列表 . . . 242
12.17自动变量的列表 . . . 242
12.18变量扩展的列表 . . . 242
12.19源代码转换工具列表 . . . 244

Abstract

这本书是自由的；你可以在与 Debian 自由软件指导方针（DFSG）兼容的任意版本的 GNU 通用公共许可证的条款下重新分发和修改本书。

序言

Debian 参考手册（版本 2.153） (2026-11-29 05:38:00 UTC) 旨在为作为一份安装后用户指南，为 Debian 系统的使用与管理提供宽泛的概览。

本书的目标读者：愿意学习 shell 脚本，但是不准备为了理解 GNU/Linux 系统是如何运作的而阅读其所有 C 语言源代码的人。

如需系统安装指导信息，请见：

- Debian GNU/Linux 当前稳定版安装手册
- Debian GNU/Linux 当前测试版安装手册

免责声明

所有担保条款具有免责效力。所有商标均为其各自商标所有者的财产。

Debian 系统本身是一个变化的事物。这导致其文档难于及时更新并且正确。虽然是以 Debian 系统当前的 `testing`（测试版）作为写作该文档的基础，但当你阅读本文的时候，部分内容仍然可能已经过时。

请把本文档作为第二参考。本文档不能够代替任何官方指导手册。文档作者和文档贡献者对在本文档中的错误、遗漏或歧义，不承担责任后果。

什么是 **Debian**

Debian 项目是一个由个人组成的团体，该团体的成员均把创建一个自由操作系统作为共同事业。Debian 的发布具有下文所列的特征。

- 承诺软件自由：Debian 社群契约和 Debian 自由软件指导方针（DFSG）
- 基于因特网上无酬劳的志愿者的工作发布：https://www.debian.org
- 大量预编译的高质量软件包
- 专注于稳定性和安全性，同时易于获取安全更新
- 在 `testing` 版仓库中注重软件包最新版本的平滑升级
- 支持大量硬件架构

Debian 系统中的自由软件来自 GNU, Linux, BSD, X, ISC, Apache, Ghostscript, Common Unix Printing System，Samba, GNOME, KDE, Mozilla, LibreOffice, Vim, TeX, LaTeX, DocBook, Perl, Python, Tcl, Java, Ruby, PHP, Berkeley DB, MariaDB, PostgreSQL, SQLite, Exim, Postfix, Mutt, FreeBSD, OpenBSD, Plan 9 以及许多更加独立的自由软件项目。Debian 将上述各种各样的自由软件集成到一个系统里面。

关于本文档

指导原则

写作本文档时，遵循下列指导原则。

- 仅提供概览，而忽略边界情况。（**Big Picture** 原则）
- 保持文字简短紧凑。（**KISS** 原则）
- 不重复造轮子。（使用链接指向已有参考）
- 专注于使用非图形的工具和控制台。（使用 **shell** 例子）
- 保持客观。（使用 popcon 等等。）

提示
我试图阐明操作系统底层和体系结构的各方面内容。

预备知识



警告
阅读本文档，你需要通过自己的努力去查找本文档未提及的问题答案。本文档仅仅提供有效的起点。

你必须自己从以下原始材料查找解决方案。

- Debian 网站（https://www.debian.org）上的通用信息
- "`/usr/share/doc/`*`package_name`*" 目录下的文档
- Unix 风格的 **manpage**: "`dpkg -L` *`package_name`* `|grep '/man/man.*/'`"
- GNU 风格的 **info page**: "`dpkg -L` *`package_name`* `|grep '/info/'`"
- 错误报告：https://bugs.debian.org/*package_name*
- Debian Wiki（https://wiki.debian.org/）用于变化和特定的话题
- 国际开放标准组织的的单一 UNIX 规范 UNIX 系统主页上
- 自由的百科全书：维基百科（https://www.wikipedia.org/）
- Debian 管理员手册
- 来自 Linux 文档项目 (TLDP) 的 HOWTO

注意
软件包的详细文档，你需要安装软件包名用"`-doc`" 作为后缀名的相应文档包来得到。

排版约定

本文通过如下使用 bash(1) shell 命令例子的简要方式来提供信息。

```
# command-in-root-account
$ command-in-user-account
```

这些 shell 提示符区分了所使用的帐户。为了可读性，在本手册中 shell 提示符相关的环境变量被设置为“PS1='\$'”和“PS2=' '”。这与实际安装的系统所使用的 shell 提示符很有可能会不同。

所有的命令示例都是运行在英语语言环境“LANG=en_US.UTF8”下的。请不要期望命令示例中像 *command-in-root-accoun* 和 *command-in-user-account* 这样的占位符会被翻译。这么做是为了保持所有翻译示例都是最新的。

注意
参见在 bash(1) 中对环境变量“$PS1”和“$PS2”的解释。

要求系统管理员执行的操作，须用祈使句描述，如“在 shell 中输入命令字符串后，键入 Enter 键。”

这些描述列或类似信息在表格有一个名词短语，后面会紧跟软件包短描述，这些短语会省略掉前面的”a” 和”the”。它们也可以包含一个不定式短语作名词短语，在联机帮助的短命令描述约定后面不带”to”。有些人可能觉得这看起来有点可笑，这里故意保留这种风格是为了让文档看起来尽可能的简单。这些名词短语在短命令描述约定里并不会采用首字母大写的方式。

注意
无论专有名词和命令名位于何处，保持其英文字母大小写不变。

在文本段落中引用的命令片断由双引号括起来的打印机字体进行标记，就像”`aptitude safe-upgrade`”。

在文本段落中引用的来自配置文件的文本数据由双引号括起来的打印机字体进行标记，就像”`deb-src`”。

命令和置于其后的圆括号内的手册页章节数（可选），由打字机字体进行标记，就像 bash(1)。我们鼓励您这样通过输入以下命令来获得信息。

```
$ man 1 bash
```

manpage 会在打字机字体后面括号中显示 manpage 页章节号，如 sources.list(5)。建议你通过键入以下命令来获取帮助信息。

```
$ man 5 sources.list
```

info page 页是由双引号之间的打字机字体来标注，如 `info make`。建议你通过键入以下的命令来获取帮助信息。

```
$ info make
```

文件名将由双引号括起来的打印机字体进行标记，就像”`/etc/passwd`”。对于配置文件，你可以输入下列的命令来获取它的信息。

```
$ sensible-pager "/etc/passwd"
```

目录名将由双引号括起来的打印机字体进行标记，就像”`/etc/apt/`”。你可以输入下列的命令来浏览目录的内容。

```
$ mc "/etc/apt/"
```

软件包名称将由打印机字体进行标记，就像 `vim`。你可以输入下列的命令来获取它的信息。

```
$ dpkg -L vim
$ apt-cache show vim
$ aptitude show vim
```

一个文档可能通过文件名来指示它的位置，文件名将由双引号括起来的打印机字体进行标记，就像”/usr/share/doc/bas
和”/usr/share/doc/base-passwd/users-and-groups.html”，或通过它的 URL，就像 https://www.debian.org。
你可以通过输入下列命令来阅读文档。

```
$ zcat "/usr/share/doc/base-passwd/users-and-groups.txt.gz" | sensible-pager
$ sensible-browser "/usr/share/doc/base-passwd/users-and-groups.html"
$ sensible-browser "https://www.debian.org"
```

环境变量将由双引号括起来的打印机字体进行标记，并带有”$” 前缀，就像”$TERM”。你可以输入下列命令来获取它的当前值。

```
$ echo "$TERM"
```

popcon 流行度

popcon 数据被用来客观地衡量每个包的流行度。它的下载时间为 2026-11-28 09:48:51 UTC，包含了超过 229395 个二进制软件包和 26 个架构的全部 299780 份提交。

注意
请注意 amd64 不稳定（unstable）版的软件仓库中只包含当前 78709 软件包。popcon 数据包含许多旧系统安装报告。

以“V:”开头表示“votes”的 popcon 数值计算方式为“1000 * （当前运行在 PC 上的包的 popcon 提交）/ （总的 popcon 提交）”。

以“I:”开头表示“安装数”的 popcon 数值计算方式为“1000 * （当前安装在 PC 上的包的 popcon 提交）/ （总的 popcon 提交）”。

注意
流行度评比 popcon 数据不应视为对包的重要性的绝对度量。有许多因素可以影响统计数据。例如，参与流行度评比的某些系统可能有像“/usr/bin”的目录，挂载的时候带“noatime”选项以提升系统性能，这样的系统有效的禁用了“投票（vote）”功能。

软件包大小

软件包的大小数据同样表明了对每个包的客观衡量。它基于“apt-cache show”或“aptitude show”命令（目前在 amd64 架构的不稳定版）报告的“安装大小：”。报告的大小单位是 KiB（Kibibyte= 表示 1024 Bytes 的单位）。

注意
包大小是一个小数值的包可能显示了这个在“不稳定”版的包是一个虚拟包，它包含关于依赖关系的重要内容，会安装其他的包。虚拟包使能平稳过度或分割一个包。

注意
包大小后面跟着”(*)” 表明这个软件包在不稳定版本中是缺失的同时使用了实验性版本中的软件包大小来替代。

给本文档报告 Bug

如果你发现本文档有任何问题，请使用 reportbug(1) 向 debian-reference 软件包报告 bug。对纯文件版本或源代码的改进建议，请使用”diff -u” 包含在 bug 报告里面。

一些对新使用者的提醒

这里给出对新用户的一些提醒信息：

- 备份你的数据
  - 参见第 10.2 节。
- 妥善保存你的密码和安全信息
- KISS（保持简单而傻瓜式）
  - 不要在系统中过度设计（overengineering）
- 阅读你的日志文件
  - 第一条错误信息才是最重要的
- RTFM（阅读手册与指导）
- 在问问题前，先在互联网上搜索
- 当不是必须要使用 root 的时候，就不要使用 root
- 不要胡乱折腾软件包管理系统
- 不要输入任何你不理解的命令
- （在完全地检查过安全问题之前）不要随意修改文件权限
- 在测试过你所做的修改之前不要关闭 root shell
- 总是保留一个替代的启动媒介 (U 盘、CD-ROM, ⋯)

一些对新使用者的引导

从 Debian 邮件列表来的一些有趣引文，说不定可以帮助新使用者启蒙。

- “这是 Unix。它给你足够的绳索来吊死你自己。”--- Miquel van Smoorenburg <miquels at cistron.nl>
- “Unix 是用户友好的⋯⋯它仅仅选择谁是它的朋友。”--- Tollef Fog Heen <tollef at add.no>

维基百科文章”Unix 哲学” 列出了一些有趣的指导。

Chapter 1

GNU/Linux 教程

我认为学习一个计算机系统，就像学习一门新的外语。虽然教程和文档是有帮助的，但你必须自己练习。为了帮助你平滑起步，我详细说明一些基本要点。

Debian GNU/Linux 中最强大的设计来自 Unix 操作系统，一个多用户多任务的操作系统。你必须学会利用这些特性以及 Unix 和 GNU/Linux 的相似性。

别回避面向 Unix 的文档，不要只是依赖于 GNU/Linux 文档，这样做会剥夺你了解许多有用的信息。

注意
如果你在任何类 Unix 系统中使用过一段时间的命令行工具，你可能已经掌握了这份文档中的内容。那请把它当做一个实战检验和回顾。

1.1 控制台基础

1.1.1 shell 提示符

启动系统之后，如果你没有安装 GUI（例如 GNOME 或者 KDE），那么你会看到字符登录界面。假设你的主机名为 `foo`，那么登录提示符将如下所示。

如果你安装了一个 GUI 环境，那么你仍然能够用 Ctrl-Alt-F3 进入基于字符的登录提示符，同时你能通过 Ctrl-Alt-F2 回到 GUI 环境（更多详情请参阅下文第 1.1.6 节）。

```
foo login:
```

在登录提示符下，你输入你的用户名，例如 `penguin`，然后按回车键，接下来输入你的密码并再次按回车键。

注意
遵循 Unix 传统，Debian 系统下的用户名和密码是大小写敏感的。用户名通常由小写字母组成。第一个用户账号通常在安装期间进行创建。额外的用户账号由 root 用户用 adduser(8) 创建。

系统以保存在"`/etc/motd`" 中的欢迎信息（Message Of The Day）来开始，同时显示一个命令提示符。

```
Debian GNU/Linux 12 foo tty3

foo login: penguin
Password:
```

```
Linux foo 6.5.0-0.deb12.4-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.5.10-1~bpo12+1 (2023-11-23) ↩
     x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Last login: Wed Dec 20 09:39:00 JST 2023 on tty3
foo:~$
```

现在，你就在 shell 下。shell 解析你的命令。

1.1.2 GUI 下的 shell 提示符

如果你在安装 Debian 的过程中，安装了一个 GUI 环境，那么你在启动系统后将使用图形登录界面。输入你的用户名和密码可以登录到非特权用户帐号。使用 Tab 键（跳格键）可以在用户名和密码之间移动，也可以使用鼠标主要键点击。

要在 GUI（图形用户界面）环境下获得 shell 提示符，你必须启动一个 x 终端模拟器程序，例如 gnome-terminal(1)、konsole(1) 或 xterm(1)。在 GNOME 桌面环境下，你可以按超级键（Windows 键），在搜索提示里输入”terminal” 来打开终端。

在其它一些桌面系统（如 `fluxbox`）下面，可能没有明显的开始菜单入口。如果是这种情况，试下右击桌面屏幕并希望能有弹出菜单。

1.1.3 root 账户

root 账户也被称作超级用户或特权用户。用这个账户，你能够履行下面的系统管理任务。

- 读、写和删除系统上的任何文件，不顾它们的文件权限
- 设置系统上任何文件的所有者和权限
- 设置系统上任何非特权用户的密码
- 免用户密码登录任何帐户

无限权力的 root 账户，要求你慎重和负责任的使用。



警告
千万不要和其他人共享 root 密码.

注意
一个文件（包括硬件设备，如 CD-ROM 等，这些对 Debian 系统来说都只是一个文件）的权限可能会导致非 root 用户无法使用或访问它。虽然在这种情况下，使用 root 帐户是一个快速的方法，但正确的解决方法应该是对文件权限和用户组的成员进行合适的设置（参见第 1.2.3 节）。

1.1.4 root shell 提示符

这里有一些基本的方法可以让你在输入 root 密码后获得 root 的 shell 提示符。

- 在字符界面的登录提示符，键入 `root` 作为用户名登录。
- 在任意用户的 shell 提示符下输入“`su -l`”。
  - 这不会保存当前用户的环境设定。
- 在任意用户的 shell 提示符下输入“`su`”。
  - 这会保存当前用户的一些环境设定。

1.1.5 GUI 系统管理工具

如果你的桌面菜单没有使用适当权限启动 GUI（图形用户界面）的自动化管理工具，你可以在终端模拟器（例如 gnome-terminal(1)、konsole(1) 或 xterm(1)）中 root 的 shell 提示符下启动它。参见第 1.1.4 节和第 7.9 节。



警告
永远不要在显示管理器（例如 gdm3(1)）的提示符下输入 `root` 来使用 root 账户启动 GUI 显示/会话管理器。永远不要在显示关键信息的 X Window 下运行不受信任的远程 GUI 程序，因为它可能会监听你的 X 屏幕。

1.1.6 虚拟控制台

在默认的 Debian 系统中，有 6 个可切换的类 VT100 字符控制台，可以直接在 Linux 主机上启动 shell。除非你处于 GUI 环境下，否则你可以同时按下左 `Alt` 键和 `F1`—`F6` 之一的键在虚拟控制台间切换。每一个字符控制台都允许独立登录账户并提供多用户环境。这个多用户环境是伟大的 Unix 的特性，很容易上瘾。

如果你处于 GUI 环境中，你可以通过 `Ctrl-Alt-F3` 键前往字符控制台 3，也就是同时按下左 `Ctrl` 键、左 `Alt` 键和 `F3` 键。你可以按下 `Alt-F2` 回到 GUI 环境，它一般运行在虚拟控制台 2。

你也可以使用命令行切换到另一个虚拟控制台，例如切换到控制台 3。

```
# chvt 3
```

1.1.7 怎样退出命令行提示符

在命令行输入 `Ctrl-D`，即同时按下左侧`-Ctrl-`键和 `d-`键，即可关闭 shell 活动。如果你正处于字符控制台，你将会返回到登录提示行。尽管这些控制字符“control D” 使用了大写字母，你并不需要按住 Shift-键。`Ctrl-D` 也可以简写为 `^D`。或者，你也可以键入“exit”退出命令行。

如果你位于 `x` 终端模拟器 (1) 中，你可以使用这个关闭 `x` 终端模拟器窗口。

1.1.8 怎样关闭系统

就像任何其他的现代操作系统一样，Debian 会通过内存中的缓存数据进行文件操作以提高性能，因此在电源被安全地关闭前需要适当的关机过程，通过将内存中的数据强制写入硬盘来维持文件的完整性。如果软件的电源控制可用，那么关机过程中会自动关闭系统电源。（否则，你可能需要在关机过程之后按电源键几秒钟。）

在普通多用户模式模式下，可以使用命令行关闭系统。

```
# shutdown -h now
```

在单用户模式下，可以使用命令行关闭系统。

```
# poweroff -i -f
```

参见第 6.3.8 节。

1.1.9 恢复一个正常的控制台

当做了一些滑稽的事（例如“cat 二进制文件”）后，屏幕会发狂，你可以在命令行输入“reset”。你可能无法在屏幕上看到你输入的命令。你也可以输入“clear”来清屏。

1.1.10 建议新手安装的额外软件包

尽管连无需任何桌面环境的 Debian 系统最小安装都提供了基本的 Unix 功能，但对新手而言，使用 apt-get(8) 安装一些基于字符终端的命令行和 curses 软件包（例如 mc 和 vim）依旧是一个不错的主意。

```
# apt-get update
 ...
# apt-get install mc vim sudo aptitude
 ...
```

如果你已经安装了这些软件包，那么不会有新的软件包被安装。

软件包	流行度	大小	说明
mc	V:42, I:172	1590	文本模式的全屏文件管理器
sudo	V:739, I:870	6817	给普通用户授予部分 root 权限的程序
vim	V:73, I:330	4415	Unix 文本编辑器 Vi 的改进版，一个程序员的文本编辑器（标准版）
vim-tiny	V:55, I:979	1931	Unix 文本编辑器 Vi 的改进版，一个程序员的文本编辑器（精简版）
emacs-nox	V:3, I:12	46564	GNU 项目的 Emacs，基于 Lisp 的扩展文本编辑器
w3m	V:11, I:125	2853	文本模式的万维网浏览器
gpm	V:8.0, I:8.7	524	Unix 风格的文本控制台复制粘贴工具（守护进程）

Table 1.1: 有趣的文本模式程序包列表

您也可以考虑阅读一些其他的信息文档。

软件包	流行度	大小	说明
doc-debian	I:887	185	Debian 项目文档，（Debian 常见问题）和其它文档
debian-policy	I:6.4	5147	Debian 策略手册和相关文档
developers-reference	V:0.2, I:2.0	2656	Debian 开发者指导方针和信息
debmake-doc	I:0.32	11807	Debian 维护者手册
debian-history	I:0.49	6513	Debian 项目历史
debian-faq	I:886	791	Debian 常见问题

Table 1.2: 软件包信息文档列表

你可以用下面的命令安装这些包。

```
# apt-get install package_name
```

1.1.11 额外用户账号

如果你不想用你自己的主用户账户来进行下面的练习操作，你可以使用下面的方式创建一个练习用户账户，比如说，创建一个用户名为 `fish` 的账号。

```
# adduser fish
```

回答所有问题。

这将创建一个名为 `fish` 的新账号。在你练习完成后，你可以使用下面的命令删除这个用户账号和它的用户主目录。

```
# deluser --remove-home fish
```

在非 Debian 和特别的 Debian 系统，上面的活动需要使用底层的 useradd(8) 和 userdel(8) 工具代替。

1.1.12 sudo 配置

对于典型的单用户工作站，例如运行在笔记本电脑上的桌面 Debian 系统，通常简单地配置 sudo(8) 来使为非特权用户（例如用户 `penguin`）只需输入用户密码而非 root 密码就能获得管理员权限。

```
# echo "penguin  ALL=(ALL) ALL" >> /etc/sudoers
```

另外，可以使用下列命令使非特权用户（例如用户 `penguin`）无需密码就获得管理员权限。

```
# echo "penguin  ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers
```

这些技巧只对你管理的单用户工作站中那个唯一的用户有用。



警告
在多用户工作站中不要建立这样的普通用户账户，因为它会导致非常严重的系统安全问题。



小心
在上述例子中，用户 `penguin` 的密码及账号要有和 root 账号密码同样多的保护。
在这种情况下，管理员权限被赋予那些有权对工作站进行系统管理任务的人。永远不要让你的公司行政管理部门或你的老板进行管理（例如给予他们权限），除非他们获得了授权并有这样的能力。

注意
为了对受限的设备和文件提供访问权限，你应该考虑使用组来提供受限访问，而不是通过 sudo(8) 来使用 `root` 权限。
随着越来越细致周密的配置，sudo(8) 可以授予一个共享系统上的其它用户有限的管理权限而不共享 root 密码。这可以帮助对有多个管理员的主机进行责任追究，你可以了解到是谁做什么。另一方面，你可能不想任何人有这样的权限。

1.1.13 动手时间

现在你已经准备好在 Debian 系统上开工了，只要你使用非特权用户账号就不会有风险。

我们使用下面的方式，把 Debian 系统当作一个类 Unix 系统来学习。

- 第 1.2 节 (基本概念)
- 第 1.3 节 (生存方式)
- 第 1.4 节 (基本方式)
- 第 1.5 节 (shell 机制)
- 第 1.6 节 (文本处理方式)

1.2 类 Unix 文件系统

在 GNU/Linux 和其他类 Unix 操作系统中，文件被组织到目录中。所有的文件和目录排放在以“/”为根的巨大的树里。叫它树是因为如果你画出文件系统，它看起来就像一棵树，但是它是颠倒过来的。

这些文件和目录可以分散在多个设备中。mount(8) 用于把某个设备上找到的文件系统附着到巨大的文件树上。相反的，umount(8) 把它再次分离。在最近的 Linux 内核里，mount(8) 带某些参数时可以把文件树的一部分绑定到另外的地方，或者可以把文件系统挂载为共享的、私有的、从设备、或不可绑定的。对每个文件系统支持的挂载选项可以在`/usr/share/doc/linux-doc-*/Documentation/filesystems/`找到。

Unix 系统上叫做目录，某些其他系统上叫做文件夹。请同样留意，在任何 Unix 系统上，没有的驱动器的概念，例如“`A:`”。这只有一个文件系统，并且所有东西都包含在内。这相对于 Windows 来说是一个巨大的优点。

1.2.1 Unix 文件基础

下面是一些 Unix 文件基础。

- 文件名是区分大小写的。也就是说，”`MYFILE`” 和”`MyFile`” 是不同的文件。
- 根目录意味着文件系统的根，简单的称为“`/`”，不要把它跟 root 用户的家目录“`/root`”混淆了。
- 每个目录都有一个名字，它可以包含任意字母或除了`/`”以外的符号。根目录是个特例。它的名字是“`/`”(称作“斜线”或“根目录”)，并且它不能被重命名。
- 每个文件或目录都被指定一个全限定文件名，绝对文件名，或路径，按顺序给出必须经过的目录从而到达相应目录。这三个术语是同义的。
- 所有的全限定文件名以“`/`”目录开始，并且在每个目录或文件名之间有一个“`/`”。第一个“`/`”是最顶层目录，其他的“`/`”用于分隔跟着的子目录。直到到达最后的入口，即实际文件的名称。这些话可能会令人困惑。用下面这个全限定文件名作为例子：“`/usr/share/keytables/us.map.gz`”。不过，人们也把它的基名“`us.map.gz`”单独作为文件名。
- 根目录有很多分支，例如“`/etc/`”和“`/usr/`”。这些子目录依次分出更多的子目录，例如“`/etc/systemd/`”和“`/usr/local/`”。这整体叫做“目录树”。你可以把一个绝对文件名想象成从“`/`”这棵树的基到某个分支（一个文件）的结尾的一条路径。你也听到人们谈论目录树，就好像它是一个包含所有直系后代的“家庭”树的一个图，这个图叫做根目录（“`/`”）：因此子目录有父目录，并且一条路径显示了一个文件完整的祖先。也有相对路径从其他地方开始，而不是从根目录。你应该还记得目录“`..`/”指向父目录。这个术语也适用于其他类似目录的结构，如分层数据结构。
- 对于一个物理设备，是没有一个特定的目录路径名来对应的组成部分. 这不同于 RT-11, CP/M,OpenVMS,MS-DOS,AmigaOS, 以及微软的 Windows，这些系统存在一个路径包含了一个设备名字，比如”`C:\`”。(尽管如此, 路径条目确实存在引用了物理设备作为正常的文件系统的一部分. 参考第 1.2.2 节。)

注意
虽然你可以在文件名中使用任意的字幕或者符号, 但是在实际情况下这样做是一个坏主意. 最好避免使用一些在命令行里面含有特殊意义的字符, 比如空格, 制表符, 换行符, 和其它的特殊字符: `{ } ( ) [ ] ' ` " \ / >< | ; ! #&^ * % @ $`. 如果你想有一个区分度良好的命名, 比较好的选择是利用时期, 连字符和下划线. 你也可以每个单词的首字母大写, 这叫大驼峰命名法, 比如这样"`LikeThis`". 经验丰富的 Linux 用户会趋向于在文件名中不使用空格.

注意
这个"root" 可能既表示" 超级用户 root" 又表示" 根目录"(/root) . 应该根据上下文确定它的用法.

注意
单词 **path** 不仅表示包含全限定文件名, 也可能表示命令搜索的路径. 通常路径真实的意思是需要通过上下文来明确.

关于文件层次的最佳详细实践在文件系统层次标准 ("`/usr/share/doc/debian-policy/fhs/fhs-2.3.txt.gz`" 和 `hier` (7)). 你应该记住以下的一些标准作为开始学习的步骤.

目录	目录用途
`/`	根目录
`/etc/`	系统范围的配置文件
`/var/log/`	系统日志文件
`/home/`	所有非特权用户的用户目录

Table 1.3: 重要目录的用途列表

1.2.2 文件系统深入解析

按照 **UNIX** 系统的传统，Debian GNU / Linux 的文件系统是在物理数据存储设备诸如磁盘或其他存储设备上，与硬件设备的交互，如控制台和远程串口终端都是以统一的方式呈现在"`/ dev /`"下面。

每个文件、目录、命名管道（一种两个程序间共享数据的方法）或 Debian GNU/Linux 系统上的物理设备都有一个叫做索引节点（inode）的数据结构，描述了其相关属性，例如拥有它的用户（所有者），它属于的组，最后一次访问时间，等等。把所有东西都表示在文件系统中的想法是来源于 Unix，现代的 Linux 内核则将这个思路进行了扩展。现在，甚至有关计算机上正在运行的进程的信息都可以在文件系统中找到。

这个对物理实体和内部进程的统一和抽象是非常强大的，因为这允许我们用同样的命令对许多完全不同的设备进行同样的操作。甚至可以通过向链接到运行进程的特殊文件写入数据来改变内核的运行方式。

提示
如果您需要识别文件树和物理实体之间的对应关系，请尝试不带参数运行 mount(8)。

1.2.3 文件系统权限

类 Unix 系统的文件系统权限被定义给三类受影响的用户。

- 拥有这个文件的用户（**u**）
- 这个文件所属组的其他用户（**g**）

- 所有其余的用户（**o**），同样称为“世界”和“所有人”

对文件来说，每个对应权限允许下列动作。

- 可读（**r**）权限允许所有者检查文件的内容。
- 可写（**w**）权限允许所有者修改文件内容。
- 可执行（**x**）权限允许所有者把文件当做一个命令运行。

对于目录来说，每个对应权限允许下列动作。

- 可读（**r**）权限允许所有者列出目录内的内容。
- 可写（**w**）权限允许所有者添加或删除目录里面的文件。
- 可执行（**x**）权限允许所有者访问目录里的文件。

在这里，一个目录的可执行权限意味着不仅允许读目录里的文件，还允许显示他们的属性，例如大小和修改时间。ls(1) 用于显示文件和目录的权限信息（更多）。当运行时带有“`-l`”选项，它将按给定顺序显示下列信息。

- 文件类型（第一个字母）
- 文件的访问权限（9 个字符，三个字符组成一组按照用户、组、其他的顺序表示）
- 链接到文件的硬链接数
- 文件所有者的用户名
- 这个文件所属的组名
- 以字符（字节）为单位的文件大小
- 文件的日期和时间（mtime）
- 文件的名字

字符	说明
`-`	普通文件
`d`	目录
`l`	符号链接
`c`	字符设备节点
`b`	块设备节点
`p`	命名管道
`s`	套接字

Table 1.4: “`ls -l`”输出的第一个字符列表

chown(1) 用于 root 账户修改文件的所有者。chgrp(1) 用于文件的所有者或 root 账户修改文件所属的组。chmod(1) 用于文件的所有者或 root 账户修改文件和文件夹的访问权限。操作一个 `foo` 文件的基本语法如下。

```
# chown newowner foo
# chgrp newgroup foo
# chmod  [ugoa][+-=][rwxXst][,...] foo
```

例如，你可以按照下面使一个目录树被用户 `foo` 所有，并共享给组 `bar`。

```
# cd /some/location/
# chown -R foo:bar .
# chmod -R ug+rwX,o=rX .
```

有三个更加特殊的权限位。

- **Set-User-ID(SUID)** 位（**s** 或 **S** 替换用户的 **x**）
- **Set-Group-ID(SGID)** 位（**s** 或 **S** 替换组的 **x**）
- 粘滞位（**t** 或 **T** 替代其他用户的 **x**）

如果“`ls -l`”对这些位的输出是大写字母，则表示这些输出下面的执行位未设置。

给一个可执行文件设置 **Set-User-ID** 位将允许一个用户以他自己的 ID 运行这个可执行文件（例如 **root** 用户）。类似的，给一个可执行文件设置了 **Set-Group-ID** 位将允许一个用户以文件所属组的 ID 运行该文件。（例如 **root** 组）。由于这些设置可能导致安全风险，设置它们为可用的时候需要格外留意。

在一个目录上设置“**Set-Group-ID**”将打开类 BSD 的文件创建计划，所有在目录里面创建的文件将属于目录所属的组。

给一个目录设置“粘滞位”将保护该目录内的文件不被其所有者之外的一个用户删除。为了保护一个在像“`/tmp`”这样所有人可写或同组可写的目录下文件内容的安全，不仅要去除可写权限，还要给其所在目录设置粘滞位。否则，该文件可以被任意对其所在目录有写权限的用户删除并创建一个同名的新文件。

这里有一点有趣的文件权限例子。

```
$ ls -l /etc/passwd /etc/shadow /dev/ppp /usr/sbin/exim4
crw------T 1 root root   108, 0 Oct 16 20:57 /dev/ppp
-rw-r--r-- 1 root root     2761 Aug 30 10:38 /etc/passwd
-rw-r----- 1 root shadow   1695 Aug 30 10:38 /etc/shadow
-rwsr-xr-x 1 root root   973824 Sep 23 20:04 /usr/sbin/exim4
$ ls -ld /tmp /var/tmp /usr/local /var/mail /usr/src
drwxrwxrwt 14 root root  20480 Oct 16 21:25 /tmp
drwxrwsr-x 10 root staff  4096 Sep 29 22:50 /usr/local
drwxr-xr-x 10 root root   4096 Oct 11 00:28 /usr/src
drwxrwsr-x  2 root mail   4096 Oct 15 21:40 /var/mail
drwxrwxrwt  3 root root   4096 Oct 16 21:20 /var/tmp
```

chmod(1) 有另一种数值模式来描述文件权限。这种数字模式使用 3 到 4 位八进制（底为 8）数。

数字	说明
第一个可选数字	**Set-User-ID** (=4), **Set-Group-ID** (=2) 和粘滞位 (=1) 之和
第二个数字	用户的可读 (=4), 可写 (=2) 和可执行 (=1) 权限之和
第三个数字	组权限同上
第四个数字位	其他用户权限同上

Table 1.5: chmod(1) 命令文件权限的数字模式

这听起来很复杂实际上相当简单。如果你把“`ls -l`”命令输出的前几列（2-10），看成以二进制（底为 2）表示文件的权限（“-”看成 0，“rwx”看成 1），你应该可以理解用数字模式值的最后 3 位数字对文件权限的八进制表示。

尝试下列例子

```
$ touch foo bar
$ chmod u=rw,go=r foo
$ chmod 644 bar
$ ls -l foo bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:39 bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:35 foo
```

提示
如果你需要在 shell 脚本中访问“`ls -l`”显示的信息，你需要使用相关命令，如 test(1)，stat(1) 和 readlink(1)。shell 内置命令，如“[”或“`test`”，可能也会用到。

1.2.4 控制新建文件的权限：umask

什么权限将应用到新建文件受 shell 内置命令 umask 的限制。参见 dash(1)，bash(1)，和 bash-builtins(7)。

```
(file permissions) = (requested file permissions) & ~(umask value)
```

umask 值	创建的文件权限	创建的目录权限	用法
0022	-rw-r--r--	-rwxr-xr-x	仅所属用户可写
0002	-rw-rw-r--	-rwxrwxr-x	仅所属组可写

Table 1.6: **umask** 值举例

Debian 默认使用用户私人组（UPG）。每当一个新用户添加到系统的时候都会创建一个 UPG。UPG 的名字和创建它的用户相同，这个用户是这个 UPG 的唯一成员。自从每个用户都有自己的私人组之后，把 umask 设置成 0002 变得更安全了。（在某些 Unix 变体中，把所有普通用户设置到一个叫 **users** 的组是非常常见的做法，在这种情况下，出于安全考虑把 umask 设为 0022 是一个好主意）

提示
通过把“umask 002”写入 ~/.bashrc 文件打开 UPG。

1.2.5 一组用户的权限（组）

警告
在做重启或者类似行为前，确保保存没有保存的修改。

你能够增加一个用户 penguin 到 bird 组，在两个步骤内：

- 使用下面中的一个改变组配置：
  - 执行”sudo usermod -aG bird penguin”。
  - 执行”sudo adduser penguin bird”。（只在典型的 Debian 系统）
  - 执行”sudo vigr” 编辑 /etc/group 和”sudo vigr -s” 编辑 /etc/gshadow 来追加 penguin 到 bird 行。
- 使用下面中的一个来应用配置：
  - 冷重启再登录。（最佳选项）
  - 通过 GUI（图形用户界面）菜单登录和注销（这可能无法在现代桌面环境下工作。）

你能够从 bird 组移除用户 penguin，用两个步骤：

- 使用下面中的一个改变组配置：
  - 执行”sudo usermod -rG bird penguin”。
  - 执行”sudo deluser penguin bird”。（只在典型的 Debian 系统）
  - 执行”sudo vigr” 编辑 /etc/group 和”sudo vigr -s” 编辑 /etc/gshadow 来移除 bird 行里面的 penguin。
- 使用下面中的一个来应用配置：
  - 冷重启再登录。（最佳选项）
  - 执行”kill -TERM -1” 并做一些修复行为，比如”systemctl restart NetworkManager.service”。

– 通过 GUI（图形用户界面）菜单注销，不是 Gnome 桌面的一个选项。

在现代桌面系统，任何热重启尝试是真实的冷重启的脆弱替代。

注意
或者，你可以通过添加一行“`auth optional pam_group.so`”到“`/etc/pam.d/common-auth`”以及配置“`/etc/security/group.conf`”，使得在身份验证过程动态添加用户到组。（参见第 4 章。）

在 Debian 系统中，硬件设备是另一种文件。如果你从一个用户账户访问某些设备出现问题，例如 USB 盘和 CD-ROM，你需要使这个用户成为相关组的成员。

一些著名的由系统提供的组允许其成员不需要 `root` 权限访问某些特定的文件和设备。

组	可访问文件和设备的描述
`dialout`	完全及直接的访问串口端口（“`/dev/ttyS[0-3]`”）
`dip`	有限的访问串口，创建到信任点的拨号 **IP** 连接
`cdrom`	CD-ROM, DVD+/-RW 驱动器
`audio`	音频设备
`video`	视频设备
`scanner`	扫描仪
`adm`	系统监控日志
`staff`	一些用于初级管理工作的目录：“`/usr/local`”，“`/home`”

Table 1.7: 关于文件访问的由系统提供的著名组列表

提示
你需要属于 `dialout` 组才能重配置调制解调器、拨号到任意地方，等等。但如果 `root` 用户在“`/etc/ppp/peers/`”为受信任点创建了预定义配置文件的话，你只需要属于 `dip` 组，就可以创建拨号 **IP** 来连接到那些受信任的点上，需使用的命令行工具包括 pppd(8)、pon(1) 以及 poff(1)。

某些著名的由系统提供的组允许它们的成员不带 `root` 权限运行特定的命令。

组	可访问命令
`sudo`	使用超级用户权限执行任何命令
`lpadmin`	执行命令以从打印机数据库添加、修改、移除打印机

Table 1.8: 著名的由系统提供用于特定命令运行的组列表

由系统提供的用户和组的完整列表，参见由 `base-passwd` 包提供的“`/usr/share/doc/base-passwd/users-and-gro`中，当前版本的“用户和组”。

用户和组系统的管理命令，参见 passwd(5)，group(5)，shadow(5)，newgrp(1)，vipw(8)，vigr(8)，以及 pam_group(8)。

1.2.6 时间戳

GNU/Linux 文件有三种类型的时间戳。

注意
ctime 不是文件创建时间。

类型	含义（历史上 **Unix** 的定义）
mtime	文件修改时间 (`ls -l`)
ctime	文件状态修改时间 (`ls -lc`)
atime	文件最后被访问的时间 (`ls -lu`)

Table 1.9: 时间戳类型列表

注意
atime 在 GNU/Linux 系统上的真实值可能和历史上 Unix 的定义有所不同。

- 覆盖一个文件，将会改变该文件所有的 **mtime**, **ctime**, 和 **atime** 属性。
- 改变文件的所有者或者权限，将改变文件的 **ctime** 和 **atime** 属性。
- 在历史上的 Unix 系统中，读取一个文件将改变文件的 **atime** 属性。
- 读一个文件，将改变文件的 **atime** 属性；在 GNU/Linux 系统上，这仅发生在其文件系统使用“`strictatime`”参数挂载的情况下。
- 如果 GNU/Linux 系统的文件系统使用”`relatime`” 选项挂载，第一次读文件，或者随后读文件，将改变该文件的 **atime** 属性. (从 Linux 2.6.30 开始的默认行为)
- 如果 GNU/Linux 系统的文件系统使用”`noatime`” 挂载，则读一个文件，不会改变这个文件的 **atime** 属性。

注意
为了在正常的使用场景中能够提升文件系统的读取效率，新增了”`noatime`” 和”`relatime`” 这两个加载选项。如使用了”`strictatime`” 选项，即使简单的文件读操作都伴随着更新 **atime** 属性这个耗时的写操作。但是 **atime** 属性除了 mbox(5) 文件以外却很少用到。详情请看 mount(8)。

使用 touch(1) 命令修改已存在文件的时间戳。

对于时间戳，在非英语区域（“`fr_FR.UTF-8`”），`ls` 命令输出本地化字符串。

```
$ LANG=C  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
$ LANG=en_US.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
$ LANG=fr_FR.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 oct. 16 21:35 foo
```

提示
参考第 9.3.4 节自定义“`ls -l`”输出。

1.2.7 链接

有两种方法把一个文件“`foo`”链接到一个不同的文件名“`bar`”。

- Hard link
  - 对现有文件重复名称
  - ”`ln foo bar`”

- 符号链接或 symlink
  - 通过名字指向另一个文件的特殊文件
  - "ln -s foo bar"

请参阅下面的示例，rm 命令结果中链接数的变化和细微的差别。

```
$ umask 002
$ echo "Original Content" > foo
$ ls -li foo
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 foo
$ ln foo bar     # hard link
$ ln -s foo baz  # symlink
$ ls -li foo bar baz
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 foo
$ rm foo
$ echo "New Content" > foo
$ ls -li foo bar baz
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1450183 -rw-rw-r-- 1 penguin penguin 12 Oct 16 21:48 foo
$ cat bar
Original Content
$ cat baz
New Content
```

硬链接可以在同一个文件系统内创建，并共用同一个 inode 号，由 ls(1) 带"-i"选项显示。

符号链接总是名义上具有"rwxrwxrwx"的文件访问权限，如上面例子所示，实际的有效访问权限由它所指向的文件确定。



小心
除非你有非常好的理由，否则不要创建一个复杂的符号链接或硬链接通常是个好主意。符号链接的逻辑组合可能导致文件系统噩梦般的无限循环。

注意
通常使用符号链接比使用硬链接更合适，除非你有一个好理由使用硬链接。

"."目录链接到它所在的目录，因此任何新建目录的链接数从 2 开始。".."目录链接到父目录，因此目录的链接数随着新的子目录的创建而增加。

如果你刚从 Windows 迁移到 Linux，你很快将清楚 Unix 的文件名链接相较于 Windows 最相近的"快捷方式"是多么精心设计的。由于它是在文件系统中实现的，应用无法看到链接文件跟原始文件之间的区别。在硬链接这种情况，这真的是毫无差别。

1.2.8 命名管道（先进先出）

命名管道是一个像管道一样的文件。你把内容放进了文件，它从另一端出来。因此，它被称为 FIFO，即先进先出：你从管道这端先放进去的东西会从另一端先出来。

如果对一个命名管道进行写入操作，写入的过程不会被终止，直到写入的信息从管道中被读取出来。读取过程将会持续到没有信息可以读取为止。管道的大小始终是零，它不存储数据，它只是连接两个过程，像 shell 提供的"|" 语法功能一样。然而，一旦管道有了名称，这两个进程就可以不必在同一个命令行，甚至由同一个用户运行。管道是 Unix 的一个非常有影响力的创新。

尝试下列例子

```
$ cd; mkfifo mypipe
$ echo "hello" >mypipe & # put into background
[1] 8022
$ ls -l mypipe
prw-rw-r-- 1 penguin penguin 0 Oct 16 21:49 mypipe
$ cat mypipe
hello
[1]+  Done                    echo "hello" >mypipe
$ ls mypipe
mypipe
$ rm mypipe
```

1.2.9 套接字

套接字被广泛应用于所有的互联网通信，数据库和操作系统本身。它类似于命名管道（FIFO）并且允许进程之间甚至不同计算机之间进行信息交换。对于套接字，这些进程不需要在同一时间运行，也不需要是同一个父进程的子进程。它是进程间通信（IPC）的一个节点。信息的交换可能会通过网络发生在不同主机之间。最常见的两种是 互联网套接字和 UNIX 域套接字 。

提示
通过"`ss -t -u -a`"（来自 `iproute2` 软件包）命令可以很方便的查看系统已经打开了哪些套接字。

1.2.10 设备文件

设备文件包括系统的物理设备和虚拟设备，如硬盘、显卡、显示屏、键盘。虚拟设备的一个例子是控制台，用"`/dev/console`"来描述。

设备文件有两种类型。

- 字符设备
  - 每次访问一个字符
  - 一个字符等于一个字节
  - 如键盘、串口…
- 块设备
  - 通过更大的单元–块，进行访问
  - 一个块 > 一个字节
  - 如硬盘等…

你可以读写块设备文件，尽管该文件可能包含二进制数据，读取后显示出无法理解的乱码。向文件写入数据，有时可以帮助定位硬件连接故障。比如，你可以将文本文件导入打印机设备"`/dev/lp0`"，或者将调制解调命令发送到合适的串口"`/dev/ttyS0`"。但是，除非这些操作都小心完成，否则可能会导致一场大灾难。所以要特别小心。

注意
常规访问打印机，使用 lp(1)。

设备的节点数可以通过执行 ls(1) 得到，如下所示。

```
$ ls -l /dev/sda /dev/sr0 /dev/ttyS0 /dev/zero
brw-rw---T  1 root disk     8,  0 Oct 16 20:57 /dev/sda
brw-rw---T+ 1 root cdrom   11,  0 Oct 16 21:53 /dev/sr0
crw-rw---T  1 root dialout  4, 64 Oct 16 20:57 /dev/ttyS0
crw-rw-rw-  1 root root     1,  5 Oct 16 20:57 /dev/zero
```

- "/dev/sda" 的主设备号是 8，次设备号是 0。它可以被 disk 群组的用户读写。
- "/dev/sr0" 的主设备号是 11，次设备号是 0。它可以被 cdrom 群组的用户读写。
- "/dev/ttyS0" 的主设备号是 4，次设备号是 64。它可以被 dailout 群组的用户读写。
- "/dev/zero" 的主设备号是 1，次设备号是 5。它可以被任意用户读写。

在现代 Linux 系统中，处在"/dev" 之下的文件系统会自动被 udev() 机制填充。

1.2.11 特殊设备文件

还有一些特殊的设备文件。

设备文件	操作	响应描述
/dev/null	读取	返回"文件结尾字符（EOF）"
/dev/null	写入	无返回（一个无底的数据转存深渊）
/dev/zero	读取	返回"\0 空字符"（与 ASCII 中的数字 0 不同）
/dev/random	读取	从真随机数产生器返回一个随机字符，供应真熵（缓慢）
/dev/urandom	读取	从能够安全加密的伪随机数产生器返回一个随机字符
/dev/full	写入	返回磁盘已满（ENOSPC）错误

Table 1.10: 特殊设备文件列表

这些特别设备文件经常和 shell 数据重定向联合使用（参考第 1.5.8 节）。

1.2.12 procfs 和 sysfs

procfs 和 sysfs 两个伪文件系统，分别加载于"/proc" 和"/sys" 之上，将内核中的数据结构暴露给用户空间。或者说，这些条目是虚拟的，他们打开了深入了解操作系统运行的方便之门。

目录"/proc" 为每个正在运行的进程提供了一个子目录，目录的名字就是进程标识符（PID）。需要读取进程信息的系统工具，如 ps()，可以从这个目录结构获得信息。

"/proc/sys" 之下的目录，包含了可以更改某些内核运行参数的接口。（你也可以使用专门的 sysctl() 命令修改，或者使用其预加载/配置文件"/etc/sysctl.d/*.conf"。）

当人们看到这个特别大的文件"/proc/kcore" 时，常常会惊慌失措。这个文件于你的的电脑内存大小相差不多。它被用来调试内核。它是一个虚拟文件，指向系统内存，所以不必担心它的大小。

"/sys" 以下的目录包含了内核输出的数据结构，它们的属性，以及它们之间的链接。它同时也包含了改变某些内核运行时参数的接口。

参考"proc.txt(.gz)"，"sysfs.txt(.gz)"，以及其他相关的 Linux 内核文档("/usr/share/doc/linux-doc-*/Doc
这些文件由 linux-doc-* 软件包提供。

1.2.13 tmpfs

tmpfs 是一个临时文件系统，它的文件都保存在虚拟内存中。必要时，位于内存页缓存的 tmpfs 数据可能被交换到硬盘中的交换分区。

系统启动早期阶段，"/run" 目录挂载为 tmpfs。这样即使"/" 挂载为只读，它也是可以被写入的。它为过渡态文件提供了新的存储空间，同时也替代了 Filesystem Hierarchy Standar2.3 版中说明的目录位置：

- "/var/run" → "/run"
- "/var/lock" → "/run/lock"
- "/dev/shm" → "/run/shm"

参考"tmpfs.txt(.gz)", 文件位于 Linux 内核文档("/usr/share/doc/linux-doc-*/Documentation/filesystems 目录之下，由软件包 linux-doc-* 提供。

1.3 Midnight Commander (MC)

Midnight Commander (MC) 是一个 Linux 终端或其它终端环境下的 GNU 版" 瑞士军刀" 。它为新手们提供了一个菜单式样的终端使用体验，这比标准的 Unix 命令更易于学习。

你可能需要按照下面的命令来安装标题为" mc " 的 Midnight Commander 软件包。

```
$ sudo apt-get install mc
```

使用 mc(1) 命令那个来浏览 Debian 系统。这是最好的学习方式。请使用光标键和回车键来翻看一些感兴趣的内容。

- "/etc" 及其子目录
- " /var/log " 及其子目录
- " /usr/share/doc " 及其子目录
- " /usr/sbin " 和" /usr/bin "

1.3.1 自定义 MC

为了在退出 MC 的时候更改目录并 cd 到其它目录，我建议修改"~/.bashrc" 包含一个由 mc 包提供的脚本。

```
. /usr/lib/mc/mc.sh
```

查看 mc(1) (在"-P" 选项里) 的原因。(如果你不能理解我这里说所讲的，你可以稍后回头再看)

1.3.2 启动 MC

MC 可以这样启动起来。

```
$ mc
```

MC 通过菜单覆盖了所有的文件操作，因此而让用户更省心省力。只需要按 F1 就可以跳转到帮助界面。你只需要按光标键和功能键就可以使用 MC。

> 注意
> 某些终端比如 gnome-terminal(1)，功能键的按键触发消息可能会被终端程序截取。在 gnome-terminal 里，可以通过" 首选项" →"通用"-> " 快捷键" 菜单设置来禁用这些特征。

如果你遇到字符编码问题，显示出来都是乱码，通过添加"-a" 到 MC 命令行或许有助于避免此类问题。

如果这样不能解决 MC 中的显示问题，可以参考第 8.3.1 节.

1.3.3 MC 文件管理

默认的两个目录面板里包含了文件列表。另一个有用的模式是设置右边窗口为” 信息” 来读取文件访问权限信息。接下来是一些必要的快捷键。守护进程 gpm(8) 运行的时候，你也可以在字符命令行里用鼠标来操作。(在 MC 里进行复制和粘贴操作的时候一定要按住 shift 键。)

快捷键	键绑定功能
F1	帮助菜单
F3	内部文件查看器
F4	内部编辑器
F9	激活下拉菜单
F10	退出 Midnight Commander
Tab	在两个窗口间移动
Insert 或 Ctrl-T	用于多文件操作的标记文件，如副本
Del	删除文件 (注意---设置 MC 为安全删除模式)
光标键	自我解释

Table 1.11: MC 快捷键绑定

1.3.4 MC 命令行技巧

- cd 命令在选中的屏幕中改变目录。
- Ctrl-Enter or Alt-Enter 拷贝文件名到命令行。使用 cp(1) 和 mv(1) 两个命令来进行处理。
- Alt-Tab 显示文件名自动补全提示。
- 通过添加 MC 命令参数可以指定开始目录；例如，”mc /etc /root”。
- Esc + n-key → Fn (即 Esc + 1 → F1 等；Esc + 0 → F10)
- 先按 Esc 键和同时按 Alt 是一样；例如, 输入 Esc + c 和同时 Alt-C 是一样的。Esc 被称为 meta 键，有时候也称之为”M-”。

1.3.5 MC 内部编辑器

这个内置编辑器有一个有意思的粘贴方案。按 F3 开始选择起始点，再按 F3 选择终点并高亮选择区。此刻你可以移动你的光标，使用 F6 将选区移动到当前光标下，F5 则将选区复制到当前光标下。F2 保存文件。F10 退出。多数光标键以直观的方式工作。

这个编辑器可以直接以下面的命令打开一个文件。

```
$ mc -e filename_to_edit
```

```
$ mcedit filename_to_edit
```

这不是一个多窗口编辑器，但是能通过复用终端来达到同样的效果。在两个窗口间复制，需要用到 Alt-F*n* 来切换虚拟终端并使用”File→Insert file” 或者”File→Copy to file” 来移动文本。

内部编辑器可以被外部编辑器替代。

同样，许多程序使用环境变量 $EDITOR 或 $VISUAL 来决定编辑器的使用。如果你准备使用 vim(1) 或者 nano(1) 来开始，你或许需要将下面的代码加入”~/.bashrc” 来对 mcedit 进行设置。

```
export EDITOR=mcedit
export VISUAL=mcedit
```

如果可能的话我推荐用”vim”。

如果你使用 vim(1) 并不顺手，你可以在大部分系统中继续使用 mcedit(1) 来进行工作。

1.3.6 MC 内部查看器

MC 是一个非常智能的查看器。这是一个在文档中搜索文本的好工具。我经常使用它在/usr/share/doc 目录中查找文件。这是浏览大量 Linux 信息的最快方式。这个查看器可以通过下列命令中的任何一个来直接启动。

```
$ mc -v path/to/filename_to_view
```

```
$ mcview path/to/filename_to_view
```

1.3.7 自动启动 MC

在文件中输入回车，用适当的程序来处理文件的内容 (查看第 9.4.11 节)。这是 MC 一个非常方便的用法。

文件类型	对回车键的响应
可执行文件	执行命令
帮助文档	管道内容查看器软件
html 文件	管道内容网页浏览器
"*.tar.gz" 和"*.deb" 文件	浏览其内容就像查看子目录一样

Table 1.12: MC 中对回车键的响应

为让这些查看器和虚拟文件特征生效，可查看的文件不能够被设置为可执行。使用 chmod(1) 或通过 MC 文件菜单改变他们的状态。

1.3.8 MC 中的虚拟文件系统

MC 能够跨因特网访问文件。在菜单按 F9, "Enter" 和"h" 来激活 Shell 文件系统。按"sh://[user@]machine[:options]/[的形式输入 URL, 就会看起来像本地使用 ssh 一样来检索远程目录。

1.4 类 Unix 工作环境基础

虽然 MC 差不多可以让你做任何事情，但学会从 shell 提示下使用命令行工具也是非常重要的，可以让你变得熟悉类 Unix 工作环境。

1.4.1 登录 shell

因登录 shell 可以被一些系统初始化程序使用，请谨慎的把登录 shell 保持为 bash(1) ，并避免把它转换为 chsh(1)。

如果你想使用不同的 shell 交互提示符，从 GUI（图形用户界面）的终端模拟器来设置；或者从 ~/.bashrc 启动，比如说，在里面放置"exec /usr/bin/zsh -i -l" 或"exec /usr/bin/fish -i -l"。

提示
虽然类 POSIX 共享基本语法，但他们在 shell 变量和全局扩展等基本事情上，行为可以不同。细节请查阅他们的文档。

在本教程中，交互式的 shell 总是指 bash.

软件包	流行度	大小	**POSIX shell**	说明
bash	V:900, I:1000	7309	是	Bash: GNU Bourne Again SHell (事实上的标准)
bash-completion	V:41, I:964	1952	N/A	bash shell 编程补全
dash	V:933, I:999	211	是	Debian Almquist Shell, 擅长 shell 脚本
zsh	V:38, I:67	2571	是	Z shell: 有许多增强的标准 shell
tcsh	V:4, I:14	1366	No	TENEX C Shell: 一个 Berkeley csh 的增强版本
mksh	V:5.0, I:7.3	7713	是	Korn shell 的一个版本
csh	V:1.0, I:5.0	348	No	OpenBSD C Shell, Berkeley csh 的一个版本
sash	V:0.3, I:4.6	1249	是	有内置命令的 Stand-alone shell (并不意味着标准的"/usr/bin/sh")
ksh	V:0.2, I:7.5	65	是	Korn shell 的真正的 AT&T 版本
rc	V:0.08, I:0.69	182	No	AT&T Plan 9 rc shell 的一个实现
posh	V:0.01, I:0.21	195	是	Policy-compliant Ordinary SHell 策略兼容的普通 shell(pdksh 派生)

Table 1.13: shell 程序列表

1.4.2 定制 bash

你可以通过"~/.bashrc"来定制 bash(1) 的行为。

尝试下列例子。

```
# enable bash-completion
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# CD upon exiting MC
. /usr/lib/mc/mc.sh

# set CDPATH to a good one
CDPATH=.:/usr/share/doc:~:~/Desktop:~
export CDPATH

PATH="${PATH+$PATH:}/usr/sbin:/sbin"
# set PATH so it includes user's private bin if it exists
if [ -d ~/bin ] ; then
  PATH="~/bin${PATH+:$PATH}"
fi
export PATH

EDITOR=vim
export EDITOR
```

提示
你可以在第 9 章中的第 9.3.6 节找到更多关于 bash 的定制技巧。

提示
`bash-completion` 软件包能够让 `bash` 进行命令补全.

1.4.3 特殊按键

在类 Unix 环境，有一些具有特殊含义的按键。请注意，普通的 Linux 字符控制台，只有左手边的 `Ctrl` 和 `Alt` 键可以正常工作。其中有几个值得记住的按键。

快捷键	描述
`Ctrl-U`	删除光标前到行首的字符
`Ctrl-H`	删除光标前的一个字符
`Ctrl-D`	终止输入（如果你在使用 shell，则退出 shell）
`Ctrl-C`	终止一个正在运行的程序
`Ctrl-Z`	通过将程序移动到后台来暂停程序
`Ctrl-S`	停止屏幕输出
`Ctrl-Q`	激活屏幕输出
`Ctrl-Alt-Del`	重启/关闭系统，参见 inittab(5)
左 `Alt` 键（可选择同时按下 `Windows-key`）	Emacs 和相似 UI 的元键（meta-key）
`Up-arrow`	开始在 `bash` 中查看命令历史
`Ctrl-R`	开始在 `bash` 的增量命令历史中搜索
`Tab`	在 `bash` 命令行中补全文件名
`Ctrl-V Tab`	在 `bash` 命令行中输入 `Tab` 而不是进行补全

Table 1.14: bash 的按键绑定列表

提示
`Ctrl-S` 的终端功能可能被 stty(1) 禁用。

1.4.4 鼠标操作

Debian 系统针对文本的鼠标操作混合 2 种风格，外加一些新的方法：

- 传统的 Unix 鼠标操作方式：
  - 使用 3 个按钮（单击）
  - 使用主要键
  - 由 X 应用，如 `xterm`，以及文本应用在控制台中使用
- 现代 GUI（图形用户界面）鼠标操作方式：
  - 使用 2 个按钮（拖动 + 单击）
  - 使用主要键和剪贴板
  - 用于现代的 GUI（图形用户界面）应用，比如 `gnome-terminal`

这里，主要键的选择会高亮文本范围。在终端程序内，使用 `Shift-Ctrl-C` 来代替，这样可以避免终止一个运行的程序。

在现代滚轮鼠标上的中央滚轮，被认为是中间键，并可以被当做中间键使用。在 2 键鼠标系统的情况下，同时按左键和右键就相当于按中间键。

为了在 Linux 字符控制台中使用鼠标，您需要让 gpm(8) 作为后台守护进程（daemon）运行。

操作	响应
左击并拖动鼠标	主要键的选择作为选择范围
单击左键	主要键的位置作为选择范围的开头
单击右键（传统方式）	主要键的位置作为选择范围的结尾
单击右键（现代方式）	依赖上下文的菜单（剪切、拷贝、粘贴）
点击中键或者 Shift-Ins	在光标处插入主要键的选择
Ctrl-X	剪切主要键的选择到剪贴板
Ctrl-C (在终端是 Shift-Ctrl-C)	拷贝主要键的选择到剪贴板
Ctrl-V	粘贴剪切板的内容到光标处

Table 1.15: Debian 上的鼠标操作和相关按键操作列表

1.4.5 分页程序

less(1) 命令是一个增强版的分页程序（文件内容查看器）。它按照指定的命令参数或标准输出来读取文件。在用 less 命令查看的时候如果需要帮助可以按“h”。它的功能比 more(1) 命令更丰富，通过在脚本的开头执行”eval $(lesspipe)”或”eval $(lessfile)”它的功能还能变得更加强大。详细请参考”/usr/share/doc/less/LESSOPEN”。”-R” 选项可以实现原始的字符输出还可以启用 ANSI 颜色转义序列。详细请参考 less(1)。

> 提示
> 在 less 命令中，输入”h” 来查看帮助屏幕，输入”/” 或”?” 来搜索字符串，输入”-i” 来改变大小写敏感性。

1.4.6 文本编辑器

在使用类 Unix 系统过程中，各种类似于 Vim 或 Emacs 的工具，你应该精通其中的一个。

我认为习惯于使用 Vim 命令是一个明智的选择，因为 Linux/Unix 系统里一般都附带了 Vi 编辑器。(实际上最初的 vi 以及后来的 nvi 这类工具程序很常见。因为在 Vim 里提供了 F1 帮助键，在同类工具中它的功能更强大，所以我选择 Vim 而不是其它新出的一些工具。)

假设你不是用 Emacs 就是用 XEmacs 作为你的编辑器，其实还有更好的选择，尤其是在编程的时候。Emacs 还有很多其他的特点，包括新手导读，目录编辑器，邮件客户端等等。当编写脚本或程序的时候，它能自动识别当前工作模式所对应的格式，让使用更加便利。一些人甚至坚持认为 Linux 系统里最需要配备的就是 Emacs。花十分钟来学习 Emacs 可以为后面的工作剩下更多时间。在此强烈推荐学习使用 Emacs 时候直接使用 GNU Emacs 参考手册。

在实践应用中所有这些程序都会有一个教程，输入”vim” 和 F1 键就可以启动 Vim。建议你最好阅读一下前面的 35 行。移动光标到”|tutor|” 并按 Ctrl-] 就可以看到在线培训教程。

> 注意
> 好的编辑器，像 Vim 和 Emacs，可以处理 UTF-8 及其它不常用编码格式的文本。有个建议就是在 GUI（图形用户界面）环境下使用 UTF-8 编码，并安装要求的程序和字体。编辑器里可以选择独立于 GUI（图形用户界面）环境的编码格式。关于多字节文本可以查阅参考文档。

1.4.7 设置默认文本编辑器

Debian 有许多不同的编辑器。我们建议安装上面提到的 vim 软件包。

Debian 通过命令“/usr/bin/editor”提供了对系统默认编辑器的统一访问，因此其它程序（例如 reportbug(1)）可以调用它。你可以通过下列命令改变它。

```
$ sudo update-alternatives --config editor
```

对于新手，我建议使用“/usr/bin/vim.basic”代替“/usr/bin/vim.tiny”，因为它支持格式高亮。

提示
许多程序使用环境变量“$EDITOR”或“$VISUAL”来决定使用那个编辑器（参见第 1.3.5 节和第 9.4.11 节）。出于 Debian 系统的一致性考虑，它们被设置到“/usr/bin/editor”。（在历史上，“$EDITOR”是“ed”，“$VISUAL”是“vi”。）

1.4.8 使用 vim

最近的 vim(1) 用完全的”nocompatible” 选项启动自己，进入到普通模式。[1]

模式	按键	操作
NORMAL	:help|only	显示帮助文件
NORMAL	:e filename.ext	打开新的缓冲区来编辑 filename.ext
NORMAL	:w	把目前的缓冲区改写到原始文件
NORMAL	:w filename.ext	写入当前缓冲区到 filename.ext
NORMAL	:q	退出 vim
NORMAL	:q!	强制退出 vim
NORMAL	:only	关闭所有其它分割打开的窗口
NORMAL	:set nocompatible?	检查 vim 是否在完全的 nocompatible 模式
NORMAL	:set nocompatible	设置 vim 到完全的 nocompatible 模式
NORMAL	i	进入插入模式
NORMAL	R	进入替代模式
NORMAL	v	进入可视模式
NORMAL	V	进入可视行模式
NORMAL	Ctrl-V	进入可视块模式
除了 TERMINAL-JOB 外	ESC-键	进入普通模式
NORMAL	:term	进入 TERMINAL-JOB 模式
TERMINAL-NORMAL	i	进入 TERMINAL-JOB 模式
TERMINAL-JOB	Ctrl-W N (或者 Ctrl-\ Ctrl-N)	进入 TERMINAL-NORMAL 模式
TERMINAL-JOB	Ctrl-W :	在 TERMINAL-NORMAL 模式里进入 Ex-模式

Table 1.16: 基本的 Vim 按键列表

请使用”vimtutor” 程序来学习 vim，通过一个交互式的指导课程。

vim 程序基于模式输入的按键来改变它的行为。在插入-模式和替代-模式下，输入的按键大部分进入了缓冲区。移动光标大部分在普通-模式下完成。交互选择在可视-模式下完成。在普通-模式下输入”:” ，改变它的模式进入到 Ex-模式。Ex-接受命令。

提示
Vim 和 **Netrw** 软件包可以一起使用。Netrw 同时支持在本地和网络读写文件，浏览目录。用”vim .” (一个点作为参数) 来尝试 Netrw，在”:help netrw” 读取它的文档。

vim 的高级配置，参见第 9.2 节。

1即使旧的 vim 也能够启动完全的”nocompatible” 模式，通过使用”-N” 选项启动。

1.4.9 记录 shell 活动

shell 命令的输出有可能滚动出了屏幕，并可能导致你无法再查看到它。将 shell 活动记录到文件中再来回顾它是个不错的主意。当你执行任何系统管理任务时，这种记录是必不可少的。

提示
新版本的 Vim (version>=8.2) 能够被用来清晰的记录 shell 活动，使用 `TERMINAL-JOB`-模式。参见第 1.4.8 节。

记录 shell 活动的基本方法是在 script(1) 下运行 shell。

尝试下列例子

```
$ script
Script started, file is typescript
```

在 `script` 下使用任何 shell 命令。

按 `Ctrl-D` 来退出 `script`。

```
$ vim typescript
```

参见第 9.1.1 节。

1.4.10 基本的 Unix 命令

让我们来学习基本的 Unix 命令。在这里，我指的是一般意义上的“UNIX”。任何 UNIX 克隆系统通常都会提供等价的命令。Debian 系统也不例外。如果有一些命令不像你想的那样起作用，请不要担心。如果 shell 中使用了别名，其对应的命令输出会不同。这些例子并不意味着要以这个顺序来执行。

尝试使用非特权用户账号来使用下列的命令。

注意
Unix 有一个惯例，以“`.`”开头的文件将被隐藏。它们一般为包含了配置信息和用户首选项的文件。
对于 `cd` 命令，参见 bash-builtins(7)。
基本的 Debian 系统的默认分页程序是 more(1)，它无法往回滚动。通过命令“`apt-get install less`” 安装 `less` 软件包后，less(1) 会成为默认的分页程序，它可以通过方向键往回滚动。
上面命令”ps aux | grep -e ”[e]xim4*”” 中正则表达式里的”[” 和”]” 使得 grep 可以避免匹配到自身。正则表达式中的”4*” 表示字符”4” 出现 0 次或多次，因此 grep 可以匹配”exim” 和”exim4”。虽然”*” 同时出现在 shell 文件名通配符和正则表达式中，但它们的含义不同。可以从 grep(1) 学习正则表达式。

作为训练，请使用上述的命令来遍历目录并探究系统。如果你有任何有关控制台命令的问题，请务必阅读手册。

尝试下列例子

```
$ man man
$ man bash
$ man builtins
$ man grep
$ man ls
```

手册的风格可能让人有点难以习惯，因为它们都相当简洁，尤其是比较老旧、非常传统的那些手册。但是，一旦你习惯了它，你来欣赏它们的简洁。

请注意，许多类 Unix 命令（包含来自 GNU 和 BSD 的）都可以显示简短的帮助信息，你可以使用下列的其中一种方式来查看它（有时不带任何参数也可以）。

```
$ commandname --help
$ commandname -h
```

命令	说明
pwd	显示当前/工作目录的名称
whoami	显示当前的用户名
id	显示当前用户的身份（名称、uid、gid 和相关组）
file *foo*	显示“*foo*”文件的文件类型
type -p *commandname*	显示命令的文件所处位置“*commandname*”
which *commandname*	,,
type *commandname*	显示“*commandname*”命令的相关信息
apropos *key-word*	查找与“*key-word*”有关的命令
man -k *key-word*	,,
whatis *commandname*	用一行解释“*commandname*”命令
man -a *commandname*	显示“*commandname*”命令的解释（Unix 风格）
info *commandname*	显示“*commandname*”命令相当长的解释（GNU 风格）
ls	显示目录内容（不包含以. 点号开头的文件和目录）
ls -a	显示目录内容（包含所有文件和目录）
ls -A	显示目录内容（包含几乎所有文件和目录，除了“..”和“.”）
ls -la	显示所有的目录内容，并包含详细的信息
ls -lai	显示所有的目录内容，并包含 inode 和详细的信息
ls -d	显示当前目录下的所有目录
tree	使用树状图显示目录内容
lsof *foo*	列出处于打开状态的文件”*foo*”
lsof -p *pid*	列出被某进程打开的文件: ”*pid*”
mkdir *foo*	在当前目录中建立新目录“*foo*”
rmdir *foo*	删除当前目录中的“*foo*”目录
cd *foo*	切换到当前目录下或变量“$CDPATH”中的“*foo*”目录
cd /	切换到根目录
cd	切换到当前用户的家目录
cd */foo*	切换到绝对路径为“*/foo*”的目录
cd ..	切换到上一级目录
cd ~*foo*	切换到用户“*foo*”的家目录
cd -	切换到之前的目录
</etc/motd pager	使用默认的分页程序来显示“/etc/motd”的内容
touch *junkfile*	建立一个空文件“*junkfile*”
cp *foo bar*	将一个现有文件“*foo*”复制到一个新文件“*bar*”
rm *junkfile*	删除文件“*junkfile*”
mv *foo bar*	将一个现有文件“*foo*”重命名成“*bar*”（“*bar*”必须不存在）
mv *foo bar*	将一个现有文件“*foo*”移动到新的位置“*bar/foo*”（必须存在“*bar*”目录）
mv *foo bar/baz*	移动一个现有文件“*foo*”到新位置并重命名为“*bar/baz*”（必须存在“*bar*”目录，且不存在“*bar/baz* 文件”）
chmod 600 *foo*	使其他人无法读写现有文件“*foo*”（并且所有人都无法执行该文件）
chmod 644 *foo*	使其他人对现有文件“*foo*”可读但不可写（并且所有人都无法执行该文件）
chmod 755 *foo*	使其他人对“*foo*”可读而不可写（并且所有人都能执行该文件）
find . -name *pattern*	使用 shell “*pattern*”查找匹配的文件名（速度较慢）
locate -d . *pattern*	使用 shell “*pattern*”查找匹配的文件名（速度较快，使用定期生成的数据库）
grep -e "*pattern*" *.html	在当前目录下以“.html”结尾的所有文件中，查找匹配“*pattern*”的文件并显示
top	全屏显示进程信息，输入“q”退出
ps aux | pager	显示所有正在运行的进程的信息（BSD 风格）
ps -ef | pager	显示所有正在运行的进程的信息（Unix system-V 风格）
ps aux | grep -e "[e]xim4*"	显示所有正在运行“exim”和“exim4”的进程
ps axf | pager	显示所有正在运行的进程的信息（ASCII 风格）
kill *1234*	杀死 ID 为“*1234*”的进程
gzip *foo*	使用 Lempel-Ziv 编码（LZ77）将“*foo*”压缩为“*foo*.gz”
gunzip *foo*.gz	将“*foo*.gz”解压为“*foo*”
bzip2 *foo*	使用 Burrows-Wheeter 块排序压缩算法和 Huffman 编码将“*foo*”压缩为“*foo*.bz2”（压缩效果比 gzip 更好）
bunzip2 *foo*.bz2	将“*foo*.bz2”解压为“*foo*”
xz *foo*	使用 Lempel-Ziv-Markov 链算法将“*foo*”压缩为“*foo*.xz”（压缩效果比 bzip2 更好）
unxz *foo*.xz	将“*foo*.xz”解压为“*foo*”
tar -xvf *foo*.tar	从“*foo*.tar”档案中提取文件

1.5 简单 shell 命令

现在你已经对如何使用 Debian 系统有了一些感觉。让我们深入了解 Debian 系统中命令执行的机制。在这里，我已经为新手简化了现实。有关确切的解释，请参见 bash(1)。

一般的命令由有序的组件构成。

1. 设置变量值（可选）
2. 命令名
3. 参数（可选）
4. 重定向（可选：> , >> , < , << 等等）
5. 控制操作（可选：&& ，|| ，换行符, ; , & , (,)）

1.5.1 命令执行和环境变量

一些环境变量的值会改变部分 Unix 命令的行为。

环境变量的默认值由 PAM 系统初始化，其中一些会被某些应用程序重新设定。

- PAM(可插拔身份验证模块)系统的模块，比如 pam_env 模块，可以通过 /etc/pam.conf”、”/etc/environment” 和”/etc/default/locale” 设置环境变量。
- 从 systemd 用户模式单元(参见 systemd.unit(5))运行的现代 GUI(图形用户界面)环境，通过在”~/.config/environmen 目录下的文件来设置它的环境变量。
- 用户特有的程序初始化时，可以重新设置在”~/.profile”、”~/.bash_profile” 和”~/.bashrc” 中设置的环境变量。

1.5.2 “$LANG”变量

默认的语言环境是在”$LANG” 环境变量中定义，它在安装的时候配置为”LANG=xx_YY.UTF-8”，或者在接下来的 GUI（图形用户界面）中配置，例如在 GNOME 中是，” 设置” → ” 区域 & 语言” → ” 语言” / ” 格式”。

> 注意
> 目前建议最好用变量”$LANG” 来配置系统环境变量，只有在逼不得已的情况下才用 $LC_*” 开头的变量。

“$LANG”变量的完整的语言环境值由 3 部分组成：“xx_YY.ZZZZ”。

语言环境值	说明
xx	ISO 639 语言代码（小写）例如“en”
YY	ISO 3166 国家代码（大写）例如“US”
ZZZZ	编码，总是设置为“UTF-8”

Table 1.18: 语言环境值的 3 个部分

使用 shell 命令行按顺序执行下列典型的命令。

语言环境推荐	语言（地区）
en_US.UTF-8	英语（美国）
en_GB.UTF-8	英语（大不列颠）
fr_FR.UTF-8	法语（法国）
de_DE.UTF-8	德语（德国）
it_IT.UTF-8	意大利语（意大利）
es_ES.UTF-8	西班牙语（西班牙）
ca_ES.UTF-8	加泰隆语（西班牙）
sv_SE.UTF-8	瑞典语（瑞典）
pt_BR.UTF-8	葡萄牙语（巴西）
ru_RU.UTF-8	俄语（俄国）
zh_CN.UTF-8	汉语（中华人民共和国）
zh_TW.UTF-8	汉语（中国台湾）
ja_JP.UTF-8	日语（日本）
ko_KR.UTF-8	韩语（韩国）
vi_VN.UTF-8	越南语（越南）

Table 1.19: 语言环境推荐列表

```
$ echo $LANG
en_US.UTF-8
$ date -u
Wed 19 May 2021 03:18:43 PM UTC
$ LANG=fr_FR.UTF-8 date -u
mer. 19 mai 2021 15:19:02 UTC
```

这里，date(1) 程序执行时使用了不同的语言环境值。

- 第一个命令，“$LANG”设置为系统的默认语言环境值“en_US.UTF-8”。
- 第二个命令，“$LANG”设置为法语的 UTF-8 语言环境值“fr_FR.UTF-8”。

大多数的命令在执行时并没有预先定义环境变量。对于上面的例子，你也可以选择如下的方式。

```
$ LANG=fr_FR.UTF-8
$ date -u
mer. 19 mai 2021 15:19:24 UTC
```

> 提示
> 提交一个 BUG 报告的时候，如果使用的是非英语的环境，在”LANG=en_US.UTF-8” 语言环境环境下对命令进行运行和检查会更好一些。

对于语言环境配置的细节，参见第 8.1 节。

1.5.3 ”$PATH” 变量

当你在 Shell 里输入命令的时候，Shell 会在”$PATH” 变量所包含的目录列表里进行搜索，”$PATH” 变量的值也叫作 Shell 的搜索路径。

在默认的 Debian 安装过程中，所使用的用户账号的”$PATH” 环境变量可能不包括”/usr/sbin” 和”/usr/sbin” 目录。例如，ifconfig 命令就需要指定完整的路径”/usr/sbin/ifconfig”。(类似地，ip 命令是在”/usr/bin” 目录下)

可以在 Bash 脚本文件”~/.bash_profile” 或”~/.bashrc” 中改变”$PATH” 环境变量的值。

1.5.4 ”$HOME” 变量

很多命令在用户目录中都存放了用户指定的配置，然后通过配置的内容来改变它的执行方式，用户目录通常用”$HOME” 变量来指定。

”$HOME” 变量的值	程序运行环境
/	初始进程执行的程序（守护进程）
/root	root 用户权限 Shell 执行的程序
/home/*normal_user*	普通用户权限 Shell 执行的程序
/home/*normal_user*	普通用户 GUI 桌面菜单执行的程序
/home/*normal_user*	用 root 用户权限来执行程序”sudo program”
/root	用 root 用户权限执行程序”sudo -H program”

Table 1.20: ”$HOME” 变量值列表

> 提示
> Shell 扩展”~/” 为转入当前用户的主目录，也就是”$HOME/”。Shell 扩展”~foo/” 为 foo 的目录，也就是”/home/foo/”。

如果 $HOME 对你的程序不可用，参见第 12.1.5 节。

1.5.5 命令行选项

一些命令附带参数。这些参数以”-” 或”--” 开头，通常称之为选项，用来控制命令的执行方式。

```
$ date
Thu 20 May 2021 01:08:08 AM JST
$ date -R
Thu, 20 May 2021 01:08:12 +0900
```

这里的命令参数”-R” 改变 date(1) 命令输出为 RFC2822 标准的日期字符格式。

1.5.6 Shell 通配符

经常有这种情况你期望命令成串自动执行而不需要挨个输入，将文件名扩展为 **glob**，(有时候被称为通配符)，以此来满足这方面的需求。

shell glob 模式	匹配规则描述
*	不以”.” 开头的文件名
.*	以”.” 开头的文件名（在 Bash 5.2 及以后，”.” 和”..” 默认被排除）
?	精确字符
[…]	包含在括号中的任意字符都可以作为精确字符
[a-z]	”a” 到”z” 之间的任意一个字符都可以作为精确字符
[!…]	除了包含在括号中的任意字符 (”!” 除外) 的其它字符，都可以作为一个字符

Table 1.21: Shell glob 模式

注意
在路径名中的"/" 不能够被上面表格中的 shell 全局模式匹配。

尝试下列例子

```
$ mkdir foo; cd foo; touch b1.txt b2.txt b3.c b4.h .b5.txt ..b6.txt
$ mkdir bar; cd bar; touch b1.txt b2.txt b3.c b4.h .b5.txt ..b6.txt ; cd -
$ echo b*.txt
b1.txt b2.txt
$ echo b*
b1.txt b2.txt b3.c b4.h bar
$ echo */*.txt
bar/b1.txt bar/b2.txt
$ echo *.[hc]
b3.c b4.h
$ echo .*
.b5.txt ..b6.txt
$ echo b[!1-3]*
b4.h bar
$ echo b*/b[!1-3]*
bar/b4.h
$ cd ..; rm -rf junk
```

参见 glob(7)。

注意
与 shell 通用的文件名匹配方式不同，使用" `-name` " 选项的 `find` (1)，其 shell 模式" `*` "，匹配以" `.` " 开始的文件名.（新 POSIX 的特性)

注意
BASH 可以使用内置的 shopt 选项如" `dotglob` "，" `noglob` "," `nocaseglob` "," `nullglob` "," `extglob` " 定制全局行为, 使用 `bash (1)` 查看详细说明。

1.5.7 命令的返回值

每个命令都会返回它的退出状态（变量："$?"）作为返回值。

命令的退出状态	数字返回值	逻辑返回值
success	zero, 0	**TRUE**
error	non-zero, -1	**FALSE**

Table 1.22: 命令的退出代码

尝试下列例子。

```
$ [ 1 = 1 ] ; echo $?
0
$ [ 1 = 2 ] ; echo $?
1
```

注意
请注意，**success** 是逻辑 **TRUE** ，0（zero）则是它的值。这有些不直观，需要在这里提一下。

1.5.8 典型的顺序命令和 shell 重定向

让我们试着记住下面 Shell 命令里部分命令行所使用的命令习语。

命令常见用法	说明
`command &`	在子 shell 的后台中执行 `command`
`command1 | command2`	通过管道将 `command1` 的标准输出作为 `command2` 的标准输入（并行执行）
`command1 2>&1 | command2`	通过管道将 `command1` 的标准输出和标准错误作为 `command2` 的标准输入（并行执行）
`command1 ; command2`	依次执行 `command1` 和 `command2`
`command1 && command2`	执行 `command1`；如果成功，按顺序执行 `command2`（如果 `command1` 和 `command2` 都执行成功了，返回 success）
`command1 || command2`	执行 `command1`；如果不成功，按顺序执行 `command2`（如果 `command1` 或 `command2` 执行成功，返回 success）
`command > foo`	将 `command` 的标准输出重定向到文件 `foo`（覆盖）
`command 2> foo`	将 `command` 的标准错误重定向到文件 `foo`（覆盖）
`command >> foo`	将 `command` 的标准输出重定向到文件 `foo`（附加）
`command 2>> foo`	将 `command` 的标准错误重定向到文件 `foo`（附加）
`command > foo 2>&1`	将 `command` 的标准输出和标准错误重定向到文件 `foo`
`command < foo`	将 `command` 的标准输入重定向到文件 `foo`
`command << delimiter`	将 `command` 的标准输入重定向到下面的命令行，直到遇到“`delimiter`”（here document）
`command <<- delimiter`	将 `command` 的标准输入重定向到下面的命令行，直到遇到“`delimiter`”（here document，命令行中开头的制表符会被忽略）

Table 1.23: Shell 命令常见用法

Debian 系统是一个多任务的操作系统。后台任务让用户能够在一个 shell 中执行多个程序。后台进程的管理涉及 shell 的内建命令：`jobs`、`fg`、`bg` 和 `kill`。请阅读 bash(1) 中的章节：“SIGNALS”、“JOB CONTROL”和“bash-builtins(1)”。

尝试下列例子

```
$ </etc/motd pager
```

```
$ pager </etc/motd
```

```
$ pager /etc/motd
```

```
$ cat /etc/motd | pager
```

尽管 4 个 shell 重定向的例子都会显示相同的结果，但最后一个例子毫无意义地运行了额外的 `cat` 命令浪费了资源。

shell 允许你使用 `exec` 通过任意一个文件描述符来打开文件。

```
$ echo Hello >foo
$ exec 3<foo 4>bar  # open files
$ cat <&3 >&4       # redirect stdin to 3, stdout to 4
$ exec 3<&- 4>&-    # close files
$ cat bar
Hello
```

预定义的文件描述符 0-2。

设备	说明	文件描述符
stdin	标准输入	0
stdout	标准输出	1
stderr	标准错误	2

Table 1.24: 预定义的文件描述符

1.5.9 命令别名

你可以为经常使用的命令设置一个别名。

尝试下列例子

```
$ alias la='ls -la'
```

现在，“la”是“ls -al”的简写形式，并同样会以长列表形式列出所有的文件。

你可以使用 alias 来列出所有的别名（参见 bash(1) 中的“SHELL BUILTIN COMMANDS”）。

```
$ alias
...
alias la='ls -la'
```

你可以使用 type 来确认命令的准确路径或类型（参见 bash(1) 中的“SHELL BUILTIN COMMANDS”）。

尝试下列例子

```
$ type ls
ls is hashed (/bin/ls)
$ type la
la is aliased to ls -la
$ type echo
echo is a shell builtin
$ type file
file is /usr/bin/file
```

ls 在最近被使用过，而“file”没有，因此“ls”标记为“hashed”（被录入哈希表），即 shell 有一个内部的记录用来快速访问“ls”所处的位置。

> 提示
> 参见第 9.3.6 节。

1.6 类 Unix 的文本处理

在类 Unix 的工作环境中，文本处理是通过使用管道组成的标准文本处理工具链完成的。这是另一个重要的 Unix 创新。

1.6.1 Unix 文本工具

这里有一些在类 Unix 系统中经常使用到的标准文本处理工具。

- 没有使用正则表达式：
  - cat(1) 连接文件并输出全部的内容。

  - tac(1) 连接文件并反向输出。
  - cut(1) 选择行的一部分并输出。
  - head(1) 输出文件的开头。
  - tail(1) 输出文件的末尾。
  - sort(1) 对文本文件的行进行排序。
  - uniq(1) 从已排序的文件中移除相同的行。
  - tr(1) 转换或删除字符。
  - diff(1) 对文件的行进行对比。
- 默认使用基础正则表达式（**BRE** ）：
  - ed(1) 是一个原始行编辑器。
  - sed(1) 是一个流编辑器。
  - grep(1) 匹配满足 pattern 的文本。
  - vim(1) 是一个屏幕编辑器。
  - emacs(1) 是一个屏幕编辑器。（有些扩展的 **BRE** ）
- 使用扩展的正则表达式（**ERE** ）：
  - awk(1) 进行简单的文本处理。
  - egrep(1) 匹配满足多个 pattern 的文本。
  - `tcl`(3tcl) 可以进行任何你想得到的文本处理：参见 re_syntax(3) 。经常与 `tk`(3tk) 一起使用。
  - perl(1) 可以进行任何你想得到的文本处理。参见 perlre(1) 。
  - `pcre2-utils` 软件包中的 pcre2grep(1) 可以匹配满足 Perl 兼容正则表达式（PCRE）模式的文本。
  - 带有 `re` 模块的 python3(1) 可以进行任何你想得到的文本处理。参见“`/usr/share/doc/python3/html/index.html`

如果你不确定这些命令究竟做了什么，请使用“`man command`”来自己把它搞清楚吧。

注意
排序的顺序和表达式的范围取决于语言环境。如果你想要获得一个命令的传统行为，可以使用 **C** 语言环境或 **C.UTF-8** 语言环境代替原来的 **UTF-8** 语言环境（参见第 8.1 节）。

注意
Perl 正则表达式（perlre(1) ）、Perl 兼容正则表达式（PCRE）和 Python 的 `re` 模块提供的正则表达式与一般的 **ERE** 相比多了许多通用的扩展。

1.6.2 正则表达式

正则表达式被使用在许多文本处理工具中。它们类似 shell 的通配符，但更加复杂和强大。

正则表达式描述要匹配的模式，它是由文本字符和元字符构成的。

元字符仅仅是带有特殊含义的字符。它们有两种主要的形式，**BRE** 和 **ERE** ，使用哪种取决于上述的文本工具。

emacs 中的正则表达式基本上是 **BRE** 但含有 **ERE** 中的元字符“+”和“?”。因此，在 `emacs` 中没必要使用“\”来转义它们。

grep(1) 可以使用正则表达式来进行文本搜索。

尝试下列例子

BRE	ERE	正则表达式的描述
\ . [ ] ^ $ *	\ . [ ] ^ $ *	通用的元字符
\+ \? \(\) \{ \} |		BRE 独有的“\”转义元字符
	+ ? ( ) { } |	ERE 独有的不需要“\”转义的元字符
c	c	匹配非元字符“c”
\c	\c	匹配一个字面意义上的字符“c”，即使“c”本身是元字符
.	.	匹配任意字符，包括换行符
^	^	字符串的开始位置
$	$	字符串的结束位置
\<	\<	单词的开始位置
\>	\>	单词的结束位置
[abc…]	[abc…]	匹配在“abc...”中的任意字符
[^abc…]	[^abc…]	匹配除了“abc...”中的任意字符
r*	r*	匹配零个或多个“r”
r\+	r+	匹配一个或多个“r”
r\?	r?	匹配零个或一个“r”
r1|r2	r1|r2	匹配一个“r1”或“r2”
\(r1|r2\)	(r1|r2)	匹配一个“r1”或“r2“，并作为括号内的正则表达式

Table 1.25: BRE 和 ERE 中的元字符

```
$ egrep 'GNU.*LICENSE|Yoyodyne' /usr/share/common-licenses/GPL
GNU GENERAL PUBLIC LICENSE
GNU GENERAL PUBLIC LICENSE
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
```

提示
参见第 9.3.6 节。

1.6.3 替换表达式

对于替换表达式，一些字符有特殊的含义。

替换表达式	替换表达式替换的文本
&	正则表达式所匹配的内容（在 emacs 中使用 \& ）
\n	前 n 个括号的正则表达式匹配的内容（“n”是数字）

Table 1.26: 替换表达式

对 Perl 替换字符串来说，应使用“$&”而非“&”，应使用“$n”而非“\n”。

尝试下列例子

```
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/=&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
sed -E -e 's/(1[a-z]*)[0-9]*(.*)$/=&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
```

```
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/=$&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/\2===\1/'
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
sed -E -e 's/(1[a-z]*)[0-9]*(.*)$/\2===\1/'
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/$2===$1/'
zzzefg3hij4===1abc
```

请特别注意这些括号正则表达式的格式，以及这些被匹配的文本在不同的工具中是如何被替换的。

这些正则表达式在一些编辑器中也可以用来移动光标和替换文本。

在 shell 命令行行末的反斜杠“\”会跳脱一个换行符（作为空白符），并将光标移动到下一行的行首。

请阅读所有相关手册来学习这些命令。

1.6.4 正则表达式的全局替换

ed(1) 命令可以在“file”中将所有的“FROM_REGEX”替换成“TO_TEXT”。

```
$ ed file <<EOF
,s/FROM_REGEX/TO_TEXT/g
w
q
EOF
```

sed(1) 命令可以在“file”中将所有的“FROM_REGEX”替换成“TO_TEXT”。

```
$ sed -i -e 's/FROM_REGEX/TO_TEXT/g' file
```

vim(1) 命令可以通过使用 ex(1) 命令在“file”中将所有的“FROM_REGEX”替换成“TO_TEXT”。

```
$ vim '+%s/FROM_REGEX/TO_TEXT/gc' '+update' '+q' file
```

提示
上面的“c”标志可以确保在每次替换时都进行交互式的确认。

多个文件（“file1”，“file2”和“file3”）可以使用 vim(1) 或 perl(1) 通过正则表达式进行类似的处理。

```
$ vim '+argdo %s/FROM_REGEX/TO_TEXT/gce|update' '+q' file1 file2 file3
```

提示
上面的“e”标志是为了防止“No match”错误中断替换。

```
$ perl -i -p -e 's/FROM_REGEX/TO_TEXT/g;' file1 file2 file3
```

在 perl(1) 例子中，”-i” 是在每一个目标文件的原处编辑，”-p” 是表示循环所有给定的文件。

提示
使用参数“-i.bak”代替“-i”，可以在文件名后添加“.bak”再保存。对于复杂的替换，这使得从错误中恢复变得容易。

注意
ed(1) 和 vim(1) 使用 **BRE** ；perl(1) 使用 **ERE** 。

1.6.5 从文本文件的表格中提取数据

下面有一个文本文件“DPL”，里面含有 2004 年以前 Debian 项目的领导者名字和起始日期，并以空格分隔。

```
Ian     Murdock   August  1993
Bruce   Perens    April   1996
Ian     Jackson   January 1998
Wichert Akkerman  January 1999
Ben     Collins   April   2001
Bdale   Garbee    April   2002
Martin  Michlmayr March   2003
```

提示
参见“Debian 简史”获取最新的历届 Debian 领导人信息。

Awk 经常被用来从这种类型的文件中提取数据。

尝试下列例子

```
$ awk '{ print $3 }' <DPL                   # month started
August
April
January
January
April
April
March
$ awk '($1=="Ian") { print }' <DPL          # DPL called Ian
Ian     Murdock   August  1993
Ian     Jackson   January 1998
$ awk '($2=="Perens") { print $3,$4 }' <DPL # When Perens started
April 1996
```

Shell （例如 Bash ）也可以用来分析这种文件。

尝试下列例子

```
$ while read first last month year; do
    echo $month
  done <DPL
... same output as the first Awk example
```

内建命令 `read` 使用“`$IFS`”（内部域分隔符）中的字符来将行分隔成多个单词。

如果你将“`$IFS`”改变为“`:`”，你可以很好地使用 shell 来分析“`/etc/passwd`”。

```
$ oldIFS="$IFS"   # save old value
$ IFS=':'
$ while read user password uid gid rest_of_line; do
    if [ "$user" = "bozo" ]; then
      echo "$user's ID is $uid"
    fi
  done < /etc/passwd
bozo's ID is 1000
$ IFS="$oldIFS"   # restore old value
```

(如果要用 Awk 做到相同的事，使用“FS=':'”来设置域分隔符。)

IFS 也被 shell 用来分割参数扩展、命令替换和算术扩展的结果。这不会出现在双引号或单引号中。IFS 的默认值为空格、*tab* 和换行符。

请谨慎使用 shell 的 IFS 技巧。当 shell 将脚本的一部分解释为对它的输入时，会发生一些奇怪的事。

```
$ IFS=":,"                          # use ":" and "," as IFS
$ echo IFS=$IFS,   IFS="$IFS"       # echo is a Bash builtin
IFS=  , IFS=:,
$ date -R                           # just a command output
Sat, 23 Aug 2003 08:30:15 +0200
$ echo $(date -R)                   # sub shell --> input to main shell
Sat  23 Aug 2003 08 30 36 +0200
$ unset IFS                         # reset IFS to the default
$ echo $(date -R)
Sat, 23 Aug 2003 08:30:50 +0200
```

1.6.6 用于管道命令的小片段脚本

下面的脚本作为管道的一部分，可以做一些细致的事情。

脚本片段（在一行内输入）	命令效果
`find /usr -print`	找出“`/usr`”下的所有文件
`seq 1 100`	显示 1 到 100
`| xargs -n 1 command`	把从管道过来的每一项作为参数，重复执行命令
`| xargs -n 1 echo`	把从管道过来的，用空格隔开的项，分隔成多行
`| xargs echo`	把从管道过来的所有行合并为一行
`| grep -e regex_pattern`	从管道过来，包含有 *regex_pattern* 的行，提取出来
`| grep -v -e regex_pattern`	把从管道过来，不包含有 *regex_pattern* 的行，提取出来
`| cut -d: -f3 -`	把从管道过来，用“`:`”分隔的第三列提取出来 (passwd 文件等。)
`| awk '{ print $3 }'`	把用空格隔开的第三列提取出来
`| awk -F'\t' '{ print $3 }'`	把用 tab 键隔开的第三列提取出来
`| col -bx`	删除退格键，扩展 tab 键为空格键
`| expand -`	扩展 tab 键到空格键
`| sort| uniq`	排序并删除重复行
`| tr 'A-Z' 'a-z'`	将大小字母转换为小写字母
`| tr -d '\n'`	将多行连接为一行
`| tr -d '\r'`	删除换行回车符
`| sed 's/^/# /'`	在每行行首增加一个“#” 符
`| sed 's/\.ext//g'`	删除“`.ext`”
`| sed -n -e 2p`	显示第二行
`| head -n 2 -`	显示最前面两行
`| tail -n 2 -`	显示最后两行

Table 1.27: 管道命令的小片段脚本列表

使用 find(1) 和 xargs(1)，单行 shell 脚本能够在多个文件上循环使用，可以执行相当复杂的任务。参见第 10.1.5 节和第 9.4.9 节.

当使用 shell 交互模式变得太麻烦的时候，请考虑写一个 shell 脚本 (参见第 12.1 节).

Chapter 2

Debian 软件包管理

注意
这一章假定最新的稳定版的代号为：trixie 。
在本文档中，APT 系统的数据源总称为源列表。能够在”/etc/apt/sources.list” 文件、”/etc/apt/sources.list.d/*.list” 文件或”/etc/apt/sources.list.d/*.sources” 文件的任何地方定义。

2.1 Debian 软件包管理的前提

2.1.1 Debian 软件包管理

Debian 是一个志愿者组织，它建立一致的自由软件的预编译二进制包并从档案库中分发它们。

许多远程镜像站提供了 HTTP 和 FTP 的方式来访问 Debian 档案库。也可以使用 CD-ROM/DVD 。

目前 Debian 的软件包管理系统是高级软件包工具 (APT)，它能够使用所有这些资源。

Debian 软件包管理系统，当使用适当时，可以让用户从档案库安装统一设置的二进制软件包到系统中。现在，有 78709 个可用于 amd64 架构的软件包。

Debian 软件包管理系统有丰富的历史，有许多可供选择的前端用户程序和后端访问方式。现在，我们推荐下列的这些。

- apt(8) 用于所有的交互式命令行操作，包含软件包的安装、移除和版本升级。
- apt-get(8) 用于从脚本中调用 Debian 软件包管理系统。它在 apt 不可用时也可作为一个备选选项（常见于较旧的 Debian 系统）。
- aptitude(8) 使用一个交互式的文本界面来管理已安装的软件包和搜索可用的软件包。

2.1.2 软件包配置

下面是 Debian 系统软件包配置的一些要点。

- 对于现代桌面环境来说，在软件包配置修改和软件包升级后重启系统是确保系统正常运行的好主意。
- Debian 尊重系统管理员的手动配置。换句话说，软件包配置系统不会为了方便而去更改那些配置。

软件包	流行度	大小	说明
dpkg	V:909, I:1000	5086	用于 Debian 的底层软件包管理系统（基于文件的）
apt	V:882, I:1000	5025	使用命令行管理软件包的 APT 前端：apt/apt-get/apt-cache
aptitude	V:32, I:154	4837	使用全屏控制台交互式管理软件包的 APT 前端：aptitude(8)
tasksel	V:42, I:985	352	用来安装选择的任务的 APT 前端: tasksel(8)
unattended-upgrades	V:142, I:190	374	用于 APT 的增强软件包，会自动安装安全更新
gnome-software	V:159, I:260	4732	GNOME 软件中心 (图形化的 APT 前端)
synaptic	V:31, I:262	7264	图形化的软件包管理工具（GTK 的 APT 前端）
apt-utils	V:383, I:996	1187	APT 实用程序：apt-extracttemplates(1) 、apt-ftparchive(1) 和 apt-sortpkgs(1)
apt-listchanges	V:451, I:895	563	软件包历史更改提醒工具
apt-listbugs	V:4.7, I:6.4	512	在每次 APT 安装前列出严重的 bug
apt-file	V:14, I:52	76	APT 软件包搜索工具——命令行界面
apt-rdepends	V:0.4, I:4.2	39	递归列出软件包依赖

Table 2.1: Debian 软件包管理工具列表

- 每个软件包都带有自己的配置脚本，它使用标准用户接口 debconf(7) 来帮助软件包初始化安装过程。
- Debian 开发者通过软件包配置脚本，尽力使你能有一个完美的升级体验。
- 系统管理员可以使用软件包工具的全部功能。但在默认的安装中会禁用那些具有安全风险的。
- 如果你手动激活了一些具有安全隐患的服务，你有责任遏制风险。
- 高深的配置可以由系统管理员手动启用。这可能会对用于系统配置的通用流行帮助程序造成干扰。

2.1.3 基本的注意事项



警告
不要从任何的混合套件中安装软件包。它可能会打破软件包的一致性，这需要你要深厚的系统管理知识，例如 ABI 编译器、库版本和解释器特性等等。

Debian 系统管理员中的新手应该保持在只进行安全更新的 **stable** 版本。直到你十分了解 Debian 系统前，你应当遵循下列的预防措施。

- 在源文件中不要包含 **testing** 或 **unstable** 。
- 在源文件里不要在标准的 Debian 中混合使用其它非 Debian 的档案库，例如 Ubuntu 。
- 不要建立“/etc/apt/preferences”。
- 不了解会造成的全部影响，就不要通过配置文件改变软件包管理工具的默认行为。
- 不要使用“dpkg -i *random_package*”安装任何软件包。
- 绝不使用“dpkg --force-all -i *random_package*”安装任何软件包。
- 不要删除或修改“/var/lib/dpkg/”中的文件。

- 不要让从源码直接安装的程序覆盖系统文件。
  - 如果需要的话，将它们安装到“/usr/local”或“/opt”中。

对 Debian 软件包管理系统，违背上面的预防措施，会导致不兼容影响，可能会使你的系统无法使用。

运行关键任务服务器的严谨的 Debian 系统管理员，应该使用额外的预防措施。

- 没有在安全的条件下使用你特定的配置进行彻底地测试，就不要从 Debian 安装任何软件包（包含安全更新）。
  - 你作为系统管理员要对你的系统负责到底。
  - Debian 系统长久的稳定史并无法保证什么。

2.1.4 持续升级的生活



小心
对于你的生产服务器，建议使用带有安全更新的 stable 套件。对于你只进行有限管理的桌面 PC 也是同样如此。

尽管我在上面进行了警告，我知道本文档的许多读者希望可以运行更新的 testing 或 unstable 版。

菩萨使用下面的内容拯救一个人，使他从挣扎于持续升级地狱的因果报应中脱困，并让他达到 Debian 的极乐世界。

这个列表面向自己管理的桌面环境。

- 使用 testing 版，实际上，它是自动滚动发布的，由 Debian 档案库的 QA 质量架构来管理，比如：Debian 持续集成、只上传源代码实践和库转换跟踪。在 testing 版中的软件包被更新得足够频繁来提供全部最新的特性。
- 在源列表里面设置 testing 版相应的代码名为套件名（在 trixie-作为-stable 版的发布周期时，是”forky”）。
- 大概在主版本发布一个月后，仅仅在你自己评估了形势后，才手动更新源文件里的这个代码名到新的版本号。对于这个更新，Debian 用户和开发者邮件列表也是好的信息来源。

使用 unstable 版是不推荐的。unstable 版对开发者调试软件包合适，但对普通的桌面使用而言，会有使你暴露在不必要的风险中的倾向。尽管 Debian 系统的 unstable 版在大多数时候看起来都非常稳定，但会有一些软件包问题，并且它们中的一部分是不容易解决的。

这里有一些基本预防措施意见，确保简单快速地从 Debian 软件包的 bug 中恢复。

- 使用下面的第 3.2 节制作紧急救援系统。
- 通过将 Debian 系统的 stable 版安装到另一个分区，可以使系统能够进行双启动
- 考虑安装 apt-listbugs ，这可以在升级之前检查 Debian Bug 跟踪系统（BTS）的信息
- 对软件包系统的基础设施有足够的了解来解决问题



小心
如果你无法做到这些预防措施中的任何一个，那你可能还没做好使用 testing 和 unstable 版的准备。

2.1.5 Debian 档案库基础

提示
Debian 档案库官方政策的定义参见 Debian 政策文档，第 2 章——Debian 档案库。

让我们从系统用户的角度来看看 Debian 档案库。

对于系统用户，是使用 APT 系统来访问 Debian 档案库。

APT 系统定义它的数据源作为源列表，在 sources.list(5) 里面描述。

对于使用典型的 HTTP 访问的 trixie 系统，源列表是在"/etc/apt/sources.list.d/debian.sources" 中由现代的如下的 deb822 式样提供：

```
Types: deb deb-src
URIs: http://deb.debian.org/debian/
Suites: trixie trixie-updates
Components: main non-free-firmware contrib non-free

Types: deb deb-src
URIs: http://security.debian.org/debian-security/
Suites: trixie-security
Components: main non-free-firmware contrib non-free
```

提示
如果源列表是在"/etc/apt/sources.list" 或"/etc/apt/sources.list.d/*.list" 文件中提供的老旧单行式样，使用它们更新：

```
$ sudo apt modernize-sources
```

deb822 式样的源文件的关键点如下。

- 它的定义文件在"/etc/apt/sources.list.d/*.sources" 文件里。
- 由空格隔开的每个多行块，定义了 APT 系统的数据源。
- "Types:" 章节定义列表类型，即"deb" 和"deb-src"。
- "URIs:" 章节定义 Debian 档案库 URI 的根地址。
- "Suites:" 章节定义了发行版名称列表，名称可以使用套件名或代号。
- "Components:" 章节定义 Debian 档案库中有效档案库名称列表。

如果只是用 aptitude，它不访问源代码相关的元数据，"deb-src"定义可以安全地省略。这可以加速档案库元数据的更新。

URL 可以是"https://", "http://", "ftp://", "file://",……

"#" 开头的行是注释，被忽略。

这里，我倾向于使用代号"trixie"或"forky" 来代替套件名"stable"或"testing" ，以避免下一个 stable 版本发布时出现意外。

提示
如果在上述的例子中，使用了 "sid" 代替 "trixie"，那么源列表中用于安全更新的 "deb: http://security.debian.org/ …" 这行或它的 deb822 等价内容就不需要了。因为没有用于 "sid"（unstable）的安全更新的档案库。

档案库 **URL**	套件名	代码名	仓库用途
http://deb.debian.org/debian/	`stable`	`trixie`	在扩展检查后，类似静态的 `stable` 发布
http://deb.debian.org/debian/	`stable-updates`	`trixie-updates`	稳定版的紧急更新（重要）
http://security.debian.org/debian-security/	`stable-security`	`trixie-security`	稳定版的安全更新（重要）
http://deb.debian.org/debian/	`testing`	`forky`	在表面检查和短期等待后的动态 `testing` 发布
http://deb.debian.org/debian/	`unstable`	`sid`	在最少的检查和不等待的动态 `unstable` 发布
http://deb.debian.org/debian/	`experimental`	N/A	开发者预发布实验版本（可选，只适用于开发者）
http://deb.debian.org/debian/	`stable-proposed-updates`	`trixie-proposed-updates`	用于下一个稳定版 `stable` 的点版本（小版本）发布的更新（可选）
http://deb.debian.org/debian/	`stable-backports`	`trixie-backports`	大部分从 `testing` 版中随机收集和重新编译的软件包（可选）
http://security.debian.org/debian-security/	`testing-security`	`forky-security`	这个没有积极支持，不被安全团队使用

Table 2.2: Debian 档案库站点列表

在 `trixie` 发布后，下面是配置文件所使用的 Debian 档案库站点的 URL 和套件名或代号的列表。



小心
只有带有安全更新的纯净的 **`stable`** release 版本可以提供最佳的稳定性。运行大多数 **`stable`** release 版本的软件包之中混合一些来自 **`testing`** 或 **`unstable`** release 版本的软件包会比运行纯净的 **`unstable`** release 版本冒更大的风险，这是因为库版本的不匹配导致的。如果在 **`stable`** release 版本下你真的需要一些程序的最新版本，请使用来自 stable-updates 和 backports （参见第 2.7.4 节）的软件包。使用这些软件包时必须额外小心。



小心
在"deb" 行中，你只需列出 `stable`，`testing` 或者 `unstable` 套件中的一个即可，如果你在"deb" 行中混合了 `stable`，`testing` 和 `unstable` 套件，APT 程序的执行速度将会变慢并且只有最新的档案库是有用的。只有在"`/etc/apt/preferences`" 文件带有明确目标的时候，混合的列表才是有意义的。（查看第 2.7.7 节）。

提示
对于使用 `stable` 套件的 Debian 系统而言，在源列表中包含带有"`http://security.debian.org/`"的内容是不错的主意。它会启用安全更新。

注意
Debian 安全团体将会修正 `stable` 档案库的安全缺陷。这些行为是十分严格可靠的。`testing` 档案库中的缺陷，不一定会被 Debian 测试安全团体修正。由于一些原因，这些行为相对 `stable` 档案库没有那么严格，您可能需要等待已修正的 `unstable` 软件包移植到 `testing`。`unstable` 档案库的缺陷，交由各个维护者修改。经常维护的 `unstable` 软件包通常处于相当好的状况，因为它利用了上流最新的安全修正。有关 Debian 怎样处理安全缺陷，请参见 Debian 安全常问问题 。

区域	软件包数量	软件包组件标准
`main`	77218	遵从 Debian 自由软件指导方针（DFSG），并且不依赖于 `non-free`
`non-free-firmware`	85	不符合 Debian 自由软件指导方针（DFSG），正常的系统安装过程中必需要用到的固件
`contrib`	365	遵从 Debian 自由软件指导方针（DFSG），但依赖于 `non-free`
`non-free`	1041	不遵从 Debian 自由软件指导方针（DFSG），并且不在 `non-free-firmware`

Table 2.3: Debian 归档区域（area）列表

上述软件包的数量是 amd64 架构的。`main` 区域提供 Debian 系统（参见第 2.1.6 节）。

通过把你的浏览器指向档案库 URL，这些 URL 在 `dists` 或 `pool` 之后是各不相同的，Debian 档案库能够被有规划的组织。

发行版可以用套件或代号来指定。发行版在许多文档中也被当做是套件的同义词。套件和代号的关系总结如下。

时间	**suite = `stable`**	**suite = `testing`**	**suite = `unstable`**
在 `trixie` 发布后	codename = `trixie`	codename = `forky`	codename = `sid`
在 `forky` 发布后	codename = `forky`	codename = `duke`	codename = `sid`

Table 2.4: 套件和代号的关系

代号的历史参见 Debian FAQ: 6.2.1 以前用过哪些代号名？

在较严格的 Debian 档案术语，“部分 section”这一词特指按应用领域来分类的软件包类别。（但是，主要部分 (”main section”) 这一词有时会用来描述 Debian 档案区中，名为”main 主要” 的区域。）

Debian 开发者（DD）每次上传软件包到 `unstable` 档案库（通过 incoming 处理），都必须确保上传的软件包与最新的 `unstable` 档案库中的最新软件包兼容。

如果 DD 故意打破重要的库升级等的这种兼容性，这通常会在 Debian 开发者邮件列表等进行公告。

在 Debian 档案库维护脚本将软件包从 `unstable` 档案库移动到 `testing` 档案库前，档案库维护脚本不仅检查时间 (约 2-10 天）和软件包的 RC bug 报告的状态，还尝试确保它们可以和最新的 `testing` 档案库中的软件兼容。这个过程使得 `testing` 档案库非常正确可用。

通过由发布团队领导的逐步冻结档案库的过程，并进行一些手动干预，使 `testing` 档案库完全一致，无缺陷。然后，将旧的 `testing` 档案库的代码名称分配给新的 `stable` 档案库，并为新的 `testing` 档案库创建新的代码名称。新的 `testing` 档案库最初的内容和新发布的 `stable` 档案库的内容完全相同。

`unstable` 和 `testing` 档案库都可能会遭受由以下几个因素导致的临时的小故障。

- 损坏的软件包被上传到档案库（多见于 `unstable` ）
- 延迟接受新的软件包到档案库（多见于 `unstable`)
- 档案库时间同步问题（`testing` 和 `unstable` ）
- 手动干预档案库，例如移除软件包（多见于 `testing` ）等。

因此，如果你决定使用这些档案库，你应该能够修复或忍受这些类型的小故障。



小心

在新的 `stable` 版本发布后的几个月，大多数桌面用户应该使用带有安全更新的 `stable` 档案库，即使他们通常使用 `unstable` 或 `testing` 档案库。在这个过渡期中，`unstable` 和 `testing` 档案库不适合大多数人。你使用 `unstable` 档案库的系统是很难保持良好的工作状态的，因为它会遭受核心软件包的大量升级狂潮。`testing` 档案库不大有用，因为它包含有和没有安全支持的 `stable` 档案库相同的内容（Debian testing 安全公告 2008-12）。一个月左右的时间后，如果你仔细点的话，`unstable` 或 `testing` 档案库或许可以使用。

提示
跟踪 testing 档案库时，由一个已移除的软件包引起的问题通常可以安装 unstable 档案库中相同的软件包（已修复 bug) 来解决。

档案库的定义参见 Debian 政策文档。

- 部分
- ” 优先级”
- ” 基本系统”
- ” 极重要的软件包”

2.1.6 Debian 是 100% 的自由软件

Debian 是 100% 的自由软件，因为：

- Debian 默认只安装自由软件，这尊重了用户的自由。
- Debian 在 main 中只提供自由软件。
- Debian 建议只运行来自 main 的自由软件。
- 在 main 中的软件包，没有依赖或推荐在 non-free 或 non-free-firmware 或 contrib 中的软件包。

有人想知道下列的两个事实是否互相矛盾。

- “Debian 将始终是 100% 的自由软件”。（Debian 社群契约中的第一条）
- Debian 服务器上有一些 non-free-firmware、non-free 和 contrib 软件包。

因为下列原因，这并不矛盾。

- Debian 系统具有 100% 的自由，并且它的软件包位于 Debian 服务器的 main 区域。
- Debian 系统之外的软件包位于 Debian 服务器的 non-free、non-free-firmware 和 contrib 区域。

在 Debian 社群契约的第 4 条和第 5 条对这进行了明确的解释：

- 我们将优先考虑我们的用户及自由软件
  - 我们由我们的用户及自由软件社群的需要所导向。我们将优先考虑他们的利益。我们将在多种计算环境中支持我们的用户的操作需要。我们不反对在 Debian 系统上使用非自由软件，我们也不会尝试向创建和使用这部分软件的用户索取费用。我们允许他人，在没有我们的资金的参与下，制造包括 Debian 以及商业软件的增值套件。为了达成这些目标，我们将提供集成的、高质量的、100% 自由的软件，而不附加任何可能阻止在这些方面使用的法律限制。
- 哪些作品不符合我们的自由软件规范
  - 我们知道，某些我们的用户需要使用不符合 Debian 自由软件指导方针的作品。我们为这些作品，在我们的档案库中留出了”non-free”、”non-free-firmware” 和”contrib” 目录。在这些目录下的软件包，并不属于 Debian 系统尽管它们已被配置成可以在 Debian 下使用。我们鼓励光盘制造商阅读这些目录下的软件的许可证，以判断他们是否可以在光盘中发行这些软件。所以，尽管非自由软件并非 Debian 系统的一部分，我们仍支持它们的使用，并且我们为非自由软件提供了公共资源 (诸如我们的缺陷跟踪系统以及邮件列表)。Debian 官方媒介可以包括固件，固件不是 Debian 系统的一部分，这是一个例外，能够让 Debian 用于需要这些固件的硬件上。

注意
在目前的 Debian 社群契约（Debian Social Contract） 1.2 版本第 5 条条款的实际文本和上面的文本有稍微不同。在不改变 Debian 社群契约实际内容下，这个文字调整让本用户文档在逻辑上保持一致。

用户应该了解使用 non-free 、non-free-firmware 和 contrib 中的软件包所需要冒的风险：

- 使用类似的软件包会失去自由
- 失去 Debian 对软件包的支持（这些软件包无法访问源代码，Debian 不能进行完全的支持。）
- 污染你 100% 自由的 Debian 系统

Debian 自由软件指导方针为 Debian 设立了自由软件标准。Debian 对软件包中的软件做了最广泛的解释，包含文档、固件、图标和图形数据。这使得 Debian 的自由软件标准非常严格。

典型的 non-free、non-free-firmware 和 contrib 软件包包含了下列类型的自由分发的软件包：

- 在 GNU Free Documentation License 下的文档包，包含不变的部分，比如 GCC 和 Make 的。（大多数都可以在 non-free/doc 找到.)
- 包含没有源代码的二进制数据的固件软件包，例如在第 9.10.5 节中作为 non-free-firmware 列出的软件包。(多见于 non-free-firmware/kernel 部分。)
- 游戏和字体软件包，对商业使用和/或内容修改进行了限制。

请注意，non-free、non-free-firmware 和 contrib 软件包的数量少于 main 软件包的 2%。允许访问 non-free、non-free-firmware 和 contrib 并不会模糊软件包的来源。使用 aptitude(8) 的全屏交互式界面可以提供完全的可见性和完全的控制，可以让你决定安装来自某个部分的软件包，来使你的系统保持自由。

2.1.7 软件包依赖关系

Debian 系统通过其控制文件字段中的版本化二进制依赖声明机制来提供一致的二进制软件包集合。下面有一些它们的简单定义。

- “依赖”
  - 绝对的依赖，所有在这里列出的软件包都必须同时或提前安装。
- ” 预依赖”
  - 类似于 Depends，但列出的软件包必须提前完成安装。
- ” 推荐”
  - 这里表示一个强，但不是绝对的依赖关系。大多数用户不会想要这个包，除非在这里列出的所有包都已经安装。
- ” 建议”
  - 较弱的依赖。这个软件包的大多数用户可能会从安装所列的软件包中受益，但没有它们也可以有适当的功能。
- ” 增强”
  - 这里表明一个像建议的弱依赖关系，不装也没关系。
- ” 破损”
  - 表明一个软件包不兼容一些版本规范。一般的解决方法就是升级列出的所有软件包。
- ” 冲突”

  - 这表明了绝对的不兼容。为了安装这个软件包必须移除所有列出的软件包。
- ” 替代”
  - 这表明这个文件安装的文件会替代所列的软件包的文件。
- ” 提供”
  - 表明这个软件包会提供所列的软件包所有的文件和功能。

注意
请注意，同时将“Provides”、“Conflicts”和“Replaces”定义到一个虚拟的软件包是一个明智的配置。这确保了在任何一个时间只能安装一个提供该虚拟包的真正软件包。

包含源代码依赖关系的官方定义位于 the Policy Manual: Chapter 7 - Declaring relationships between packages。

2.1.8 包管理的事件流

这是 APT 提供的软件包管理的简单事件流摘要。

- 更新（“`apt update`”、“`aptitude update`”或“`apt-get update`”）：
  1. 从远程档案库获取档案库元数据
  2. 重建和更新 APT 使用的本地元数据
- 升级（“`apt upgrade`”和“`apt full-upgrade`”，或“`aptitude safe-upgrade`”和“`aptitude full-upgrade`”，或“`apt-get upgrade`”和“`apt-get dist-upgrade`”）：
  1. 选择候选版本，它所安装的软件包通常都是最新的可用版本（例外参见第 2.7.7 节）
  2. 解决软件包依赖关系
  3. 如果候选版本与已安装的版本不同，会从远程档案库获取所选择的二进制软件包
  4. 解包所获取的二进制软件包
  5. 运行 **preinst** 脚本
  6. 安装二进制文件
  7. 运行 **postinst** 脚本
- 安装（“`apt install ...`”、“`aptitude install ...`” 或者”`apt-get install …`”）：
  1. 选择命令行中列出的包
  2. 解决软件包依赖关系
  3. 从远程服务器获取已选二进制包
  4. 解包所获取的二进制软件包
  5. 运行 **preinst** 脚本
  6. 安装二进制文件
  7. 运行 **postinst** 脚本
- 移除（”`apt remove …`”, “`aptitude remove …`”或“`apt-get remove …`”）：
  1. 选择命令行中列出的包
  2. 解决软件包依赖关系
  3. 运行 **prerm** 脚本
  4. 移除已安装的文件，除了配置文件

5. 运行 **postrm** 脚本

- 清除（"`apt purge`","`aptitude purge …`"或"`apt-get purge …`"）：
  1. 选择命令行中列出的包
  2. 解决软件包依赖关系
  3. 运行 **prerm** 脚本
  4. 移除已安装的文件，包含配置文件
  5. 运行 **postrm** 脚本

这里，为了大局，我特意省略了技术细节。

2.1.9 对包管理问题的第一个回应

你应该阅读优良的官方文档。第一个阅读的文档是 Debian 特定的"`/usr/share/doc/`*`package_name`*`/README.Debian`"。同时也应该查询"`/usr/share/doc/`*`package_name`*`/`"中的其它文档。如果你设置 shell 为第 1.4.2 节，输入下列命令。

```
$ cd package_name
$ pager README.Debian
$ mc
```

你可能需要安装以"`-doc`"后缀命名的对应文档软件包来获取详细的信息。

如果你在使用一个特定的软件包时出现了问题，一定要首先检查 Debian bug 跟踪系统（BTS）网站。

网站	命令
Debian bug 跟踪系统（BTS）的主页	`sensible-browser "https://bugs.debian.org/"`
软件包名称已知的 bug 报告	`sensible-browser "https://bugs.debian.org/`*`package_name`*`"`
bug 编号已知的 bug 报告	`sensible-browser "https://bugs.debian.org/`*`bug_number`*`"`

Table 2.5: 解决特定软件包问题的主要网站

使用 Google 搜索，在关键字中包含"`site:debian.org`"，"`site:wiki.debian.org`"，"`site:lists.debian.org`"等等。

当你要发送一份 bug 报告时，请使用 reportbug(1) 命令。

2.1.10 如何挑选 **Debian** 软件包

当遇到 2 个以上的类似的软件包时，先前没有经过反复的尝试，你不知道安装哪一个的时候，应该用常识来判断。我认为以下几点是首选的软件包应该具有的特征。

- 重要性：是 > 否
- 类型：main > contrib > non-free
- 优先级：需要 > 重要 > 标准 > 可选 > 额外
- 任务：在任务下有软件包的列表信息，例如"桌面环境"
- 软件包是被与之有依赖关系的软件包所选择的（例如 `gcc` 依赖 `gcc-16`）
- 流行度：在投票或者安装指数上有着更高的分数

- 更新日志：维护者经常的更新
- BTS (缺陷跟踪系统): 没有 RC 级别的缺陷（没有危险、重大严重的缺陷）
- BTS (缺陷跟踪系统): 有维护者对缺陷报告反馈
- BTS (缺陷跟踪系统): 有着更多的近期修复的 bug 数目
- BTS (缺陷跟踪系统): 遗留的非严重（non-wishlist）缺陷数量较少

Debian 是一个使用分布式开发模式的志愿项目，它的档案库包含了许多不同关注点和不同质量的软件包。你必须做出自己的选择。

2.1.11 怎样和不一致的要求协作

无论你决定使用哪个 Debian 系统套件，你仍然希望运行在那个套件里不存在的程序版本。即使你在其它 Debian 套件里面，或者在其它非 Debian 的资源里面，找到这个程序的二进制软件包，它们的要求可能和你目前的 Debian 系统不一致。

在第 2.7.7 节里描述的 **apt-pinning** 等技术，虽然你能够用它来调整软件包管理系统来安装这类不同步的二进制软件包，但这样的调整方法只有有限的使用场景，因为它们可能破坏那些程序和你的系统。

在单独安装这类不同步的软件包之前，你需要查找所有存在的和你目前 Debian 系统兼容的安全技术替代方案。

- 使用相应的沙盒化的上游二进制软件包来安装这样的程序（参见第 7.7 节）。
  - 许多常见的 GUI（图形用户界面）程序，比如 LibreOffice 和 GNOME 应用，会有 Flatpak、Snap 或 AppImage 软件包存在。
- 建立一个 chroot 或类似的环境来在里面运行这样的程序（参见第 9.11 节）。
  - CLI 命令能够在和它兼容的 chroot 下轻松执行（参见第 9.11.4 节）。
  - 不重启机器，能够轻松的尝试多个完整的桌面环境（参见第 9.11.5 节）。
- 自己构建需要的二进制软件包版本，和你目前 Debian 系统兼容。
  - 这是一个不轻松的任务 (参见第 2.7.13 节)。

2.2 基础软件包管理操作

在 Debian 系统中有许多基于 APT 的软件包管理工具可以在 Debian 系统上进行基于仓库的软件包管理操作。在这里，我们将介绍 3 种基本的软件包管理工具：`apt`,`apt-get` / `apt-cache` 和 `aptitude` 。

对于涉及软件包安装或更新软件包元数据的软件包管理操作，你必须有 root 权限。

2.2.1 `apt` vs. `apt-get` / `apt-cache` vs. `aptitude`

尽管 `aptitude` 是作者主要使用的一个非常好的可互动工具，但你应该知道下列警示：

- 不建议在新版本发布后在 `stable` Debian 系统上使用 `aptitude` 命令来进行跨版本的系统升级。
  - 建议使用”`apt full-upgrade`” 或“`apt-get dist-upgrade`”来进行这个操作。参见 Bug #411280。
- `aptitude` 命令有时候会为了 `testing` 或 `unstable` Debian 系统升级清除大量软件包。
  - 这个情况吓坏了许多的系统管理员。请不要惊慌。
  - 这似乎大多数是由元软件包的依赖或推荐的软件包版本偏差造成的，例如 `gnome-core`。

– 要解决这个问题，可以在 aptitude 命令菜单中选择“取消待执行的动作”，退出 aptitude，并使用“apt full-upgrade”。

apt-get 和 apt-cache 是最基础的基于 APT 的软件包管理工具。

- apt-get 和 apt-cache 只提供命令行用户界面。
- apt-get 是进行跨版本的主系统升级等操作的最合适工具。
- apt-get 提供了一个强大的软件包依赖解析器。
- apt-get 对硬件资源的要求不高。它消耗更少的内存并且运行速度更快。
- apt-cache 提供了一个标准的正则表达式来搜索软件包名称和描述。
- apt-get 和 apt-cache 可以使用 /etc/apt/preferences 来管理软件包的多个版本，但这非常繁琐。

apt 命令是一个用于软件包管理的高级命令行界面。它基本上是 apt-get、apt-cache 和类似命令的一个封装，被设计为针对终端用户交互的界面，它默认启用了某些适合交互式使用的选项。

- apt 工具在用户使用 apt install 安装软件包时提供了一个友好的进度条。
- 在成功安装下载的软件包后，apt 将默认删除缓存的.deb 软件包。

提示
建议用户使用新的 apt(8) 命令用于交互式的使用场景，而在 shell 脚本中使用 apt-get(8) 和 apt-cache(8) 命令。

---

aptitude 命令是最通用的基于 APT 的软件包管理工具。

- aptitude 提供了一个全屏的交互式文本用户界面。
- aptitude 同样也提供了一个命令用户界面。
- aptitude 是用于日常软件包管理（例如检查已安装的软件包和搜索可用的软件包）的最合适工具。
- aptitude 对硬件资源的要求更高。它消耗更多的内存并且运行速度更慢。
- aptitude 提供一个增强的正则表达式来搜索所有的软件包元数据。
- aptitude 可以管理软件包的多个版本，并且不使用 /etc/apt/preferences，这会十分直观。

2.2.2 命令行中的基础软件包管理操作

下面是使用 apt(8), aptitude(8) 和 apt-get(8) / apt-cache(8) 的命令行基本软件包管理操作。

apt / apt-get 和 aptitude 能够混用，没有大问题。

“aptitude why *regex*”可以通过“aptitude -v why *regex*”列出更多的信息。类似的信息可以通过”apt rdepends *package*” 或“apt-cache rdepends *package*”获取。

当 aptitude 命令在命令行模式下启动后遇到了一些问题（例如软件包冲突），你可以在之后的提示中按下“e”键切换到全屏的交互模式。

注意
虽然 aptitude 命令提供了丰富的功能，例如增强的软件包解析器，但它的复杂程度导致了（或可能导致）一些退步，例如 Bug #411123、Bug #514930 及 Bug #570377。如有疑问，请使用 apt,apt-get 和 apt-cache 命令来替代 aptitude 命令。

你可以在“aptitude”后面使用的命令选项。

更多内容参见 aptitude(8) 和位于“/usr/share/doc/aptitude/README”的“aptitude 用户手册”。

apt 语法	**aptitude** 语法	**apt-get** / **apt-cache** 语法	说明
`apt update`	`aptitude update`	`apt-get update`	更新软件包档案库元数据
`apt install foo`	`aptitude install foo`	`apt-get install foo`	安装“foo”软件包的候选版本以及它的依赖
`apt upgrade`	`aptitude safe-upgrade`	`apt-get upgrade`	安装已安装的软件包的候选版本并且不移除任何其它的软件包
`apt full-upgrade`	`aptitude full-upgrade`	`apt-get dist-upgrade`	安装已安装的软件包的候选版本，并且需要的话会移除其它的软件包
`apt remove foo`	`aptitude remove foo`	`apt-get remove foo`	移除“foo”软件包，但留下配置文件
`apt autoremove`	N/A	`apt-get autoremove`	移除不再需要的自动安装的软件包
`apt purge foo`	`aptitude purge foo`	`apt-get purge foo`	清除“foo”软件包的配置文件
`apt clean`	`aptitude clean`	`apt-get clean`	完全清除本地仓库的软件包检索文件
`apt autoclean`	`aptitude autoclean`	`apt-get autoclean`	清除本地仓库中过时软件包的软件包检索文件
`apt show foo`	`aptitude show foo`	`apt-cache show foo`	显示“foo”软件包的详细信息
`apt search` 正则表达式	`aptitude search` *regex*	`apt-cache search` *regex*	搜索匹配 *regex* 的软件包
N/A	`aptitude why` *regex*	N/A	解释匹配 *regex* 的软件包必须被安装的原因
N/A	`aptitude why-not` *regex*	N/A	解释匹配 *regex* 的软件包不必安装的原因
`apt list --manual-installed`	`aptitude search '~i!~M'`	`apt-mark showmanual`	列出手动安装的软件包

Table 2.6: 使用apt(8), aptitude(8) 和apt-get(8) / apt-cache(8) 的命令行基本软件包管理操作

命令选项	说明
`-s`	模拟命令的结果
`-d`	仅下载，不进行安装/更新
`-D`	在自动安装和删除前，显示简要的说明

Table 2.7: aptitude(8) 中重要的命令选项

2.2.3 aptitude 的交互式使用

要使用交互式的软件包管理，你可以像下面那样以交互模式启动 aptitude 。

```
$ sudo aptitude -u
Password:
```

这将更新档案库信息的本地副本，并以菜单的形式全屏显示软件包列表。aptitude 将它的配置放在“~/.aptitude/config”。

提示
如果你想用 root 的配置而非使用者的，可以在上面的例子中使用“sudo -H aptitude …”代替“sudo aptitude …”。

提示
当 aptitude 以交互模式启动时，会自动设置待执行的动作。如果您不喜欢，您可以通过菜单：“动作”→ “取消待执行的动作”来取消它。

2.2.4 aptitude 的按键绑定

在全屏模式下浏览软件包状态和设置动作的按键如下。

快捷键	键绑定功能
F10 或 Ctrl-t	菜单
?	显示按键帮助（更加完整的清单）
F10 → 帮助 → 用户手册	显示用户手册
u	更新软件包档案库信息
+	标记该软件包以便升级或安装
-	标记该软件包以便移除（保留配置文件）
_	标记该软件包以便清除（移除配置文件）
=	将软件包设为保持状态
U	标记所有可升级包（动作如同 **full-upgrade**）
g	开始下载并安装所选择包
q	退出该界面并保存变更
x	退出该界面并清除变更
Enter	查看软件包的信息
C	查看软件包的变更记录
l	变更软件包的显示限制
/	搜寻匹配的第一个软件包
\	重复上一个搜索

Table 2.8: aptitude 的按键绑定

可以通过命令行指定文件名称，也可以通过按“l”或“/”之后在菜单提示下输入下列所述的 aptitude 正则表达式。aptitude 正则表达式可以使用“~n”开头后接软件包名称的字符串来精确匹配软件包名称。

提示
你需要在可视化界面中按下“U”键让所有的已安装软件包升级到可用版本。否则只有选中的软件包和一些与之有依赖关系的软件包才能被升级到可用版本。

2.2.5 aptitude 软件包视图

aptitude(8) 全屏交互模式下，软件包列表里的软件包会像下面的例子那样显示。

```
idA   libsmbclient                              -2220kB 3.0.25a-1  3.0.25a-2
```

该行的从左到右的含义如下。

- “状态”标签（第一个字母）
- “动作”标签（第二个字母）
- “自动”标签（第三个字母）
- 软件包名称
- 该“动作”对磁盘空间的变化
- 软件包当前版本
- 软件包可用版本

> 提示
> 您可以在帮助菜单中找到完整的标签列表，按“?”即可在帮助菜单底部显示。

可用版本的选择是依据当前的本地首选项（参见 apt_preferences(5) 和第 2.7.7 节）。

软件包视图的几种类型都可以在“视图”菜单下找到。

视图	视图描述
`Package View`	参见表 2.10 (默认)
`Audit Recommendations`	列出推荐安装但还没有安装的软件包
`Flat Package List`	不分类地列出软件包 (用于正则表达式)
`Source Package View`	列出由源代码软件包分组的软件包

Table 2.9: aptitude 视图

标准“软件包视图”分类软件包的方法与带有一些额外功能的 `dselect` 有点像。

分类	视图描述
`Upgradable Packages`	按照 `section` → `area` → 软件包的顺序显示列出软件包
`New Packages`	,,
`Installed Packages`	,,
`Not Installed Packages`	,,
`Obsolete and Locally Created Packages`	,,
`Virtual Packages`	列出同样功能的软件包
`Tasks`	列出一个特定任务所需的不同功能的软件包

Table 2.10: 标准软件包视图的分类

> 提示
> 软件集视图可以用来为你的任务选出最佳的软件包。

2.2.6 aptitude 搜索方式选项

aptitude 提供了几个可以使用正则表达式来搜索软件包的选项。

- shell 命令行：
  - “`aptitude search '`*`aptitude_regex`*`'`”列出安装状态、软件包名称和匹配软件包的剪短描述
  - “`aptitude show '`*`package_name`*`'`”列出软件包的详细描述
- 全屏交互模式:
  - ”`l`” 可以限制匹配软件包的视图
  - ”`/`” 搜索匹配的软件包
  - ”`\`” 向后搜索匹配的软件包
  - ”`n`” 查找下一个
  - ”`N`” 查找上一个

提示
字符串 *package_name* 被看作软件包名称的精确字符串匹配，除非它是以”~” 开头的正则表达式。

2.2.7 aptitude 正则表达式

aptitude 正则表达式是类 mutt 的拓展 **ERE**（参见第 1.6.2 节），`aptitude` 具体的特殊匹配规则扩展如下。

- 正则表达式使用的是 **ERE**，就跟 egrep(1)、awk(1) 和 perl(1) 这些典型的类 Unix 文本工具中所使用的“^”、“`.*`”、“$”等是相同的。
- 依赖关系 *type* 是一种特定的软件包相互关系（depends、predepends、recommends、suggests、conflicts、replaces、provides）。
- 默认的 *type* 依赖关系是“depends”。

提示
当 *regex_pattern* 为空字符串时，请立即在命令后面添加”`~T`”。

下面是一些快捷方式。

- ”`~P`*`term`*” == ”`~Dprovides:`*`term`*”
- ”`~C`*`term`*” == ”`~Dconflicts:`*`term`*”
- ”…`~W term`” == ”`(…|term)`”

用户熟悉 `mutt` 的快速选择，因为 mutt 的灵感来源于表达式语法。参见“用户手册”“`/usr/share/doc/aptitude/README`”中的“SEARCHING, LIMITING, AND EXPRESSIONS”。

注意
`lenny` 版本的 aptitude(8) 中，新的长格式语法，例如“`?broken`”，在正则表达式中可以用来等效为它旧的短格式“`~b`”。现在空格字符“ ”被认为是除了波浪字符“~”外的另一个正则表达式终止字符。新的长格式语法参见“用户手册”。

扩展匹配规则描述	正则表达式
匹配软件包名称	~n*regex_name*
匹配描述	~d*regex_description*
匹配软件集名称	~t*regex_task*
匹配 debtag	~G*regex_debtag*
匹配维护者	~m*regex_maintainer*
匹配软件包的 section	~s*regex_section*
匹配软件包版本	~V*regex_version*
匹配档案库	~A{trixie,forky,sid}
匹配来源	~O{debian,…}
匹配优先级	~p{extra,important,optional,required,standard}
匹配必要的软件包	~E
匹配虚拟软件包	~v
匹配新的软件包	~N
匹配待执行的动作	~a{install,upgrade,downgrade,remove,purge,hold,keep}
匹配已安装软件包	~i
匹配带有 **A** 标签的已安装软件包（自动安装的软件包）	~M
匹配不带有 **A** 标签的已安装软件包（管理员选择的软件包）	~i!~M
匹配已安装并且是可升级的软件包	~U
匹配已删除但未清除的软件包	~c
匹配已移除，已清除或可移除的软件包	~g
匹配破坏依赖关系的软件包	~b
匹配破坏 *type* 依赖关系的软件包	~B 类型
匹配 *pattern* 软件包的 *type* 依赖关系	~D[类型:] 模式
匹配 *pattern* 软件包破坏的 *type* 依赖关系	~DB[类型:] 模式
匹配依赖于 *pattern* 软件包的 *type* 依赖的软件包	~R[类型:] 模式
匹配依赖于 *pattern* 软件包破坏的 *type* 依赖的软件包	~RB[类型:] 模式
匹配其它已安装软件包所依赖的软件包	~R~i
匹配没有被其它已安装软件包所依赖的软件包	!~R~i
匹配其它已安装软件包所依赖或建议安装的软件包	~R~i|~Rrecommends:~i
匹配 *pattern* 过滤版本之后的软件包	~S 过滤模式
匹配所有软件包（真）	~T
不匹配软件包（假）	~F

Table 2.11: aptitude 正则表达式

2.2.8 aptitude 的依赖解决

如果通过菜单“F10 → 选项 → 首选项 → 正在处理依赖关系”进行相应的设置，则在 aptitude 中选择一个软件包时，不仅会将其“Depends:”列表中的软件包选上，“Recommends:”列表中的软件包也会被选上。在 aptitude 下，这些自动安装的软件包在不再需要时会自动移除。

aptitude 命令中控制“自动安装”行为的标签也可以通过 apt 软件包中的 apt-mark(8) 命令来设置。

2.2.9 软件包活动日志

你可以在日志文件里查询到软件包活动历史。

文件	内容
/var/log/dpkg.log	dpkg 级的软件包活动日志
/var/log/apt/term.log	通用 APT 活动日志
/var/log/aptitude	aptitude 命令活动日志

Table 2.12: 软件包活动日志文件

事实上，很难从这些日志上快速获得有用的信息。较简便的方法参见第 9.3.9 节。

2.3 aptitude 操作范例

下面是一些 aptitude(8) 的操作范例。

2.3.1 查找感兴趣的软件包

你可以根据 aptitude 这个包管理工具中的软件包描述或者是任务面板下的列表信息，来查找你所需要的软件包。

2.3.2 通过正则表达式匹配软件包名称来列出软件包

下面的命令列出了通过正则表达式匹配软件包名称来列出软件包。

```
$ aptitude search '~n(pam|nss).*ldap'
v   libnss-ldap                                                 -
p   libnss-ldapd                                                - NSS module for using LDAP  ↩
    as a naming service
v   libpam-ldap                                                 -
p   libpam-ldapd                                                - PAM module for using LDAP  ↩
    as an authentication service
```

这种方式查找精确的软件包名称很方便。

2.3.3 使用正则表达式匹配浏览

在” 新扁平软件包列表” 中使用”l” 提示查看，正则表达式”~dipv6” 可以限制性地匹配软件描述，并交互式地展示信息。

2.3.4 完整地清理已删除软件包

您能清除所有已移除软件包的剩余配置文件。

检查以下命令的结果。

```
# aptitude search '~c'
```

如果您确认所列出的软件包应当被完整删除，请运行以下命令。

```
# aptitude purge '~c'
```

您可能想要在交互模式中做类似的操作进行细粒度的控制。

在” 新软件包视图” 使用”l” 提示并输入正则匹配式”~c” ，这将仅匹配软件包，比如，” 移除但不清空配置” 。所有符合匹配的软件包可以在顶层标题上使用”[” 显示。

当您在顶层标题如“未安装的包”中输入“_”，当前标题下的软件包只有匹配正则式才会被清除。您还可以使用“=”来交互式地排除软件包以避免删除它们。

这种技术方便易用且适用于许多其他的命令键。

2.3.5 调整自动/手动安装状态

下面是调整软件包的自动/手动安装状态的方法（在使用非 aptitude 软件包管理器之后）。

1. 用 root 以交互模式运行 aptitude。
2. 用”u” 命令更新可用的软件包列表，”U” 命令标记所有可升级的软件包以执行升级，”f” 命令清除新软件包列表，”g” 命令执行所有可升级的软件包以执行升级。
3. 按下“l”，并输入“~i(~R~i|~Rrecommends:~i)”来限制软件包的显示，按下“M”将“已安装软件包”的状态改为自动安装。
4. 按下“l”，并输入“~prequired|~pimportant|~pstandard|~E”来限制软件包的显示，按下“m”将“已安装软件包”的状态改为手动安装。
5. 按下“l”，并输入“~i!~M”来限制软件包的显示，在“已安装软件包”上按下“[”来陈列无用的软件包，按下“-”将它们移除。
6. 按下“l”，并输入“~i”来限制软件包的显示，之后在“软件集”上按下“m”将那些软件包标记为手动安装。
7. 退出 aptitude。
8. 用 root 用户执行”apt-get -s autoremove|less” 命令，来查看有那些软件包是不再需要的。
9. 在交互模式下重启 aptitude 程序，用“m”命令标记所需要的软件包。
10. 用 root 用户重新执行”apt-get -s autoremove|less” 这个命令来复查移除的包中是不是只含有自己所希望移除的软件包。
11. 用 root 用户执行”apt-get autoremove|less” 命令来自动移除不再需要的软件包。

在你所需要执行的”Tasks” 上，运行”m” 命令是一个可选的操作，目的就是为了防止大量软件包被卸载的情况出现。

2.3.6 全面的系统升级

注意
当你迁移到新的发行版的时候，虽然正如下面所描述的那样，Debian 是可升级的，但是你还是应该考虑纯净的安装新的系统。这给了你机会去移除废弃的软件包同时还可以接触到最新软件包的完美集合体。当然，在做迁移之前，你也应该对你的系统做完整的备份，并把它移到安全的地方去（查看第 10.2 节)。“我”也建议用不同的分区做另外一个启动项，来实现平稳的升级。

你可以通过改变源列表的内容使之指向新的发行版所在地址的方法来进行系统的全面升级，然后运行”`apt update; apt dist-upgrade`” 命令。

在 `trixie`-作为-`stable` 发布循环中, 从 `stable` 升级到 `testing` 或者 `unstable`，你应该用”`forky`” 或者”`sid`” 替换源列表文件里的”`trixie`” 示例，参考第 2.1.5 节。

事实上，由于一些软件包版本变迁的问题，你可能会遇到一些困难，主要是由于软件包的依赖问题。升级之后的差异越大，你越有可能遇到麻烦。在新版本发行后，系统从旧的 `stable` 过渡到新的 `stable`，你可以查看 Release Notes 然后按照里面的步骤去做，来尽可能的减少麻烦。

在它正式发布之前，你决定要从先前的 `stable` 迁移到将要发布的 `testing`，这里没有 Release Notes 可以帮到你。在前一个 `stable` 发布以后， `stable` 发行版跟将要发布的 `testing` 发行版之间的差异可能变得相当大同时也使得升级系统变得更加的复杂。

在全面升级系统的时候，你应该谨慎的操作，同时你也应该从邮件列表中获取最新的资料然后根据你的常识作出正确的判断。

1. 查看先前的“发行说明”。
2. 备份整个系统 (尤其是数据和配置信息)。
3. 当 bootloader 坏了的时候，手边应该有可以引导电脑启动的存储介质。
4. 事先通知系统上的用户。
5. 用 `script`(1) 记录升级的过程。
6. 用“unmarkauto”命令来保留你想要的软件包，例如”`aptitude unmarkauto vim`” 这个命令是用来防止移除 vim 这个软件的。
7. 为了减少软件包之间可能会发生的冲突，应该尽量减少要安装的软件包的数目，例如，移除桌面环境这个软件包。
8. 移除”`/etc/apt/preferences`” 文件（禁用 **apt-pinning**)。
9. 试着一步步的升级：`oldstable` → `stable` → `testing` → `unstable`.
10. 升级源列表文件，使其指向新的档案库然后运行”`aptitude update`” 命令。
11. 可选的安装选项，首先是新的 **core packages**，例如“`aptitude install perl`”。
12. 运行”`apt-get -s dist-upgrade`” 命令来评估升级造成的影响。
13. 最后运行”`apt-get dist-upgrade`” 命令。



小心
在 `stable` 版本升级的时候，跳过主要的 Debian 发行版是不明智的。



小心
在先前的“发行手册”里，GCC、Linux 内核、initrd-tools、Glibc、Perl、APT 工具链等等，有一些关于系统全面升级的重要注意事项。



小心
“发行说明”不一定覆盖所有可能情况。如果你修改了底层配置，下一次升级可能会因为”... 升级后的段错误...”而严重失败。

关于 `unstable` 版本的日常升级，查看第 2.4.3 节。

2.4 高级软件包管理操作

2.4.1 命令行中的高级软件包管理操作

下面列出了一些其它的软件包管理操作，这些操作对于 `aptitude` 过于高级或缺失所需的功能。

注意
对于一个支持多架构的软件包，你可能需要为一些命令指定架构名称。例如，使用“`dpkg -L libglib2.0-0:amd64`”来列出 `amd64` 架构的 `libglib2.0-0` 软件包的内容。



小心
系统管理员应该小心使用低级的软件包工具（例如“`dpkg -i …`”和“`debi …`”），它们不会自动处理所需的软件包依赖。dpkg 的命令行选项“`--force-all`”和类似的选项（参见 dpkg(1)）只适用于高手。没有完全理解它们的效果却使用它们会破坏你的整个系统。

请注意以下几点。

- 所有的系统配置和安装命令都需要以 root 运行。
- 不同于使用正则表达式的 `aptitude` （参见第 1.6.2 节），其它的软件包管理命令使用类似于 shell glob 的通配符（参见第 1.5.6 节）。
- apt-file(1) 由 `apt-file` 软件包提供，并且需要先运行“`apt-file update`”。
- configure-debian(8) 由 `configure-debian` 软件包提供，它运行 dpkg-reconfigure(8) 作为后端。
- dpkg-reconfigure(8) 使用 debconf(1) 作为后端来运行软件包脚本。
- ”`apt-get build-dep`” 、”`apt-get source`” 和”`apt-cache showsrc`” 命令需要源列表中存在”`deb-src`” 条目。
- dget(1) 、debuild(1) 和 debi(1) 需要 `devscripts` 软件包。
- 参见第 2.7.13 节里使用”`apt-get source`” 的打包（重打包）过程。
- `make-kpkg` 命令需要 `kernel-package` 软件包（参见第 9.10 节）。
- 通用打包参见第 12.9 节.

命令	操作
`COLUMNS=120 dpkg -l` *`package_name_pattern`*	列出用于错误报告的已安装软件包的状态
`dpkg -L` *`package_name`*	显示一个已安装软件包的内容
`dpkg -L` *`package_name`* `| egrep '/usr/share/man/man.*/.+'`	列出一个已安装软件包的 man 手册页
`dpkg -S` *`file_name_pattern`*	列出匹配文件名的已安装软件包
`apt-file search` *`file_name_pattern`*	列出档案库中匹配文件名的软件包
`apt-file list` *`package_name_pattern`*	列出档案库中匹配的软件包的内容
`dpkg-reconfigure` *`package_name`*	重新配置软件包
`dpkg-reconfigure -plow` *`package_name`*	通过最详细的方式来重新配置软件包
`configure-debian`	以全屏菜单的形式重新配置软件包
`dpkg --audit`	部分安装软件包的审计系统
`dpkg --configure -a`	配置所有部分安装的软件包
`apt-cache policy` *`binary_package_name`*	显示一个二进制软件包的可用版本、优先级和档案库信息
`apt-cache madison` *`package_name`*	显示一个软件包的可用版本和档案库信息
`apt-cache showsrc` *`binary_package_name`*	显示一个二进制软件包的源代码软件包信息
`apt-get build-dep` *`package_name`*	安装构建软件包所需要的软件包
`aptitude build-dep` *`package_name`*	安装构建软件包所需要的软件包
`apt-get source` *`package_name`*	（从标准档案库）下载源代码
`dget` *`dsc 文件的 URL`*	（从其它档案库）下载源代码软件包
`dpkg-source -x` *`package_name_version-debian.revision.dsc`*	从源代码软件包集合（“`*.orig.tar.gz`”和“`*.debian.tar.gz`”/“`*.diff.gz`”）中构建代码树
`debuild binary`	从本地的源代码树中构建软件包
`make-kpkg kernel_image`	从内核源代码树中构建一个内核软件包
`make-kpkg --initrd kernel_image`	从启用了 initramfs 的内核代码树中构建一个内核软件包
`dpkg -i` *`package_name_version-debian.revision_arch`*`.deb`	安装一个本地的软件包到系统中
`apt install /path/to/`*`package_filename`*`.deb`	安装本地软件包到系统中，同时尝试自动解决依赖
`debi` *`package_name_version-debian.revision_arch`*`.dsc`	安装本地软件包到系统中
`dpkg --get-selections '*' >selection.txt`	保存 `dpkg` 级别的软件包选择状态信息
`dpkg --set-selections <selection.txt`	使用 `dpkg` 设置软件包选择状态
`echo` *`package_name`* `hold | dpkg --set-selections`	使用 `dpkg` 将一个软件包的包选择状态设置为 **hold**（相当于”`aptitude hold` 包名”）

Table 2.13: 高级软件包管理操作

2.4.2 验证安装的软件包文件

已经安装 debsums 软件包的，能使用 debsums(1) 命令通过"/var/lib/dpkg/info/*.md5sums"文件中的 MD5sum 值，验证已安装的文件。参见第 10.3.5 节来获得 MD5sum 是怎样工作的信息。

注意
因为 MD5sum 数据库可能被侵入者篡改，debsums(1) 作为安全工具使用有限。这种工具用于校验管理者造成的本地修改或媒体错误造成的损坏是很不错的。

2.4.3 预防软件包故障

许多用户更想使用 Debian 系统的 **testing** （或 **unstable**）版本，因为它有新的功能和软件包。但这会使得系统更容易遇到严重的软件包问题。

安装软件包 `apt-list bugs` 可以避免您的系统遭遇严重 bugs，在通过 APT 系统升级时，它会自动检查 Debian BTS 里的严重 bug。

安装 `apt-listchanges` 软件包，在使用 APT 系统升级时它会在"`NEWS.Debian`"中提供重要新闻。

2.4.4 搜索软件包元数据

尽管近来浏览 Debian 网站 https://packages.debian.org/ 是搜索软件包元数据更加简单的方法，但我们依旧来看看更传统的方法。

grep-dctrl(1) 、grep-status(1) 和 grep-available(1) 命令被用来搜索具有 Debian 软件包控制文件格式的任何文件。

"`dpkg -S` *`file_name_pattern`*"能够被用来搜索包含该文件的软件包名称，其匹配的名称是由 dpkg 安装的。但它会忽略维护者的脚本创建的文件。

如果你需要对 dpkg 元数据进行更复杂的搜索，你需要在"`/var/lib/dpkg/info/`"目录下运行"`grep -e regex_patter`
`*`"命令。这会使你在软件包脚本和安装查询文本中搜索提及的单词。

如果你想递归查找软件包依赖，你应该使用 apt-rdepends(8) 。

2.5 Debian 软件包内部管理

让我们来学习 Debian 软件包管理的内部工作原理。这应该能够帮助你独立解决一些软件包问题。

2.5.1 档案库元数据

每个发行版的元数据文件都保存在 Debian 镜像站的"`dist/`*`codename`*"下面，例如"`http://deb.debian.org/debian/`"档案库的结构可以通过网络浏览器来浏览。其中有 6 种关键的元数据。

为了减少网络流量，在最近的档案库中，这些元数据存储为压缩了的差分文件。

2.5.2 顶层"**Release**"文件及真实性

提示
顶层"`Release`"文件用于签署 **secure APT** 系统下的归档文件。

文件	位置	内容
`Release`	发行版的顶层	档案库描述和完整性信息
`Release.gpg`	发行版的顶层	"`Release`" 文件的签名文件，使用档案库密钥签名
`Contents-`*`architecture`*	发行版的顶层	列出在相关架构中所有软件包的全部文件
`Release`	每个发行版/区域/架构组合的顶部	归档描述使用 `apt_preferences(5)` 的规则
`Packages`	每个发行版/区域/二进制架构组合的顶部	连接 `debian/control` 获得二进制包
`Sources`	每个发行版/区域/源代码组合的顶部	连接 `debian/control` 获取源代码包

Table 2.14: Debian 档案库元数据的内容

每个 Debian 档案库的网址都有一个这样的“`Release`”文件，例如“`http://deb.debian.org/debian/dists/unstable`”，内容如下。

```
Origin: Debian
Label: Debian
Suite: unstable
Codename: sid
Date: Sat, 14 May 2011 08:20:50 UTC
Valid-Until: Sat, 21 May 2011 08:20:50 UTC
Architectures: alpha amd64 armel hppa hurd-i386 i386 ia64 kfreebsd-amd64 kfreebsd-i386 mips ↩
     mipsel powerpc s390 sparc
Components: main contrib non-free
Description: Debian x.y Unstable - Not Released
MD5Sum:
 bdc8fa4b3f5e4a715dd0d56d176fc789 18876880 Contents-alpha.gz
 9469a03c94b85e010d116aeeab9614c0 19441880 Contents-amd64.gz
 3d68e206d7faa3aded660dc0996054fe 19203165 Contents-armel.gz
...
```

> 注意
> 在第 2.1.5 节里，你能够发现我使用”suite” 和”codename” 的逻辑。“发行版”被用来同时谈及”suite” 和”codename”. 所有由档案库提供的归档”area” 名，会被列在”Components” 下.

顶层文件”`Release`” 的完整性，是由叫 secure apt 的加密架构来验证，在 apt-secure(8) 中进行描述。

- 加密签名文件”`Release.gpg`” 是由顶层授权文件”`Release`” 和加密的 Debian 档案库公钥创建。
- 公开的 Debian 档案库公钥能够通过安装 `debian-archive-keyring` 软件包来安装到本地。
- **secure APT** 系统自动验证下载的顶层文件”`Release`” 的完整性。加密验证过程用到了”`Release.gpg`” 文件和本地安装的 Debian 档案库公钥.
- 所有”`Packages`” 和”`Sources`” 文件的完整性是由在顶层”`Release`” 文件里的 MD5sum 值来验证。所有软件包文件的完整性由”`Packages`” 和”`Sources`” 文件里的 MD5sum 值来验证. 参见 debsums(1) 和第 2.4.2 节.
- 因加密签名验证比计算 MD5sum 值消耗更多的 CPU，使用 MD5sum 值来验证每一个软件包，使用加密签名来验证顶层的”`Release`” 文件，这种方式提供较好安全性的同时，也有比较好的性能 (参见第 10.3 节).

如果源列表条目特别指定了”`signed-by`” 选项，它下载的顶层”`Release`” 文件使用这个指定的公钥来验证。这在当源列表包含有非 Debian 档案库时有用。

当然，你能够使用 `gpg` 手工验证”`Release`” 的完整性，使用”`Release.gpg`” 文件和在 ftp-master.debian.org 上公布的 Debian 档案库公钥。

2.5.3 档案库层的“**Release**”文件

提示
档案库层的“Release”文件将用作 apt_preferences(5) 的规则。

归档层次的”Release” 文件，其全部归档位置在源列表中指定，如以下的”http://deb.debian.org/debian/dists/u
或”http://deb.debian.org/debian/dists/sid/main/binary-amd64/Release”。

```
Archive: unstable
Origin: Debian
Label: Debian
Component: main
Architecture: amd64
```



小心
对于”Archive:” 章节，系列名称 (”stable”，”testing”，”unstable”, ···) 用于 Debian archive ，而代号 (”trusty”, ”xenial”, ”artful”, ···) 用于 Ubuntu archive。

对于部分档案库, 比如说 experimental 和 trixie-backports, 它们包含的软件包不会被自动安装，这是因为有额外的行，例如在”http://deb.debian.org/debian/dists/experimental/main/binary-amd64/Release” 里面有如下额外的一行。

```
Archive: experimental
Origin: Debian
Label: Debian
NotAutomatic: yes
Component: main
Architecture: amd64
```

请注意，普通的档案库没有”NotAutomatic: yes”, 默认的 Pin-Priority 值是 500, 而对于有”NotAutomatic: yes” 的特殊档案库, 默认的 Pin-Priority 值是 1 (参见 apt_preferences(5) 和第 2.7.7 节).

2.5.4 获取用于软件包的元数据

当使用 APT 工具时，如 aptitude, apt-get, synaptic, apt-file, auto-apt，我们需要更新包含 Debian 档案库信息元数据的本地拷贝。这些本地拷贝的文件名称，和在源列表文件里面的 distribution, area, architecture 相应名称一致。(参见第 2.1.5 节).

- ”/var/lib/apt/lists/deb.debian.org_debian_dists_*distribution*_Release”
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_*distribution*_Release.gpg”
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_*distribution_area*_binary-*architecture*_Pac
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_*distribution_area*_source_Sources”
- ”/var/cache/apt/apt-file/deb.debian.org_debian_dists_*distribution*_Contents-*architecture*.g (apt-file)

前 4 种类型的文件是所有相关的 APT 命令共享的，并且可以通过“apt-get update”或“aptitude update”在命令行中进行更新。如果在源列表中有相应的“deb”行，则“软件包”元数据会进行更新。如果在源列表中有相应的“deb-src” 行，则“源代码”元数据会进行更新。

”Packages” 和”Sources” 的元数据文件包含有“Filename:”字段，指向二进制和源代码包文件的位置。目前，这些软件包都统一放在”pool/” 目录树下，这样可以改善跨版本发布的传输。

“软件包”元数据的本地副本可以使用 aptitude 来进行交互式的搜索。专门的搜索命令 grep-dctrl(1) 可以搜索“软件包”和“源代码”元数据的本地副本。

”Contents-*architecture*” 元数据的本地拷贝，能够被”apt-file update” 更新，它的位置和其它 4 个不同。参见 apt-file(1). (auto-apt 的”Contents-*architecture*.gz” 文件的本地拷贝默认也使用不同的位置。)

2.5.5 APT 的软件包状态

除了远程获取元数据，lenny 之后的 APT 工具还会将它在本地产生的安装状态信息保存在“/var/lib/apt/extended_st
中，APT 会使用它们来追踪自动安装的所有软件包。

2.5.6 aptitude 的软件包状态

除了远程获取元数据，aptitude 命令还会将它在本地产生的安装状态信息保存在“/var/lib/aptitude/pkgstates”中，这些信息只能被 aptitude 使用。

2.5.7 获取的软件包的本地副本

所有通过 APT 机制远程获取的软件包都被保存在“/var/cache/apt/archives”中，直到它们被清除。

aptitude 的这个缓存文件清理策略，能够在”Options” → ”Preferences” 下设置，也可以通过它的菜单，”Actions” 下的”Clean package cache” 或”Clean obsolete files” 来执行强制清理。

2.5.8 Debian 软件包文件名称

Debian 软件包文件有特定的名称结构。

软件包类型	名称结构
二进制软件包（亦称 deb）	*package-name_upstream-version-debian.revision_architec*
用于 debian-installer 的二进制软件包（亦称 udeb）	*package-name_upstream-version-debian.revision_architec*
源代码软件包（上游源代码）	*package-name_upstream-version-debian.revision*.orig.tar
1.0 源代码软件包 (Debian 改变)	*package-name_upstream-version-debian.revision*.diff.gz
3.0 (quilt 补丁管理工具) 源代码软件包 (Debian 改变)	*package-name_upstream-version-debian.revision*.debian.t
源代码软件包（说明）	*package-name_upstream-version-debian.revision*.dsc

Table 2.15: Debian 软件包的名称结构

提示
这里仅叙述了基本的源码包格式。更多内容请参考 dpkg-source(1)。

注意
你可以用 dpkg(1) 提供的命令检查软件包版本，例如，”dpkg --compare-versions 7.0 gt 7.~pre1 ; echo $?” .

名称组件	可用的字符（正则表达式）	存在状态
package-name	[a-z0-9][-a-z0-9.+]+	必需
epoch:	[0-9]+:	可选
upstream-version	[-a-zA-Z0-9.+:]+	必需
debian.revision	[a-zA-Z0-9.+~]+	可选

Table 2.16: Debian 软件包名称中每一个组件可以使用的字符

注意
debian-installer (d-i) 使用 udeb 作为它的二进制软件包的文件扩展名，而非普通的 deb。一个 udeb 软件包是从 deb 软件包中剥离了一些不必要的内容（例如文档），从而节省空间同时也放宽软件包政策的要求。deb 和 udeb 软件包会共享相同的软件包结构。“u”表示微小。

2.5.9 dpkg 命令

dpkg(1) 是 Debian 软件包管理中最底层的工具。它非常强大，必须小心使用。
当安装名为“*package_name*”的软件包时，dpkg 会按照下列的顺序处理它。

1. 解包 deb 文件（等同于“ar -x”）
2. 使用 debconf(1) 执行“*package_name*.preinst”
3. 将软件包安装到系统中（等同于“tar -x”）
4. 使用 debconf(1) 执行“*package_name*.postinst”

debconf 系统提供带有 I18N 和 L10N （第 8 章）支持的标准化用户交互。

文件	内容说明
/var/lib/dpkg/info/*package_name*.conffiles	列出配置文件。（使用者可修改的）
/var/lib/dpkg/info/*package_name*.list	列出软件包安装的所有文件和目录
/var/lib/dpkg/info/*package_name*.md5sums	列出软件包安装的文件的 MD5 哈希值
/var/lib/dpkg/info/*package_name*.preinst	软件包安装之前运行的软件包脚本
/var/lib/dpkg/info/*package_name*.postinst	软件包安装之后运行的软件包脚本
/var/lib/dpkg/info/*package_name*.prerm	软件包移除之前运行的软件包脚本
/var/lib/dpkg/info/*package_name*.postrm	软件包移除之后运行的软件包脚本
/var/lib/dpkg/info/*package_name*.config	用于 debconf 系统的软件包脚本
/var/lib/dpkg/alternatives/*package_name*	update-alternatives 命令使用的替代信息
/var/lib/dpkg/available	所有软件包的可用性信息
/var/lib/dpkg/diversions	dpkg(1) 使用的文件移动信息，由 dpkg-divert(1) 设置
/var/lib/dpkg/statoverride	dpkg(1) 使用的文件状态改变信息，由 dpkg-statoverride(1) 设置
/var/lib/dpkg/status	所有软件包的状态信息
/var/lib/dpkg/status-old	“/var/lib/dpkg/status”文件的第一代备份
/var/backups/dpkg.status*	第二代备份，以及“/var/lib/dpkg/status”文件更旧的备份

Table 2.17: dpkg 创建的重要文件

“status”文件也被例如 dpkg(1)、“dselect update”和“apt-get -u dselect-upgrade”等工具使用。
专门的搜索命令 grep-dctrl(1) 可以搜索“status”和“available”元数据的本地副本。

提示
在 debian 安装器环境下, udpkg 命令用于打开 udeb 软件包，udpkg 命令是 dpkg 命令的一个精简版本.

2.5.10 update-alternatives 命令

Debian 系统使用 update-alternatives(1) 让用户可以不受干扰地安装多种重叠的程序。例如，如果同时安装了 vim 和 nvi 软件包，你可以使 vi 命令选择运行 vim。

```
$ ls -l $(type -p vi)
lrwxrwxrwx 1 root root 20 2007-03-24 19:05 /usr/bin/vi -> /etc/alternatives/vi
$ sudo update-alternatives --display vi
...
$ sudo update-alternatives --config vi
  Selection    Command
 -----------------------------------------------
      1        /usr/bin/vim
*+    2        /usr/bin/nvi

Enter to keep the default[*], or type selection number: 1
```

Debian 选择系统在"/etc/alternatives/"目录里通过符号链接来维持它的选择。选择进程使用"/var/lib/dpkg/alter
目录里面的相应文件。

2.5.11 dpkg-statoverride 命令

当安装一个软件包时，由 dpkg-statoverride(1) 命令提供的状态修改，是告诉 dpkg(1) 对文件使用不同的属主或权限的一个方法。如果使用了"--update" 选项，并且文件存在，则该文件会被立即设置为新的属主和模式。



小心
系统管理员使用 chmod 或 chown 命令直接修改某个软件包文件的属主或权限，在下次软件包升级时，将会被重置。

注意
本人在此使用了文件一词，但事实上也可用于 dpkg 所处理的任何文件系统对象，包括目录，设备等。

2.5.12 dpkg-divert 命令

dpkg-divert(1) 命令提供的文件转移功能，是强制 dpkg(1) 不将文件安装到其默认位置，而是安装到被转移的位置。dpkg-divert 专用于软件包维护脚本。不建议系统管理员随意使用它。

2.6 从损坏的系统中恢复

当运行测试版或不稳定版系统，系统管理员会遇到从错误的软件包管理进行恢复的情形。



小心
下面的一些方法具有很高的风险。在此先对你进行警告！

2.6.1 不兼容旧的用户配置

如果一个桌面 GUI 程序在重要的上游版本升级后变得不稳定，你应该怀疑这是旧的本地配置文件（由它创建的）所导致的。如果它在新建的用户账号下运行稳定，那么这个假设就得到了证实。（这是一个打包的 bug 并且打包者通常会避免它。）

为了恢复稳定，你应该移除相应的本地配置文件并重新启动 GUI 程序。你可能需要阅读旧的配置文件内容以便之后恢复配置信息。（别将它们删得太快了。）

2.6.2 软件包数据缓存错误

软件包数据缓存错误，能够造成奇怪的错误，比如 APT 的"GPG error: ... invalid: BADSIG ..."。

你应该通过"`sudo rm -rf /var/lib/apt/*` "删除所有缓存的数据，然后重新尝试。(如果使用了 `apt-cacher-ng`，你还应运行"`sudo rm -rf /var/cache/apt-cacher-ng/*` "。)

2.6.3 使用 **dpkg** 命令进行救援

因 `dpkg` 是非常底层的软件包工具，它能够在没有网络连接时运行。

让我们假设 `foo` 软件包损坏了，需要被修复。

你可以在软件包缓存目录："`/var/cache/apt/archives/`"中找到旧的 `foo` 软件包的无 bug 版本。（如果找不到，你可以从档案库 https://snapshot.debian.org/ 中下载它，或从具有软件包缓存功能的机器中拷贝它。）

如果你能够启动系统，你可以通过下列命令来安装它。

```
# dpkg -i /path/to/foo_old_version_arch.deb
```

如果由于依赖问题，无法用这种方式安装软件包，并且你真的必须真么做，你可以使用 `dpkg` 的"`--ignore-depends`"、"`--f`和其它选项来无视依赖。如果你这么做了，之后你必须认真努力地修复依赖关系。更多细节参见 dpkg(8)。

注意
如果你的系统严重损坏了，你应该将系统完整备份到一个安全的地方（参见第 10.2 节）并进行一次全新的安装。这是耗时较少且效果较好的办法。

提示
如果你系统损坏较小，你也可以使用更高层的 APT 系统来降级整个系统，就像第 2.7.11 节中做的那样。

2.6.4 缺少依赖导致的安装失败

如果你通过"`sudo dpkg -i ...`" 强制安装一个软件包到系统，而不安装它所依赖的所有软件包，这个软件包将作为"部分安装"而失败。

你应当安装所有依赖的软件包，重复使用"`sudo dpkg -i ...`" 或通过使用：

```
# apt --fix-broken install
```

然后，使用下列命令来配置所有部分安装的软件包。

```
# dpkg --configure -a
```

2.6.5 具有相同文件的不同软件包

文档级的软件包管理系统，比如说 aptitude(8) 或 apt-get(8), 使用软件包依赖，当出现相同文件时，不会尝试去安装软件包。(参见第 2.1.7 节).

软件包维护者的错误，或者系统管理员配置了不一致的档案库混合源，(参见第 2.7.6 节)，都会出现不正确的软件包依赖情况。如果在出现相同文件的情况下，你通过 aptitude(8) 或 apt-get(8) 安装软件包，dpkg(1) 在对软件包解包时，确定会给调用程序返回错误，并不会覆盖已经存在的文件。



小心
使用第三方软件包会导致重大的系统风险，因为其通过使用 root 权限运行维护者脚本能够对你的系统做任何事。dpkg(1) 命令只防止解包时的覆盖行为。

可以先通过删除旧的令人讨厌的软件包，*old-package*，来解决这类错误的安装问题。

```
$ sudo dpkg -P old-package
```

2.6.6 修复损坏的软件包脚本

当软件包脚本中的一个命令由于某些原因返回错误，脚本也将由于错误而退出，软件包管理系统忽略它们的行为，并导致部分安装的软件包。当一个软件包在它的删除脚本中有错误时，该软件包将会成为不可能删除的软件包，处理这些问题，都会变得相当棘手。

对于“*package_name*”的软件包脚本问题，你应该查看下列的软件包脚本。

- ”/var/lib/dpkg/info/*package_name*.preinst”
- ”/var/lib/dpkg/info/*package_name*.postinst”
- ”/var/lib/dpkg/info/*package_name*.prerm”
- ”/var/lib/dpkg/info/*package_name*.postrm”

使用下列的方法，以 root 编辑损坏的软件包脚本。

- 在行首添加“#”可以禁用出错的行
- 在出错行的行尾添加“|| true”可以强制返回成功

然后，使用下列命令来配置所有部分安装的软件包。

```
# dpkg --configure -a
```

2.6.7 恢复软件包选择数据

如果“/var/lib/dpkg/status”因为某种原因出现错误，Debian 系统会丢失软件包选择数据并受到严重影响。寻找位于“/var/lib/dpkg/status-old”或“/var/backups/dpkg.status.*”中旧的“/var/lib/dpkg/status”文件。

给“/var/backups/”分配一个单独的分区是一个好习惯，因为这个目录包含了许多重要的系统数据。

对于严重的损坏，我建议备份系统后重新安装。即使失去“/var/”中的所有数据，你依旧可以从“/usr/share/doc/”目录恢复一些信息来引导你进行新的安装。

重新安装最小（桌面）系统。

```
# mkdir -p /path/to/old/system
```

将旧系统挂载到“/path/to/old/system/”。

```
# cd /path/to/old/system/usr/share/doc
# ls -1 >~/ls1.txt
# cd /usr/share/doc
# ls -1 >>~/ls1.txt
# cd
# sort ls1.txt | uniq | less
```

然后你就可以根据软件包名称来进行安装了。（可能会有一些非软件包名称，例如“texmf”。）

2.7 软件包管理技巧

出于简化，在 trixie 发布后，在这个章节的源列表例子，使用单行式样在”/etc/apt/sources.list” 里表示。

2.7.1 上传软件包的是谁？

尽管“/var/lib/dpkg/available”和“/usr/share/doc/package_name/changelog”中列出的维护者姓名提供了关于“软件包运作的幕后者是谁”这一问题的一些信息，但软件包的实际上传者依旧不明。devscripts 软件包中的 who-uploads(1) 可以识别 Debian 源软件包的实际上传者。

2.7.2 限制 APT 的下载带宽

如果你想限制 APT 的下载带宽到 800Kib/sec（=100KiB/sec），你应该像下面那样设置 APT 的配置参数。

```
APT::Acquire::http::Dl-Limit "800";
```

2.7.3 自动下载和升级软件包

apt 软件包有自己的 cron 脚本“/etc/cron.daily/apt”，它支持自动下载软件包。可以安装 unattended-upgrades 软件包来增强这个脚本，使它能够自动升级软件包。可以通过“/etc/apt/apt.conf.d/02backup”和“/etc/apt/apt.co 中的参数来进行自定义，相关说明位于“/usr/share/doc/unattended-upgrades/README”中。

unattended-upgrades 软件包主要用于 stable 系统的安全更新。如果自动升级损坏 stable 系统的风险小于被入侵者利用已被安全更新修复的安全漏洞，你应该考虑使用自动更新，配置参数如下。

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::Unattended-Upgrade "1";
```

如果你运行的是 testing 或 unstable 系统，你应该不会想要使用自动更新，因为它肯定会在某天损坏系统。即使位于这样的 testing 或 unstable 情况下，你可能依旧想提前下载软件包以节省交互式升级的时间，其配置参数如下。

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::Unattended-Upgrade "0";
```

2.7.4 更新和向后移植

stable-updates ("trixie-updates"，在 trixie-作为-stable 发布循环) 和 backports.debian.org 档案库提供了 stable 版软件包更新。

为了去使用这些档案库，你需要在"/etc/apt/sources.list" 文件里写入如下所示的档案库列表。

```
deb http://deb.debian.org/debian/ trixie main non-free-firmware contrib non-free
deb http://security.debian.org/debian-security trixie-security main non-free-firmware ↩
    contrib non-free
deb http://deb.debian.org/debian/ trixie-updates main non-free-firmware contrib non-free
deb http://deb.debian.org/debian/ trixie-backports main non-free-firmware contrib non-free
```

并不需要在"/etc/apt/preferences" 文件中显式设置 Pin-Priority 值. 当新的包可用时，默认配置提供了更合理的更新 (请见第 2.5.3 节).

- 所有已安装的旧软件包都可以通过 trixie-updates 档案库升级到新软件包。
- 只有从 trixie-backports 档案库中手动安装的旧软件包才会通过 trixie-backports 档案库升级到新软件包。

当你想要从 trixie-backports 档案库中手动的安装一个名叫"*package-name*" 的软件及其依赖包的时候，你应该在目标档案库之前加一个"-t" 参数。

```
$ sudo apt-get install -t trixie-backports package-name
```



警告
不要从 backports.debian.org 档案库安装太多软件包。它能够造成软件包依赖复杂。替代解决方案参见第 2.1.11 节。

2.7.5 外部软件包档案库



警告
你应当小心，外部软件包获取你系统的 root 权限。你应当只使用可信赖的外部软件包档案库。替代方案参见第 2.1.11 节。

你能够使用安全 APT 来使用 Debian 兼容的外部软件包档案库，将它加入到源列表，并把它的档案库密钥放入"/etc/apt/trusted.gpg.d/" 目录。参见 sources.list(5) 和 apt-secure(8) 。

2.7.6 不使用 **apt-pinning** 的混合源档案库软件包



小心
从混合源档案库中安装软件包是不被 Debian 官方发行版所支持的，除了官方支持的档案库的特殊组合以外，例如 stable 的 security updates 和 stable-updates。

这里有一个列子，在原有只跟踪 testing 的场景，操作包含在 unstable 里发现的新的上游软件包版本。

1. 临时更改"`/etc/apt/sources.list`"文件，使之指向单一的"`unstable`"发行版路径。
2. 运行"`aptitude update`"命令。
3. 运行"`aptitude install` *`package-name`*"命令。
4. 恢复到原始"`/etc/apt/sources.list`"文件，使之指向 `testing` 路径。
5. 运行"`aptitude update`"命令。

使用这个手工方法，你不需要创建"`/etc/apt/preferences`"文件，也不需要担心 **apt-pinning**。但这个方法仍然是非常麻烦的。



小心
当使用混合档案源的时候，因为 Debian 不会确保软件之间的兼容性，所以你必须自己去解决兼容性问题。如果软件之间存在不兼容性，系统可能会损坏。你必须能够判断这些操作所需的技术要求。使用任意混合的档案源是完全可选的操作，我并不鼓励你去使用它。

从不同的档案库中安装软件包的一般规则如下。

- 非二进制软件包 ("`Architecture: all`") 的安装是更保险的。
  - 文档软件包：没有特别的要求
  - 解释程序的软件包：兼容的解释器必须是可用的
- 二进制软件包 (non "`Architecture: all`") 通常会面临很多障碍，它的安装不保险的。
  - 库文件版本的兼容性（包括"`libc`"）
  - 与之相关的有用的程序版本的兼容性
  - 内核 ABI 的兼容性
  - C++ ABI 的兼容性
  - …

注意
为了使软件包的安装变得更保险，一些商业的非自由的二进制程序包可能会提供完整的静态链接库。你还是应该检查 ABI 的兼容性问题等等。

注意
除非为了短期避免破坏软件包，从非 Debian 档案库安装二进制软件包通常是一个坏的主意。你需要查找所有存在的和你目前 Debian 系统兼容的安全技术替代方案。（参见第 2.1.11 节）。

2.7.7 使用 apt-pinning 调整获选版本



警告
新手用 **apt-pinning** 命令肯定会造成比较大的问题。你必须避免使用这个技术，除非确实需要它。参见第 2.1.11 节用于替代方案。

没有"`/etc/apt/preferences`"文件，APT 系统使用版本字符串来选择最新的可用版本作为候选版本。这是通常的状态，也是 APT 系统最推荐的使用方法。所有官方支持的档案库集合，并不要求"`/etc/apt/preferences`"文件，因此，一些不应当被作为自动更新源的软件包，被标记为 **NotAutomatic**，并被适当处理。

提示
版本字符串的比较规则可以被验证，例子如下，"`dpkg --compare-versions ver1.1 gt ver1.1~1; echo $?`" (参见 dpkg(1))。

如果经常从混合源档案库中安装软件包 (参见第 2.7.6 节), 你可以通过创建"`/etc/apt/preferences`" 文件并且在其中写入关于调整候选版本的软件包选取规则的合适条目 (如 apt_preferences(5) 中所示) 来自动化这些复杂的操作。这被称为 **apt-pinning**。

当使用 **apt-pinning** 命令时，因为 Debian 不会确保软件之间的兼容性，所以你必须自己确认其兼容性。**apt-pinning** 是完全可选的操作，我并不建议去使用它。

档案库层级的 Release 文件 (参见第 2.5.3 节) 使用 apt_preferences(5) 的规则. 对于 Debian 通用档案库和 Debian 安全档案库，**apt-pinning** 只在"suite" 名下工作。(这点和 Ubuntu 档案库不同.) 例如，你在"`/etc/apt/preferences`" 文件里面，可以使用"`Pin: release a=unstable`" ，但不能使用"`Pin: release a=sid`".

当使用非 Debian 的档案库作为 **apt-pinning** 的一部分时，你应该检查它们的用途和可信度。例如，Ubuntu 和 Debian 是不能混在一起的。

注意
即使不创建"`/etc/apt/preferences`" 文件，在不用 **apt-pinning** 命令的情况下，你也可以进行相当复杂的系统工作 (参见第 2.6.3 节和第 2.7.6 节)。

如下是关于 **apt-pinning** 技术的简化说明。

可用的软件包源在"`/etc/apt/sources.list`" 文件里面定义，APT 系统从可用的软件包源里面选择 Pin-Priority 值最大的，作为升级软件包的候选版本。如果一个软件包的 Pin-Priority 大于 1000，这个版本限制为只能升级，关闭了软件包降级功能（参见第 2.7.11 节）。

每个软件包的 Pin-Priority 值是在"`/etc/apt/preferences`" 文件中的"Pin-Priority" 条目中定义或者是使用它的默认值。

Pin-Priority	**apt-pinning** 对软件包的影响
1001	安装该软件包，即使是一个降级软件包的指令
990	用作目标发行版档案库的默认值
500	用作常规档案库的默认值
100	用于 **NotAutomatic** 和 **ButAutomaticUpgrades** 档案库的默认值
100	用于已安装软件包
1	用于 **NotAutomatic** 档案库的默认值
-1	即使被推荐，也绝不安装这个软件包

Table 2.18: 用于 **apt-pinning** 技术的值得注意的 Pin-Priority 值列表。

目标版本档案仓库，能够由命令行选项设置，例如: "`apt-get install -t testing some-package`"

NotAutomatic 和 **ButAutomaticUpgrades** 的档案是由档案库服务器上档案层级的 Release 文件来设置，(参见第 2.5.3 节)，同时包含"`NotAutomatic: yes`" 和"`ButAutomaticUpgrades: yes`". 而 **NotAutomatic** 档案也是由档案库服务器上的档案层级的 Release 文件来设置，但只包含"`NotAutomatic: yes`".

来自多个档案源的软件包的 **apt-pinning** 情况可以通过"`apt-cache policy` *package*" 命令显示。

- "`Package pin:`" 开头的行，列出了软件包版本的 **pin** ，如果 *package* 相关的 pin 已经定义, 例如, "`Package pin: 0.190`".
- 没有"`Package pin:`" 的行存在，如果没有 *package* 相关的定义。
- 与 *package* 相关的 Pin-Priority 值列在所有版本字符串的右边，比如，"`0.181 700`".
- "`0`" 是列在所有版本字符串的右边，如果没有 *package* 相关的定义。例如, "`0.181 0`".
- 档案库 (在"`/etc/apt/preferences`" 文件作为"`Package: *`" 定义) 的 Pin-Priority 值，列在所有档案库路径的左边，例如，"`100 http://deb.debian.org/debian/ trixie-backports/main Packages`".

2.7.8 阻止推荐的软件包的安装



警告
新手用 **apt-pinning** 命令肯定会造成比较大的问题。你必须避免使用这个技术，除非确实需要它。参见第 2.1.11 节用于替代方案。

如果不想要引入推荐的特定软件包，你必须创建"`/etc/apt/preferences`" 文件并且像如下所示的那样在文件的顶部明确列出这些软件包。

```
Package: package-1
Pin: version *
Pin-Priority: -1

Package: package-2
Pin: version *
Pin-Priority: -1
```

2.7.9 使用带有 **unstable** 软件包的 **testing** 版本



警告
新手用 **apt-pinning** 命令肯定会造成比较大的问题。你必须避免使用这个技术，除非确实需要它。参见第 2.1.11 节用于替代方案。

如下是一个关于 **apt-pinning** 技术的例子，当使用 `testing` 的时候，实现 `unstable` 中的特定的较新的上游版本软件包的日常升级。你应该按如下所示的在"`/etc/apt/sources.list`" 文件中列出所有需要的档案库。

```
deb http://deb.debian.org/debian/ testing main contrib non-free
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://security.debian.org/debian-security testing-security main contrib
```

按如下所示的设置"`/etc/apt/preferences`" 文件。

```
Package: *
Pin: release a=unstable
Pin-Priority: 100
```

当想要在此配置下从 `unstable` 档案库中安装"*`package-name`*" 软件及它的依赖包时，你执行带有"`-t`" 选项 (`unstable` 的 Pin-Priority 值变为 990) 的转换目标发行版的命令。

```
$ sudo apt-get install -t unstable package-name
```

在此配置下，执行"`apt-get update`"和"`apt-get dist-upgrade`"(或者"`aptitude safe-upgrade`"和"`aptitude full-upgrade`") 命令, 会从 `testing` 档案库升级那些从 `testing` 档案库安装的软件包并且从 `unstable` 档案库升级那些从 `unstable` 档案库中安装的软件包。



小心
小心不要去移除"`/etc/apt/sources.list`" 文件中的"`testing`" 档案库。如果文件中没有"`testing`" ，APT 系统会使用更加新的 `unstable` 档案库升级软件包。

提示
我通常会在上述操作后，马上注释掉”/etc/apt/sources.list” 文件中的”unstable” 档案库记录。这避免了因为处理”/etc/apt/sources.list” 文件中的众多记录而造成的升级缓慢虽然同时也阻止了那些从 unstable 档案库中安装的软件包通过 unstable 升级。

提示
如果”/etc/apt/preferences” 文件中”Pin-Priority: 1” 替代了”Pin-Priority:100”，即使”/etc/apt/sources.list” 文件中的”testing” 记录被删除了，Pin-Priority 值为 100 的已安装软件包也不会通过 unstable 档案库升级。

如果你希望自动跟踪 unstable 里某些特殊的软件包，而在安装时不再使用初始化选项”-t unstable” ，你必须创建”/etc/apt/preferences” 文件，并在该文件顶部按下面的方式清晰的列出所有那些软件包。

```
Package: package-1
Pin: release a=unstable
Pin-Priority: 700

Package: package-2
Pin: release a=unstable
Pin-Priority: 700
```

如下是为每个特定的软件包设置 Pin-Priority 值。例如，为了使用最新的 unstable 的英文版”Debian Reference”，你应该在”/etc/apt/preferences” 文件中写入以下条目。

```
Package: debian-reference-en
Pin: release a=unstable
Pin-Priority: 700

Package: debian-reference-common
Pin: release a=unstable
Pin-Priority: 700
```

提示
即使你使用的是 stable 档案库，**apt-pinning** 技术仍然是有效的。根据我以前的经验，从 unstable 档案库安装的文档包一直是安全的。

2.7.10 使用带有 `experimental` 软件包的 `unstable` 版本



警告
新手用 **apt-pinning** 命令肯定会造成比较大的问题。你必须避免使用这个技术，除非确实需要它。参见第 2.1.11 节用于替代方案。

这是使用 **apt-pinning** 的另一个示例，该示例主要使用 unstable 源，但包含了 experimental 源, 该源可用于安装上游更新的软件包. 需要包含在”/etc/apt/sources.list” 文件中的列表如下。

```
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://deb.debian.org/debian/ experimental main contrib non-free
deb http://security.debian.org/ testing-security main contrib
```

由于 experimental 源是非自动（**NotAutomatic**）的源 (参见第 2.5.3 节)，其默认的 Pin-Priority 值被设置为 1 (<<100) . 并不需要在”/etc/apt/preferences” 文件中设置 Pin-Priority 值，只需要指定 experimental 源，除非你需要在下次更新时自动升级时更新特定软件包.

2.7.11 紧急降级



警告
新手用 **apt-pinning** 命令肯定会造成比较大的问题。你必须避免使用这个技术，除非确实需要它。参见第 2.1.11 节用于替代方案。



小心
降级在 Debian 设计上就不被官方支持。仅仅是在紧急恢复过程中需要做的一部分工作。尽管憎恨这种情形，但降级在很多场景下工作得也不错。对于重要系统，你应当在恢复操作后备份所有重要数据，并从零开始重新安装一个新的系统。

你可以通过控制候选版本从新的档案库降级到旧的档案库（参见第 2.7.7 节），从而使损坏的系统恢复。下面是一种懒惰的方法，可以避免许多冗长的“`dpkg -i broken-package_old-version.deb`”命令（参见第 2.6.3 节）。

搜索“`/etc/apt/sources.list`”文件中像下面那样使用 `unstable` 的行。

```
deb http://deb.debian.org/debian/ sid main contrib non-free
```

使用下面的行替换它，从而改为使用 `testing`。

```
deb http://deb.debian.org/debian/ forky main contrib non-free
```

按如下所示的设置”`/etc/apt/preferences`” 文件。

```
Package: *
Pin: release a=testing
Pin-Priority: 1010
```

运行“`apt-get update; apt-get dist-upgrade`”使整个系统的软件包强制降级。

在紧急降级后，移除“`/etc/apt/preferences`”这个特殊的文件。

提示
这是一个好方法，移除（不是清除！）尽可能多地软件包，来减少依赖问题。你可能需要手动移除和安装一些软件包来使系统降级。需要特别注意 Linux 内核、引导程序、udev、PAM、APT 和网络相关的软件包以及它们的配置文件。

2.7.12 equivs 软件包

如果你从源代码编译了一个程序来代替 Debian 软件包，最好将它做成一个真正的本地 Debian 软件包（`*.deb`）并使用私人档案库。

如果你选择从源代码编译一个程序并将它安装到“`/usr/local`”，你可能需要使用 `equivs` 作为最后步骤来满足缺失的软件包依赖。

```
Package: equivs
Priority: optional
Section: admin
Description: Circumventing Debian package dependencies
 This package provides a tool to create trivial Debian packages.
 Typically these packages contain only dependency information, but they
 can also include normal installed files like other packages do.
 .
```

```
One use for this is to create a metapackage: a package whose sole
purpose is to declare dependencies and conflicts on other packages so
that these will be automatically installed, upgraded, or removed.
.
Another use is to circumvent dependency checking: by letting dpkg
think a particular package name and version is installed when it
isn't, you can work around bugs in other packages' dependencies.
(Please do still file such bugs, though.)
```

2.7.13 移植一个软件包到 **stable** 系统



小心
由于系统差异，不能够保证这里描述的过程能够工作，而不需要额外的手工处理。

对于部分升级的 `stable` 系统，使用源软件包在运行环境中重新构建一个软件包是不错的选择。这可以避免因为依赖关系导致大量软件包升级。

在 `stable` 系统的“`/etc/apt/sources.list`”文件中添加下列条目。

```
deb-src http://deb.debian.org/debian unstable  main contrib non-free
```

如下安装编译所需的软件包并下载源软件包。

```
# apt-get update
# apt-get dist-upgrade
# apt-get install fakeroot devscripts build-essential
# apt-get build-dep foo
$ apt-get source foo
$ cd foo*
```

如果需要向后移植，可以从 backport 的软件包中更新一些工具链软件包，例如 `dpkg` 和 `debhelper`。

执行下列命令。

```
$ dch -i
```

更新软件包版本，例如在“`debian/changelog`”中附加一个“`+bp1`”

像下面那样构建软件包并将它们安装到系统中。

```
$ debuild
$ cd ..
# debi foo*.changes
```

2.7.14 用于 **APT** 的代理服务器

因为镜像整个 Debian 档案库的子区会浪费硬盘和网络带宽，当你管理许多 LAN 上的系统时，为 APT 部署一个本地代理服务器是个好主意。APT 可以通过配置来使用通用 web（http）代理服务器，例如 `squid`（参见第 6.5 节），细节参见 apt.conf(5) 和“`/usr/share/doc/apt/examples/configure-index.gz`”。环境变量“`$http_proxy` 会覆盖“`/etc/apt/apt.conf`”文件中设置的代理服务器。

这里有一些 Debian 档案库的专用代理工具。你应该在使用它们之前检查 BTS。



小心
当 Debian 重构它的档案库结构时，这些专用的代理工具往往需要软件包维护者重写代码，并可能在一段时间内无法使用。另一方面，通用 web （http）代理服务器更强健并且更容易应对这种改变。

软件包	流行度	大小	说明
apt-cacher	V:0.27, I:0.32	265	为 Debian 软件包和源代码文件进行缓存代理（Perl 程序）
apt-cacher-ng	V:3.8, I:3.9	2063	分发软件包的缓存代理（C++ 编译的程序）

Table 2.19: Debian 档案库的专用代理工具

2.7.15 更多关于软件包管理的文档

你可以从下面的文档中了解软件包管理的更多信息。

- 软件包管理的主要文档：
  - aptitude(8), dpkg(1), tasksel(8), apt(8), apt-get(8), apt-config(8), apt-secure(8), sources.list(5), apt.conf(5), and apt_preferences(5);
  - 来自 apt-doc 软件包的“/usr/share/doc/apt-doc/guide.html/index.html”和“/usr/share/doc/apt-doc
  - 来自 aptitude-doc-en 软件包的“/usr/share/doc/aptitude/html/en/index.html”。
- Debian 档案库的官方详细文档：
  - ”Debian Policy Manual Chapter 2 - The Debian Archive”，
  - ”Debian Developer’s Reference, Chapter 4 - Resources for Debian Developers 4.6 The Debian archive”，
  - ”The Debian GNU/Linux FAQ, Chapter 6 - The Debian FTP archives”。
- 为 Debian 用户构建一个 Debian 软件包的教程：
  - “Debian 维护者指南”。

Chapter 3

系统初始化

作为系统管理员，粗略地了解 Debian 系统的启动和配置方式是明智的。尽管准确的细节在安装的软件包及对应的文档中，但这些知识对我们大多数人来说都是必须掌握的。

下面是 Debian 系统初始化的要点概述。由于 Debian 系统在不断发展，您应该参考最新的文档。

- Debian Linux 内核手册是关于 Debian 内核的主要信息来源。
- bootup(7) 介绍了基于 `systemd` 的系统启动流程。（近期的 Debian）
- boot(7) 介绍了基于 UNIX System V Release 4 的系统启动流程。（旧版的 Debian）

3.1 启动过程概述

计算机系统从上电事件到能为用户提供完整的操作系统（OS）功能为止，需要经历几个阶段的启动过程。

为简便起见，笔者将讨论范围限定在具有默认安装的典型 PC 平台上。

典型的启动过程像是一个四级的火箭。每一级火箭将系统控制权交给下一级。

- 第 3.1.1 节
- 第 3.1.2 节
- 第 3.1.3 节
- 第 3.1.4 节

当然，这些阶段可以有不同的配置。比如，你编译了自己的内核，则可能会跳过迷你 Debian 系统的步骤。因此，在读者亲自确认之前，请勿假定自己系统的情况也是如此。

3.1.1 第一阶段：UEFI

Unified Extensible Firmware Interface (UEFI) 统一可扩展固件接口定义了启动管理器作为 UEFI 规范的一部分。当一个计算机打开电源，启动管理器是启动流程的第一阶段，它检查启动配置并基于启动配置的设置，执行特定的操作系统引导加载程序或操作系统内核（通常是引导加载程序）。启动配置通过变量存储在 NVRAM，变量包括指示操作系统引导加载程序或操作系统内核的文件系统路径的变量。

EFI system partition (ESP) EFI 系统分区是一个数据存储设备分区，在计算机里用来遵照 UEFI 规范。当计算机打开电源时，由 UEFI 固件来访问，它存储了 UEFI 应用程序和这些应用程序运行所需要的文件，包括操作系统的引导加载程序。（在老的 PC 系统，存放在 MBR 里的 BIOS 可以用来代替。）

3.1.2 第二阶段：引载加载程序

引导加载程序是启动过程的第二阶段，由 UEFI 启动。引导加载程序将系统内核映像和 initrd 映像加载到内存并将控制权交给它们。initrd 映像是根文件系统映像，其支持程度依赖于所使用的引导加载程序。

Debian 系统通常使用 Linux 内核作为默认的系统内核。当前的 5.x Linux 内核的 initrd 映像在技术上是 initramfs（初始 RAM 文件系统）映像。

有许多引导加载程序和配置选项存在。

软件包	流行度	大小	**initrd**	引导加载程序	说明
grub-efi-amd64	I:472	142	支持	GRUB UEFI	可智能识别磁盘分区和文件系统，例如 vfat、ext4…（UEFI）
grub-pc	V:17, I:504	479	支持	GRUB 第 2 版	可智能识别磁盘分区和文件系统，例如 vfat、ext4…（BIOS）
grub-rescue-pc	V:0.04, I:0.55	7269	支持	GRUB 第 2 版	此为 GRUB 第 2 版的可引导修复映像（CD 和软盘）（PC / BIOS 版本）
grml-rescueboot	V:0.1, I:1.1	39	N/A	GRUB 第 2 版插件	grml-rescueboot 增加即用的 Linux ISO 镜像到 grub2 启动菜单
syslinux	V:3, I:30	325	支持	Isolinux	可识别 ISO9660 文件系统。引导 CD 使用此项。
syslinux	V:3, I:30	325	支持	Syslinux	可识别 MSDOS 文件系统（FAT）。引导软盘使用此项。
loadlin	V:0.02, I:0.36	87	支持	Loadlin	新系统从 FreeDOS 或 MSDOS 中启动。
mbr	V:0.2, I:2.6	47	不支持	Neil Turton 的 MBR	此为取代 MSDOS MBR 的自由软件。只可识别硬盘分区。

Table 3.1: 引导加载程序列表

对于 UEFI 系统，GRUB2 首先读取 ESP 分区，使用”`/boot/efi/EFI/debian/grub.cfg`” 里面 `search.fs_uuid` 指定的 UUID 来确定 GRUB2 菜单配置文件”`/boot/grub/grub.cfg`” 所在的分区。

GRUB2 菜单配置文件的关键部分看起来像：

```
menuentry 'Debian GNU/Linux' ... {
        load_video
        insmod gzio
        insmod part_gpt
        insmod ext2
        search --no-floppy --fs-uuid --set=root fe3e1db5-6454-46d6-a14c-071208ebe4b1
        echo    'Loading Linux 5.10.0-6-amd64 ...'
        linux   /boot/vmlinuz-5.10.0-6-amd64 root=UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1 ↩
             ro quiet
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-5.10.0-6-amd64
}
```

对于这部分的 `/boot/grub/grub.cfg`，这个菜单条目的意义如下。

在 Debian 系统，”`/boot/grub/grub.cfg`” 是由安装的 GRUB 软件包 (如：`grub-efi-amd64`) 管理，不赞成用户直接修改这个文件。应当定制”`/etc/grub.d/`” 和”`/etc/default/grub`” 来代替修改配置文件。配置 GRUB 配置文件，更新 NVRAM 变量来自动启动到 Debian：

```
# dpkg-reconfigure grub-efi-amd64
```

设置	值
GRUB2 模块加载	`gzio`, `part_gpt`, `ext2`
使用的根文件系统分区	由 `UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1` 指定的分区标识
内核镜像文件在根文件系统中的路径	`/boot/vmlinuz-5.10.0-6-amd64`
使用的内核启动参数	”`root=UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1 ro quiet`”
initrd 镜像文件在根文件系统中的路径	`/boot/initrd.img-5.10.0-6-amd64`

Table 3.2: `/boot/grub/grub.cfg` 文件上面部分菜单条目意义

提示
通过删除”`/etc/default/grub`” 里面 `GRUB_CMDLINE_LINUX_DEFAULT` 的 `quiet` 值，你能够显示内核启动日志信息。

提示
把图形文件放到”`/boot/grub/`” 来增加 GRUB 背景图。

参见“`info grub`”、grub-install(8)、grub-mkconfig(8)。

3.1.3 第三阶段：迷你 **Debian** 系统

迷你 Debian 系统是启动流程的第三阶段，由引导加载程序启动。它会在内存中运行系统内核和根文件系统。这是启动流程的一个可选准备阶段。

注意
“迷你 Debian 系统”是笔者自创的术语，用于在本文档中描述启动流程的第三个阶段。这个系统通常被称为 initrd 或 initramfs 系统。内存中类似的系统在 Debian 安装程序中使用。

`/init` 程序是内存中的根文件系统上执行的第一个程序。这个程序在用户空间把内核初始化，并把控制权交给下一阶段。迷你 Debian 系统能够在主引导流程之前添加内核模块或以加密形式挂载根文件系统，使引导流程更加灵活。

- 如果 initramfs 是由 `initramfs-tools` 创建，则”`/init`” 程序是一个 shell 脚本程序。
  - 通过给内核添加“`break=init`” 等启动参数，你可以中断这部分启动流程以获取 root shell。更多中断条件请参见”`/init`“脚本。这个 shell 环境已足够成熟，你可通过它很好地检查机器的硬件。
  - 迷你 Debian 系统中可用的命令是精简过的，且主要由一个称为 busybox(1) 的 GNU 工具提供。
- 如果 initramfs 是由 `dracut` 创建，则”`/init`” 程序是一个二进制 `systemd` 程序。
  - 迷你 Debian 系统中可用的命令是一个精简过的 systemd(1) 环境。



小心
当在一个只读的根文件系统上时，使用 `mount` 命令需要添加 `-n` 选项。

3.1.4 第四阶段：常规 **Debian** 系统

常规 Debian 系统是启动流程的第四阶段，由迷你 Debian 系统启动。迷你 Debian 系统的内核在此环境下继续运行。根文件系统将由内存切换到物理存储设备上。

init 程序是系统执行的第一个程序（PID=1），它启动其它各种程序以完成主引导流程。init 程序的默认路径是”`/usr/sbin/init`“，但可通过内核启动参数修改，例如”`init=/path/to/init_program`”。

在 Debian 8 jessie（2015 年发布）版本后，”`/usr/sbin/init`” 是一个到”`/lib/systemd/systemd`” 的符号链接。

提示
你的系统中实际使用的 init 命令可以使用“`ps --pid 1 -f`”命令确认。

软件包	流行度	大小	说明
`systemd`	V:925, I:981	11401	基于事件且支持并发的 init(8) 守护进程（可替代 `sysvinit`）
`cloud-init`	V:4.6, I:6.4	3267	云实例架构的初始化系统
`systemd-sysv`	V:921, I:984	77	`systemd` 需用的用以代替 `sysvinit` 的手册页和符号链接
`init-system-helpers`	V:754, I:989	132	在 `sysvinit` 和 `systemd` 之间进行转换的帮助工具
`initscripts`	V:14, I:49	197	用于初始化和关闭系统的脚本
`sysvinit-core`	V:3.1, I:3.5	368	类 System V 的 init(8) 工具
`sysv-rc`	V:27, I:52	85	类 System V 的运行级别修改机制
`sysvinit-utils`	V:734, I:1000	100	类 System V 的实用工具（startpar(1)，bootlogd(8)，……）
`lsb-base`	V:193, I:281	12	Linux 标准规范 3.2 版的 init 脚本功能
`insserv`	V:32, I:52	132	利用 LSB init.d 脚本依赖性来组织启动步骤的工具
`kexec-tools`	V:1.6, I:5.3	320	用于 kexec(8) 重启（热启动）的 kexec 工具
`systemd-bootchart`	V:0.17, I:0.49	131	启动流程性能分析器
`mingetty`	V:0.1, I:2.3	36	仅包含控制台的 getty(8)
`mgetty`	V:0.13, I:0.44	318	可智能调制解调的 getty(8) 替代品

Table 3.3: Debian 系统启动工具列表

提示
有关启动流程加速的最新信息，请参见 Debian 维基：启动流程加速词条。

3.2 应急救援系统



警告
在没有一个应急救援系统的情况下，不要执行一个和启动过程相关的系统管理任务。

应急救援系统能够让我们执行有挑战性的任务，比如：

- 从引导管理器损坏的安装中启动系统

- 修复启动引导器损坏的安装
- 从损坏的无法启动的系统中提取数据
- 编辑文件系统、磁盘分区和涉及 root 文件系统的 LVM 卷

典型的，一个应急救援系统用一个 ISO 镜像文件提供，并写到一个可移动存储介质，比如：

- USB 闪存盘准备作为第 9.7.2 节
- CD / DVD 准备作为第 9.7.7 节

为了简化，下面提到的例子用 U 盘作为示例，但是 CD 或 DVD 也能够很好的使用。

提示
你可以按需改变一些 UEFI NVRAM 变量来启动在可移动存储介质上的任意引导管理器。

3.2.1 USB 上的 GRUB UEFI 应急救援系统

”USB 上的 GRUB UEFI 应急救援系统” 插入，打开系统电源，GRUB UEFI 应急救援系统的菜单就能够启动。

这个” USB 上的 GRUB UEFI 应急救援系统” 是事先将 Super Grub2 Disk ISO 镜像写入到 U 盘来准备。

在安装其它操作系统等启动管理器配置损坏的场景，可以这样修复：

- 启动 GRUB UEFI 应急救援系统，自动发现已安装的可启动系统。
- 从 GRUB 菜单启动已经安装的 Debian 系统。
- 在 Linux root shell 提示符：

```
# dpkg-reconfigure grub-efi-amd64
```

注意
可启动的 GRUB 应急救援 ISO 镜像也能够通过下面的”`info grub-mkrescue`” 来生成。它提供了一个命令行的 GRUB shell 提示符，但是不提供自动发现已经安装的可启动的系统。

3.2.2 USB 上的 Linux 即用救援系统

“Linux 即用应急救援系统 U 盘”插入，打开系统电源，能够启动 Linux 即用应急救援系统。

这个”USB 上的 Linux 即用应急救援系统” 能够通过予先将一个基于 Debian 的 Linux 即用 ISO 镜像写入到一个 U 盘来准备好。这里是一些这种 Linux 即用镜像的列子。

- Debian 即用镜像
- Kali Linux 即用镜像
- Grml 即用 Linux

这里是一些 Linux 即用应急管理系统的使用场景：

- 修复由于安装其它操作系统等原因导致损坏的引导管理器配置：

  - 启动 Linux 即用救援系统。
  - 挂载无法启动的 Debian 系统上包含 root 文件系统的分区到“/mnt”。
  - 在 Linux root shell 提示符：

```
# chroot /mnt; dpkg-reconfigure grub-efi-amd64
```

- 修复损坏的 dpkg 软件包：
  - 启动 Linux 即用救援系统。
  - 挂载已经安装有损坏 dpkg 软件包的 Debian 系统中，包含 root 文件系统的分区到”/mnt”。
  - 在 Linux root shell 提示符：

```
# dpkg --root /mnt -i /mnt/var/cache/apt/archives/dpkg_old_version_amd64.deb
```

- 执行通常被禁止的操作，比如在安装好的系统上进行文件系统操作（参见第 9.6 节）。



警告
你的 Linux 即用系统的 GUI（图形用户界面）屏幕在不活动时，可以被锁屏。

提示

- 设置你的密码，在你一开始使用 Linux 即用系统时，是一个好的注意。
- 你可以设置你的密码，例如在 Linux 虚拟控制台的用户 shell 提示符下用”sudo passwd *user*”（参见第 1.1.6 节）。
- 一些 Linux 即用系统可以设置默认的” 用户/密码”: ”Debian Live” = ”user/live”, ”Kali Linux Live” = ”kali/kali”

3.2.3 从 GRUB 启动的 Linux 即用应急救援系统

Linux 即用应急救援系统能够从 GRUB 菜单条目启动。使用下面的方式来准备这个 GRUB 配置：

- 安装 grml-rescueboot 软件包
- 找到一个在它镜像里面有”/boot/grub/loopback.cfg” 的 Linux 即用 ISO 镜像。
  - 在第 3.2.2 节里面提到的 ISO 镜像，在它们的镜像里面有”/boot/grub/loopback.cfg” 。
- 拷贝这些 Linux 即用 ISO 镜像到”/boot/grml/” 目录。
- 更新 GRUB 菜单：

```
# dpkg-reconfigure grub-efi-amd64
```

紧接着打开系统电源，GRUB 显示候选 Linux 即用应急救援系统的菜单。

3.3 Systemd

3.3.1 Systemd 初始化

当 Debian 系统启动，`/usr/sbin/init` 符号链接到的 `/usr/lib/systemd` 作为初始系统进程 (`PID=1`) 启动，该进程由 root (`UID=0`) 所有。参见 systemd(1)。

`systemd` 初始化进程基于单元配置文件 (参见 systemd.unit(5)) 来并行派生进程，这些单元配置文件使用声明样式来书写，代替之前的类 SysV 的过程样式。这些单元配置文件从下面的一系列路径来加载 (参见 systemd-system.conf(5)) 。

派生的进程被放在一个单独的 Linux control groups，在单元后命名，它们属于一个私有的 systemd 层级结构 (参见 cgroups 和第 4.7.5 节)。

系统模式单元从 systemd.unit(5) 描述中的"System Unit Search Path" 加载。主要部分是按照下列优先权顺序：

- "`/etc/systemd/system/*`": 管理员创建的系统单元文件
- "`/run/systemd/system/*`": 运行时单元文件
- "`/lib/systemd/system/*`": 发行版软件包管理器安装的系统单元文件

他们的相互依赖关系通过"`Wants=`", "`Requires=`", "`Before=`", "`After=`", …等指示来配置，(参见 systemd.unit(5) 里的"MAPPING OF UNIT PROPERTIES TO THEIR INVERSES")。资源控制也是被定义 (参见 systemd.resource-control(5)).

根据单元配置文件的后缀来区分它们的类型：

- ***.service** 描述由 `systemd` 控制和监管的进程. 参见 systemd.service(5).
- ***.device** 描述在 sysfs(5) 里面作为 udev(7) 设备树展示的设备。参见 systemd.device(5).
- ***.mount** 描述由 `systemd` 控制和监管的文件系统挂载点。参见 systemd.mount(5).
- ***.automount** 描述由 `systemd` 控制和监管的文件系统自动挂载点。参见 systemd.automount(5).
- ***.swap** 描述由 `systemd` 控制和监管的 swap 文件或设备。参见 systemd.swap(5).
- ***.path** 描述被 `systemd` 监控的路径，用于基于路径的活动。参见 systemd.path(5).
- ***.socket** 描述被 `systemd` 控制和监管的套接字，用于基于套接字的活动。参见 systemd.socket(5).
- ***.timer** 描述被 `systemd` 控制和监管的计时器，用于基于时间的活动。参见 systemd.timer(5).
- ***.slice** 管理 cgroups(7) 的资源。参见 systemd.slice(5).
- ***.scope** 使用 `systemd` 的总线接口来程序化的创建，用以管理一系列系统进程。参见 systemd.scope(5).
- ***.target** 把其它单元配置文件分组，在启动的时候，来创建同步点。参见 systemd.target(5).

系统启动时(即，init)，`systemd` 进程会尝试启动"`/lib/systemd/system/default.target`(通常是到"`graphical.ta`的符号链接)。首先，一些特殊的 target 单元 (参见 systemd.special(7))，比如"`local-fs.target`"、"`swap.target`"和"`cryptsetup.target`" 会被引入以挂载文件系统。之后，其它 target 单元也会根据单元依赖关系而被引入。详细情况，请阅读 bootup(7)。

`systemd` 提供向后兼容的功能。在"`/etc/init.d/rc[0123456S].d/[KS]`*name*" 里面的 SysV 风格的启动脚本仍然会被分析；telinit(8) 会被转换为 systemd 的单元活动请求。



小心
模拟的运行级别 2 到 4 全部被符号链接到了相同的"`multi-user.target`"。

3.3.2 Systemd 登录

当一个用户通过 gdm3(8)、sshd(8) 等登录到 Debian 系统,`/lib/systemd/system --user` 作为用户服务管理器进程启动，并由相应的用户所有。参见 systemd(1)。

`systemd` 初始化进程基于单元配置文件 (参见 systemd.unit(5)) 来并行派生进程，这些单元配置文件使用声明样式来书写，代替之前的类 SysV 的过程样式。这些单元配置文件从下面的一系列路径来加载 (参见 systemd-system.conf(5)) 。

用户模式单元从 systemd.unit(5) 描述中的”User Unit Search Path” 加载。主要部分是按照下列优先权顺序：

- ”`~/.config/systemd/user/*`”: 用户配置单元文件
- ”`/etc/systemd/user/*`”: 管理员创建的用户单元文件
- ”`/run/systemd/user/*`”: 用户运行时单元文件
- ”`/lib/systemd/user/*`”: 发行版软件包管理器安装的用户单元文件

这些是和第 3.3.1 节用同样的方式管理。

3.4 内核消息

在控制台上显示的内核错误信息，能够通过设置他们的阈值水平来配置。

```
# dmesg -n3
```

错误级别值	错误级别名称	说明
0	KERN_EMERG	系统不可用
1	KERN_ALERT	行为必须被立即采取
2	KERN_CRIT	危险条件
3	KERN_ERR	错误条件
4	KERN_WARNING	警告条件
5	KERN_NOTICE	普通但重要的条件
6	KERN_INFO	信息提示
7	KERN_DEBUG	debug 级别的信息

Table 3.4: 内核错误级别表

3.5 系统消息

在 `systemd` 下, 内核和系统的信息都通过日志服务 `systemd-journald.service` (又名 `journald`) 来记录，放在”`/var/log/journal`” 下的不变的二进制数据，或放在”`/run/log/journal/`” 下的变化的二进制数据. 这些二进制日志数据，可以通过 journalctl(1) 命令来访问。例如，你可以显示从最后一次启动以来的日志，按如下所示：

```
$ journalctl -b
```

在 `systemd` 下，系统日志工具 rsyslogd(8) 可以被卸载。如果安装了它，它会改变它的行为来读取易失性二进制日志数据(代替在 systemd 之前默认的”`/dev/log`”)并创建传统的永久性 ASCII 系统日志数据。”`/etc/default/rsyslog`” 和”`/etc/rsyslog.conf`” 能够自定义日志文件和屏幕显示。参见 rsyslogd(8) 和 rsyslog.conf(5)，也可以参见第 9.3.2 节。

操作	命令片段
查看从最后一次启动开始的系统服务和内核日志	"`journalctl -b --system`"
查看从最后一次启动开始的当前用户的服务日志	"`journalctl -b --user`"
查看从最后一次启动开始的"`$unit`" 工作日志	"`journalctl -b -u $unit`"
查看从最后一次启动开始的"`$unit`" 的工作日志 ("`tail -f`" 式样)	"`journalctl -b -u $unit -f`"

Table 3.5: 典型的 `journalctl` 命令片段列表

3.6 系统管理

`systemd` 不仅仅提供系统初始化，还用 systemctl(1) 命令提供通用的系统管理操作。

这里, 上面例子中的"`$unit`"，可以是一个单元名（后缀`.service` 和`.target` 是可选的)，或者，在很多情况下，也可以是匹配的多个单元 (shell 式样的全局通配符"`*`", "`?`", "`[]`"，通过使用 fnmatch(3) ，来匹配目前在内存中的所有单元的基本名称).

上面列子的系统状态改变命令，通常是通过"`sudo`" 来处理，用以获得需要的系统管理权限。

"`systemctl status $unit|$PID|$device`" 的输出使用有颜色的点 ("●") 来概述单元状态，让人看一眼就知道。

- 白色的"●" 表示一个" 不活动" 或" 变为不活动中" 的状态。
- 红色的"●" 表示"失败"或者"错误"状态。
- 绿色"●" 表示"活动"、"重新加载中"或"激活中"状态。

3.7 其它系统监控

这里是 `systemd` 下其它零星的监控命令列表。请阅读包括 cgroups(7) 在内的相关的 man 手册页。

3.8 系统配置

3.8.1 主机名

内核维护系统主机名。在启动的时候，通过 `systemd-hostnamed.service` 启动的系统单位设置系统的主机名，此主机名保存在"`/etc/hostname`"。这个文件应该只包含系统主机名，而不是全称域名。

不带参数运行 hostname(1) 命令可以打印出当前的主机名。

3.8.2 文件系统

硬盘和网络文件系统的挂载选项可以在"`/etc/fstab`" 中设置，参见 fstab(5) 和第 9.6.7 节。

加密文件系统的配置设置在"`/etc/crypttab`"中。参见 crypttab(5)

软 RAID 的配置 mdadm(8) 设置在"`/etc/mdadm/mdadm.conf`". 参见 mdadm.conf(5).

操作	命令片段
列出所有存在的单元类型	"systemctl list-units --type=help"
列出内存中所有 target 单元	"systemctl list-units --type=target"
列出内存中所有 service 单元	"systemctl list-units --type=service"
列出内存中所有 device 单元	"systemctl list-units --type=device"
列出内存中所有 mount 单元	"systemctl list-units --type=mount"
列出内存中所有 socket 单元	"systemctl list-sockets"
列出内存中所有 timer 单元	"systemctl list-timers"
启动"$unit"	"systemctl start $unit"
停止"$unit"	"systemctl stop $unit"
重新加载服务相关的配置	"systemctl reload $unit"
停止和启动所有"$unit"	"systemctl restart $unit"
启动"$unit" 并停止所有其它的	"systemctl isolate $unit"
转换到" 图形" (图形界面系统)	"systemctl isolate graphical"
转换到" 多用户" (命令行系统)	"systemctl isolate multi-user"
转换到" 应急模式" (单用户命令行系统)	"systemctl isolate rescue"
向"$unit" 发送杀死信号	"systemctl kill $unit"
检查"$unit" 服务是否是活动的	"systemctl is-active $unit"
检查"$unit" 服务是否是失败的	"systemctl is-failed $unit"
检查"$unit\|$PID|device" 的状态	"systemctl status $unit\|$PID|$device"
显示"$unit\|$job" 的属性	"systemctl show $unit\|$job"
重设失败的"$unit"	"systemctl reset-failed $unit"
列出所有单元服务的依赖性	"systemctl list-dependencies --all"
列出安装在系统上的单元文件	"systemctl list-unit-files"
启用"$unit" (增加符号链接)	"systemctl enable $unit"
禁用"$unit" (删除符号链接)	"systemctl disable $unit"
取消遮掩"$unit" (删除到"/dev/null" 的符号链接)	"systemctl unmask $unit"
遮掩"$unit" (增加到"/dev/null" 的符号链接)	"systemctl mask $unit"
获取默认的 target 设置	"systemctl get-default"
设置默认 target 为"graphical" (图形系统)	"systemctl set-default graphical"
设置默认的 target 为"multi-user" (命令行系统)	"systemctl set-default multi-user"
显示工作环境变量	"systemctl show-environment"
设置环境变量"variable" 的值为"value"	"systemctl set-environment variable=value"
取消环境变量"variable" 的设置	"systemctl unset-environment variable"
重新加载所有单元文件和后台守护进程 (daemon)	"systemctl daemon-reload"
关闭系统	"systemctl poweroff"
关闭和重启系统	"systemctl reboot"
挂起系统	"systemctl suspend"
休眠系统	"systemctl hibernate"

Table 3.6: 典型的 systemctl 命令片段列表

操作	命令片段
显示每一个初始化步骤所消耗的时间	"systemd-analyze time"
列出所有单元的初始化时间	"systemd-analyze blame"
加载"$unit" 文件并检测错误	"systemd-analyze verify $unit"
简洁的显示用户调用会话的运行时状态信息	"loginctl user-status"
简洁的显示调用会话的运行时状态信息	"loginctl session-status"
跟踪 cgroups 的启动过程	"systemd-cgls"
跟踪 cgroups 的启动过程	"ps xawf -eo pid,user,cgroup,args"
跟踪 cgroups 的启动过程	读取"/sys/fs/cgroup/" 下的 sysfs

Table 3.7: systemd 下其它零星监控命令列表



警告
每次启动的时候，在挂载了所有文件系统以后，"/tmp", "/var/lock", 和"/var/run" 中的临时文件会被清空。

3.8.3 网络接口初始化

对于使用 systemd 的现代 Debian 桌面系统，网络接口通常由两个服务进行初始化：lo 接口通常在"networking.servic
处理，而其它接口则由"NetworkManager.service"处理。

参见第 5 章来获取怎样来配置它们的信息。

3.8.4 云系统初始化

云系统实例可以由"Debian Official Cloud Images" 或类似的镜像启动。对于这样的系统实例，主机名、文件系统、网络、语言环境、SSH 密钥、用户和组等个性化信息，可以使用 cloud-init 和 netplan.io 软件包提供的功能来配置，利用多个数据源，放在原始系统镜像里面的文件和在启动过程中提供的外部数据。这些软件包使用 YAML 数据来声明系统配置。

更多信息参见"Cloud Computing with Debian and its descendants", "Cloud-init documentation" 和第 5.4 节。

3.8.5 调整 sshd 服务的个性化例子

使用默认安装，通过 systemd 启动的过程中，在 network.target 启动后，很多网络服务 (参见第 6 章) 作为后台守护进程（daemon）启动。"sshd" 也不列外。让我们修改为按需启动"sshd" 作为一个定制化的例子。

首先，禁用系统安装的服务单元。

```
$ sudo systemctl stop sshd.service
$ sudo systemctl mask sshd.service
```

传统 Unix 服务的按需套接字激活（on-demand socket activation）系统由 inetd (或 xinetd) 超级服务来提供。在 systemd 下, 相同功能能够通过增加 ***.socket** 和 ***.service** 单元配置文件来启用。

sshd.socket 用来定义一个监听的套接字

```
[Unit]
Description=SSH Socket for Per-Connection Servers

[Socket]
ListenStream=22
Accept=yes
```

```
[Install]
WantedBy=sockets.target
```

`sshd@.service` 作为 `sshd.socket` 匹配的服务文件

```
[Unit]
Description=SSH Per-Connection Server

[Service]
ExecStart=-/usr/sbin/sshd -i
StandardInput=socket
```

然后重新加载。

```
 $ sudo systemctl daemon-reload
```

3.9 udev 系统

从 Linux 内核 2.6 版开始，udev 系统提供了自动硬件发现和初始化机制。(参见 udev(7)). 在内核发现每个设备的基础上，udev 系统使用从 sysfs 文件系统 (参见第 1.2.12 节) 的信息启动一个用户进程，使用 modprobe(8) 程序 (参见第 3.10 节) 加载支持它所要求的内核模块, 创建相应的设备节点。

提示
如果由于某些理由，"`/lib/modules/`*`kernel-version`*`/modules.dep`" 没有被 depmod(8) 正常生成，模块可能不会被 udev 系统按期望的方式加载。执行"`depmod -a`" 来修复它。
"`/etc/fstab`" 里面的挂载规则, 设备节点不必需是静态的。你能够使用 UUID 来挂载设备，来代替"`/dev/sda`" 之类的设备名. 参见第 9.6.3 节.

由于 udev 系统是一个正在变化的事物，我在其它文档进行了详细描述，在这里只提供了最少的信息。



警告
不要尝试用 udev 规则里面的 `RUN`（在 udev(7) 提到）长期运行程序，比如说备份脚本。请创建一个适当的 systemd.service(5) 文件并激活它来替代。参见第 10.2.3.2 节。

3.10 内核模块初始化

通过 modprobe(8) 程序添加和删除内核模块，使我们能够从用户进程来配置正在运行的 Linux 内核。udev 系统 (参见第 3.9 节) 自动化它的调用来帮助内核模块初始化。

存在一些非硬件模块及特殊的硬件驱动模块(如下所述)，需要通过预加载的方式进行加载；具体做法是在"`/etc/modules-`目录下创建一个以`.conf` 为后缀的文件，并将这些模块逐行写入其中（参见 modules-load.d(5)）。

- TUN/TAP 模块提供虚拟的 Point-to-Point 网络设备 (TUN) 和虚拟的 Ethernet 以太网网络设备 (TAP),
- netfilter 模块提供 netfilter 防火墙能力 (iptables(8), 第 5.7 节),
- watchdog timer 驱动模块。

modprobe(8) 程序的配置文件是按 modprobe.conf(5) 的说明放在"`/etc/modprobes.d/`" 目录下，(如果你想避免自动加载某些内核模块，考虑把它们作为黑名单放在"`/etc/modprobes.d/blacklist`" 文件里.)

"`/lib/modules/`*`version`*`/modules.dep`" 文件由 depmod(8) 程序生成，它描述了 modprobe(8) 程序使用的模块依赖性.

注意
如果你在启动时出现模块加载问题，或者 modprobe(8) 时出现模块加载问题, "`depmod -a`" 可以通过重构"`modules.dep`" 来解决这些问题。

---

modinfo(8) 程序显示 Linux 内核模块信息。

lsmod(8) 程序以好看的格式展示"`/proc/modules`" 的内容, 显示当前内核加载了哪些模块。

提示
你能够精确识别你系统上的硬件。参见第 9.5.3 节.
你可以在启动时配置硬件来激活期望的硬件特征。参见第 9.5.4 节.
你可以重新编译内核来增加你的特殊设备的支持。参见第 9.10 节.

Chapter 4

认证和访问控制

当用户（或程序）需要访问系统时，需要进行认证，确认身份是受信任。



警告
PAM 的配置错误可能会锁住你的系统。你必须有一个准备好的救援 CD，或者设立一个替代的 boot 分区。为了恢复系统，你需要使用它们启动系统并纠正错误。

4.1 一般的 Unix 认证

一般的 Unix 认证由 PAM （Pluggable Authentication Modules，即可插入的验证模块） 下的 pam_unix(8) 模块提供。它的 3 个重要文件如下，其内的条目使用“:”分隔。

文件	权限	用户	组	说明
/etc/passwd	-rw-r--r--	root	root	（明文的）用户账号信息
/etc/shadow	-rw-r-----	root	shadow	安全加密的用户账号信息
/etc/group	-rw-r--r--	root	root	组信息

Table 4.1: pam_unix(8) 使用的 3 个重要配置文件

“/etc/passwd”包含下列内容。

```
 ...
user1:x:1000:1000:User1 Name,,,:/home/user1:/bin/bash
user2:x:1001:1001:User2 Name,,,:/home/user2:/bin/bash
 ...
```

如 passwd(5) 中所述，这个文件中被“:”分隔的每项含义如下。

- 登录名
- 密码形式说明
- 数字形式的用户 ID
- 数字形式的组 ID
- 用户名或注释字段

- 用户家目录
- 可选的用户命令解释器

“/etc/passwd”的第二项曾经被用来保存加密后的密码。在引入了“/etc/shadow”后，该项被用来说明密码形式。

内容	说明
（空）	无需密码的账号
x	加密后的密码保存在“/etc/shadow”

Table 4.2: “/etc/passwd”第二项的内容

“/etc/shadow”包含下列内容。

```
 ...
user1:$1$Xop0FYH9$IfxyQwBe9b8tiyIkt2P4F/:13262:0:99999:7:::
user2:$1$vXGZLVbS$ElyErNf/agUDsm1DehJMS/:13261:0:99999:7:::
 ...
```

如 shadow(5) 中所述，这个文件中被“:”分隔的每项含义如下。

- 登录名
- 加密后的密码（开头的“1”表示使用 MD5 加密。“*”表示无法登录。）
- 最后一次修改密码的时间，其表示从 1970 年 1 月 1 日起的天数
- 允许用户再次修改密码的天数间隔
- 用户必须修改密码的天数间隔
- 密码失效前的天数，在此期间用户会被警告
- 密码失效后的天数，在次期间密码依旧会被接受
- 账号失效的时间，其表示从 1970 年 1 月 1 日起的天数
- …

“/etc/group”包含下列内容。

```
group1:x:20:user1,user2
```

如 group(5) 中所述，这个文件中被“:”分隔的每项含义如下。

- 组名称
- 加密后的密码（不会被真正使用）
- 数字形式的组 ID
- 使用“,”分隔的用户名列表

注意
“/etc/gshadow”为“/etc/group”提供了与“/etc/shadow”相似的功能，但没有被真正地使用。

注意
如果”auth optional pam_group.so” 这行添加到了”/etc/pam.d/common-auth”，并且在”/etc/security/group.conf” 里进行了设置，一个用户的实际组就可以被动态添加。参见 pam_group(8).

注意
base-passwd 软件包包含了一份用户和组的官方文档：“/usr/share/doc/base-passwd/users-and-groups.htm

4.2 管理账号和密码信息

下面是一些管理账号信息的重要命令。

命令	功能
getent passwd *user_name*	浏览“*user_name*”的账号信息
getent shadow *user_name*	浏览用户”*user_name*” 隐藏的账户信息
getent group *group_name*	浏览“*group_name*”的组信息
passwd	管理账号密码
passwd -e	为激活的账号设置一次性的密码
chage	管理密码有效期信息

Table 4.3: 管理账号信息的命令

其中的一些功能只能被 root 使用。密码和数据的加密参见 crypt(3)。

注意
在设置了 PAM 和 NSS 的系统上（例如 Debian salsa 机器），本地的“/etc/passwd”、“/etc/group”和“/etc/shadow”可能不会被系统激活使用。上述的命令即使处于这种环境下依旧是有效的。

4.3 好密码

在系统安装时建立一个账号或使用 passwd(1) 命令时，你应该选择一个好密码，它应该由 6 到 8 个字符组成，其中包含下列根据 passwd(1) 设定的每个组合中的一个或多个字符。

- 小写字母
- 数字 0 到 9
- 标点符号



警告
密码中不要使用可以猜到的词。账号名、身份证号码、电话号码、地址、生日、家庭成员或宠物的名字、字典单词、简单的字符序列（例如“12345”或“qwerty”）等都是糟糕的选择。

4.4 设立加密的密码

下面是一些用于生成加盐的加密密码的独立工具。

软件包	流行度	大小	命令	功能
whois	V:25, I:199	384	mkpasswd	具备 crypt(3) 库所有特性的前端
openssl	V:853, I:996	2564	openssl passwd	计算密码哈希 (OpenSSL). passwd(1ssl)

Table 4.4: 生成密码的工具

4.5 PAM 和 NSS

现代的类 Unix 系统（例如 Debian 系统）提供 PAM（Pluggable Authentication Modules，插入式验证模块） 和 NSS（Name Service Switch，名称服务切换） 机制给本地系统管理员，使他们能够配置自己的系统。它们的功能可以概括为以下几点。

- PAM 给应用软件提供了一个灵活的认证机制，因此涉及到了密码数据的交换。
- NSS 提供了一个灵活的名称服务机制，它经常被 C 标准库使用，使例如 ls(1) 和 id(1) 这样的程序获得用户和组名称。

PAM 和 NSS 系统必须保持配置一致。

PAM 和 NSS 系统中重要的软件包如下。

软件包	流行度	大小	说明
libpam-modules	V:945, I:1000	892	插入式验证模块（基础服务）
libpam-ldapd	V:8, I:13	82	允许 LDAP 接口的插入式验证模块
libpam-systemd	V:788, I:972	849	用于 logind 注册用户会话的可插入式验证模块（Pluggable Authentication Module）
libpam-doc	I:7.7	1492	插入式验证模块（html 和文本文档）
libc6	V:924, I:1000	5483	GNU C 库：同样提供“名称服务切换”服务的共享库
glibc-doc	I:4.9	3927	GNU C 库：帮助页面
glibc-doc-reference	I:3.0	14764	GNU C 库：参考手册，有 info、pdf 和 html 格式（non-free）
libnss-mdns	V:314, I:505	142	用于解析组播 DNS 名称的 NSS 模块
libnss-ldapd	V:9, I:16	138	NSS 模块，用于使用 LDAP 作为一个名称服务的

Table 4.5: PAM 和 NSS 系统中重要的软件包

- libpam-doc 中“The Linux-PAM System Administrators’ Guide”是了解 PAM 配置的必要文档。
- glibc-doc-reference 中的“System Databases and Name Service Switch”是了解 NSS 配置的重要文档。

注意
你可以使用“aptitude search 'libpam-|libnss-'”命令查看更多的相关软件包。NSS 缩写也可能意味着“Network Security Service，网络安全服务”，它不同于“Name Service Switch，名称服务切换”。

注意
PAM 是用来为每个程序使用系统范围的默认值来初始化环境变量的最基础方法。

在 systemd 下，libpam-systemd 软件包被安装用来管理用户登录，通过为 logind 在 systemd 控制组层级中注册用户会话来实现。参见 systemd-logind(8)、logind.conf(5) 和 pam_systemd(8)。

4.5.1 PAM 和 NSS 访问的配置文件

下面是一些 PAM 和 NSS 访问的重要配置文件。

配置文件	功能
`/etc/pam.d/`*`program_name`*	为“*`program_name`*”程序设置 PAM 配置；参加 pam(7) 和 pam.d(5)
`/etc/nsswitch.conf`	为每个服务条目设置 NSS 配置。参见 nsswitch.conf(5)
`/etc/nologin`	通过 pam_nologin(8) 模块限制用户登录
`/etc/securetty`	通过 pam_securetty(8) 模块限制 root 访问 tty
`/etc/security/access.conf`	通过 pam_access(8) 模块设置访问限制
`/etc/security/group.conf`	通过 pam_group(8) 模块设置基于组的限制
`/etc/security/pam_env.conf`	通过 pam_env(8) 模块设置环境变量
`/etc/environment`	通过带有“`readenv=1`”参数的 pam_env(8) 模块设置额外的环境变量
`/etc/default/locale`	通过带有“`readenv=1 envfile=/etc/default/locale`”参数的 pam_env(8) 模块设置语言环境值（在 Debian 系统中）
`/etc/security/limits.conf`	通过 pam_linits(8) 模块设置资源限制（ulimit、core 等等）
`/etc/security/time.conf`	通过 pam_time(8) 模块设置时间限制
`/etc/systemd/logind.conf`	设置 `systemd` 的登录管理器配置 (参见 logind.conf(5) 和 systemd-logind.service(8))

Table 4.6: PAM 和 NSS 访问的配置文件

密码选择的限制是通过 PAM 模块 pam_unix(8) 和 pam_cracklib(8) 来实现的。它们可以通过各自的参数进行配置。

> 提示
> PAM 模块在文件名中使用后缀“`.so`”。

4.5.2 现代的集中式系统管理

现代的集中式系统管理可以使用集中式的轻量目录访问协议（LDAP）服务器进行部署，从而通过网络管理许多类 Unix 和非类 Unix 系统。轻量目录访问协议的开源实现是 OpenLDAP 软件。

LDAP 服务器使用带有 PAM 和 NSS 的 `libpam-ldapd` 和 `libnss-ldapd` 软件包为 Debian 系统提供账号信息。需要一些动作来启用 LDAP（我没有使用过这个设置，并且下面的信息纯粹是第二手的信息。请在这种前提下阅读下列内容。）。

- 你通过运行一个程序，例如独立的 LDAP 守护进程 slapd(8)，来建立集中式的 LDAP 服务器。
- 你在“`/etc/pam.d/`”目录中的 PAM 配置文件里，使用“`pam_ldap.so`”替代默认值“`pam_unix.so`”。
- 你在“`/etc/nsswitch.conf`”文件中改变 NSS 配置，使用“`ldap`”替代默认值（“`compat`”或“`file`”）。
- 为了密码的安全，你必须让 `libpam-ldapd` 使用 SLL（或 TLS） 连接。
- 为了确保 LDAP 网络开销数据的完整性，你必须让 `libpam-ldapd` 使用 SLL（或 TLS） 连接。
- 为了减少 LDAP 网络流量，你应该在本地运行 nscd(8) 来缓存任何 LDAP 搜索结果。

参见由 `libpam-doc` 软件包提供的 nsswitch.conf(5)、pam.conf(5)、ldap.conf(5) 和“`/usr/share/doc/libpam-doc/html/`”中的文档，以及 `glibc-doc` 软件包提供的“`info libc 'Name Service Switch'`”。

类似地，你可以使用其它方法来设置另一种集中式的系统。

- 同 Windows 系统集成用户和组。
  - 通过 `winbind` 和 `libpam_winbind` 软件包访问 Windows domain 服务.
  - 参见 winbindd(8) 和 Integrating MS Windows Networks with Samba。
- 同古老的类 Unix 系统集成用户和组。
  - 通过 `nis` 软件包访问 NIS (之前叫 YP) 或 NIS+。
  - 参见 The Linux NIS(YP)/NYS/NIS+ HOWTO。

4.5.3 “为什么 GNU su 不支持 wheel 组”

这是在旧的“`info su`”底部 Richard M. Stallman 所说的一句名言。别担心：Debian 系统中当前的 `su` 命令使用了 PAM，这样当在“`/etc/pam.d/su`”中启用了带有“`pam_wheel.so`”的行后，就能够限制非 wheel 组的用户 `su` 到 `root` 组的能力。

4.5.4 严格的密码规则

安装 `libpam-pwquality` 软件包你能够强制使用严格的密码规则。

在一个典型的 GNOME 系统，将会安装 `libpam-gnome-keyring`，”`/etc/pam.d/common-password`” 看起来像：

```
# here are the per-package modules (the "Primary" block)
password  requisite     pam_pwquality.so retry=3 minlen=8 difok=3
password  [success=1 default=ignore]  pam_unix.so obscure use_authtok try_first_pass  ↩
    yescrypt
# here's the fallback if no module succeeds
password  requisite     pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password  required      pam_permit.so
# and here are more per-package modules (the "Additional" block)
password  optional  pam_gnome_keyring.so
# end of pam-auth-update config
```

4.6 安全认证

> 注意
> 这里的信息也许不能完全满足你的安全需求，但这里应当是一个好的起点。

4.6.1 确保互联网上的的密码安全

许多流行的传输层服务都使用纯文本来传输包括密码验证信息在内的各类消息。使用纯文本在公网上传输密码是很糟糕的做法，因为这样传输的密码很容易在网上被他人截获。为了确保整个沟通过程，包括密码信息在内都使用加密传输来确保安全，您可以在“传输层安全（Transport Layer Security，TLS）”协议或者其前身，“安全套接字层（Secure Sockets Layer，SSL）”协议之上运行这些服务。

加密消耗 CPU 时间。作为对 CPU 有益的替代方案，你可以保持使用纯文本通讯，仅仅使用安全认证协议加密密码，比如说：POP 使用”Authenticated Post Office Protocol” (APOP)，SMTP 和 IMAP 使用”Challenge-Response

不安全的服务名	端口	安全的服务名	端口
www (http)	80	https	443
smtp (mail)	25	ssmtp (smtps)	465
ftp-data	20	ftps-data	989
ftp	21	ftps	990
telnet	23	telnets	992
imap2	143	imaps	993
pop3	110	pop3s	995
ldap	389	ldaps	636

Table 4.7: 安全和不安全的服务端口列表

Authentication Mechanism MD5" (CRAM-MD5)。(你的邮件客户端通过互联网上你的邮件服务器发送邮件时，最近流行使用新的递交端口 587 来代替传统的 SMTP 端口 25，这样可以避免在使用 CRAM-MD5 认证自己时，网络提供商阻塞 25 端口。)

传统的基于密码的认证仍然有固有的安全问题，比如密码在多个网站重复使用。参见其它安全方案的认证方式：基于时间的单次密码 Time-based one-time password (TOTP), Universal 2nd Factor (U2F), Web 认证 (WebAuthn, 通常指"passkey")

4.6.2 安全 Shell

安全 Shell (SSH) 程序使用安全认证来提供不安全网络上两个不可信任主机之间的安全加密通讯。它由 OpenSSH 客户端, ssh(1), 和 OpenSSH 后台守护进程（daemon），sshd(8) 组成.SSH 使用端口转发特性，可以给 POP 和 X 之类的不安全的协议通讯建立隧道，使其可以在互联网上安全传输。

客户端可以使用如下方式来认证自己：基于主机的认证、公钥认证、质疑应答认证、密码认证。使用公钥认证，可以实现远程免密码登录。参见第 6.3 节.

4.6.3 互联网额外的安全方式

即使你运行 Secure Shell (SSH) 和 Point-to-point tunneling protocol (PPTP) 这样的安全服务，在互联网上，仍然有机会使用野蛮暴力猜测密码攻击进入。使用防火墙策略 (参见第 5.7 节)，并和下面的安全工具一起，可以提升安全形势。

软件包	流行度	大小	说明
knockd	V:0.7, I:1.5	110	小的 port-knock 后台守护进程（daemon）knockd(1) 和客户端 knock(1)
fail2ban	V:101, I:112	2275	禁用造成多个认证错误的 IP

Table 4.8: 提供额外安全方式的工具列表

4.6.4 root 密码安全

为阻止人们使用 root 权限访问你的机器，你需要做下面的操作。

- 防止对系统存储设备的物理访问 (HDD / SSD / ...)
- 锁住 UEFI/ BIOS 来阻止从可移动介质启动
- 为 GRUB 交互式会话设置密码

- 锁住 GRUB 菜单，禁止编辑

如果可以物理访问系统存储设备，则可以使用下面的步骤，相对简单的重置密码。

1. 将系统存储设备拿到一个可以设置 UEFI/BIOS 从 USB 启动的电脑。
2. 使用应急救援介质启动系统 (参见第 3.2.2 节)。
3. 用读写访问挂载根分区。
4. 编辑根分区的"`/etc/passwd`" 文件，使 `root` 账户条目的第二段为空。

对于 `grub-rescue-pc` ，即使用紧急介质启动的电脑，如果有编辑 GRUB 菜单条目 (参见第 3.1.2 节) 的权限，在启动时，使用下面的步骤更加简单。

1. 使用内核参数启动系统来修改一些事情，比如说，"`root=/dev/sda6 rw init=/bin/sh`".
2. 编辑"`/etc/passwd`" 文件，使 `root` 账户条目的第二段为空。
3. 重启系统。

系统的 root shell 现在可以无密码访问了。

注意
一旦某人拥有 root shell 访问权限，他能够访问任何内容，并可以重设系统上的任何密码。此外，他可以使用 `john` 和 `crack` 等软件包的暴力破解工具来比较所有用户的密码 (参见第 9.5.10 节)。被破解的密码，可以用来和其它系统进行比较。

为避免这些相关问题，仅有的理论上的软件解决方案是使用 dm-crypt 和 initramfs (参见第 9.9 节) 加密 root 分区 (或"`/etc`" 分区) 。这样的话，你总是需要密码来启动系统。

4.7 其它的访问控制

在密码基于认证和文件权限之外，系统也有其它的访问控制。

注意
参见第 9.4.16 节来限制内核的安全警告密钥（SAK）功能。

4.7.1 访问控制列表 (ACLs)

访问控制列表 ACL 是在第 1.2.3 节里面解释的普通权限的一个超集。

在现代桌面环境中，你会遇到 ACL 行为。比如，当一个已经格式化的 USB 存储设备自动挂载到"`/media/penguin/USBSTI`
一个普通用户 `penguin` 能够执行：

```
 $ cd /media/penguin
 $ ls -la
total 16
drwxr-x---+ 1 root    root    16 Jan 17 22:55 .
drwxr-xr-x  1 root    root    28 Sep 17 19:03 ..
drwxr-xr-x  1 penguin penguin 18 Jan  6 07:05 USBSTICK
```

在第 11 列的"+" 表示 ACL 在使用。如果没有 ACL，一个普通用户 penguin 应该不能够像这样列出目录内容，因为 penguin 不在 root 组。你能够按如下方式查看 ACL：

```
 $ getfacl .
# file: .
# owner: root
# group: root
user::rwx
user:penguin:r-x
group::---
mask::r-x
other::---
```

这里：

- "user::rwx"、"group::---" 和"other::---" 相应的为普通所有者、组和其它人的权限。
- ACL "user:penguin:r-x" 运行普通用户 penguin 有"r-x" 权限。这个能够让"ls -la" 列出目录内容。
- ACL "mask::r-x" 设置上层绑定权限。

更多信息参见"POSIX Access Control Lists on Linux"、acl(5)、getfacl(1) 和 setfacl。

4.7.2 sudo

sudo(8) 程序是为了使一个系统管理员可以给用户受限的 root 权限并记录 root 活动而设计的。sudo 只需要一个普通用户的密码。安装 sudo 软件包并通过设置"/etc/sudoers"中的选项来使用它。参见"/usr/share/doc/sudo/examples/su 和第 1.1.12 节中的配置示例。

我将 sudo 用于单用户系统（参见第 1.1.12 节）是为了防止自己可能做出的愚蠢行为。就我个人而言，我认为使用 sudo 会比使用 root 账号操作系统来得好。例如，下列命令将"*some_file*"的拥有者改变为"*my_name*"。

```
$ sudo chown my_name some_file
```

当然如果你知道 root 密码（比如自行安装 Debian 的用户所做的），任何用户账号都可以使用"su -c"让任何命令以 root 运行。

4.7.3 PolicyKit

PolicyKit 是在类 Unix 操作系统中控制整个系统权限的一个操作系统组件。

较新的 GUI 图形界面程序设计时便考虑到了不作为特权进程来运行。它们通过 PolicyKit 来和特权进程通信，从而执行管理操作。

在 Debian 系统中，PolicyKit 限制了属于 sudo 组的用户账号的这种操作。

参见 polkit(8)。

4.7.4 限制访问某些服务端的服务

对系统安全而言，尽可能的禁用服务程序，是一个好的主意。网络服务是危险的。有不使用的服务，不管是直接由后台守护进程（daemon）激活，还是通过 super-server 程序激活，都被认为是安全风险。

许多程序, 比如说 sshd(8), 使用基于 PAM 的访问控制。也还有许多方式来限制访问一些服务端的程序。

- 配置文件: "/etc/default/*program_name*"

- 后台守护进程（daemon）的 Systemd 服务单元配置
- PAM (Pluggable Authentication Modules)
- super-server 使用"`/etc/inetd.conf`"
- TCP wrapper 使用"`/etc/hosts.deny`" 和"`/etc/hosts.allow`" , tcpd(8)
- Sun RPC" 使用 `/etc/rpc.conf`"
- atd(8) 使用"`/etc/at.allow`" 和"`/etc/at.deny`"
- crontab(1) 使用"`/etc/cron.allow`" 和"`/etc/cron.deny`"
- Network firewall 或 netfilter 框架

参见第 3.6 节、第 4.5.1 节和第 5.7 节。

提示
NFS 和其它基于 RPC 的程序，需要激活 Sun RPC 服务。

提示
如果你远程访问最新的 Debian 系统有问题，看下在"`/etc/hosts.deny`" 里是否存在"`ALL: PARANOID`" 这样讨厌的配置，请把它注释掉。(但是你必须注意这种行为所带来的安全风险。)

4.7.5 Linux 安全特性

Linux 内核已经发展和支持在传统的 UNIX 实现里面没有的安全特征。

Linux 支持扩展属性，扩展了传统的 UNIX 属性 (参见 xattr(7))。

Linux 把传统的超级用户相关的特权分开到不同的单元，被称为 capabilities(7)，它能够独立的启用和禁用。从 2.2 版本内核开始，Capabilities 是一个线程独立的属性。

Linux Security Module (LSM) 安全模块框架提供了一个多方面的安全检查机制，和新的内核扩展关联。例如：

- AppArmor
- Security-Enhanced Linux (SELinux)
- Smack (Simplified Mandatory Access Control Kernel)
- Tomoyo Linux

这些扩展紧缩的权力模型比普通的类 Unix 安全模型策略更加严格，甚至 root 的权力也被限制。建议你阅读 kernel.org 上的 Linux 安全模块（LSM）框架文档。

Linux 的 namespaces 封装了一个全局系统资源到一个抽象的概念，全局系统资源在 namespace 内对进程可见，并且 namespace 有它们自己的全局资源隔离实例。对其它进程全局资源的可见性的改变是，同一个 namespace 的成员可见，但是对非同一个 namespace 的其它进程不可见。从内核 5.6 版本起，有 8 种 namespaces (参见 namespaces(7), unshare(1), nsenter(1))。

在 Debian 11 Bullseye (2021) 中, Debian 使用 unified cgroup hierarchy（统一 cgroup 层级架构）(亦称为 cgroups-v2)。

namespaces 同 cgroups 一起来隔离它们的进程，允许资源控制的使用示例是：

- Systemd。参见第 3.3.1 节。
- 沙盒环境。参见第 7.7 节。
- Linux 容器，比如 Docker、LXC。参见第 9.11 节。

这些功能不能够通过第 4.1 节实现。这些高级话题大部分超出了本介绍文档的范围。

Chapter 5

网络设置

提示
关于 Debian 专属的网络手册，请查看 Debian 管理员手册—网络配置。

提示
systemd 环境下，可以用 networkd 来配置网络。请参考 systemd-networkd(8)。

5.1 基本网络架构

让我们来回顾一下现代 Debian 操作系统中的基本网络架构。

5.1.1 主机名解析

主机名解析，目前也是由 NSS (名字服务转换 Name Service Switch) 机制来支持。这个解析的流程如下。

1. "`/etc/nsswitch.conf`" 文件里的"`hosts: files dns`" 这段规定主机名解析顺序。(代替"`/etc/host.conf`" 文件里的"`order`" 这段原有的功能。)
2. `files` 方式首先被调用。如果主机名在"`/etc/hosts`" 文件里面发现，则返回所有有效地址并退出。("`/etc/host.con`" 文件包含"`multi on`".)
3. `dns` 方式被调用。如果主机名通过查询"`/etc/resolv.conf`" 文件里面写的互联网域名系统 Domain Name System (DNS) 来找到，则返回所有有效地址并退出。

一个典型的工作站在安装时就会设置主机名，例如"*host_name*" 和设置为空字符串的可选域名。这样，"`/etc/hosts`" 看起来像下文所示的这样。

```
127.0.0.1 localhost
127.0.1.1 host_name

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

软件包	流行度	大小	类型	说明
network-manager	V:429, I:472	7366	配置：：NM	NetworkManager（守卫进程）：自动管理网络
network-manager-gnome	V:38, I:136	19	配置：：NM	NetworkManager（GNOME 前端）
netplan.io	V:2.4, I:8.7	340	配置：：NM+networkd	Netplan (生成器): 统一的，声明网络接口到 NetworkManager 和 systemd-networkd 后端
ifupdown	V:686, I:973	204	配置：：ifupdown	用来启动/关闭网络的标准工具（Debian 特有）
pppoeconf	V:0.2, I:3.6	176	配置：：辅助	配置助手，以便于使用 PPPoE 连接
wpasupplicant	V:431, I:520	4337	配置：：辅助	WPA 和 WPA2 客户端支持（IEEE 802.11i)
wpagui	V:0.2, I:1.2	780	配置：：辅助	wpa_supplicant Qt 图形界面客户端
wireless-tools	V:196, I:263	232	配置：：辅助	操控 Linux 无线扩展的工具
iw	V:42, I:478	332	配置：：辅助	配置 Linux 无线设备的工具
iproute2	V:802, I:988	4279	配置：：iproute2	iproute2, IPv6 和其他高级网络配置：ip(8),tc(8) 等等
iptables	V:402, I:635	2408	配置：：Netfilter	封包过滤和网络地址转换管理工具（Netfilter）
nftables	V:301, I:889	189	配置：：Netfilter	封包过滤和网络地址转换管理工具（Netfilter）({ip,ip6,arp,eb}tables 的后续替代者)
iputils-ping	V:222, I:998	188	测试	测试能否连接远程主机，通过主机名或 IP 地址（iproute2）
iputils-arping	V:2, I:17	53	测试	测试能否连接远程主机，通过 ARP 地址
iputils-tracepath	V:2, I:19	50	测试	跟踪访问远程主机的路径
ethtool	V:103, I:252	1106	测试	显示或更改以太网设备的设定
mtr-tiny	V:5, I:38	182	测试：：底层	追踪连接远程主机的路径（文本界面）
mtr	V:4, I:39	231	测试：：底层	追踪连接远程主机的路径（文本界面和 GTK 界面）
gnome-nettool	V:0.5, I:7.9	2480	测试：：底层	获取常见网络信息的工具（GNOME)
nmap	V:27, I:180	4946	测试：：底层	网络映射/端口扫描（Nmap，控制台）
tcpdump	V:24, I:181	1418	测试：：底层	网络流量分析（Tcpdump，控制台）
wireshark	V:4, I:38	17442	测试：：底层	网络流量分析（Wireshark,GTK）
tshark	V:3, I:23	615	测试：：底层	网络流量分析 (控制台）
tcptrace	V:0.2, I:1.5	407	测试：：底层	根据 tcpdump 的输出生成的连接数据统计
dnsutils	V:4, I:137	23	测试：：底层	BIND 软件包提供的网络客户端程序：nslookup(1),nsupdate(1),dig(1)
dlint	V:0.1, I:1.9	52	测试：：底层	利用域名服务器查询来查看 DNS 域信息
dnstracer	V:0.1, I:1.0	63	测试：：底层	跟踪 DNS 查询直至源头

Table 5.1: 网络配置工具一览表

每一行由 IP 地址开始，接下来是相关联的主机名.

在这个例子的第二行 `127.0.1.1` IP 地址也许不会在其它类 Unix 系统发现。Debian Installer 为没有永久 IP 地址的系统创建这个条目，作为某些软件（如 GNOME）的一个变通方法，见文档 bug #719621.

host_name 匹配在"`/etc/hostname`" 里定义的主机名。(参见第 3.8.1 节)。

对于有永久 IP 地址的系统，这个永久 IP 地址应当代替这里的 `127.0.1.1`。

对于有永久 IP 地址和有域名系统 Domain Name System (DNS) 提供完全资格域名 fully qualified domain name (FQDN) 的系统，规范名 *host_name.domain_name* 应当被用来代替 *host_name*.

如果 `resolvconf` 软件包没有安装，"`/etc/resolv.conf`" 是一个静态文件。如果安装了，它是一个符号链接。此外，它包含有解析策略的初始化信息。如 DNS 是 IP="`192.168.11.1`", 则包含如下。

```
nameserver 192.168.11.1
```

`resolvconf` 软件包使这个"`/etc/resolv.conf`" 文件成为一个符号链接，并通过钩子脚本自动管理其内容。

对于典型 adhoc 局域网环境下的 PC 工作站，除了基本的 `files` 和 `dns` 方式之外，主机名还能够通过 Multicast DNS (mDNS) 进行解析。

- Avahi 提供 Debian 下的组播 DNS 发现框架。
- 它和 Apple Bonjour / Apple Rendezvous 相当.
- `libnss-mdns` 插件包提供 mDNS 的主机名解析，GNU C 库 (glibc) 的 GNU 名字服务转换 Name Service Switch (NSS) 功能支持 mDNS。
- "`/etc/nsswitch.conf`" 文件应当有像"`hosts: files mdns4_minimal [NOTFOUND=return] dns`" 这样的一段 (其它配置参见 `/usr/share/doc/libnss-mdns/README.Debian`)。
- 一个使用".local" pseudo-top-level domain 后缀的主机名解析，是用 IPv4 地址"`224.0.0.251`" 或 IPv6 地址"`FF02::FB`" 发送一个组播 UDP 包的 mDNS 查询信息。

注意
域名系统 Domain Name System 中的扩展通用顶级域名 expansion of generic Top-Level Domains (gTLD) 还在进行中。在局域网内，选择一个域名时，请提防名字冲突 name collision。

注意
使用软件包，比如 `libnss-resolve` 联合 `systemd-resolved`，或者 `libnss-myhostname`，或者 `libnss-mymachine`, 并在"`/etc/nsswitch.conf`" 文件里面的"`hosts`" 行里相应列出 mymachines 或 myhostname 关键字，这将忽略我们上面讨论的传统网络配置。更多信息参见 nss-resolve(8)、systemd-resolved(8)、nss-myhostname(8) 和 nss-mymachines(8)。

5.1.2 网络接口名称

systemd 使用"`enp0s25`" 之类的" 可预测网络接口名称"。

5.1.3 局域网网络地址范围

让我们重新提醒下在 rfc1918 里规定的局域网 local area networks (LANs)IPv4 32 位地址在各类地址的保留范围. 这些地址保证不会与因特网上专有的地址冲突。

注意
IP 地址书写中有冒号的是 IPv6 地址，例如，"`::1`" 是 `localhost` 本地主机。

类别	网络地址	子网掩码	子网掩码/位数	子网数
A	10.x.x.x	255.0.0.0	/8	1
B	172.16.x.x —172.31.x.x	255.255.0.0	/16	16
C	192.168.0.x —192.168.255.x	255.255.255.0	/24	256

Table 5.2: 网络地址范围列表

注意
如果这些地址分配到一个主机，那么这个主机一定不能够直接访问互联网，必须通过一个作为网关的代理服务或通过网络地址转换 Network Address Translation (NAT). 消费局域网环境，宽带路由器通常使用 NAT。

5.1.4 网络设备支持

尽管 Debian 系统支持大多数硬件设备，但依旧有一些网络设备需要 DFSG non-free 固件来支持它们。参见第 9.10.5 节。

5.2 现代的桌面网络配置

对于使用 systemd 的现代 Debian 桌面系统，网络接口通常由两个服务进行初始化：lo 接口通常在“networking.servic
处理，而其它接口则由“NetworkManager.service”处理。

Debian 可以通过后台守护进程（daemon）管理软件来管理网络连接，例如 NetworkManager (NM)（network-manager 和相关软件包）。

- 它们有自己的 GUI 和命令行程序来作为用户界面。
- 它们有自己的后台守护进程（daemon）作为它们的系统后端。
- 它们使你可以简单地将系统连接到网络。
- 它们使你可以简单地管理有线和无线网络的配置。
- 它们允许你配置网络而不依赖传统的 ifupdown 软件包。

注意
不要在服务器上使用这些自动网络配置工具。它们主要针对于笔记本电脑上的移动桌面用户。

这些现代的网络配置工具需要进行适当的配置，以避免与传统 ifupdown 软件包发生冲突，它的配置文件位于“/etc/network/interfaces”。

5.2.1 图形界面的网络配置工具

Debian 系统 NM 的官方文档位于“/usr/share/doc/network-manager/README.Debian”。

本质上，如下操作即可完成桌面的网络配置。

1. 通过下列命令使桌面用户 foo 归属“netdev”组（另外，例如 GNOME 和 KDE 这样的现代桌面环境会通过 D-bus 自动完成该操作）。

```
$ sudo usermod -a -G netdev foo
```

2. 使“/etc/network/interfaces”的配置保持下面那样简洁。

```
auto lo
iface lo inet loopback
```

3. 通过下列命令重新启动 NM。

```
$ sudo systemctl restart NetworkManager
```

4. 通过图形界面配置网络。

注意
只有不列在“/etc/network/interfaces”中的接口会被 NM 管理，以避免与 ifupdown 的冲突。

提示
如果你想扩展 NM 的网络配置功能，请寻找适当的插件模块和补充软件包，例如 network-manager-openconnect、network-manager-openvpn-gnome、network-manager-pptp-gnome、mobile-broadband-provider-info 等等。

5.3 没有图像界面的现代网络配置

使用 systemd 的系统中，可以在 /etc/systemd/network/ 里配置网络。参见 systemd-resolved(8)、resolved.conf(5) 和 systemd-networkd(8)。

这个允许在没有图像界面的情况下配置现代网络。

DHCP 客户端的配置可以通过创建”/etc/systemd/network/dhcp.network” 文件来进行设置。例如：

```
[Match]
Name=en*

[Network]
DHCP=yes
```

一个静态网络配置能够通过创建”/etc/systemd/network/static.network” 来设置. 比如：

```
[Match]
Name=en*

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
```

5.4 现代云网络配置

云的现代网络配置可以使用 cloud-init 和 netplan.io 软件包 (参见第 3.8.4 节)。

netplan.io 软件包支持把 systemd-networkd 和 NetworkManager 作为它的网络配置后端，能够使用 YAML 数据声明网络配置。当你改变 YAML:

- 运行"`netplan generate`" 命令，从 YAML 生成所有必须的后端配置。
- 运行"`netplan apply`" 命令应用生成的配置到后端。

参见"Netplan documentation", netplan(5), netplan-generate(8) 和 netplan-apply(8)。

也可以参见"Cloud-init documentation" (特别是围绕"Configuration sources" 和"Netplan Passthrough") 了解 `cloud-init` 是怎样能够集成替代的数据源到 `netplan.io` 配置。

5.4.1 使用 DHCP 的现代云网络配置

DHCP 客户端的配置可以通过创建一个数据源文件"`/etc/netplan/50-dhcp.yaml`" 来进行设置：

```
network:
  version: 2
  ethernets:
    all-en:
      match:
        name: "en*"
      dhcp4: true
      dhcp6: true
```

5.4.2 使用静态 IP 的现代云网络配置

一个静态网络配置能够通过创建一个数据源文件"`/etc/netplan/50-static.yaml`" 来设置：

```
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 192.168.0.15/24
      routes:
        - to: default
          via: 192.168.0.1
```

5.4.3 使用 Network Manager 的现代云网络配置

使用 Network Manager 架构的网络客户端配置，可以通过创建一个数据源文件"`/etc/netplan/00-network-manager.y`
来进行设置：

```
network:
  version: 2
  renderer: NetworkManager
```

5.5 底层网络配置

在 Linux 上的底层网络配置，使用 iproute2 程序 (ip(8), ⋯) .

5.5.1 Iproute2 命令

Iproute2 命令集提供完整的底层网络配置能力。有个从旧的 net-tools 命令集到新的 iproute2 命令集的转换表。

参见 ip(8) 和 Linux 高级路由和流量控制（Linux Advanced Routing & Traffic Control）.

旧的 **net-tools**	新的 **iproute2**	操作
ifconfig(8)	`ip addr`	一个设备上的协议（IP 或 IPv6）地址
route(8)	`ip route`	路由表条目
arp(8)	`ip neigh`	ARP 或 NDISC 缓存条目
`ipmaddr`	`ip maddr`	多播地址
`iptunnel`	`ip tunnel`	IP 隧道
nameif(8)	ifrename(8)	基于 MAC 地址的网络接口名
mii-tool(8)	ethtool(8)	以太网设备设置

Table 5.3: 从旧的 `net-tools` 命令集到新的 `iproute2` 命令集转换表

5.5.2 安全的底层网络操作

你可以按下面的方式安全的使用底层网络命令，这些命令不会改变网络配置。

命令	说明
`ip addr show`	显示活动的网络接口连接和地址状态
`route -n`	用数字地址显示全部路由表
`ip route show`	用数字地址显示全部路由表
`arp`	显示当前 ARP 缓存表的内容
`ip neigh`	显示当前 ARP 缓存表的内容
`plog`	显示 ppp 后台守护进程（daemon）日志
`ping yahoo.com`	检查到"`yahoo.com`"的因特网连接
`whois yahoo.com`	在域名数据库里面检查谁注册了"`yahoo.com`"
`traceroute yahoo.com`	跟踪到"`yahoo.com`"的因特网连接
`tracepath yahoo.com`	跟踪到"`yahoo.com`"的因特网连接
`mtr yahoo.com`	跟踪到"`yahoo.com`"的因特网连接（重复的）
`dig [@dns-server.com] example.com [{a|mx|any}]`	查询由"`dns-server.com`"提供服务的"`example.com`"域名的 DNS 记录："a", "mx" 或"any" 记录
`iptables -L -n`	查看包过滤
`ss -a`	找出所有打开的端口
`ss -l`	找出监听端口
`ss -l -t -n`	找出 TCP 监听端口（数字的）
`dlint example.com`	查询"`example.com`"的 DNS zone 信息

Table 5.4: 底层网络命令列表

提示
部分底层网络配置工具放在"`/usr/sbin/`"目录。你可以像"`/usr/sbin/ifconfig`"这样使用完整命令路径，或把"`/usr/sbin`"加到"`~/.bashrc`"文件列出的"`$PATH`"环境变量里。

5.6 网络优化

通用的网络优化超出了本文的范围。我提及消费等级连接相关的主题。

5.6.1 找出最佳 MTU

网络管理器通常会自动设置最佳最大传输单元 (MTU) 。

软件包	流行度	大小	说明
iftop	V:6, I:83	93	显示一个网络接口上的带宽使用信息
iperf	V:2, I:31	23	互联网协议带宽测量工具
ifstat	V:0.5, I:5.1	52	接口统计监控
bmon	V:2, I:19	154	便携式带宽监视器和网速估计工具
ethstatus	V:0.2, I:2.3	41	快速测量网络设备吞吐的脚本
bwm-ng	V:1.2, I:8.8	95	小巧简单的控制台带宽监测器
ethstats	V:0.06, I:0.35	21	基于控制台的以太网统计监视器

Table 5.5: 网络优化工具列表

在一些场景中，在用 ping(8) 加上"`-M do`" 选项发送各种大小的 ICMP 报文数据包进行实验后，你希望可以手动设置 MTU。MTU 是最大可完成没有 IP 分片的数据包大小加上 28 字节（IPv4）或 48 字节（IPv6）。下面的列子，发现 IPv4 连接的 MTU 是 1460，IPv6 连接的 MTU 是 1500。

```
$ ping -4 -c 1 -s $((1500-28)) -M do www.debian.org
PING  (149.20.4.15) 1472(1500) bytes of data.
ping: local error: message too long, mtu=1460

---  ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

$ ping -4 -c 1 -s $((1460-28)) -M do www.debian.org
PING  (130.89.148.77) 1432(1460) bytes of data.
1440 bytes from klecker-misc.debian.org (130.89.148.77): icmp_seq=1 ttl=50 time=325 ms

---  ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 325.318/325.318/325.318/0.000 ms
$ ping -6 -c 1 -s $((1500-48)) -M do www.debian.org
PING www.debian.org(mirror-csail.debian.org (2603:400a:ffff:bb8::801f:3e)) 1452 data bytes
1460 bytes from mirror-csail.debian.org (2603:400a:ffff:bb8::801f:3e): icmp_seq=1 ttl=47  ↩
    time=191 ms

--- www.debian.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 191.332/191.332/191.332/0.000 ms
```

这个过程是路径 MTU (PMTU) 发现 (RFC1191) ，tracepath(8) 命令能够自动完成这个。

网络环境	**MTU**	基本原理
拨号连接 (IP: PPP)	576	标准的
以太网连接 (IP: DHCP 或固定)	1500	默认标准值

Table 5.6: 最佳 MTU 值的基本指引方法

除了这些基本的指引方法外，你还应当知道下面的信息。

- 使用任何隧道方式 (VPN 等.) 的最佳 MTU 需要进一步减去它们上面的头部。
- MTU 值不应当超过通过实验验证的 PMTU 值。
- 当遇到其它限制的时候，较大的 MTU 值通常比较好。

最大分片大小 (MSS) 是另外一种衡量包大小的方法。MSS 和 MTU 的关系如下.

- 对于 IPv4，MSS = MTU - 40
- 对于 IPv6，MSS = MTU - 60

注意
基于 iptables(8) (参见第 5.7 节) 的优化，能够通过 MSS 来压缩包大小，路由器会用到 MMS 。参见 iptables(8) 中的”TCPMSS” .

5.6.2 WAN TCP 优化

现代大带宽和高延时的 WAN，TCP 吞吐量能够通过调整 TCP 缓冲大小的参数，在”TCP 调整” 里，来最大化。到目前为止，当前 Debian 默认设置能够很好的服务好我的 1G bps 光纤到户 LAN 连接。

5.7 Netfilter 网络过滤框架

Netfilter 使用 Linux 内核模块 (参见第 3.10 节) 提供状态防火墙和网络地址转换 (NAT) 框架。

软件包	流行度	大小	说明
nftables	V:301, I:889	189	封包过滤和网络地址转换管理工具（Netfilter）({ip,ip6,arp,eb}tables 的后续替代者)
iptables	V:402, I:635	2408	netfilter 管理工具 (iptables(8) 用于 IPv4, ip6tables(8) 用于 IPv6)
arptables	V:0.1, I:1.6	102	netfilter 管理工具 (arptables(8) 用于 ARP)
ebtables	V:14, I:22	276	netfilter 管理工具 (ebtables(8) 用于以太网桥)
iptstate	V:0.1, I:1.6	130	持续性监控 netfilter 状态 (和 top(1) 相似)
ufw	V:87, I:110	862	Uncomplicated Firewall (UFW) 是一个管理 netfilter 防火墙的程序
gufw	V:5, I:10	3682	Uncomplicated Firewall (UFW) 的图像用户界面
firewalld	V:16, I:22	2581	firewalld 是一个动态管理防火墙的程序，支持网络区域
firewall-config	V:0.8, I:3.0	1096	firewalld 图像用户界面
shorewall-init	V:0.13, I:0.31	88	Shoreline 防火墙初始化
shorewall	V:2.2, I:4.5	3090	Shoreline 防火墙, netfilter 配置文件生成器
shorewall-lite	V:0.03, I:0.05	71	Shoreline 防火墙, netfilter 配置文件生成器 (精简版)
shorewall6	V:0.7, I:1.2	1334	Shoreline 防火墙, netfilter 配置文件生成器 (IPv6 版本)
shorewall6-lite	V:0.01, I:0.01	71	Shoreline 防火墙, netfilter 配置文件生成器 (IPv6, 精简版)

Table 5.7: 防火墙工具列表

参见 netfilter.org ”nftables” 项目，`nftables` 软件包中新的用户空间命令 nft(8)。（Linux 内核 >= 3.13）

参见 netfilter.org ”iptables” 项目，在 `iptables` 软件包和同类软件包里面老的用户空间命令 iptables(8) 和同类命令。（Linux kernel >= 2.4.x）

你能从 shell 手工交互式的配置 netfilter，使用”`nft list ruleset > file`” 或 iptables-save(8) 保存当前状态, 当系统重启时，通过 init 脚本调用”`nft -f file`” 或 iptables-restore(8) 来恢复。

像 shorewall 这样的配置帮助脚本能够使这个过程变得更简单。

参见 Netfilter 文档上的文档 (或在”`/usr/share/doc/iptables/html/`” 里面的文档)。

- Linux Networking-concepts HOWTO
- Linux 2.4 Packet Filtering HOWTO
- Linux 2.4 NAT HOWTO
- nftables HOWTO

Chapter 6

网络应用

建立网络连接后（参加第 5 章），你可以运行各种网络应用。

提示
对于现代的 Debian 网络基础设施的具体说明，阅读 Debian 管理员手册——网络基础设施。

提示
在某些 ISP 下，如果你启用“两步验证”，你可能需要获取一个应用密码以从你的程序访问 POP 和 SMTP 服务。你也可能需要事先允许你的主机 IP 进行访问。

6.1 网页浏览器

有许多网页浏览器软件包，使用超文本传输协议（HTTP）访问远程内容。

软件包	流行度	大小	类型	网络浏览器说明
chromium	V:37, I:101	324490	X	Chromium，（来自 Google 的开源浏览器）
firefox	V:15, I:21	317809	, ,	Firefox，（来自 Mozilla 的开源浏览器，仅在 Debian Unstable 中可用）
firefox-esr	V:188, I:423	299668	, ,	Firefox ESR（Firefox 延长支持版本）
epiphany-browser	V:2.0, I:9.7	6737	, ,	GNOME，兼容 HIG，Epiphany
konqueror	V:26, I:110	8191	, ,	KDE，Konqueror
dillo	V:0.5, I:3.8	2203	, ,	Dillo，（基于 FLTK 的轻量级浏览器）
w3m	V:11, I:125	2853	文本	w3m
lynx	V:30, I:454	2031	, ,	Lynx
elinks	V:2, I:14	1839	, ,	ELinks
links	V:2, I:19	2344	, ,	Links（纯文本）
links2	V:0.8, I:9.2	5497	图像	Links（没有 X 的控制台图像）

Table 6.1: 网页浏览器列表

6.1.1 伪装用户代理字符串

为了访问一些过度限制的网站，你可能需要伪装网页浏览器程序返回的 User-Agent 字符串。参见：

- MDN Web Docs: userAgent
- Chrome Developers: Override the user agent string
- How to change your user agent
- How to Change User-Agent in Chrome, Firefox, Safari, and more
- How to Change Your Browser's User Agent Without Installing Any Extensions
- How to change the User Agent in Gnome Web (epiphany)

6.1.2 浏览器扩展

所有现代的 GUI（图形用户界面）浏览器支持基于 browser extension 的源代码，它在按 web extensions 变成标准化。

6.2 邮件系统

本章节关注于消费者级互联网连接的典型的移动工作站。



小心
如果你想设置邮件服务器来直接通过互联网交换邮件，你应该最好阅读一下这个基本文档。

6.2.1 电子邮件基础

电子邮件由三个部分组成，消息的信封，邮件头及邮件正文。

- SMTP 用电子邮件信封上的"To" 和"From" 信息来投递邮件。(信封上的"From" 信息也被叫做退回地址, 例如 From_ 等等)。
- 电子邮件头的"To" 和"From" 信息，显示在电子邮件客户端上. (在大部分情况下，这些信息是跟电子邮件信封一致，但并不全是这样。)
- 覆盖邮件头和正文数据的电子邮件消息格式被多用途互联网邮件扩展 (MIME) 扩展，从纯文本的 ASCII 到其它字符编码，包括作为附件的音频、视频、图像和应用程序。

功能全面的基于电子邮件客户端的 GUI 程序使用基于 GUI 的直观的配置，提供下列所有功能。

- 为了处理正文数据类型及其编码，它创建和使用多用途互联网邮件扩展 (MIME) 来解释邮件标头和邮件正文。
- 它使用旧的基础访问认证或现代的 OAuth 2.0 向 ISP（互联网服务提供商）的 SMTP 和 IMAP 服务器认证它自己。(对于 OAuth 2.0, 通过桌面环境设置来设置它，例如，"Settings" -> "Online Accounts".)
- 它发送消息到 ISP 的智能主机的 SMTP 服务监听的消息递交端口（587）。
- 从 TLS/IMAP4 端口（993）接收存储在 ISP 的服务器上的消息。
- 它能够通过他们的属性过滤邮件。
- 它能够提供额外的功能：联系人、日历、任务、备忘录。

软件包	流行度	大小	类型
evolution	V:27, I:227	521	X GUI 程序 (GNOME, groupware 套件)
thunderbird	V:39, I:100	304278	X GUI 程序 (GTK，Mozilla Thunderbird)
kmail	V:43, I:101	25384	X GUI 程序 (KDE)
mutt	V:11, I:78	7378	很有可能与 vim 一起使用的字符终端程序
mew	V:0.01, I:0.14	2319	(x)emacs 下的字符终端程序

Table 6.2: 邮件用户代理列表 (MUA)

6.2.2 现代邮件服务限制

现代邮件服务器有一些限制来最小化暴露滥用（不希望和未被要求的电子邮件）问题。

- 在消费者级的网络上运行 SMTP 服务器来直接可靠的发送邮件到远端主机是不现实的。
- 一个邮件能够被任何主机静悄悄的拒绝，即使路由到了目的地，除非它尽可能看起来是经过认证的。
- 期望单个智能主机可靠的发送不相关的源邮件地址到远程主机，这是不现实的。

这是因为：

- 从消费者级网络提供的主机连接到互联网的 SMTP 端口（25）已经被封锁了。
- 从互联网的 SMTP 端口（25）连接到消费者级网络提供的主机已经被封锁了。
- 从消费者级网络提供的主机发出到互联网的消息，只能够通过消息递交端口（587）发送。
- 像域名密钥识别邮件 (DKIM)、发信者策略框架 (SPF) 和基于域名的消息认证、报告和反应（DMARC）这样的反垃圾邮件技术广泛用于电子邮件过滤。
- 域名密钥识别邮件服务可能会用于你的通过 smarthost 的电子邮件发送。
- 智能主机可以在邮件头重写源电子邮件地址为你的邮件账户，来阻止电子邮件欺诈。

6.2.3 历史邮件服务端期望

一些在 Debian 上的程序，它们默认期望访问 `/usr/sbin/sendmail` 命令来发送邮件，或者从一个个性化设置的 UNIX 系统邮件服务器来发送邮件，实现历史的功能：

- 邮件是由纯文本文件创建。
- 邮件是由 `/usr/sbin/sendmail` 命令处理。
- 对于目的地址为同一主机，`/usr/sbin/sendmail` 命令进行邮件的本地分发，将邮件附在 `/var/mail/$username` 文件后。
  - 期望这个特征的命令：`apt-listchanges`, `cron`, `at`, ...
- 对于目的地址在远程主机，`/usr/sbin/sendmail` 命令远程传输邮件到目的主机，使用 SMTP 发现 DNS MX 记录。
  - 期望这个特征的命令：`popcon`, `reportbug`, `bts`, ...

6.2.4 邮件传输代理 (MTA)

在 Debian 12 Bookworm 后，在没有 mail transfer agent (MTA) 程序的情况下，Debian 移动工作站可以基于电子邮件客户端，配置为全功能的 GUI （图像用户界面）。

以往的 Debian 会安装某个 MTA 程序来支持期望 `/usr/sbin/sendmail` 命令的程序。移动工作站上这样的 MTA 必须和第 6.2.2 节第 6.2.3 节协同工作。

对于移动工作站，典型的 MTA 选择是 `exim4-daemon-light` 或 `postfix`，并选择类似这样的安装选项：“Mail sent by smarthost; received via SMTP or fetchmail”。这些是轻量 MTA 和”`/etc/aliases`” 匹配。

提示
配置 `exim4` 来发送互联网邮件，多个源电子邮件地址使用多个相应的智能主机，这是不寻常的。如果一些程序需要这样的能力，使用 `msmtp` 来设置他们，它比较容易来设置多个源电子邮件地址。然后给主 MTA 仅仅保留单个电子邮件地址。

软件包	流行度	大小	说明
`exim4-daemon-light`	V:197, I:206	1777	Exim4 邮件传输代理 (MTA：Debian 默认的)
`exim4-daemon-heavy`	V:4.5, I:4.8	1955	Exim4 邮件传输代理 (MTA：灵活的替代品)
`exim4-base`	V:201, I:211	1687	Exim4 文档 (文本) 和通用文件
`exim4-doc-html`	I:0.94	3809	Exim4 文档 (html)
`exim4-doc-info`	I:0.51	649	Exim4 文档 (info)
`postfix`	V:99, I:104	4277	Postfix 邮件传输代理 (MTA：安全的替代品)
`postfix-doc`	I:4.0	5030	Postfix 文档 (html+text)
`sasl2-bin`	V:4.5, I:9.4	370	Cyrus SASL API 实现 (实现 postfix SMTP 认证)
`cyrus-sasl2-doc`	I:0.68	2140	Cyrus SASL - 文档
`msmtp`	V:10, I:15	822	轻量 MTA
`msmtp-mta`	V:7.8, I:9.8	139	轻量 MTA (sendmail 兼容扩展到 `msmtp`)
`nullmailer`	V:7.2, I:7.6	483	部分功能 MTA，没有本地邮件
`ssmtp`	V:3.8, I:5.6	134	部分功能 MTA，没有本地邮件
`sendmail-bin`	V:9.4, I:9.7	1954	全功能 MTA(如果你已经对它熟悉)
`git-email`	V:0.4, I:8.6	1229	git-send-email(1) 发送系列补丁邮件程序

Table 6.3: 基础的邮件传输代理相关的软件包列表

6.2.4.1 exim4 的配置

对于那些通过 smarthost 的网络邮件，你应该按如下所示的 (重新) 配置 `exim4-*` 软件包。

```
$ sudo systemctl stop exim4
$ sudo dpkg-reconfigure exim4-config
```

配置”General type of mail configuration” 时，选择”mail sent by smarthost; received via SMTP or fetchmail”。

设置”System mail name:” 为默认的 FQDN (参见第 5.1.1 节)。

设置”IP-addresses to listen on for incoming SMTP connections:” 为默认的”127.0.0.1; ::1”。

”Other destinations for which mail is accepted:” 选项留空。

”Machines to relay mail for:” 选项留空。

设置”IP address or host name of the outgoing smarthost:” 为”smtp.hostname.dom:587”。

设置”Hide local mail name in outgoing mail?” 选项为”No”。(或者像第 6.2.4.3 节描述的那样使用 `/etc/email-addresses` 代替)

选择如下所示的其中一个来回答”Keep number of DNS-queries minimal (Dial-on-Demand)?”。

- ”No” 如果启动的时候，系统就连上了互联网。
- ”Yes” 如果启动的时候，系统没有连上互联网。

设置”Delivery method for local mail:” 选项为”mbox format in /var/mail/”。

”Split configuration into small files?:” 选项设为”Yes”。

通过修改”`/etc/exim4/passwd.client`” 文件，来创建用于 smarthost 的密码条目。

```
$ sudo vim /etc/exim4/passwd.client
 ...
$ cat /etc/exim4/passwd.client
^smtp.*\.hostname\.dom:username@hostname.dom:password
```

配置 exim4(8)，在”`/etc/default/exim4`” 文件中写入”`QUEUERUNNER='queueonly'`”，”`QUEUERUNNER='nodaemon'`” 等等，来最小化系统资源使用。（可选的）

通过如下所示的启动 `exim4`。

```
$ sudo systemctl start exim4
```

”`/etc/exim4/passwd.client`” 文件中的主机名不应该是别名，你应该按如下所示的检查真正的主机名。

```
$ host smtp.hostname.dom
smtp.hostname.dom is an alias for smtp99.hostname.dom.
smtp99.hostname.dom has address 123.234.123.89
```

我在”`/etc/exim4/passwd.client`” 文件中使用正则表达式来绕过别名问题。即使 ISP 更改了别名所指向的主机名，SMTP AUTH 还是可能工作的。

你能够通过如下所示的手动更新 `exim4` 配置:

- 更新”`/etc/exim4/`” 目录下的 `exim4` 配置文件。
  - 创建”`/etc/exim4/exim4.conf.localmacros`” 来设置宏命令和修改”`/etc/exim4/exim4.conf.template`” 文件。(没有分割的配置）
  - 在”`/etc/exim4/exim4.conf.d`” 子目录中创建新文件或编辑已存在的文件。(分割的配置）
- 运行”`systemctl reload exim4`”.



小心
如果 debconf 询问”Keep number of DNS-queries minimal (Dial-on-Demand)?” 这个问题时，选择了”No” (默认值)，那么启动 `exim4` 会花很长时间并且系统在启动的时候不会连接到互联网。

请阅读”`/usr/share/doc/exim4-base/README.Debian.gz`” 官方指导和 update-exim4.conf(8)。



警告
从所有的实践考虑，使用带 STARTTLS 的 SMTP 端口 587，或者 SMTPS (SSL 之上的 SMTP) 端口 465, 代替纯 SMTP 端口 25。

6.2.4.2 带有 **SASL** 的 **postfix** 配置

对于通过 smarthost 的网络邮件，你应该首先阅读 postfix 文档和关键的手册页。

命令	功能
postfix(1)	Postfix 控制程序
postconf(1)	Postfix 配置工具
postconf(5)	Postfix 配置参数
postmap(1)	Postfix 查找表维护
postalias(1)	Postfix 别名数据库维护

Table 6.4: 重要的 postfix 手册页列表

你应该按如下所示的 (重新) 配置 `postfix` 和 `sasl2-bin` 软件包。

```
$ sudo systemctl stop postfix
$ sudo dpkg-reconfigure postfix
```

选择”Internet with smarthost”。

设置”SMTP relay host (blank for none):” 为”`[smtp.hostname.dom]:587`” 并按如下所示配置。

```
$ sudo postconf -e 'smtp_sender_dependent_authentication = yes'
$ sudo postconf -e 'smtp_sasl_auth_enable = yes'
$ sudo postconf -e 'smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd'
$ sudo postconf -e 'smtp_sasl_type = cyrus'
$ sudo vim /etc/postfix/sasl_passwd
```

为 smarthost 创建密码条目。

```
$ cat /etc/postfix/sasl_passwd
[smtp.hostname.dom]:587     username:password
$ sudo postmap hush:/etc/postfix/sasl_passwd
```

通过如下所示的启动 `postfix`。

```
$ sudo systemctl start postfix
```

`dpkg-reconfigure` 会话中使用的”`[`” 和”`]`” 和”`/etc/postfix/sasl_passwd`” 确保不去检查 MX 记录而是直接使用指定的明确主机名。参见”`/usr/share/doc/postfix/html/SASL_README.html`” 里面的”Enabling SASL authentication in the Postfix SMTP client” 条目。

6.2.4.3 邮件地址配置

这里有一些用于邮件传输、投递和用户代理的邮件地址配置文件。

文件	功能	应用
`/etc/mailname`	用于 (外发) 邮件的默认主机名	Debian 专用的，mailname(5)
`/etc/email-addresses`	用于外发邮件的主机名伪装	exim(8) 专用的，exim4-config_files(5)
`/etc/postfix/generic`	用于外发邮件的主机名伪装	postfix(1) 专用的，postmap(1) 命令执行后激活。
`/etc/aliases`	用于接收邮件的账户别名	通用的，newaliases(1) 命令执行后激活。

Table 6.5: 与邮件地址相关的配置文件列表

”`/etc/mailname`” 文件中的 **mailname** 通常是全称域名 (FQDN)，这个全程域名将会被解析成主机的 IP 地址。对于没有可解析成 IP 地址的主机名的移动工作站，设置 **mailname** 为”`hostname -f`” 的值。(这对于 `exim4-*` 和 `postfix` 都是安全有效的选择。)

提示
”/etc/mailname” 中的内容被许多非 MTA 程序用作它们的默认行为。对于 mutt，在 ~/muttrc 文件中设置”hostname” 和”from” 变量来覆盖 **mailname** 值。对于 devscripts 软件包的程序，例如 bts(1) 和 dch(1)，导出环境变量”$DEBFULLNAME” 和”$DEBEMAIL” 的值来覆盖它。

提示
popularity-contest 软件包一般以 FQDN 形式的 root 账户发送邮件。你需要像 /usr/share/popularity-contest/default.conf 文件中描述的那样去设置 /etc/popularity-contest.conf 文件中的 MAILFROM 值。否则，你的邮件会被 smarthost SMTP 服务器拒绝。尽管这些过程很乏味，这种方法比为所有通过 MTA 并且是以 root 用户发送的邮件重写源地址更安全。这也可以被其他守护进程或者是 cron 脚本使用。

当设置 **mailname** 为”hostname -f” 的值时，通过 MTA 的源邮件地址的伪装可以通过如下所示的来实现。

- 用于 exim4(8) 的”/etc/email-addresses” 文件，exim4-config_files(5) 手册页中有关于它的解释
- 用于 postfix(1) 的”/etc/postfix/generic” 文件，generic(5) 手册页中有关于它的解释

对于 postfix，接下来的额外步骤需要执行。

```
# postmap hash:/etc/postfix/generic
# postconf -e 'smtp_generic_maps = hash:/etc/postfix/generic'
# postfix reload
```

你能够通过如下所示的来测试邮件地址配置。

- exim(8) 用 -brw, -bf, -bF, -bV, ... 选项
- postmap(1) 用 -q 选项。

提示
Exim 带有一些有用的程序，例如 exiqgrep(8) 和 exipick(8)。参见”dpkg -L exim4-base|grep man8/” 来获得可用的命令。

6.2.4.4 基础 MTA 操作

这里有一些基础的 MTA 操作。有一些可能会通过 sendmail(1) 的兼容性接口来实现。

提示
往”/etc/ppp/ip-up.d/*” 里写一个刷新所有邮件的脚本会是个不错的主意。

6.3 服务器远程访问和工具 (SSH)

Secure SHell (SSH) 是因特网上的安全连接方式。在 Debian 里面，有一个叫 OpenSSH 的免费 SSH 版本，在 openssh-client 和 openssh-server 包里。

对于用户来讲, ssh(1) 功能比 telnet(1) 更加智能和安全. 不像 telnet 命令, ssh 命令不会在遇到 telnet 的退出字符 (初始默认是 CTRL-]) 时停止.

虽然 shellinabox 不是一个 SSH 程序，它列在这里作为远程终端访问的一个有趣的替代。

连接到远程 X 客户端程序，参见：第 7.9 节。

exim 命令	**postfix** 命令	说明
`sendmail`	`sendmail`	从标准输入读取邮件并且安排投递 (`-bm`)
`mailq`	`mailq`	列出带有状态和队列 ID 的邮件队列 (`-bq`)
`newaliases`	`newaliases`	初始化别名数据库 (`-I`)
`exim4 -q`	`postqueue -f`	刷新等待邮件 (`-q`)
`exim4 -qf`	`postsuper -r ALL deferred; postqueue -f`	刷新所有邮件
`exim4 -qff`	`postsuper -r ALL; postqueue -f`	刷新甚至已经冻结的邮件
`exim4 -Mg queue_id`	`postsuper -h queue_id`	通过邮件的队列 ID 来冻结它
`exim4 -Mrm queue_id`	`postsuper -d queue_id`	通过邮件的队列 ID 来移除它
N/A	`postsuper -d ALL`	移除所有邮件

Table 6.6: 基础 MTA 操作列表

软件包	流行度	大小	工具	说明
`openssh-client`	V:709, I:997	3611	ssh(1)	SSH 客户端
`openssh-server`	V:781, I:824	3615	sshd(8)	SSH 服务端
`ssh-askpass`	I:15	103	ssh-askpass(1)	请求用户输入 ssh-add 密码 (简单的 X)
`ssh-askpass-gnome`	V:0.4, I:2.9	46	gnome-ssh-askpass(1)	请求用户输入 ssh-add 密码 (GNOME)
`ssh-askpass-fullscreen`	V:0.08, I:0.36	41	ssh-askpass-fullscreen(1)	请求用户输入 ssh-add 密码 (GNOME)，有更好看的界面
`shellinabox`	V:0.60, I:0.88	528	shellinaboxd(1)	浏览器访问 VT100 终端模拟器网页服务器

Table 6.7: 服务器远程访问和工具列表



小心
如果你的 SSH 是从因特网来访问，参见第 4.6.3 节。

提示
请使用 screen(1) 程序来让远程 shell 在中断的连接上存活 (参见第 9.1.2 节).

6.3.1 SSH 基础

OpenSSH SSH 后台守护进程（daemon）只支持 SSH 2 协议。

请阅读"`/usr/share/doc/openssh-client/README.Debian.gz`"、ssh(1)、sshd(8)、ssh-keygen(1)、ssh-add(1) 和 ssh-agent(1).



警告
如果想要运行 OpenSSH 服务，"`/etc/ssh/sshd_not_to_be_run`" 必须不存在。
不要打开基于 rhost 的认证 (`/etc/ssh/sshd_config` 中的 `HostbasedAuthentication`)。

配置文件	配置文件描述
`/etc/ssh/ssh_config`	SSH 客户端默认, 参见 ssh_config(5)
`/etc/ssh/sshd_config`	SSH 服务端默认, 参见 sshd_config(5)
`~/.ssh/authorized_keys`	该账户连接到这个服务器上的客户端使用的默认 SSH 公钥
`~/.ssh/id_rsa`	用户的 SSH-2 RSA 私钥
`~/.ssh/id_`*`key-type-name`*	用户的 SSH-2 密钥，*key-type-name* 为 `ecdsa`、`ed25519` 等

Table 6.8: SSH 配置文件列表

从客户端启动一个 ssh(1) 连接.

命令	说明
`ssh username@hostname.domain.ext`	使用默认模式连接
`ssh -v username@hostname.domain.ext`	有详细信息的默认连接模式
`ssh -o PreferredAuthentications=password username@hostname.domain.ext`	SSH 2 版本，强制使用密码
`ssh -t username@hostname.domain.ext passwd`	在远程主机上运行 `passwd` 命令来更新密码

Table 6.9: SSH 客户端启动例子列表

6.3.2 远程主机上的用户名

如果你在本地和远程主机上使用相同的用户名，你能够省略输入"`username@`"。

即使在本地和远程主机使用不同的用户名，你可以使用"`~/.ssh/config`" 来省略输入用户名. 对于 Debian Salsa 服务器，使用账户名"`foo-guest`", 你可以设置"`~/.ssh/config`" 包含下面的内容。

```
Host salsa.debian.org people.debian.org
User foo-guest
```

6.3.3 免密码远程连接

使用"PubkeyAuthentication" (SSH-2 协议)，人们可以避免记住远程系统的密码.

在远程系统的"/etc/ssh/sshd_config" 里，设置相应的条目, "PubkeyAuthentication yes"。

在本地生成授权秘钥对，并安装公钥到远程系统。

```
$ ssh-keygen -t rsa
$ cat .ssh/id_rsa.pub | ssh user1@remote "cat - >>.ssh/authorized_keys"
```

你可以在"~/.ssh/authorized_keys" 里给条目增加选项来限制主机和运行特定的命令。参见 sshd(8)"AUTHORIZED_KEYS FILE FORMAT" 。

6.3.4 处理其它 **SSH** 客户端

其它平台上有一些免费的 SSH 客户端。

环境	免费 **SSH** 程序
Windows	puTTY (PuTTY：一个自由的 SSH 、Telnet 客户端) (GPL)
Windows (cygwin)	cygwin 里的 SSH(Cygwin: Get that Linux feeling - on Windows) (GPL)
Mac OS X	OpenSSH; 在终端应用中使用 ssh (GPL)

Table 6.10: 其它平台上免费 SSH 客户端列表

6.3.5 建立 **ssh** 代理

用密码来保护你的 SSH 认证私钥是安全的。如果密码没有设置，使用"ssh-keygen -p" 来设置。

把你的公钥 (比如："~/.ssh/id_rsa.pub") 放到远程主机的"~/.ssh/authorized_keys"，这个远程主机使用上面描述的基于密码的连接方式。

```
$ ssh-agent bash
$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/username/.ssh/id_rsa:
Identity added: /home/username/.ssh/id_rsa (/home/username/.ssh/id_rsa)
```

从这里执行接下来的命令，就不再需要密码。

```
$ scp foo username@remote.host:foo
```

按 ^D 来终结 ssh 代理会话。

对于 X 服务端，通常的 Debian 启动脚本会作为父进程执行 ssh-agent。所以你只需要执行一次 ssh-add。进一步的信息，请阅读 ssh-agent(1) 和 ssh-add(1).

6.3.6 从远程主机发送邮件

如果你在一个正确设置了 DNS 的服务器上有一个 SSH shell 账号，你能够将在你本地工作站上生成的邮件，作为远程服务器上的邮件，真正的从远程服务器上发送。

```
$ ssh username@example.org /usr/sbin/sendmail -bm -ti -f "username@example.org" < mail_data ↩
    .txt
```

6.3.7 SMTP/POP3 隧道的端口转发

通过 ssh 建立一个这样的管道连接，从 localhost 的 4025 端口到 remote-server 的 25 端口，并从 localhost 的 4110 端口到 remote-server 的 110 端口，请在本机执行如下命令.

```
# ssh -q -L 4025:remote-server:25 4110:remote-server:110 username@remote-server
```

这是跨越因特网建立 SMTP/POP3 服务连接的安全方法。在远程主机"/etc/ssh/sshd_config" 里设置"AllowTcpForwar
条目为"yes".

6.3.8 怎样通过 SSH 关闭远程系统

你可以使用 at(1) 命令 (参见第 9.4.13 节) 来从 SSH 终端里保护"shutdown -h now" (参见第 1.1.8 节) 操作过程。

```
# echo "shutdown -h now" | at now
```

在 screen(1) (参见第 9.1.2 节) 会话里运行"shutdown -h now"，是另外一个方法来做这同样的事情。

6.3.9 SSH 故障排查

如果你遇到问题，检查配置文件的权限并用"-v" 选项运行 ssh。

如果你是 root 账户，并有使用防火墙，使用"-p" 选项; 这可以避免使用 1 —1023 之间的服务端口.

如果 ssh 连接到远程站点突然停止工作，这也许是系统管理员胡乱操作的结果，可能是在系统维护时改变了"host_key". 在确认这个情况后，并且没有人试图用聪明的黑客技术来篡改远程主机，你可以在本机"~/.ssh/known_ho
里删除"host_key" 条目来重新获得连接。

6.4 打印服务和工具

在老的类 Unix 系统中，BSD Line printer daemon(lpd) 行打印机后台守护曾经是标准。传统的自由软件的标准打印输出格式是 PostScript (PS)。为了能够打印到非 PostScript 打印机，需要将一些过滤器系统和 Ghostscript 一道使用。参见第 11.4.1 节。

在现代的 Debian 系统中，Common UNIX Printing System 通用 UNIX 打印系统是事实上的标准。现代自由软件的标准打印输出格式是 Portable Document Format (PDF) 可移植文件格式。

CUPS 使用 Internet Printing Protocol 互联网打印协议 (IPP)。IPP 是具备双向通信能力的跨平台远程打印的事实标准。

幸亏有 CUPS 系统的文件格式依赖自动转化特征，简单的发送任何数据到 lpr 命令，都将产生期望的打印输出。(在 CUPS 里, lpr 能够通过安装 cups-bsd 软件包来获取.)

Debian 系统有一些不错的软件包用于打印服务和作为打印工具。

提示
你可以让你的 web 浏览器访问"http://localhost:631/" 来配置 CUPS 系统。

软件包	流行度	大小	端口	说明
lpr	V:1.9, I:2.2	378	printer (515)	BSD lpr/lpd (线性打印机后台守护进程 daemon)
cups	V:141, I:444	1102	IPP (631)	互联网打印 CUPS 服务器
cups-client	V:149, I:455	432	, ,	用于 CUPS 的 System V 打印机命令: lp(1), lpstat(1), lpoptions(1), cancel(1), lpmove(8), lpinfo(8), lpadmin(8), ⋯
cups-bsd	V:33, I:178	131	, ,	用于 CUPS 的 BSD 打印机命令: lpr(1), lpq(1), lprm(1), lpc(8)
printer-driver-gutenprint	V:10, I:49	1117	没有使用	CUPS 打印机驱动

Table 6.11: 打印服务和工具列表

6.5 其它网络应用服务

这里是其它网络应用服务。

软件包	流行度	大小	协议	说明
inetutils-telnetd	V:0.2, I:1.3	200	TELNET	TELNET 服务
nfs-kernel-server	V:46, I:53	886	NFS	Unix 文件共享
samba	V:112, I:123	5190	SMB	Windows 文件和打印共享
netatalk	V:0.57, I:0.79	933	ATP	Apple/Mac 文件和打印共享 (AppleTalk)
proftpd-core	V:7.6, I:8.4	5041	FTP	通用文件下载
apache2	V:168, I:205	590	HTTP	通用 web 服务器
squid	V:8.8, I:9.3	9585	, ,	通用 web 代理服务器
bind9	V:32, I:35	905	DNS	其它主机的 IP 地址
kea	I:0.51	245	DHCP	客户端自身的 IP 地址

Table 6.12: 其它网络应用服务列表

通用互联网文件系统协议 (CIFS) 和服务消息块 (SMB) 协议一样，被微软 Windows 广泛应用。

提示
参见第 4.5.2 节服务系统集成。

提示
主机名解析通常由 DNS 服务提供. 对于由 DHCP 动态分配的主机 IP 地址, 动态 DNS 能够使用 `bind9` 和 `kea` 建立主机名解析，Debian wiki 的 DDNS 页有说明.

提示
使用 `squid` 之类的代理服务器，和使用 Debian 文档库的完全本地镜像服务器相比，能够大量节省带宽。

6.6 其它网络应用客户端

这里是其它网络应用客户端。

软件包	流行度	大小	协议	说明
netcat-traditional	V:61, I:905	139	TCP/IP	TCP/IP 瑞士军刀
netcat-openbsd	V:23, I:118	105	TCP/IP	支持 IPv6、代理和 Unix 套接字的 TCP/IP 瑞士军刀
openssl	V:853, I:996	2564	SSL	安全套接字层 (SSL) 二进制和相关的加密工具
stunnel4	V:7, I:10	24	, ,	通用 SSL 封装
inetutils-telnet	V:48, I:778	282	TELNET	TELNET 客户端
nfs-common	V:151, I:188	1259	NFS	Unix 文件共享
smbclient	V:29, I:204	2117	SMB	微软 Windows 文件和打印共享客户端
cifs-utils	V:34, I:117	369	, ,	远程微软 Windows 文件系统挂载和卸载命令
wget	V:186, I:982	3784	HTTP 和 FTP	web 下载工具
curl	V:296, I:739	526	, ,	, ,
transmission-gtk	V:10, I:135	6639	BitTorrent	BitTorrent 客户端 (GTK)
transmission-qt	V:0.6, I:2.5	6522	, ,	BitTorrent 客户端 (Qt)
ktorrent	V:1.3, I:4.8	5031	, ,	BitTorrent 客户端 (Qt)
qbittorrent	V:8, I:23	16666	, ,	BitTorrent 客户端 (Qt)
bind9-host	V:143, I:940	137	DNS	来自 bind9 的 host(1), "Priority: standard"
bind9-dnsutils	V:116, I:836	521	, ,	来自 bind 的 dig(1), "Priority: standard"
ldap-utils	V:11, I:56	791	LDAP	从 LDAP 服务获取数据

Table 6.13: 网络应用客户端列表

6.7 系统后台守护进程（daemon）诊断

`telnet` 程序能够手工连接到系统后台守护进程（daemon），并进行诊断。

测试纯 POP3 服务，尝试用下面的操作

```
$ telnet mail.ispname.net pop3
```

部分 ISP 提供 TLS/SSL 加密的 POP3 服务，为了测试它，你需要用到 `telnet-ssl` 包里支持 TLS/SSL 的 `telnet` 客户端，或 `openssl` 软件包。

```
$ telnet -z ssl pop.gmail.com 995
```

```
$ openssl s_client -connect pop.gmail.com:995
```

下面的 RFCs 提供每一个系统后台守护进程（daemon）所需要的知识。

RFC	说明
rfc1939 和 rfc2449	POP3 服务
rfc3501	IMAP4 服务
rfc2821 (rfc821)	SMTP 服务
rfc2822 (rfc822)	邮件文件格式
rfc2045	多用途互联网邮件扩展 (MIME)
rfc819	DNS 服务
rfc2616	HTTP 服务
rfc2396	URI 定义

Table 6.14: 常用 RFC 列表

在”`/etc/services`” 里，描述了端口用途.

Chapter 7

GUI（图形用户界面）系统

7.1 GUI（图形用户界面）桌面环境

在 Debian 系统上，有几个功能全面的 GUI 桌面环境选择。

任务软件包	流行度	大小	说明
task-gnome-desktop	I:191	9	GNOME 桌面环境
task-xfce-desktop	I:88	9	Xfce 桌面环境
task-kde-desktop	I:93	6	KDE Plasma 桌面环境
task-mate-desktop	I:30	9	MATE 桌面环境
task-cinnamon-desktop	I:35	9	Cinnamon 桌面环境
task-lxde-desktop	I:18	9	LXDE 桌面环境
task-lxqt-desktop	I:16	9	LXQt 桌面环境
task-gnome-flashback-desktop	I:4	6	GNOME Flashback 桌面环境

Table 7.1: 桌面环境列表

提示
选择的任务元软件包的依赖软件包，在 Debian 非稳定版/测试版环境下，由于最新的软件包变迁状态，可能没有及时同步。对于 `task-gnome-desktop`，你可以按下面的方法调整软件包选择：

- 用 `sudo aptitude -u` 启动 aptitude(8)。
- 移动光标到”Tasks” 并按回车键。
- 移动光标到”End-user” 并按回车键。
- 移动光标到”GNOME” 并按回车键。
- 移动光标到 `task-gnome-desktop` 并按回车键。
- 移动光标到”Depends” 并按”m”（手工选择）。
- 移动光标到”Recommends” 并按”m” （手工选择）。
- 移动光标到”`task-gnome-desktop` 并按”-”。（删除）
- 调整选择的软件包，并删除造成软件包冲突的问题软件包。
- 按”g” 来开始安装。

本章将大部分关注 Debian 默认的桌面环境：`task-gnome-desktop`，在 wayland 上提供 GNOME。

7.2 GUI（图形用户界面）通信协议

在 GNOME 桌面使用的 GUI 通信协议可以为：

- Wayland (服务端显示协议) (原生)
- X 窗口系统核心协议 (通过 `xwayland`)

请查看 freedesktop.org 站点来了解 Wayland 架构和 X 窗口架构是如何不同。
从用户的观点，不同能够被通俗的概况为：

- Wayland 是在同一个主机上的 GUI 通信协议：新、简单、快速，不需要 setuid root 二进制
- X Window 是一个具备网络功能的 GUI 通信协议：传统、复杂、慢，需要 setuid root 二进制

对于使用 Wayland 协议的应用，由 VNC 或 RDP 来支持从一个远程主机上访问它们显示的内容。参见第 7.8 节
现代 X 服务器具有 MIT 共享内存扩展 ，他们和本地 X 客户端通过本地共享内存进行通讯. 这就绕过了网络透明的 Xlib 进程间通讯通道，获得了性能。这个情况，也是创建 Wayland 作为本地 GUI 通信协议的背景。
使用从 GNOME 终端启动的 `xeyes` 程序，你能够检查每个 GUI（图形用户界面）应用程序使用的 GUI 通信协议。

```
$ xeyes
```

- 如果鼠标是在使用 Wayland 服务端显示协议的应用程序上，比如“GNOME 终端”，眼睛不会跟随鼠标移动。
- 如果鼠标是在使用 X 窗口系统核心协议的应用程序上，比如“xterm”，眼睛会跟随鼠标移动，暴露出不是那么孤立的 X 窗口架构的特性。

到 2021 年 4 月，许多流行的 GUI 应用程序，比如 GNOME 和 LibreOffice (LO) 已经被移植到了 Wayland 服务端显示协议。我发现 `xterm`, `gitk`, `chromium`, `firefox`, `gimp`, `dia` 和 KDE 应用程序任然使用 X 窗口系统核心协议。

注意
对于 Wayland 之上的 xwayland 或原生的 X 窗口系统，这两个上面的旧的 X 服务端配置文件"/etc/X11/xorg.conf" 不应当在系统上存在。显卡和输入设备目前是由内核的 DRM、KMS 和 udev 配置。原生的 X 服务端已经重写来使用它们。参见 Linux 内核文档的"modedb default video mode support"。

7.3 GUI（图形用户界面）架构

这里是 Wayland 环境上用于 GNOME 的著名的 GUI 架构软件包。

软件包	流行度	软件包大小	说明
mutter	V:1, I:21	218	GNOME 的 mutter 窗口管理器 **[auto]**
xwayland	V:256, I:332	2653	运行在 wayland 之上的一个 X 服务端 **[auto]**
gnome-remote-desktop	V:154, I:239	2743	使用 PipeWire 的 GNOME 远程桌面后台守护进程（daemon） **[auto]**
gnome-tweaks	V:17, I:230	1145	GNOME 的高级配置设置
gnome-shell-extension-prefs	V:7, I:109	33	启用/禁用 GNOME 外壳扩展的工具

Table 7.2: 著名的 GUI 架构软件包列表

这里，"**[auto]**" 表示这些软件包在 task-gnome-desktop 安装时会自动安装。

提示
gnome-tweaks 是一个不可缺少的配置工具。例如：

- 你能强制调整声音音量，从"General（普通）"到"Over-Amplification（过分放大）"。
- 你能够强迫"Caps" 键变成"Esc" 键，从"Keyboard & Mouse" -> "Keyboard" -> "Additional Layout Option".

提示
GNOME 桌面环境的详细特征能够使用工具来配置，在按下 Super-键后，通过选择"settings", "tweaks" 或"extensions" 来启动配置。

7.4 GUI（图形用户界面）应用

现在在 Debian 上，有许多有用的 GUI 应用存在。如果在 GNOME 桌面环境中没有相应功能的软件，那么安装例如 scribus（KDE）这样的软件包是完全可以接受的。但安装过多功能重复的软件包，会使你的系统凌乱。

这里是一份捕获我眼球的 GUI（图形用户界面）程序列表。

软件包	流行度	软件包大小	类型	说明
evolution	V:27, I:227	521	GNOME	个人信息管理（群组软体和电子邮件）
thunderbird	V:39, I:100	304278	GTK	电子邮件客户端 (Mozilla Thunderbird（雷鸟）)
kontact	V:1.0, I:9.7	2416	KDE	个人信息管理（群组软体和电子邮件）
libreoffice-writer	V:92, I:417	34243	LO	文字处理软件
abiword	V:0.8, I:4.5	3596	GNOME	文字处理软件
calligrawords	V:0.3, I:2.6	7144	KDE	文字处理软件
scribus	V:1, I:12	32942	KDE	编辑 PDF 文件的 desktop publishing 编辑器
glabels	V:0.4, I:2.4	1295	GNOME	标签编辑器
libreoffice-calc	V:89, I:414	29622	LO	电子表格
gnumeric	V:2.8, I:9.7	10053	GNOME	电子表格
calligrasheets	V:0.2, I:1.6	14014	KDE	电子表格
libreoffice-impress	V:78, I:412	2501	LO	演示文稿
calligrastage	V:0.1, I:1.6	6237	KDE	演示文稿
libreoffice-base	V:18, I:65	5136	LO	数据库管理
kexi	V:0.04, I:0.73	7565	KDE	数据库管理
libreoffice-draw	V:78, I:412	11424	LO	矢量图形编辑器（绘图）
inkscape	V:12, I:73	113357	GNOME	矢量图形编辑器（绘图）
karbon	V:0.2, I:2.0	4029	KDE	矢量图形编辑器（绘图）
dia	V:1, I:16	3846	GTK	流程图和示意图编辑器
gimp	V:40, I:206	33386	GTK	位图图形编辑器 (绘图)
shotwell	V:15, I:244	6334	GTK	数码照片管理器
digikam	V:1.5, I:8.2	333	KDE	数码照片管理器
darktable	V:3, I:10	36092	GTK	摄影师的虚拟灯台和暗房
gnome-video-trimmer	V:0.16, I:0.97	1665	GNOME	无损剪切视频文件
kdenlive	V:3, I:21	22233	KDE	非线性视频编辑器
shotcut	V:1.2, I:7.5	8488	Qt 应用	非线性视频编辑器
openshot-qt	V:0.8, I:8.3	196007	Qt 应用	非线性视频编辑器
flowblade	V:0.2, I:2.1	42213	GTK	非线性视频编辑器
planner	V:0.2, I:4.3	1400	GNOME	项目管理
calligraplan	V:0.1, I:2.9	23545	KDE	项目管理
gnucash	V:2.0, I:6.4	29447	GNOME	个人会计
homebank	V:0.3, I:1.5	3277	GTK	个人会计
kmymoney	V:0.4, I:1.8	19313	KDE	个人会计
frescobaldi	V:0.3, I:1.4	14150	Qt 应用	LilyPond 乐谱音乐文本编辑器
librecad	V:1, I:13	12361	Qt 应用	计算机辅助设计 (CAD) 系统 (2D)
freecad	V:1, I:20	112	Qt 应用	计算机辅助设计 (CAD) 系统 (3D)
kicad	V:3, I:14	226336	GTK	电路图和 PCB 设计软件
xsane	V:9, I:127	1512	GTK	扫描仪前段
libreoffice-math	V:73, I:415	1980	LO	数学方程/公式编辑器
calibre	V:7, I:24	76289	KDE	电子书转换器和库管理
fbreader	V:0.7, I:5.2	4204	GTK	电子书阅读器
evince	V:76, I:280	953	GNOME	文档（pdf）阅读器
okular	V:43, I:128	4544	KDE	文档（pdf）阅读器
x11-apps	V:32, I:440	2461	单纯的 X 应用	xeyes(1) 等。

7.5 用户目录

用户目录，比如”~/Desktop”、”~/Documents”……，其默认的名字，使用系统安装时桌面环境的语言。你能够把它们重置为英语：

```
$ LANGUAGE=C xdg-user-dirs-update --force
```

然后你手工移到所有数据到新的目录。参加 xdg-user-dirs-update(1)。

通过编辑”~/.config/user-dirs.dirs”，你也能够把它们设置为任何名字。参见 user-dirs.dirs(5)。

7.6 字体

对于 Debian 的用户，有许多有用的矢量字体存在。用户关注是怎样避免冗余，怎样配置禁用部分已经安装的字体。此外，无用的字体选择可以搞乱你的 GUI（图形用户界面）应用程序菜单。

Debian 系统使用 FreeType 2.0 库来栅格化许多矢量字体格式，用于屏幕和打印：

- Type 1 (PostScript) 字体使用三次贝塞尔曲线 (差不多废弃的格式)
- TrueType 字体使用二次贝塞尔曲线 (好的选择格式)
- OpenType 字体使用三次贝塞尔曲线 (最佳选择格式)

7.6.1 基础字体

下面的编撰的表格希望帮助用户选择适当的矢量字体，并清楚的理解排版指标兼容（metric compatibility）和字形覆盖。大部分字体覆盖了所有拉丁、希腊和 Cyril 字符。最终选择的激活字体也受你的审美观影响。这些字体能够被用于屏幕显示和纸张打印。

这里：

- ”MCM” 表示” 与微软提供的字体是排版指标兼容的”
- ”MCMATC” 表示和” 微软提供的字体：Arial, Times New Roman, Courier New 排版指标兼容”
- ”MCAHTC” 表示” 和 Adobe 提供的字体: Helvetica, Times, Courier 排版指标兼容”
- 在字体类型列的数字表示对相同磅数的字体与 M 字重的相对粗细程度（译注：M 表示 Medium 适中，字体粗细程度的适中值）。
- 在 mono 字体类型列中的”P” 表示用于编程中，能够清晰的区分”0”/”O” 和”1”/”I”/”l”。
- `ttf-mscorefonts-installer` 软件包下载微软的”Core fonts for the Web” 并安装 Arial, Times New Roman, Courier New, Verdana, ... 。这些安装的字体数据，是非开源的数据。

许多开源的拉丁字体，有 URW Nimbus 家族或 Bitstream Vera 的血统痕迹。

提示
如果你的语言环境所需要的字体没有在上面的字体中涵盖，请使用 aptitude 在”Tasks” -> ”Localization” 下面检查任务软件包列表。字体软件包作为”Depends:” 或”Recommends:” 列出，在本地化任务软件包里面是首要候选软件包。

软件包	流行度	大小	**sans**	**serif**	**mono**	字体注释
fonts-cantarell	V:173, I:284	224	59	-	-	Cantarell (GNOME 3, 显示)
fonts-noto	I:146	30	61	63	40	Noto fonts (Google, 有 CJK 的多语言)
fonts-dejavu	I:381	34	58	68	40	DejaVu (GNOME 2, MCM:Verdana, 扩展 Bitstream Vera)
fonts-liberation2	V:36, I:143	15	56	60	40	Liberation 字体用于 LibreOffice (Red Hat, MCMATC)
fonts-croscore	V:19, I:35	5260	56	60	40	Chrome OS: Arimo, Tinos 和 Cousine (Google, MCMATC)
fonts-crosextra-carlito	V:21, I:88	2696	57	-	-	Chrome 操作系统: Carlito (Google, MCM:Calibri)
fonts-crosextra-caladea	V:13, I:81	347	-	55	-	Chrome 操作系统: Caladea (Google, MCM:Cambria) (只有拉丁字符)
fonts-freefont-ttf	V:77, I:200	14461	57	59	40	GNU FreeFont (扩展 URW Nimbus)
fonts-quicksand	V:210, I:450	392	56	-	-	Debian 任务桌面, Quicksand (显示，只有拉丁字符)
fonts-hack	V:33, I:135	2509	-	-	40 P	给源代码设计的一个字体 Hack (Facebook)
fonts-sil-gentiumplus	V:1, I:27	14345	-	54	-	Gentium SIL
fonts-sil-charis	V:1, I:27	6704	-	59	-	Charis SIL
fonts-urw-base35	V:187, I:533	15558	56	60	40	URW Nimbus (Nimbus Sans, Roman No. 9 L, Mono L, MCAHTC)
fonts-ubuntu	V:2.1, I:4.4	4312	58	-	33 P	Ubuntu 字体 (显示)
fonts-terminus	V:0.3, I:3.8	451	-	-	33	Cool retro terminal fonts
ttf-mscorefonts-installer	V:1, I:40	85	56?	60	40	下载微软非开源字体（见下）

Table 7.4: 著名的TrueType 和OpenType 字体列表

软件包	流行度	大小	说明
`libfreetype6`	V:582, I:996	1098	FreeType 字体栅格化库
`libfontconfig1`	V:580, I:801	344	Fontconfig 字体配置库
`fontconfig`	V:477, I:688	415	`fc-*`: Fontconfig 命令行命令
`font-manager`	V:2.2, I:6.3	1139	Font 管理器: Fontconfig GUI（图形用户界面）命令
`nautilus-font-manager`	V:0.09, I:0.28	43	Font 管理器的 Nautilus 扩展

Table 7.5: 著名的字体环境和相关软件包列表

7.6.2 字体栅格化

Debian 使用 FreeType 来栅格化字体。它的字体选择架构由 Fontconfig 字体配置库提供。

> 提示
> 一些字体软件包，比如说 `fonts-noto*`，会安装太多的字体。你可以保持某些字体软件包的安装，但在通常使用的情况下禁用。由于 Han unification 中日韩统一表意文字，一些 Unicode 码点被期望有多个字形，不希望的字形变体会被没有配置的 Fontconfig 库选择。一个最令人讨厌的情形是在 CJK 中日韩国家中的"U+3001 IDEOGRAPHIC COMMA" 和"U+3002 IDEOGRAPHIC FULL STOP"。你能够使用 GUI（图形用户界面）字体管理器 (font-manager) 简单的配置存在的字体来避免这个问题情形。

你也可以从命令行列出字体配置状态。

- 使用"`fc-match(1)`"查看 fontconfig 的默认字体
- 使用"`fc-list(1)`"查看所有可用的 fontconfig 字体

你能够从文本编辑器配置字体配置状态，但这是琐碎的。参见 fonts.conf(5)。

7.7 沙盒

Linux 上大部分 GUI（图形用户界面）应用在非 Debian 的源上，是以二进制格式存在。

- AppImage -- 任何地方运行的 Linux 应用
- FLATHUB -- Linux 应用，就是这里
- snapcraft -- Linux 应用商店

> 
> 警告
> 从这些站点来的二进制软件包，有可能包括私有的非开源软件。

对使用 Debian 的自由软件的狂热爱好者，这些二进制格式的分发，有一些存在的理由。因为这能够得到一个干净的库集合，由 Debian 提供的库和由每个应用程序相应的上游开发者使用的库，独立开来。

运行外部二进制的固有风险，能够使用沙盒环境减少，它有现代 Linux 安全特性的手段。（参见第 4.7.5 节）。

- 对于 AppImage 和一些上游站点来的二进制，在手工配置后的 firejail 里运行。
- 对于从 FLATHUB 来的二进制，在 Flatpak 里运行它们。（不需要手工配置。）
- 对于从 snapcraft 来的二进制，在 Snap 里面运行它们。（不需要手工配置。和后台守护进程（daemon）兼容。）

`xdg-desktop-portal` 软件包为通用的桌面特性提供一个标准的 API。参见 xdg-desktop-portal (flatpak) 和 xdg-desktop-portal (snap) 。

这个沙盒环境技术和在智能手机操作系统上的应用程序非常相像，这里的应用程序也是在资源访问受到控制下执行的。

一些大的 GUI（图形用户界面）应用程序，比如说 Debian 上的网页浏览器，也在内部使用了沙盒环境技术，这样让它们安全性更好。

软件包	流行度	大小	说明
flatpak	V:108, I:116	9297	Flatpak 桌面应用程序配置框架
gnome-software-plugin-flatpak	V:28, I:39	291	GNOME 软件管理器的 Flatpak 支持
snapd	V:56, I:59	62738	启用 snap 软件包的后台守护进程（daemon）和工具
gnome-software-plugin-snap	V:1.4, I:2.0	144	GNOME 软件管理器的 Snap 支持
xdg-desktop-portal	V:376, I:438	2291	Flatpak 和 Snap 的桌面集成门户
xdg-desktop-portal-gtk	V:340, I:436	715	gtk (GNOME) 的 xdg-desktop-portal 后端
xdg-desktop-portal-kde	V:81, I:111	3333	Qt (KDE) 的 xdg-desktop-portal 后端
xdg-desktop-portal-wlr	V:2.4, I:6.7	167	wlroots (Wayland) 的 xdg-desktop-portal 后端
firejail	V:1.1, I:4.2	1725	和 AppImage 一起使用的 SUID 安全沙盒程序 firejail

Table 7.6: 著名的沙盒环境和相关软件包列表

软件包	流行度	大小	协议	说明
gnome-remote-desktop	V:154, I:239	2743	RDP	GNOME 远程桌面服务端
xrdp	V:29, I:31	4757	RDP	xrdp, 远程桌面协议 (RDP) 服务器
x11vnc	V:10, I:48	1863	RFB (VNC)	x11vnc, 远程帧缓存协议 (VNC) 服务器
tigervnc-standalone-server	V:6, I:15	3176	RFB (VNC)	TigerVNC, 远程帧缓存协议 (VNC) 服务器
gnome-connections	V:7, I:154	1726	RDP, RFB (VNC)	GNOME 远程桌面客户端
remmina	V:16, I:59	1046	RDP, RFB (VNC), SPICE, SSH, ...	Remmina: GTK 远程桌面客户端
krdc	V:2, I:15	4144	RDP, RFB (VNC)	KRDC: KDE 远程桌面客户端
virt-viewer	V:5, I:36	1278	RFB (VNC), SPICE	虚拟机管理器下的客户机操作系统的 GUI 显示客户端

Table 7.7: 著名的远程访问服务端列表

7.8 远程桌面

7.9 X 服务端连接

有几种方法从远程主机上的应用连接到 X 服务端（包括本地主机的 xwayland ）。

软件包	流行度	大小	命令	说明
openssh-server	V:781, I:824	3615	sshd 使用选项 X11-forwarding	SSH 服务端（安全）
openssh-client	V:709, I:997	3611	ssh -X	SSH 客户端（安全）
xauth	V:197, I:975	81	xauth	X 授权文件工具
x11-xserver-utils	V:376, I:532	559	xhost	X 服务端访问控制

Table 7.8: 连接到 X 服务端的方式

7.9.1 X 服务端本地连接

使用 X 核心协议的本地应用，能够通过本地 UNIX 域名套接字进行本地连接，来访问本地的 X 服务端。这可以通过拥有 access cookie 的授权文件来授权。授权文件的位置通过”$XAUTHORITY” 环境变量确定，X 显示通过”$DISPLAY” 环境变量确定。由于这些环境变量通常会被自动设置，不需要另行指定。例如，下面的”gitk”。

```
username $ gitk
```

> 注意
> 对于 xwayland, XAUTHORITY 有类似”/run/user/1000/.mutter-Xwaylandauth.YVSU30” 的值。

7.9.2 X 服务端远程连接

使用 X 核心协议的远程应用访问本地的 X 服务器显示，由 X11 转发特性支持。

- 在本地主机中打开一个 gnome 终端。
- 通过下列命令，运行带 -X 选项的 ssh(1)，建立与远程站点的连接。

```
localname @ localhost $ ssh -q -X loginname@remotehost.domain
Password:
```

- 通过下列命令，在远程站点运行一个 X 应用程序，例如“gitk”。

```
loginname @ remotehost $ gitk
```

这个方法可以显示来自远程 X 客户端的输出，相当于它是通过一个本地 UNIX 域名套接字进行本地的连接。

参见介绍 SSH/SSHD 的第 6.3 节。



警告
由于安全的原因，在 Debian 系统上，远程 TCP/IP 连接到 X 服务端，是默认被禁用的。不要通过简单的设置”xhost +” 来启用它们。如果能够避免的话，也不要启用 XDMCP 连接。

7.9.3 X 服务端 chroot 连接

在同一个环境下（比如 chroot），使用 X 核心协议的应用访问同一主机的 X 服务端，授权文件无法访问，能够使用 xhost 进行安全的授权，通过使用 User-based access ，例如下面的”gitk”。

```
username $ xhost + si:localuser:root ; sudo chroot /path/to
# cd /src
# gitk
# exit
username $ xhost -
```

7.10 剪贴板

剪贴文本到剪贴板，参见第 1.4.4 节。

剪贴图像到剪贴板，参见第 11.6 节。

一些命令行的命令也能操作字符剪贴板（主要键和剪贴板）。

软件包	流行度	软件包大小	当前目标	说明
xsel	V:8, I:42	55	X	X 选择的命令行接口（剪贴板）
xclip	V:19, I:78	62	X	X 选择的命令行接口（剪贴板）
wl-clipboard	V:11, I:28	174	Wayland	wl-copy wl-paste: Wayland 剪贴板的命令行接口
gpm	V:8.0, I:8.7	524	Linux 控制台	在 Linux 控制台上捕获鼠标事件的后台守护进程（daemon）

Table 7.9: 操作字符剪贴板相关程序列表

Chapter 8

国际化和本地化

一个应用软件的多语言化 (M17N) 或本地语言支持，通过 2 个步骤完成。

- 国际化 (I18N): 使一个软件能够处理多个语言环境。
- 本地化 (L10N)：使一个软件处理一个特定的语言环境。

提示
在 multilingualization（多语言化）、internationalization（国际化）和 localization（本地化）中，有 17, 18, 或 10 个字母在”m” 和”n”, ”i” 和”n”, 或”l” 和”n” 中间，它们相应表示为 M17N, I18N 和 L10N。细节参见国际化和本地化 。

8.1 语言环境

程序支持国际化的行为，是通过配置环境变量”$LANG” 来支持本地化。语言环境的实际支持，依赖 `libc` 库提供的特性，并要求安装 `locales` 或 `locales-all` 软件包。`locales` 软件包需要被适当的初始化。

如果 `locales` 或 `locales-all` 软件包均没有安装，支持语言环境的特性丢失，系统使用 US 英语消息，并按 **ASCII** 处理数据。这个行为和”$LANG” 被设置为”LANG=”、”LANG=C” 或”LANG=POSIX” 相同。

GNOME 和 KDE 等现代软件是多语言的。他们通过处理 UTF-8 数据来实现国际化，并通过 gettext(1) 架构提供翻译信息来本地化。翻译信息可以由独立的本地化软件包来提供。

目前的 Debian 桌面 GUI（图形用户界面）系统通常在 GUI 环境中设置语言环境为”LANG=xx_YY.UTF-8”。这里，”xx” 是 ISO 639 语言代码 ，”YY” 是 ISO 3166 国家地区代码。这些值由配置桌面的 GUI 对话框来设置，并改变程序的行为。参见第 1.5.2 节

8.1.1 UTF-8 语言环境的基本原理

最简单的文本数据表达是 **ASCII**，它对英语是足够的，少于 127 个字符（使用 7 位描述）。

即使纯英文文本也可能包含非 ASCII 字符，例如微微卷曲的左右引号在 ASCII 中是不可用的。

```
b''“b''double quoted textb''”b'' is not "double quoted ASCII"
b''‘b''single quoted textb''’b'' is not 'single quoted ASCII'
```

为了支持更多字符，许多字符集和编码系统被用来支持多语言。（参见表 11.2）。

Unicode 字符集可以用 21 位码点范围来显示几乎所有人类已知的字符（例如，十六进制的 0 到 10FFFF）。

文本编码系统 UTF-8 将 Unicode 码点适配到一个合理的 8 位数据流，并大部分兼容 ASCII 数据处理系统。这个使 **UTF-8** 作为现代推荐的选择。**UTF** 表示 Unicode 转换格式（Unicode Transformation Format）。当 ASCII 纯文本数据转换为 UTF-8 数据，它有和原始 ASCII 完全一样的内容和大小。所以配置 UTF-8 语言环境不会有任何丢失。

在 UTF-8 语言环境下兼容的应用程序，你可以显示和编辑外语文本数据，在所要求的字体和输入法安装和启用后。例如在"LANG=fr_FR.UTF-8" 语言环境下，gedit(1) (GNOME 桌面的文本编辑器) 能够显示和编辑中文字符文本数据，而显示的菜单是法语。

提示
新标准的"en_US.UTF-8" 和老标准的"C"/"POSIX" 语言环境都使用标准的 US 英文消息，它们在排序等方面有细微的不同。在维护老的"C" 本地行为时，如果你不仅想处理 ASCII 字符，同时还想优雅的处理 UTF-8 编码的字符，在 Debian 上使用非标准的"C.UTF-8" 语言环境。

注意
一些程序在支持 I8N 后会消耗更多的内存。这是因为它们为了速度优化，而在内部使用 UTF-32(UCS4) 来支持 Unicode，并且每个独立于语言环境所选的 ASCII 字符数据都会消耗 4 个字节。再一次地，使用 UTF-8 语言环境并不会使你损失什么。

8.1.2 语言环境的重新配置

为了让系统访问某一语言环境，语言环境数据必须从语言环境数据库中编译。

`locales` 软件包没有包含预先编译的语言环境数据。你需要按下面的方法配置：

```
# dpkg-reconfigure locales
```

该过程包含 2 个步骤。

1. 选择所有需要的语言环境数据编译为二进制形式。（请确认至少包含一个 UTF-8 语言环境）
2. 通过创建"/etc/default/locale"来设置系统默认的语言环境值给 PAM 使用（参见第 4.5 节）。

设置在"/etc/default/locale" 里的系统范围的默认语言环境，可以被 GUI（图形用户界面）应用程序的 GUI 配置覆盖。

注意
所使用的确切传统编码系统可以通过"/usr/share/i18n/SUPPORTED" 来确认。因此，"LANG=en_US" 是"LANG=en_US.ISO-8859-1"。

`locales-all` 软件包有所有预编译的语言环境数据，但是不创建"/etc/default/locale"，你可能还需要安装 `locales` 软件包。

提示
一些 Debian 系发行版的 `locales` 软件包，包含有所有语言环境的预先编译好的语言环境数据。为了模拟这样的系统环境，你需要同时在 Debian 安装 `locales` 和 `locales-all` 软件包。

8.1.3 文件名编码

对于跨平台的数据交换 (参见第 10.1.7 节), 你需要使用特殊的编码挂载文件系统. 举个例子，不使用选项时，mount(8) 假设 vfat 文件系统使用 CP437. 你需要给文件名提供明确的挂载选项来使用 UTF-8 或 CP932.

注意
在 GNOME 这类的现代桌面环境下，当自动挂载一个热拔插 U 盘时，你可以提供这样的挂载选项。右击桌面上的图标，点击”Drive”，”Setting”, 输入”utf8” 到”Mount options:”. 当这个 U 盘下次挂载时，UTF-8 就可以了。

注意
如果你在升级一个系统，或者从老的非 UTF-8 系统迁移磁盘，非 ASCII 字符的文件名也许是使用老旧的 ISO-8859-1 或 eucJP 来编码. 请寻求文本转换工具把他们转换到 UTF-8. 参见第 11.1 节.

在默认情况下，Samba 对新的客户端 (Windows NT, 200x, XP) 使用 Unicode，但对老的客户端 (DOS 和 Windows 9x/Me) 使用 CP850. 可以在”`/etc/samba/smb.conf`” 文件里面，使用”`dos charset`” 来改变老客户端的这个默认编码。比如说，CP932 表示为日语。

8.1.4 本地化信息和翻译文档

在 Debian 系统中显示的许多文档和文本信息有翻译存在，比如错误信息、标准程序输出、菜单和手册页。GNU gettext(1) 命令工具链是大部分翻译活动的后端工具。

aptitude(8) 里，”Tasks” → ”Localization” 提供一个有用的二进制包扩展列表，给应用程序增加本地化信息和提供翻译文档。

举个例子，你可以安装 `manpages-`*`LANG`* 包来获得本地化 man 手册页信息。从”`/usr/share/man/it/`” 来读取 *programname* 意大利语的 man 手册页, 执行下面的操作。

```
LANG=it_IT.UTF-8 man programname
```

通过 $LANGUAGE 环境变量，GNU gettext 能够适应翻译语言的优先级列表。例如：

```
 $ export LANGUAGE="pt:pt_BR:es:it:fr"
```

获取更多信息，参见 `info gettext` ，阅读”The LANGUAGE variable” 章节。

8.1.5 语言环境的影响

sort(1) 和 ls(1) 的字符排序受语言环境 locale 影响。导出 `LANG=en_US.UTF-8`，排序用字典 `A->a->B->b...->Z->z` 顺序，而导出 `LANG=C.UTF-8`，排序使用 ASCII 二进制 `A->B->...->Z->a->b...` 顺序。

ls(1) 的日期格式受语言环境影响 (参见第 9.3.4 节)。

date(1) 程序的日期格式受语言环境的影响。例如：

```
 $ unset LC_ALL
 $ LANG=en_US.UTF-8 date
Thu Dec 24 08:30:00 PM JST 2023
 $ LANG=en_GB.UTF-8 date
Thu 24 Dec 20:30:10 JST 2023
 $ LANG=es_ES.UTF-8 date
jue 24 dic 2023 20:30:20 JST
 $ LC_TIME=en_DK.UTF-8 date
2023-12-24T20:30:30 JST
```

不同语言环境的数字标点不一样。比如说，英语语言环境中, 一千点一显示为"1,000.1"，而在德语语言环境中, 它显示为"1.000,1"。你可以在电子表格程序里面看到这个不同。

"$LANG" 环境变量的每一个细节特征能够通过设置"$LC_*" 变量来覆盖。这些环境变量又能够通过设置"$LC_ALL" 变量而被再次覆盖。细节参见 locale(7) man 手册页。除非你有强烈的理由创建复杂的配置，请远离他们并只使用"$LANG" 变量来设置一个 UTF-8 语言环境。

8.2 键盘输入

键盘系统能够在系统的不同层级配置。

- Linux 内核: keyboard(5)
- X 服务端：setxkbmap(1)、xkeyboard-config(5)、环境变量 XMODIFIERS
- GUI（图形用户界面）桌面环境：输入法框架：ibus, fcitx5
- 应用层：环境变量设置它的输入源：GTK_IM_MODULE、QT_IM_MODULE、QT_IM_MODULES……

输入方式框架（IM）组成：

- 输入方式引擎（IME）：实际的输入方式
- 配置：处理 IBus 和其它服务的配置，比如 IME 插件的配置
- 面板：用户界面，比如语言栏和候选表

输入多种语言到应用程序的处理流程大致如下：

```
Keyboard        UI panel    Configuration          Application
|                  ^ |           |                        ^  ^
v                  | v           v                        |  |
Linux kernel -> Input method engine (IME) -+-> Gtk, Qt ---+  |
                   | ^                     |                 |
                   | |                     +-> X11, Wayland -+
                   v |
                  IME plugin (ibus-mozc, ...)
```

8.2.1 Linux 控制台和 X 窗口的键盘输入

Debian 系统可以使用 keyboard-configuration 软件包配置多个国际化键盘布局。

```
# dpkg-reconfigure keyboard-configuration
```

对于 Linux 控制台和 X 窗口系统，这将更新在"/etc/default/keyboard" 里的配置参数。许多非 ASCII 字符，包括许多欧洲语言使用的重音字符，可以使用死键、AltGr 键和组合键来输入它们。

注意
如果 ibus 激活，即使在基于传统的 X 的桌面环境下，通过 setxkbmap 配置的传统的 X 键盘，也会被 ibus 覆盖。你能够禁用安装的 ibus，使用 im-config 设置输入法为"None"。更多信息，参见 Debian 维基：键盘。

8.2.2 Wayland 键盘输入

不像 X 窗口协议，Wayland 核心协议甚至不支持重音字符输入。流行的 Wayland 合成器，比如 GNOME 的 Mutter 或 KDE 的 KWin，实现了用于文本输入的 `text-input-unstable-v3` 扩展协议（参见当前 Wayland 协议和它们支持的状态）。

`text-input-unstable-v3` 协议和 Wayland 输入法一起工作得很好（参见 Wayland 输入法项目 post-mortem）。

- 大部分使用 GTK 或 Qt 图形库构建的 GUI（图形用户界面）应用程序支持这个 `text-input-unstable-v3`。
- 流行的输入法引擎 Input Method Engines (IME)，比如 IBus 或 Fcitx (版本 5)，能够和这个 `text-input-unstable-v3` 一起工作。
- 输入法引擎（IME）使用插件来支持多种语言的文本输入。
- 最近的输入法引擎集成了"X Keyboard Extension (XKB)"，功能和 `setxkbmap` 类似，之前是由 X 窗口提供在欧洲语言中输入重音字符支持。

8.2.3 IBus 支持的输入法

对于 GNOME，"`ibus`" 是默认的输入法引擎，它是通过 GNOME 的软件包依赖自动安装的。

- 大部分多语言键盘输入特性，能够从"GNOME 设置" 或"GNOME 调整" 中配置。
- 一些多语言键盘输入特性可能需要从 ibus-setup(1) 命令来配置。
- 存在表情符号键盘输入，通过按 `SUPER-.` " (同时按住 Windows 键和句号键)，接着一个用于每个表情的关键字和空格键。

下面是 IBus 和它的插件软件包列表。

8.2.4 Fcitx 支持的输入法

Fcitx（版本号 5）输入法框架在中国广受欢迎并与"`ibus`" 兼容。

下面是"`fcitx5`" 和它的插件软件包列表。

8.2.5 一个日语的例子

我发现在英语环境 ("`en_US.UTF-8`") 下启动日文输入法非常有用. 下面是使用 IBus 的做法:

1. 安装日文输入法软件包 `ibus-mozc`（或 `ibus-anthy` ）。
2. • 通用：执行 ibus-setup(1) → 选择"Input Method" → 单击"Add" → "Japanese" → "Mozc (or Anthy)" → 单击"Add"
   • GNOME: 选择"Settings" → "Keyboard" → "Input Sources" → 在"Input Sources" 中单击"+" → "Japanese" → "Mozc (或 Anthy)"，然后单击"Add"。
3. 你可以选择许多输入源。
4. 重新登录用户账户。
5. 右键单击 GUI 工具条图标，设置每一个输入源。
6. 使用 SUPER-SPACE 在安装的输入法之间进行切换. (SUPER 键通常是 Windows 键.)

软件包	流行度	大小	支持的语言环境
ibus	V:211, I:252	1884	使用 dbus 的输入方式框架
ibus-mozc	V:2.0, I:3.7	872	日文
ibus-anthy	V:0.45, I:0.98	8740	日文
ibus-skk	V:0.04, I:0.15	242	日文
ibus-kkc	V:0.04, I:0.18	206	日文
ibus-libpinyin	V:1.0, I:4.7	125	中文（zh_CN）
ibus-chewing	V:0.11, I:0.73	298	繁体中文（zh_TW）
ibus-libzhuyin	I:0.08	41021	繁体中文（zh_TW）
ibus-rime	V:0.31, I:0.50	76	中文 (zh_CN/zh_TW)
ibus-cangjie	V:0.01, I:0.09	235	中文（zh_HK）
ibus-hangul	V:0.3, I:1.9	261	韩文
ibus-libthai	V:0.00, I:0.05	84	泰文
ibus-table-thai	I:0.05	59	泰文
ibus-unikey	V:0.21, I:0.42	286	越南语
keyman	V:0.01, I:0.09	507	多语言：Keyman 插件，超过 2000 种语言
ibus-table	V:0.06, I:0.85	2482	IBus 表插件
ibus-m17n	V:0.2, I:1.7	373	多语言：印度语、阿拉伯语和其它

Table 8.1: IBus 和它的插件软件包列表

软件包	流行度	大小	支持的语言环境
fcitx5	V:8, I:12	712	输入方式框架和”ibus” 兼容
fcitx5-mozc	V:1.1, I:1.8	1578	日文
fcitx5-anthy	V:0.06, I:0.24	853	日文
fcitx5-skk	V:0.08, I:0.16	371	日文
fcitx5-kkc	V:0.00, I:0.07	418	日文
fcitx5-chinese-addons	I:8.7	17	中文（zh_* 的元软件包）
fcitx5-pinyin	V:4.2, I:9.4	1000	中文（zh_CN）
fcitx5-chewing	V:0.29, I:0.92	228	繁体中文（zh_TW）
fcitx5-zhuyin	I:0.07	108535	繁体中文（zh_TW）
fcitx5-rime	V:0.57, I:0.99	393	中文 (zh_CN/zh_TW)
fcitx5-table-cangjie-large	I:0.12	1292	中文（zh_HK）
fcitx5-hangul	V:0.13, I:0.28	239	韩文
fcitx5-libthai	I:0.04	119	泰文
fcitx5-table-thai	I:0.07	34	泰文
fcitx5-unikey	V:0.13, I:0.26	624	越南语
fcitx5-m17n	V:0.41, I:0.85	171	多语言：印度语、阿拉伯语和其它
fcitx5-table	V:0.3, I:9.2	428	fcitx5 表插件
fcitx5-keyman	V:0.04, I:0.05	241	多语言：Keyman 插件，超过 2000 种语言

Table 8.2: Fcitx5 和它的插件软件包列表

提示
如果你希望在日本物理键盘（shift-2 按键刻有有一个" 双引号标记）上访问只有字母表的键盘环境，在上面的过程中选择”Japanese” 。你能够在物理的“US”键盘（shift-2 按键刻有有一个 @ 标记）上使用”Japanese mozc (或 anthy)” 输入日文。

- 对于 Wayland：
  - `im-config` 软件包没有做任何事情，能够被安全的移除。
  - 你也许不需要设置环境变量，除非为了向后兼容性等原因。
  - 如果你需要设置环境变量，创建一个类似”`~/.config/environment.d/50-input-method.conf` ” 的文件来设置它们。
- 对于 X Windows：
  - 安装 `im-config` 软件包。
  - im-config(8) 的用户界面菜单入口是”Input method”。
  - 此外，从用户的 shell 来执行”`im-config`” 。
  - 如果命令从 root 账户或非 root 账号执行，im-config(8) 表现会有所不同。
  - im-config(8) 让最佳的输入法作为系统默认而不需要用户干预。

8.3 显示输出

Linux 控制台只能显示有限的字符。（你需要使用特殊的终端程序，例如 jfbterm(1)，从而在非 GUI 控制台中显示非欧洲语言。）

只要需要的字体安装并被启用，GUI（图形用户界面）环境（第 7 章）能够显示任意 UTF-8 字符。（原始字体数据的编码会被处理，并对用户来说是透明的。）

8.3.1 终端配置

Debian 系统可以使用 `console-setup` 软件包配置多个国际化控制台布局。

```
# dpkg-reconfigure console-setup
```

对于 Linux 控制台和 X 窗口系统，这将更新在”`/etc/default/console-setup`” 里的配置参数来显示许多非 ASCII 字符，包括许多欧洲语言使用的重音字符，都能够使其存在。

有几个组件可以用来配置字符控制台和 ncurses(3) 系统功能。

- “`/etc/terminfo/*/*`”文件（terminfo(5)）
- “`$TERM`”环境变量（term(7)）
- setterm(1)、stty(1)、tic(1) 和 toe(1)

如果 `xterm` 的 `terminfo` 对非 Debian 的 `xterm` 不起作用，则当你从远程登录到 Debian 系统时，你需要改变你的终端类型“`$TERM`”，从“`xterm`”更改为功能受限的版本（例如“`xterm-r6`”）。更多内容参见“`/usr/share/doc/ncurses-base/`是“`$TERM`”中最通用的。

8.3.2 东亚环境下宽度有歧义的字符

在东亚语言环境下，方框绘制、希腊字符和西里尔字符可能会显示得比你预期的样子更宽，这样会导致终端输出排列不再整齐（参见 Unicode 标准附录 #11，4.2 Ambiguous 字符）。

您可以绕过这个问题：

- `gnome-terminal`：首选项 → 配置文件 → 配置名 → 兼容性 → 宽度有歧义的字符 → 窄
- `ncurses`：设置环境变量 `export NCURSES_NO_UTF8_ACS=0`。

Chapter 9

系统技巧

这里，描述配置和管理系统的基本技巧，大部分在控制台操作。

9.1 控制台技巧

有一些工具程序来帮助你的控制台活动。

软件包	流行度	大小	说明
mc	V:42, I:172	1590	参见第 1.3 节
bsdutils	V:445, I:1000	338	script (1) 命令来记录终端会话
screen	V:49, I:187	1007	VT100/ANSI 终端模拟器混合复用的终端
tmux	V:114, I:172	1653	终端复用的备选方案（使用“Control-B”代替）
ripgrep	V:17, I:49	5284	在源代码树中快速递归搜索字符串，并自动过滤
fzf	V:10, I:37	5692	模糊的文本查找器
fzy	V:0.05, I:0.31	59	模糊的文本查找器
rlwrap	V:1, I:11	328	具备 readline 特征的命令行封装
ledit	V:0.4, I:7.1	401 401	具备 readline 特征的命令行封装
rlfe	V:0.03, I:0.49	45	具备 readline 特征的命令行封装

Table 9.1: 支持控制台活动的程序列表

9.1.1 清晰的记录 shell 活动

简单地使用 script(1) （参见第 1.4.9 节）记录 shell 活动会产生一个有控制字符的文件。这些控制字符可以按下面的方式，使用 col(1) 去掉。

```
$ script
Script started, file is typescript
```

做些操作……按 Ctrl-D 退出 script.

```
$ col -bx < typescript > cleanedfile
$ vim cleanedfile
```

有替代的方式来记录 shell 活动：

- 使用 tee (在 initramfs 的启动过程中可用)：

```
$ sh -i 2>&1 | tee typescript
```

- 使用 gnome-terminal 增加行缓冲，用滚动条查看。
- 使用 screen 和"^A H" (参见第 9.1.2 节) 来进行控制台记录。
- 使用 vim 输入":terminal" 进入终端模式。使用"Ctrl-W N" 从终端模式退出到普通模式。使用":w typescript" 将缓存写到一个文件。
- 使用 emacs 和"M-x shell", "M-x eshell", 或"M-x term" 来进入记录控制台。使用"C-x C-w" 将缓存写到文件。

9.1.2 screen 程序

screen(1) 不但允许一个终端窗口运行多个进程，还允许远程 **shell** 进程支持中断的连接. 这里是一个典型的 screen(1) 使用场景.

1. 登录到一个远程机器。
2. 在单个控制台上启动 screen。
3. 使用 ^A c ("Control-A" 接着"c") 在 screen 中创建的窗口执行多个程序.
4. 按 ^A n ("Control-A" 接着"n") 来在多个 screen 窗口间转换.
5. 突然，你需要离开你的终端，但你不想丢掉正在做的工作，而这些工作需要连接来保持。
6. 你可以通过任何方式分离 screen 会话。
   - 残忍地拔掉你的网络连接
   - 输入 ^A d ("Control-A" 接着"d") 并手工从远程连接退出登录
   - 输入 ^A DD ("Control-A" 接着"DD") 分离 screen 并退出登录
7. 你重新登录到同一个远处主机（即使从不同的终端）。
8. 使用"screen -r" 启动 screen.
9. screen 魔术般的重新附上先前所有的 screen 窗口和所有在活动运行的程序.

提示
对于拨号或者按包计费的网络连接，你可以通过 screen 节省连接费用，应为你可以在断开连接时让一个进程继续运行，当你稍后再次连接时重新附上它。

在 screen 会话里, 除了命令按键外的所有键盘输入都会被发送到当前窗口。screen 所有命令按键是通过 ^A ("Control-A") 加单个键 [加任何参数] 来输入. 这里有一些重要的命令按键需要记住。

细节参见 screen(1).

参见 tmux(1) ，了解替代命令的功能。

键绑定功能	说明
^A ?	显示帮助屏幕（显示键绑定）
^A c	创建一个新的窗口并切换到该窗口
^A n	到下一个窗口
^A p	到前一个窗口
^A 0	到 0 号窗口
^A 1	到 1 号窗口
^A w	显示窗口列表
^A a	作为键盘输入发送 Ctrl-A 到当前窗口
^A h	把当前窗口的硬拷贝写到一个文件
^A H	开始/结束当前窗口到文件的记录
^A ^X	锁定终端 (密码保护)
^A d	从终端分离 screen 会话
^A DD	分离 screen 会话并退出登录

Table 9.2: screen 键绑定列表

9.1.3 在目录间游走

在第 1.4.2 节，2 个技巧允许快速在目录间游走，在 `$CDPATH` 和 `mc` 描述。

如果你使用模糊文本过滤程序，你能够不输入精准路径。对于 `fzf` 软件包，在 `~/.bashrc` 里面包括下列内容。

```
FZF_KEYBINDINGS_PATH=/usr/share/doc/fzf/examples/key-bindings.bash
FZF_COMPLETION_PATH=/usr/share/doc/fzf/examples/completion.bash
if [ -f $FZF_KEYBINDINGS_PATH ] && [ -f $FZF_COMPLETION_PATH ]; then
  . $FZF_KEYBINDINGS_PATH
  . $FZF_COMPLETION_PATH
fi
```

例如：

- 你能够最小化的操作跳入非常深的子目录。你首先输入”`cd **`” 后按 `Tab`。然后你将被提示候选路径。输入部分路径字符串，比如 `s/d/b foo`，将会缩窄候选路径。通过有光标和回车键的 `cd`，你选择将要使用的路径。
- 你可以用最小化的操作，从命令历史里面选择一个命令。在命令行提示符下按 `Ctrl-R` 。然后你将被提示候选的命令。输入部分命令字符串，比如 `vim d`，将会缩窄候选项。使用光标和回车键选择将要使用的命令。

9.1.4 Readline 封装

一些命令，比如 `/usr/bin/dash`，它缺少命令行历史编辑能力，但在 `rlwrap` 或它的等价物下运行就能够透明的增加这样的功能。

```
$ rlwrap dash -i
```

这提供一个便利平台来测试 `dash` 的细微之处，使用类似 `bash` 的友好环境。

9.1.5 扫描源代码树

在 `ripgrep` 软件包中的 rg(1) 命令，在扫描源代码树的典型场景中，提供了一个比 grep(1) 命令更快速的替代。它充分利用了现代多核 CPU，并自动使用适当的过滤器来忽略一些文件。

9.2 定制 vim

在你通过第 1.4.8 节学习基本的 vim(1) 后, 请阅读 Bram Moolenaar 的"Seven habits of effective text editing (2000)"来理解 vim 应当怎样被使用。

9.2.1 用内部特性定制 vim

vim 的行为能够被显著的改变，通过 Ex-模式的命令，启用它的内部特性，比如"set ..." 来设置 vim 选项。

这些 Ex-模式的命令，能够在用户的 vimrc 文件里面包括，传统的"~/.vimrc" 或 git 友好的"~/.vim/vimrc"。这里有一个非常简单的例子[1]：

```
""" Generic baseline Vim and Neovim configuration (~/.vimrc)
"""   - For NeoVim, use "nvim -u ~/.vimrc [filename]"
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
let mapleader = ' '             " :h mapleader
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
set nocompatible                " :h 'cp -- sensible (n)vim mode
syntax on                       " :h :syn-on
filetype plugin indent on       " :h :filetype-overview
set encoding=utf-8              " :h 'enc (default: latin1) -- sensible encoding
""" current vim option value can be verified by :set encoding?
set backspace=indent,eol,start  " :h 'bs (default: nobs) -- sensible BS
set statusline=%<%f%m%r%h%w%=%y[U+%04B]%2l/%2L=%P,%2c%V
set listchars=eol:¶,tab:b''▫b''\ ,extends:b''▫b'',precedes:b''▫b'',nbsp:b''▫b''
set viminfo=!,'100,<5000,s100,h " :h 'vi -- bigger copy buffer etc.
""" Pick "colorscheme" from blue darkblue default delek desert elflord evening
""" habamax industry koehler lunaperche morning murphy pablo peachpuff quiet ron
""" shine slate torte zellner
colorscheme industry
""" don't pick "colorscheme" as "default" which may kill SpellUnderline settings
set scrolloff=5                 " :h 'scr -- show 5 lines around cursor
set laststatus=2                " :h 'ls (default 1)  k
""" boolean options can be unset by prefixing "no"
set ignorecase                  " :h 'ic
set smartcase                   " :h 'scs
set autoindent                  " :h 'ai
set smartindent                 " :h 'si
set nowrap                      " :h 'wrap
"set list                        " :h 'list (default nolist)
set noerrorbells                " :h 'eb
set novisualbell                " :h 'vb
set t_vb=                       " :h 't_vb -- termcap visual bell
set spell                       " :h 'spell
set spelllang=en_us,cjk         " :h 'spl -- english spell, ignore CJK
set clipboard=unnamedplus       " :h 'cb -- cut/copy/paste with other app
set hidden                      " :h 'hid
set autowrite                   " :h 'aw
set timeoutlen=300              " :h 'tm
```

vim 的键盘映射能够在用户的 vimrc 文件里面修改。例如：



小心
没有非常好的理由，请不要尝试改变默认的键绑定。

1更多精心制作的定制例子："Vim Galore", "sensible.vim", ...

```
""" Popular mappings (imitating LazyVim etc.)
""" Window moves without using CTRL-W which is dangerous in INSERT mode
nnoremap <C-H> <C-W>h
nnoremap <C-J> <C-W>j
nnoremap <C-K> <C-W>k
silent! nnoremap <C-L> <C-W>l
""" Window resize
nnoremap <C-LEFT> <CMD>vertical resize -2<CR>
nnoremap <C-DOWN> <CMD>resize -2<CR>
nnoremap <C-UP> <CMD>resize +2<CR>
nnoremap <C-RIGHT> <CMD>vertical resize +2<CR>
""" Clear hlsearch with <ESC> (<C-L> is mapped as above)
nnoremap <ESC> <CMD>noh<CR><ESC>
inoremap <ESC> <CMD>noh<CR><ESC>
""" center after jump next
nnoremap n nzz
nnoremap N Nzz
""" fast "jk" to get out of INSERT mode (<ESC>)
inoremap  jk <CMD>noh<CR><ESC>
""" fast "<ESC><ESC>" to get out of TERM mode (CTRL-\ CTRL-N)
tnoremap <ESC><ESC> <C-\><C-N>
""" fast "jk" to get out of TERM mode (CTRL-\ CTRL-N)
tnoremap jk <C-\><C-N>
""" previous/next trouble/quickfix item
nnoremap [q <CMD>cprevious<CR>
nnoremap ]q <CMD>cnext<CR>
""" buffers
nnoremap <S-H> <CMD>bprevious<CR>
nnoremap <S-L> <CMD>bnext<CR>
nnoremap [b <CMD>bprevious<CR>
nnoremap ]b <CMD>bnext<CR>
""" Add undo break-points
inoremap  , ,<C-G>u
inoremap  . .<C-G>u
inoremap  ; ;<C-G>u
""" save file
inoremap <C-S> <CMD>w<CR><ESC>
xnoremap <C-S> <CMD>w<CR><ESC>
nnoremap <C-S> <CMD>w<CR><ESC>
snoremap <C-S> <CMD>w<CR><ESC>
""" better indenting
vnoremap < <gv
vnoremap > >gv
""" terminal (Somehow under Linux, <C-/> becomes <C-_> in Vim)
nnoremap <C-_> <CMD>terminal<CR>
"nnoremap <C-/> <CMD>terminal<CR>
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
if ! has('nvim')
""" Toggle paste mode with <SPACE>p for Vim (no need for Nvim)
set pastetoggle=<leader>p
""" nvim default mappings for Vim.  See :h default-mappings in nvim
""" copy to EOL (no delete) like D for d
noremap Y y$
""" sets a new undo point before deleting
inoremap <C-U> <C-G>u<C-U>
inoremap <C-W> <C-G>u<C-W>
""" <C-L> is re-purposed as above
""" execute the previous macro recorded with Q
nnoremap Q @@
""" repeat last substitute and *KEEP* flags
nnoremap & :&&<CR>
```

```
""" search visual selected string for visual mode
xnoremap * y/\V<C-R>"<CR>
xnoremap # y?\V<C-R>"<CR>
endif
```

为了使上面的按键绑定正确地运行，终端程序需要配置：Backspace-键产生”ASCII DEL”、Delete-键产生”Escape sequence”。

其它杂项配置能够在用户的 vimrc 文件里面修改。例如：

```
""" Use faster 'rg' (ripgrep package) for :grep
if executable("rg")
  set grepprg=rg\ --vimgrep\ --smart-case
  set grepformat=%f:%l:%c:%m
endif
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""" Retain last cursor position :h '"
augroup RetainLastCursorPosition
  autocmd!
  autocmd BufReadPost *
    \ if line("'\"") > 0 && line ("'\"") <= line("$") |
    \   exe "normal! g'\"" |
    \ endif
augroup END
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""" Force to use underline for spell check results
augroup SpellUnderline
  autocmd!
  autocmd ColorScheme * highlight SpellBad term=Underline gui=Undercurl
  autocmd ColorScheme * highlight SpellCap term=Underline gui=Undercurl
  autocmd ColorScheme * highlight SpellLocal term=Underline gui=Undercurl
  autocmd ColorScheme * highlight SpellRare term=Underline gui=Undercurl
augroup END
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""" highlight tailing spaces except when typing as red (set after colorscheme)
highlight TailingWhitespaces ctermbg=red guibg=red
""" \s\+     1 or more whitespace character: <Space> and <Tab>
""" \%#\@<!  Matches with zero width if the cursor position does NOT match.
match TailingWhitespaces /\s\+\%#\@<!$/
```

9.2.2 用外部软件包定制 vim

下面能够发现有趣的外部插件软件包：

- Vim - 无所不在的文本编辑器 -- Vim 和 vim 脚本的官方上游站点
- VimAwesome -- Vim 插件列表
- vim-scripts -- Debian 软件包：一个 vim 脚本的收集

vim-scripts 软件包里面的插件软件包，能够使用用户的 vimrc 文件启用。比如：

```
packadd! secure-modelines
packadd! winmanager
" IDE-like UI for files and buffers with <space>w
nnoremap <leader>w         :WMToggle<CR>
```

新的原生 Vim 软件包系统同”git” 和”git submodule” 顺利的工作。一个这样的配置例子能够在我的 git 仓库: dot-vim 找到。本质上是这样做的：

- 通过使用"`git`"和"`git submodule`"，最新的扩展软件包，比如说"*name*"，会被放到`~/.vim/pack/*/opt/`*name*和类似的地方。
- 通过增加`:packadd!` *name* 行到用户的 vimrc 文件，这些软件包被放到`runtimepath`。
- Vim 在它的初始化时加载这些软件包到`runtimepath`。
- 在它初始化的最后，安装文档的标签被更新，使用"`helptags ALL`"。

更多信息，请使用"`vim --startuptime vimstart.log`" 启动`vim`来检查实际的执行顺序和每一个步骤消耗的时间。

是相当迷惑的看到这么多的方式[2] 来管理和加载这些外部的软件包到`vim`。检查原始的信息是最好的方法。

按键	信息
`:help package`	解释 vim 软件包机制
`:help runtimepath`	解释`runtimepath`机制
`:version`	内部状态，包括 vimrc 文件的候选
`:echo $VIM`	环境变量"`$VIM`" 用来定位 vimrc 文件的路径
`:set runtimepath?`	列出用来搜索所有运行时支持文件的目录
`:echo $VIMRUNTIME`	环境变量"`$VIMRUNTIME`" 用来定位大量系统提供的运行时支持文件

Table 9.3: `vim` 的初始化信息

9.3 数据记录和展示

9.3.1 日志后台守护进程（**daemon**）

许多传统的程序在"`/var/log/`" 目录下用文本文件格式记录它们的活动.

在一个产生很多日志文件的系统上，用 logrotate(8) 来简化日志文件的管理。

许多新的程序使用 systemd-journald(8) 日志服务的二进制文件格式来记录它们的活动，在"`/var/log/journal`" 目录下。

你能够从 shell 脚本记录数据到 systemd-journald(8) 日志，使用 systemd-cat(1) 命令。

参见第 3.5 节和第 3.4 节.

9.3.2 日志分析

这里是主要的日志分析软件 ("`~Gsecurity::log-analyzer`" 在 aptitude(8) 中).

注意
CRM114 提供语言架构来写模糊过滤器，使用了 TRE 正则表达式库 。它主要在垃圾邮件过滤器中使用，但也能够用于日志分析。

---

[2] vim-pathogen 也很流行。

软件包	流行度	大小	说明
fail2ban	V:101, I:112	2275	禁用造成多个认证错误的 IP
logwatch	V:9, I:10	2440	用 Perl 写的日志分析软件，有好的输出
awstats	V:4.7, I:7.5	6934	强大和特性全面的 web 服务器日志分析
analog	V:3, I:81	3739	web 服务器日志分析
sarg	V:0.61, I:0.67	863	生成 squid 分析报告
pflogsumm	V:1.4, I:3.4	173	Postfix 日志条目概要
fwlogwatch	V:0.11, I:0.15	487	防火墙日志分析软件
squidview	V:0.04, I:0.38	224	监控和分析 squid access.log 文件
swatch	V:0.07, I:0.28	99	有正则表达式、高亮和曲线的日志文件查看器
crm114	V:0.05, I:0.30	1371	Controllable Regex Mutilator 和垃圾邮件过滤 (CRM114)
icmpinfo	V:0.01, I:0.34	42	解释 ICMP 信息

Table 9.4: 系统日志分析软件列表

9.3.3 定制文本数据的显示

尽管例如 more(1) 和 less(1) 这样的分页程序（参见第 1.4.5 节）和用于高亮和格式的自定义工具（参见第 11.1.8 节）可以漂亮地显示文本数据，但通用的编辑器（参见第 1.4.6 节）是用途最广的，且可定制性最高。

提示
对于 vim(1) 和它的分页模式别名 view(1)，“`:set hls`”可以启用高亮搜索。

9.3.4 定制时间和日期的显示

“`ls -l`”命令默认的时间和日期显示格式取决于语言环境（相关的值参见第 1.2.6 节）。“`$LANG`”变量将被首先考虑，但它会被导出的“`$LC_TIME`”或”`$LC_ALL`” 环境变量覆盖。

每个语言环境实际的默认显示格式取决于所使用的 C 标准库的版本（`libc6` 软件包），也就是说，不同的 Debian 发行版有不同的默认情况。对于 iso-formates，参见 ISO 8601。

如果你真的想自定义超出语言环境的时间和日期显示格式，你应该通过“`--time-style`”参数或“`$TIME_STYLE`”的值来设置时间样式值（参见 ls(1)、date(1)、“`info coreutils 'ls invocation'`”）。

提示
你可以使用命令别名以避免在命令行中输入长的选项，(参见第 1.5.9 节)：

```
alias ls='ls --time-style=+%d.%m.%y %H:%M'
```

9.3.5 shell 中 echo 的颜色

大部分现代终端的 shell 中 echo 能够使用 ANSI 转义字符来显示颜色(参见“`/usr/share/doc/xterm/ctlseqs.txt.gz`”)

尝试下列例子

时间样式值	语言环境	时间和日期显示
iso	任何值	01-19 00:15
long-iso	任何值	2009-01-19 00:15
full-iso	任何值	2009-01-19 00:15:16.000000000 +0900
locale	C	Jan 19 00:15
locale	en_US.UTF-8	Jan 19 00:15
locale	es_ES.UTF-8	ene 19 00:15
+%d.%m.%y %H:%M	任何值	19.01.09 00:15
+%d.%b.%y %H:%M	C 或 en_US.UTF-8	19.Jan.09 00:15
+%d.%b.%y %H:%M	es_ES.UTF-8	19.ene.09 00:15

Table 9.5: 使用时间样式值的"ls -l" 命令的时间和日期的显示例子

```
$ RED=$(printf "\x1b[31m")
$ NORMAL=$(printf "\x1b[0m")
$ REVERSE=$(printf "\x1b[7m")
$ echo "${RED}RED-TEXT${NORMAL} ${REVERSE}REVERSE-TEXT${NORMAL}"
```

9.3.6 有颜色输出的命令

在交互式的环境下，命令的输出带颜色，能够给检查命令的输出带来便利。我在我的"~/.bashrc" 里加入了下面内容.

```
if [ "$TERM" != "dumb" ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=always'
    alias ll='ls --color=always -l'
    alias la='ls --color=always -A'
    alias less='less -R'
    alias ls='ls --color=always'
    alias grep='grep --color=always'
    alias egrep='egrep --color=always'
    alias fgrep='fgrep --color=always'
    alias zgrep='zgrep --color=always'
else
    alias ll='ls -l'
    alias la='ls -A'
fi
```

在交互式命令中，使用别名来限制颜色的影响范围。导出环境变量"export GREP_OPTIONS='--color=auto'" 也有好处，这样能够让 less(1) 之类的页面程序看到颜色。当使用管道到其它命令时，你想去掉颜色，上面列子"~/.bashrc" 中的内容，可以使用"--color=auto" 代替.

> 提示
> 在交互式的环境中，通过"TERM=dumb bash" 调用 shell ，你能够关闭这些颜色别名。

9.3.7 记录编辑器复杂的重复操作动作

你能够记录编辑器复杂的重复操作动作。

对于 Vim, 请按下面操作。

- ”qa”: 开始记录输入字符到有名字的寄存器”a”.
- …编辑器操作
- ”q”: 结束记录输入的字符。
- ”@a”: 执行寄存器”a 的内容”.

对于 Emacs, 请按下面操作。

- ”C-x (”: 开始定义一个键盘宏.
- …编辑器操作
- ”C-x)”: 结束定义一个键盘宏.
- ”C-x e”: 执行一个键盘宏.

9.3.8 记录 X 应用程序的图像

有少量方法可以记录 X 应用程序的图像，包括 xterm 显示。

软件包	流行度	大小	屏幕	命令
gnome-screenshot	V:12, I:91	1115	Wayland	GNOME 下的截屏程序
flameshot	V:7, I:17	3542	Wayland	强大的截屏程序
gimp	V:40, I:206	33386	Wayland + X	GUI 菜单中的截屏
x11-apps	V:32, I:440	2461	X	xwd(1)
imagemagick	V:7, I:266	82	X	import(1)
scrot	V:4, I:47	144	X	scrot(1)

Table 9.6: 图形图像处理工具列表

9.3.9 记录配置文件的变更

有特定的工具可以通过 DVCS 的帮助来记录配置文件的变更和在 Btrfs 上制作系统快照。

软件包	流行度	大小	说明
etckeeper	V:23, I:26	156	使用 Git（默认）、Mercurial 或 GNU Bazaar 来保存配置文件和它们的元数据
timeshift	V:7, I:13	4700	使用 rsync 或 BTRFS 快照的系统恢复工具
snapper	V:6.3, I:8.4	2461	Linux 文件系统快照管理工具

Table 9.7: 记录配置历史的软件包列表

你也可以考虑本地脚本第 10.2.3 节方案。

软件包	流行度	大小	说明
coreutils	V:914, I:1000	19751	nice(1): 用指定的调度优先权运行一个程序
bsdutils	V:445, I:1000	338	renice(1): 调整一个目前在运行的进程的调度优先权值
procps	V:845, I:998	2692	"/proc" 文件系统工具: ps(1), top(1), kill(1), watch(1), ⋯
psmisc	V:404, I:705	950	"/proc" 文件系统工具: killall(1), fuser(1), peekfd(1), pstree(1)
time	V:7, I:73	148	time(1): 运行一个程序，并从时间消耗方面来报告系统资源的使用
sysstat	V:112, I:163	1904	sar(1), iostat(1), mpstat(1), ⋯: linux 系统性能工具
isag	V:0.1, I:3.1	109	sysstat 的交互式的系统活动图
lsof	V:463, I:949	492	lsof(8): 使用"-p" 选项列出被一个系统进程打开的文件
strace	V:11, I:99	3250	strace(1): 跟踪系统调用和信号
ltrace	V:1, I:11	420	ltrace(1): 跟踪库调用
xtrace	V:0.03, I:0.65	342	xtrace(1): 跟踪 X11 客户端和服务器端之间的通信
powertop	V:38, I:231	696	powertop(1): 系统能耗使用信息
cron	V:929, I:997	248	根据 cron(8) 后台守护进程（daemon）的调度运行一个进程
anacron	V:424, I:480	114	用于非整天 24 小时运行系统的命令计划，类 cron
at	V:68, I:87	158	at(1) 或 batch(1): 在一个特定的时间运行任务或在某一系统负载下运行

Table 9.8: 监控和控制程序活动工具列表

9.4 监控、控制和启动程序活动

程序活动能够使用特殊的工具监控和控制。

> 提示
> procps 包提供了非常基础的监控、控制程序活动功能和启动程序功能。你应当把他们全部学会。

9.4.1 进程耗时

显示命令调用进程的时间消耗。

```
# time some_command >/dev/null
real    0m0.035s        # time on wall clock (elapsed real time)
user    0m0.000s        # time in user mode
sys     0m0.020s        # time in kernel mode
```

9.4.2 调度优先级

进程的调度优先级是被一个进程优先级值控制。

```
# nice  -19 top                                        # very nice
# nice --20 wodim -v -eject speed=2 dev=0,0 disk.img # very fast
```

在某些情况下，极端的进程优先级值会对系统造成伤害。小心使用这个命令。

进程优先级值	调度优先级
19	最低优先级进程
0	非常高的普通用户优先级进程
-20	root 用户非常高的优先级进程

Table 9.9: 调度优先级值列表

样式	典型的命令	特征
BSD	`ps aux`	显示%CPU %MEM
System V	`ps -efH`	显示 PPID

Table 9.10: ps 命令样式列表

9.4.3 ps 命令

在 Debian 系统上的 ps(1) 命令同时支持 BSD 和 SystemV 特征，有助于识别静态的进程活动。

对于僵尸（死了的）子进程，你能够通过”PPID” 字段的父进程 ID 来杀死它们。

pstree(1) 命令显示进程树。

9.4.4 top 命令

Debian 系统上的 top(1) 拥有丰富的特征，有助于识别进程有趣的动态行为。

它是一个交互式的全屏程序。你可以通过按”h” 键来得到它的使用帮助，按”q” 键来终止该程序。

9.4.5 列出被一个进程打开的文件

你能够通过一个进程 ID(PID) 来列出该进程所有打开的文件，例如，PID 为 1 的进程，使用下面的方式。

```
$ sudo lsof -p 1
```

PID=1 通常用于 `init` 程序.

9.4.6 跟踪程序活动

你能够跟踪程序活动，使用 strace(1), ltrace(1), xtrace(1) 来跟踪系统调用和信号、库调用、X11 客户端和服务器端之间的通信。

跟踪 `ls` 命令的系统调用。

```
$ sudo strace ls
```

提示
使用在 **/usr/share/doc/strace/examples/** 中发现的 **strace-graph** 脚本来生成一个好看的树形视图

9.4.7 识别使用文件和套接字的进程

你可以通过 fuser(1) 来识别出使用文件的进程，例如，用下面的方式识别出"/var/log/mail.log" 由哪个进程打开。

```
$ sudo fuser -v /var/log/mail.log
                     USER        PID ACCESS COMMAND
/var/log/mail.log:   root       2946 F.... rsyslogd
```

你可以看到"/var/log/mail.log" 是由 rsyslogd(8) 命令打开并写入。

你可以通过 fuser(1) 来识别出使用套接字的进程，例如，用下面的方式识别出"smtp/tcp" 由哪个进程打开。

```
$ sudo fuser -v smtp/tcp
                     USER        PID ACCESS COMMAND
smtp/tcp:            Debian-exim   3379 F.... exim4
```

现在你知道你的系统运行 exim4(8) 来处理连接到 SMTP 端口 (25) 的 TCP 连接.

9.4.8 使用固定间隔重复一个命令

watch(1) 使用固定间隔重新执行一个命令，并全屏显示输出。

```
$ watch w
```

显示哪些人登录到系统，每 2 秒钟更新一次。

9.4.9 使用文件循环来重复一个命令

通过匹配某些条件的文件来循环重复一个命令，有几种方法，例如，匹配全局模式"*.ext".

- Shell 循环方式 (参见第 12.1.4 节):

```
for x in *.ext; do if [ -f "$x"]; then command "$x" ; fi; done
```

- find(1) 和 xargs(1) 联合:

```
find . -type f -maxdepth 1 -name '*.ext' -print0 | xargs -0 -n 1 command
```

- find(1) 使用"-exec" 选项并执行命令:

```
find . -type f -maxdepth 1 -name '*.ext' -exec command '{}' \;
```

- find(1) 使用"-exec" 选项并执行一个短的 shell 脚本:

```
find . -type f -maxdepth 1 -name '*.ext' -exec sh -c "command '{}' && echo 'successful'" \;
```

上面的列子确保适当处理怪异的文件名（如包含空格）。find(1) 更多高级的用法，参见第 10.1.5 节.

9.4.10 从 GUI 启动一个程序

对于命令行界面（command-line interface，CLI），$PATH 环境变量所指定的目录中第一个匹配相应名称的程序会被执行。参见第 1.5.3 节。

对于遵从 freedesktop.org 标准的图形用户界面（graphical user interface，GUI），/usr/share/applications/ 目录中的 *.desktop 文件给每个程序的 GUI 菜单显示提供了必要的属性。遵从 Freedesktop.org xdg 菜单系统的每一个软件包，通过”/usr/share/applications/” 下”*.desktop” 提供的数据来安装它的菜单。遵从 Freedesktop.org 标准的现代桌面环境，用 xdg-utils 软件包利用这些数据生成它们的菜单。参见”/usr/share/doc/xdg-utils/README”。

举个例子，chromium.desktop 文件中为“Chromium 网络浏览器”定义了相关属性，例如程序名“Name”，程序执行路径和参数“Exec”，所使用的图标“Icon”等等（参见桌面配置项规范）。文件内容如下：

```
[Desktop Entry]
Version=1.0
Name=Chromium Web Browser
GenericName=Web Browser
Comment=Access the Internet
Comment[fr]=Explorer le Web
Exec=/usr/bin/chromium %U
Terminal=false
X-MultipleArgs=false
Type=Application
Icon=chromium
Categories=Network;WebBrowser;
MimeType=text/html;text/xml;application/xhtml_xml;x-scheme-handler/http;x-scheme-handler/ ↩
    https;
StartupWMClass=Chromium
StartupNotify=true
```

这是一个较为简单的说明。*.desktop 文件像下面那样被搜寻。

桌面环境设置 $XDG_DATA_HOME 和 $XDG_DATA_DIR 环境变量。举个例子，在 GNOME 中：

- 未设置 $XDG_DATA_HOME。（将使用默认值 $HOME/.local/share。）
- $XDG_DATA_DIRS 被设置为 /usr/share/gnome:/usr/local/share/:/usr/share/。

基准目录（参见 XDG Base Directory Specification）和应用程序目录如下所示。

- $HOME/.local/share/ → $HOME/.local/share/applications/
- /usr/share/gnome/ → /usr/share/gnome/applications/
- /usr/local/share/ → /usr/local/share/applications/
- /usr/share/ → /usr/share/applications/

*.desktop 文件将按照这个顺序在这些 applications 目录中进行搜寻。

提示
要建立一个用户自定义的 GUI 菜单项，需要在 $HOME/.local/share/applications/ 目录中添加一个 *.desktop 文件。

提示
”Exec=...” 行不会由 shell 解析。如果需要设置环境变量，使用 env(1) command。

提示
相似地，如果在这些基准目录下的 autostart 目录中建立了一个 *.desktop 文件，则 *.desktop 文件中指定的程序会在桌面环境启动时自动执行。参见 Desktop Application Autostart Specification。

提示
相似地，如果在 $HOME/Desktop 目录中建立了一个 *.desktop 文件并且桌面环境被配置为支持桌面图标启动器功能，则点击图标时指定的程序会被执行。请注意，$HOME/Desktop 目录的实际名称与语言环境有关。参见 xdg-user-dirs-update(1)。

9.4.11 自定义被启动的程序

一些程序会被另一个程序自动启动。下面是自定义该过程的方法。

- 应用程序配置菜单：
  - GNOME 桌面：“设置”→ “应用程序”→ “默认应用程序”
  - KDE 桌面：“应用开始菜单”→ “设置”→ “系统设置”→ “应用程序”→ “默认应用程序”
  - Iceweasel 浏览器：“编辑”→ “首选项”→ “应用程序”
  - mc(1)：“/etc/mc/mc.ext”
- 例如“$BROWSER”、“$EDITOR”、“$VISUAL”和“$PAGER”这样的环境变量（参见 environ(7)）
- 用于例如“editor”、“view”、“x-www-browser”、“gnome-www-browser”和“www-browser”这样的程序的 update-alternatives(1) 系统（参见第 1.4.7 节）
- “~/.mailcap”和“/etc/mailcap”文件的内容关联了程序的 MIME 类型（参见 mailcap(5)）
- “~/.mime.types”和“/etc/mime.types”文件的内容关联了 MIME 类型的文件扩展名（参见 run-mailcap(1)）

提示
update-mime(8) 会更新“/etc/mailcap” 文件，期间会用到“/etc/mailcap.order” 文件（参见 mailcap.order(5)).

提示
debianutils 软件包提供 sensible-browser(1)、sensible-editor(1) 和 sensible-pager(1)，它们可以分别对要调用的编辑器、分页程序和网络浏览器作出明智的选择。我建议你阅读那些 shell 脚本。

提示
为了在 GUI（图形用户界面）下运行例如 mutt 这样的控制台应用程序来作为你的首选应用程序，你应该像下面那样建立一个 GUI（图形用户界面）应用程序并设置“/usr/local/bin/mutt-term”为你想要启动的首选应用程序。

```
# cat /usr/local/bin/mutt-term <<EOF
#!/bin/sh
gnome-terminal -e "mutt \$@"
EOF
# chmod 755 /usr/local/bin/mutt-term
```

信号值	信号名	操作	注释
0	---	没有信号发送 (参见 kill(2))	检查进程是否运行
1	SIGHUP	终止进程	从终端断开连接（信号挂起）
2	SIGINT	终止进程	从键盘中断 (CTRL-C)
3	SIGQUIT	终止进程并触发 dump core	从键盘退出 (CTRL-\)
9	SIGKILL	终止进程	不可阻塞的 kill 信号
15	SIGTERM	终止进程	可被阻塞的终止信号

Table 9.11: kill 命令常用信号列表

9.4.12 杀死一个进程

使用 kill(1) 通过进程 ID 来杀死（发送一个信号）一个进程。

使用 killall(1) 或 pkill(1) 通过进程命令的名字或其它属性来做同样的事情。

9.4.13 单次任务时间安排

运行 at(1) 命令来安排一次性的工作。

```
$ echo 'command -args'| at 3:40 monday
```

9.4.14 定时任务安排

使用 cron(8) 来进行定时任务安排。参见 crontab(1) 和 crontab(5).

你能够作为一个普通用户定时运行一个进程，比如，foo 使用”crontab -e”命令创建一个 crontab(5) 的文件”/var/spool

这里是一个 crontab(5) 文件的列子。

```
# use /usr/bin/sh to run commands, no matter what /etc/passwd says
SHELL=/bin/sh
# mail any output to paul, no matter whose crontab this is
MAILTO=paul
# Min Hour DayOfMonth Month DayOfWeek command (Day... are OR'ed)
# run at 00:05, every day
5  0  *  * *   $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# run at 14:15 on the first of every month -- output mailed to paul
15 14 1  * *   $HOME/bin/monthly
# run at 22:00 on weekdays(1-5), annoy Joe. % for newline, last % for cc:
0 22 *   * 1-5 mail -s "It's 10pm" joe%Joe,%%Where are your kids?%.%%
23 */2 1 2 *   echo "run 23 minutes after 0am, 2am, 4am ..., on Feb 1"
```

```
5  4 *   * sun echo "run at 04:05 every Sunday"
# run at 03:40 on the first Monday of each month
40 3 1-7 * *   [ "$(date +%a)" == "Mon" ] && command -args
```

提示
对那些非连续运行的系统，安装 anacron 软件包来定时执行周期性的命令，命令在接近机器启动的时间运行，并允许有特定的时间间隔。参见 anacron(8) 和 anacrontab(5).

提示
对于定时系统维护脚本，你能够以 root 账户定时运行，把这类脚本放入"/etc/cron.hourly/", "/etc/cron.daily/", "/etc/cron.weekly/", 或"/etc/cron.monthly/". 这些脚本的执行时间，可以通过"/etc/crontab" 和"/etc/anacrontab" 来定制。

cron 后台守护进程(daemon)不存在时，Systemd 也有按时间计划运行程序的低级能力。例如，/lib/systemd/system/a 和 /lib/systemd/system/apt-daily.service 建立每天的 apt 下载行动。参见 systemd.timer(5) 。

9.4.15 基于事件的计划任务

Systemd 能够执行计划程序，不仅基于时间事件，还能够基于挂载事件。参见第 10.2.3.3 节和第 10.2.3.2 节的例子。

9.4.16 Alt-SysRq 键

按 Alt-SysRq (PrtScr) 组合键跟一个字母按键，进行不可思议的系统应急控制。

Alt-SysRq 之后的键	行为描述
k	**k**ill 杀死在当前虚拟控制台上的所有进程 (SAK)
s	**s**ync 同步刷新所有已经挂载的文件系统来避免数据损坏
u	重新以只读方式挂载所有已挂载的文件系统 (**u**mount)
r	在 X 崩溃后，从 **r**aw 模式恢复键盘

Table 9.12: 著名的 SAK 命令键列表

更多信息参见 Linux 内核用户和管理员手册 » Linux Magic System Request Key Hacks

提示
从 SSH 终端等，你能够通过向"/proc/sysrq-trigger" 写入内容来使用 Alt-SysRq 特性。例如，从 root shell 提示符运行"echo s > /proc/sysrq-trigger; echo u > /proc/sysrq-trigger" 来 **s**yncs 和 **u**mounts 所有已挂载的文件系统。

目前 (2021) Debian amd64 Linux 内核为 /proc/sys/kernel/sysrq=438=0b110110110:

- 2 = 0x2 - 启用控制台日志级别控制 (打开)
- 4 = 0x4 - 启用键盘控制 (SAK, unraw) (打开)
- 8 = 0x8 - 启用进程调试转储（debugging dumps of processes）等。（关闭）

- 16 = 0x10 - 启用 sync 命令 (打开)
- 32 = 0x20 - 启用只读重新挂载（打开）
- 64 = 0x40 - 启用进程信号 (term, kill, oom-kill) (关闭)
- 128 = 0x80 - 允许重启、关闭电源（打开）
- 256 = 0x100 - 允许调整所有 RT（实时）任务优先级（打开）

9.5 系统维护技巧

9.5.1 谁在系统里？

你可以通过下面的方法检查谁登录在系统里。

- who(1) 显示谁登录在系统里面。
- w(1) 显示谁登录在系统里面，他们正在在做什么。
- last(1) 显示用户最后登录的列表。
- lastb(1) 显示用户最后错误登录的列表。

提示
”`/var/run/utmp`” 和”`/var/log/wtmp`” 存储这样的用户信息。参见 login(1) 和 utmp(5).

9.5.2 警告所有人

你可以通过下面的方式使用 wall(1) 给登录系统的每一个人发送信息。

```
$ echo "We are shutting down in 1 hour" | wall
```

9.5.3 硬件识别

对于 PCI 类设备 (AGP, PCI-Express, CardBus, ExpressCard 等), 一开始就使用 lspci(8) (也许加上”`-nn`” 选项) 进行硬件识别比较好。

此外，你可以通过阅读”`/proc/bus/pci/devices`” 里面的内容或浏览”`/sys/bus/pci`” 下面的目录树来进行硬件识别 (参见第 1.2.12 节).

9.5.4 硬件配置

像 GNOME 和 KDE 这类现代图形桌面系统，虽然大部分硬件的配置都能够通过相应的图形配置工具来管理，但知道一些配置它们的基础方式，也是一个好的主意。

对于键盘和终端配置，参见第 8.2 节和第 8.3 节。

这里, ACPI 是一个比 APM 新的电源管理系统框架。

提示
现代系统的 CPU 频率调整功能，是由内核模块 `acpi_cpufreq` 管理的。

软件包	流行度	大小	说明
pciutils	V:279, I:994	289	Linux PCI 工具: lspci(8)
usbutils	V:146, I:894	322	Linux USB 工具: lsusb(8)
nvme-cli	V:30, I:39	2222	Linux NVMe 工具: nvme(1)
pcmciautils	V:3.8, I:5.7	92	Linux PCMCIA 工具: pccardctl(8)
scsitools	V:0.2, I:2.0	261	SCSI 硬件管理工具集: lsscsi(8)
procinfo	V:0.3, I:5.7	149	从"/proc": lsdev(8) 获得系统信息
lshw	V:12, I:87	971	硬件配置信息: lshw(1)
discover	V:26, I:527	81	硬件识别系统: discover(8)

Table 9.13: 硬件识别工具列表

软件包	流行度	大小	说明
console-setup	V:87, I:977	422	Linux 控制台字体和键盘表工具
keyd	V:1.7, I:1.8	1437	键盘按键重映射守护使用内核层的输入基元（evdev, uinput）
keyd-application-mapper	V:0.02, I:0.07	34	键盘按键重映射后台守护进程（daemon）——特定应用层的重映射
x11-xserver-utils	V:376, I:532	559	X 服务端工具: xset(1), xmodmap(1)
acpid	V:51, I:74	158	管理高级可配置和电源接口 (ACPI) 事件分发的后台守护进程（daemon）
acpi	V:6, I:69	49	显示 ACPI 设备信息的工具
sleepd	V:0.08, I:0.10	84	在笔记本空闲时，使其进入休眠状态的后台守护进程（daemon）
hdparm	V:111, I:192	246	硬盘访问优化 (参见第 9.6.9 节)
smartmontools	V:249, I:274	2455	使用 S.M.A.R.T. 控制和监控存储系统
setserial	V:3.0, I:4.7	104	串口管理工具集
memtest86+	V:1, I:19	12483	内存硬件管理工具集
scsitools	V:0.2, I:2.0	261	SCSI 硬件管理工具集
setcd	V:0.06, I:0.59	33	光驱访问优化
big-cursor	I:0.57	26	X 系统的大鼠标光标

Table 9.14: 硬件配置工具列表

9.5.5 系统时间和硬件时间

下面设置系统的硬件时间为：MM/DD hh:mm, CCYY.

```
# date MMDDhhmmCCYY
# hwclock --utc --systohc
# hwclock --show
```

Debian 系统的时间通常显示为本地时间，但硬件时间通常使用 UTC(GMT) 时间。

如果硬件时间设置为 UTC 时间，请在“`/etc/default/rcS`”里面设置“`UTC=yes`”。

下面是重新配置 Debian 系统使用的时区。

```
# dpkg-reconfigure tzdata
```

如果你希望通过网络来更新系统时间，考虑使用 `systemd-timesyncd` 和 `ntpsec` 和 `ntpsec-ntpdate` 和 `chrony` 这类包提供的 NTP 服务。

> 提示
> 在 systemd 下，是使用 `systemd-timesyncd` 来替代进行网络时间同步。参见 systemd-timesyncd(8).

参见下面内容。

- 精确时间和日期管理 HOWTO
- NTP 公共服务项目
- `ntpsec-doc` 包

> 提示
> 在 `ntp` 包里面的 ntptrace(1) 能够跟踪 NTP 服务链至原始源。

9.5.6 声音基础设施

用于现在的 Linux 的声卡设备驱动程序由高级 Linux 声音体系（Advanced Linux Sound Architecture，ALSA） 提供。ALSA 提供了兼容之前的开放声音系统（Open Sound System，OSS） 的模拟模式。

应用软件不仅可被配置为不仅直接访问声音设备，也可以通过一些标准化声音服务端系统来访问它们。目前，PulseAudio、JACK 和 PipeWire 被用作声音的服务端系统。参见 Debian 维基声音页面得到最新情况。

每个流行的桌面环境通常都有一个通用的声音引擎。每个被应用程序使用的声音引擎都可以选择连接到不同的声音服务器。

> 提示
> 使用“`cat /dev/urandom > /dev/audio`”或 speaker-test(1) 来测试扬声器（^C 停止）。

> 提示
> 如果你无法听到声音，那你的扬声器可能连接到了一个静音输出。现代的声音系统有许多输出。`alsa-utils` 软件包中的 alsamixer(1) 可以很好地配置声音和静音设置。

软件包	流行度	大小	说明
alsa-utils	V:349, I:457	2700	配置和使用 ALSA 的工具
oss-compat	V:0.5, I:8.7	21	在 ALSA 下兼容 OSS，预防“/dev/dsp not found”错误
pipewire	V:330, I:370	140	音频和视频处理引擎多媒体服务端 - 元数据包
pipewire-bin	V:336, I:370	2254	音频和视频处理引擎多媒体服务端 - 音频服务和命令行程序
pipewire-alsa	V:175, I:243	197	音频和视频处理引擎多媒体服务端 - 代替 ALSA 的音频服务
pipewire-pulse	V:304, I:350	56	音频和视频处理引擎多媒体服务端 - 代替 PulseAudio 的音频服务
pulseaudio	V:144, I:166	6616	PulseAudio 服务端
libpulse0	V:445, I:578	984	PulseAudio 客户端库
jackd	V:2, I:13	8	JACK Audio Connection Kit. (JACK) 服务器（低延迟）
libjack0	V:1.7, I:8.3	320	JACK Audio Connection Kit. (JACK) 库（低延迟）
libphonon4qt5-4	V:16, I:38	572	Phonon：KDE 声音引擎

Table 9.15: 声音软件包

环境	命令
Linux 控制台	setterm -powersave off
X 窗口（关闭屏幕保护）	xset s off
X 窗口（关闭 dpms）	xset -dpms
X 窗口（屏幕保护 GUI 配置）	xscreensaver-command -prefs

Table 9.16: 关闭屏幕保护命令列表

9.5.7 关闭屏幕保护

关闭屏幕保护，使用下面的命令。

9.5.8 关闭蜂鸣声

可以把电脑的扬声器拔掉来关闭蜂鸣声。把 pcspkr 内核模块删除，也可以做到这点。

bash(1) 用到的 readline(3) 程序，当遇到告警字符 (ASCII=7) 时，将会发生。下面的操作将阻止发生。

```
$ echo "set bell-style none">> ~/.inputrc
```

9.5.9 内存使用

对你来说，这里有两种可用的方法来得到内存的使用情况。

- “/var/log/dmesg”中的内核启动信息包含了可用内存的精确总大小。
- free(1) 和 top(1) 显示正在运行的系统中内存资源的相关信息。

下面是一个例子。

```
# grep '\] Memory' /var/log/dmesg
[    0.004000] Memory: 990528k/1016784k available (1975k kernel code, 25868k reserved, 931k ↩
     data, 296k init)
$ free -k
              total        used        free      shared     buffers      cached
```

```
Mem:        997184     976928      20256          0     129592     171932
-/+ buffers/cache:     675404     321780
Swap:      4545576          4    4545572
```

你可能会觉得奇怪：“dmesg 告诉你 free 为 990 MB，而 free -k 告诉你 free 为 320 MB。这丢失了超过 600 MB ……”。

别担心“`Mem:`”这行中“`used`”较大的值以及“`free`”较小的值，放轻松，你需要查看的是下面的那个（在上面的例子中它们是 675404 和 321780）。

对于我的 MacBook，有 1GB=1048576k 内存（显卡系统用掉一些），我看到的如下。

报告	大小
dmesg 中 total 的大小	1016784k = 1GB - 31792k
dmesg 中的 free	990528k
shell 下的 total	997184k
shell 下的 free	20256k（但有效的为 321780k）

Table 9.17: 报告的内存大小

9.5.10 系统安全性和完整性检查

糟糕的系统维护可能会暴露你的系统，导致它被外部非法使用。

对于系统安全性和完整性的检查，你需要从下面这些方面开始。

- `debsums` 软件包，参见 debsums(1) 和第 2.5.2 节。
- `chkrootkit` 软件包，参见 chkrootkit(1)。
- `clamav` 软件包家族，参见 clamscan(1) 和 freshclam(1)。
- Debian security FAQ。
- Securing Debian Manual。

软件包	流行度	大小	说明
`logcheck`	V:4.2, I:5.2	120	后台守护进程（daemon），将系统日志文件中的异常通过邮件发送给管理员
`debsums`	V:4, I:29	108	实用程序，使用 MD5 校验码对已安装软件包的文件进行校验
`chkrootkit`	V:10, I:14	988	rootkit 检测软件
`clamav`	V:8, I:37	20551	Unix 的反病毒实用程序——命令行界面
`tiger`	V:1.0, I:1.3	7800	报告系统安全漏洞
`tripwire`	V:1.1, I:1.4	5060	文件和目录完整性检测软件
`john`	V:1.1, I:7.1	469	先进的密码破解工具
`aide`	V:2.1, I:2.7	328	高级入侵环境检测——静态二进制
`integrit`	V:0.07, I:0.16	2759	文件完整性验证程序
`crack`	V:0.04, I:0.52	151	密码猜测程序

Table 9.18: 用于系统安全性和完整性检查的工具

下面是一个简单的脚本，用来检测典型的所有人可写的错误文件权限。

```
# find / -perm 777 -a \! -type s -a \! -type l -a \! \( -type d -a -perm 1777 \)
```



小心
由于 debsums 软件包使用本地存储的 MD5 校验码，因此面对恶意攻击，也不能完全相信系统安全性检测工具。

9.6 数据存储技巧

用 Linux 即用系统 (参见第 3.2.2 节和第 3.2.3 节) 启动你的系统，这将使在已经安装的系统上重新配置数据存储变得容易。

注意
对硬盘 (HDD) 的描述也适合用于其它存储设备，比如 SSD / U 盘 / 内存卡 /……。替换例子中 /dev/sda 的设备名称，用其适合的设备名称 /dev/nvme0、/dev/mmcblk0……。

如果设备在 GUI（图形用户界面）桌面系统下被自动挂载，在对它们进行操作前，你可能需要从命令行手工 umount(8) 这些设备。

9.6.1 硬盘空间使用情况

硬盘空间的使用情况可以通过 mount、coreutils 和 xdu 软件包提供的程序来评估：

- mount(8) 显示所有挂载的文件系统 (= 磁盘).
- df(1) 报告文件系统使用的硬盘空间。
- du(1) 报告目录树使用的硬盘空间。

提示
你可以将 du(8) 的输出传输给 xdu(1)，来使用它的图形交互式演示，例如“du -k . |xdu”、“sudo du -k -x / |xdu”等等。

9.6.2 硬盘分区配置

对于硬盘分区配置，尽管 fdisk(8) 被认为是标准的配置，但是 parted(8) 工具还是值得注意的。

老的 PC 使用经典的主引导记录（Master Boot Record，MBR） 方案，将硬盘分区数据保存在第一个扇区，即 LBA 扇区 0（512 字节）。

一些带有统一可扩展固件接口（Unified Extensible Firmware Interface，UEFI） 的近代 PC，包括基于 Intel 的 Mac，使用全局唯一标识分区表（GUID Partition Table，GPT） 方案，硬盘分区数据不保存在第一个扇区。

尽管 fdisk(8) 一直是硬盘分区的标准工具，但现在 parted(8) 替代了它。



小心
尽管 parted(8) 声称也能用来创建和调整文件系统，但使用维护最好的专门工具来做这些事会更为安全，例如 mkfs(8)（mkfs.msdos(8)、mkfs.ext2(8)、mkfs.ext3(8)、mkfs.ext4(8)……）和 resize2fs(8)。

软件包	流行度	大小	说明
util-linux	V:922, I:1000	4826	多种系统工具，包含 fdisk(8) 和 cfdisk(8)
parted	V:443, I:566	126	GNU Parted，硬盘分区调整程序
gparted	V:12, I:81	2313	基于 libparted 的 GNOME 分区编辑程序
gdisk	V:20, I:240	967	用于 GPT/MBR 并存的硬盘的分区编辑程序
kpartx	V:16, I:23	78	为分区建立设备映射的程序

Table 9.19: 硬盘分区管理软件包

注意
为了在 GPT 和 MBR 之间切换，你需要直接删除开头的几个块中的内容（参见第 9.8.6 节）并使用“parted /dev/sdx mklabel gpt”或“parted /dev/sdx mklabel msdos”来设置它。请注意，这里使用的“msdos”是用于 MBR。

9.6.3 使用 UUID 访问分区

尽管重新配置你的分区或可移动存储介质的激活顺序可能会给分区产生不同的名字，但你可以使用同一个 UUID 来访问它们。如果你有多个硬盘并且你的 BIOS/UEFI 没有给它们一致的设备名的话，使用 UUID 是不错的选择。

- mount(8) 命令带有“-U”选项可以使用 UUID 来挂载一个块设备，而不必使用他的文件名称，例如“/dev/sda3”。
- “/etc/fstab”（参见 fstab(5)）可以使用 UUID。
- 引载加载程序（第 3.1.2 节）也可以使用 UUID。

提示
你可以使用 blkid(8) 来查看一个特定块设备的 UUID。
你也可以使用”lsblk -f” 来检测 UUID 并查看其它信息。

9.6.4 LVM2

LVM2 是一个用于 Linux 内核的逻辑卷管理器。使用 LVM2 的话，硬盘分区可以创建在逻辑卷上来替代物理硬盘。

LVM 有下列需求。

- Linux 内核中的设备映射支持（Debian 内核默认支持）
- 用户自定义设备映射支持库（libdevmapper* 软件包）
- 用户自定义 LVM2 工具（lvm2 软件包）

请从下面的 man 手册开始了解 LVM2。

- lvm(8)：LVM2 机制的基础知识（列出了所有 LVM2 命令）
- lvm.conf(5)：LVM2 的配置文件
- lvs(8)：报告逻辑卷的相关信息
- vgs(8)：报告卷组的相关信息
- pvs(8)：报告物理卷的相关信息

9.6.5 文件系统配置

对于 ext4 文件系统, `e2fsprogs` 包提供下面的工具。

- mkfs.ext4(8) 创建新的 ext4 文件系统
- fsck.ext4(8) 检查和修复现有 ext4 文件系统
- tune2fs(8) 配置 ext4 文件系统的超级块
- debugfs(8) 交互式的调试 ext4 文件系统. (它有 `undel` 命令来恢复已经删除的文件.)

mkfs(8) 和 fsck(8) 命令是由 `e2fsprogs` 包提供的，是各种文件系统相关程序的前端。(`mkfs.fstype` 和 `fsck.fstype`). 对于 ext4 文件系统, 它们是 mkfs.ext4(8) 和 fsck.ext4(8) (它们被符号链接到 mke2fs(8) 和 e2fsck(8)).

Linux 支持的每一个文件系统，有相似的命令。

软件包	流行度	大小	说明
`e2fsprogs`	V:842, I:998	1560	ext2/ext3/ext4 文件系统工具
`btrfs-progs`	V:54, I:78	5291	Btrfs 文件系统工具
`reiserfsprogs`	V:12, I:21	473	Reiserfs 文件系统工具
`zfsutils-linux`	V:32, I:33	1905	OpenZFS 文件系统工具
`dosfstools`	V:274, I:568	310	FAT 文件系统工具. (Microsoft: MS-DOS, Windows)
`exfatprogs`	V:36, I:466	624	exFAT 文件系统工具，由三星维护。
`exfat-fuse`	V:2, I:37	73	FUSE 读写 exFAT 文件系统（微软）驱动。
`xfsprogs`	V:46, I:82	4651	XFS 文件系统工具. (SGI: IRIX)
`ntfs-3g`	V:197, I:510	1535	FUSE 读写 NTFS 文件系统（微软：Windows NT……）驱动。
`jfsutils`	V:0.4, I:6.6	1104	JFS 文件系统工具. (IBM: AIX, OS/2)
`xorriso`	V:16, I:66	378	ISO-9660 文件系统工具和来自 libburnia 的 CD/DVD 刻录工具
`wodim`	V:7, I:81	898	来自 cdrkit 的命令行 CD/DVD 刻录工具
`genisoimage`	V:17, I:150	1567	来自 cdrkit 的 ISO-9660 文件系统命令行工具
`hfsprogs`	V:0.3, I:3.0	394	HFS 和 HFS Plus 文件系统工具. (Apple: Mac OS)
`zerofree`	V:9, I:114	30	把 ext2/3/4 文件系统上空闲块设置为零的程序

Table 9.20: 文件系统管理包列表

提示
Ext4 文件系统是 Linux 系统上默认的文件系统，强烈推荐使用这个文件系统，除非你有特殊的理由不使用。
Btrfs 状态能够在 Debian wiki on btrfs 和 kernel.org wiki on btrfs 发现。它被期望作为 ext4 文件系统之后的下一个默认文件系统。
一些工具可以在没有 Linux 内核支持的情况下访问文件系统 (参见第 9.8.2 节).

9.6.6 文件系统创建和完整性检查

mkfs(8) 在 Linux 系统上创建文件系统。fsck(8) 命令在 Linux 系统上提供文件系统完整性检查和修复功能。

在文件系统创建后，Debian 现在默认不周期性的运行 `fsck`。



小心
在已经挂载的文件系统上运行 `fsck` ，一般是不安全的.

提示
在"/etc/mke2fs.conf" 里设置"enable_periodic_fsck" 并使用"tune2fs -c0 /dev/*partition_name*" 设置最大挂载数为 0，便可以在重启时，让 root 文件系统包括在内的所有文件系统上，安全的运行 fsck(8) 命令. 参见 mke2fs.conf(5) 和 tune2fs(8).
从启动脚本里面运行的 fsck(8) 命令结果，可以在"/var/log/fsck/" 目录下查看。

9.6.7 通过挂载选项优化文件系统

"/etc/fstab"中包含了基础的静态文件系统配置。例如，

```
«file system»                    «mount point» «type» «options»    «dump» «pass»
proc                                       /proc proc   defaults          0 0
UUID=709cbe4c-80c1-56db-8ab1-dbce3146d2f7 /     ext4   errors=remount-ro 0 1
UUID=817bae6b-45d2-5aca-4d2a-1267ab46ac23 none  swap   sw                0 0
/dev/scd0                          /media/cdrom0  udf,iso9660 user,noauto  0 0
```

提示
UUID（参见第 9.6.3 节）可以替代一般的块设备名称（例如"/dev/sda1"、"/dev/sda2"……）来识别一个块设备。

从 Linux 2.6.30 起，内核的默认行为是提供"relatime" 选项。

参见 fstab(5) 和 mount(8)。

9.6.8 通过超级块（superblock）优化文件系统

一个文件系统的特性可以使用 tune2fs(8) 命令通过超级块来优化。

- 执行"sudo tune2fs -l /dev/sda1"可以显示"/dev/sda1"上的文件系统超级块内容。
- 执行"sudo tune2fs -c 50 /dev/sda1" 改变"/dev/sda1" 文件系统检查 (在启动时执行 fsck) 的频率为每 50 次启动.
- 执行"sudo tune2fs -j /dev/sda1"会给文件系统添加日志功能，即"/dev/sda1"的文件系统从 ext2 转换为 ext3。（对未挂载的文件系统这么做。）
- 执行"sudo tune2fs -O extents,uninit_bg,dir_index /dev/sda1 && fsck -pf /dev/sda1"在"/dev/sd 上将它从 ext3 转换为 ext4。（对未挂载的系统这么做。）

提示
尽管 tune2fs(8) 的名字是这样的，但它不仅能用于 ext2 文件系统，也能用于 ext3 和 ext4 文件系统。

9.6.9 硬盘优化



警告
在你折腾硬盘配置之前，请检查你的硬件并阅读 hdparm(8) 的 man 手册页，因为这可能会对数据完整性造成相当大的危害。

你可以通过“hdparm -tT /dev/sda”来测试“/dev/sda”硬盘的访问速度。对于一些使用 (E)IDE 连接的硬盘，你可以使用“hdparm -q -c3 -d1 -u1 -m16 /dev/sda”来启用“(E)IDE 32 位支持”、启用“using_dma flag”、设置“interrupt-unmask flag”并设置“multiple 16 sector I/O”（危险！），从而加速硬盘访问速度。

你可以通过“hdparm -W /dev/sda”来测试“/dev/sda”硬盘的写入缓存功能。你可以使用“hdparm -W 0 /dev/sda”关闭写入缓存功能。

现代高速 CD-ROM 光驱，你可以使用”setcd -x 2” 降低速度，来读取不当压缩的 CDROM 光盘。

9.6.10 固态硬盘优化

固态硬盘（Solid state drive，SSD） 目前可以被自动检测。

在 /etc/fstab 里面，将易失性数据路径挂载为”tmpfs”，可以减少不必要的磁盘访问来阻止磁盘损耗。

9.6.11 使用 SMART 预测硬盘故障

你可以使用兼容 SMART 的 smartd(8) 后台守护进程（daemon）来监控和记录你的硬盘。

1. 在 BIOS 中启用 SMART 功能。
2. 安装 smartmontools 软件包。
3. 通过 df(1) 列出硬盘驱动并识别它们。
   - 让我们假设要监控的硬盘为“/dev/sda”。
4. 检查“smartctl -a /dev/sda”的输出，看 SMART 功能是否已启用。
   - 如果没有，通过“smartctl -s on -a /dev/sda”启用它。
5. 通过下列方式运行 smartd(8) 后台守护进程（daemon）。
   - 消除 /etc/default/smartmontools”文件中“start_smartd=yes”的注释。
   - 通过“sudo systemctl restart smartmontools”重新启动 smartd(8) 后台守护进程（daemon）。

提示
smartd(8) 后台守护进程（daemon）可以使用 /etc/smartd.conf 文件进行自定义，文件中包含了相关的警告。

9.6.12 通过 $TMPDIR 指定临时存储目录

应用程序一般在临时存储目录“/tmp”下建立临时文件。如果“/tmp”没有足够的空间，你可以通过 $TMPDIR 变量来为程序指定临时存储目录。

9.6.13 通过 LVM 扩展可用存储空间

在安装时创建在 Logical Volume Manager 逻辑卷管理 (LVM) (Linux 特性) 上的分区，它们可以容易的通过合并扩展或删除扩展的方式改变大小，而不需要在多个存储设备上进行大量的重新配置。

9.6.14 通过挂载另一个分区来扩展可用存储空间

如果你有一个空的分区（例如“/dev/sdx”），你可以使用 mkfs.ext4(1) 将它格式化，并使用 mount(8) 将它挂载到你需要更多空间的目录。（你需要复制原始数据内容。）

```
$ sudo mv work-dir old-dir
$ sudo mkfs.ext4 /dev/sdx
$ sudo mount -t ext4 /dev/sdx work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```

提示
你也可以选择挂载一个空硬盘映像文件（参见第 9.7.5 节）作为一个循环设备（参见第 9.7.3 节）。实际的硬盘使用量会随着实际存储数据的增加而增加。

9.6.15 通过“**mount --bind**”挂载另一个目录来扩展可用存储空间

如果你在另一个分区里有一个带有可用空间的空目录（例如“/path/to/emp-dir”），你可以通过带有“--bind”选项的 mount(8)，将它挂载到一个你需要更多空间的目录（例如“work-dir”）。

```
$ sudo mount --bind /path/to/emp-dir work-dir
```

9.6.16 通过 **overlay** 挂载（**overlay-mounting**）另一个目录来扩展可用存储空间

如果你在另一个分区表中有可用的空间（例如，“/path/to/empty”和“/path/to/work”），你可以在其中建立一个目录并堆栈到你需要空间的那个旧的目录（例如，“/path/to/old”），要这样做，你需要用于 Linux 3.18 版内核或更新版本（对应 Debian Stetch 9.0 或更新版本）的 OverlayFS。

```
$ sudo mount -t overlay overlay \
  -olowerdir=/path/to/old-dir,upperdir=/path/to/empty,workdir=/path/to/work
```

“/path/to/empty”和“/path/to/work”应该位于可读写的分区，从而能够写入“/path/to/old”。

9.6.17 使用符号链接扩展可用存储空间



小心
这是一个已弃用的做法。某些软件在遇到“软链接目录”时可能不会正常工作。请优先使用上文所述的“挂载”的途径。

如果你在另一个分区里有一个带有可用空间的空目录（例如“/path/to/emp-dir”），你可以使用 ln(8) 建立目录的一个符号链接。

```
$ sudo mv work-dir old-dir
$ sudo mkdir -p /path/to/emp-dir
$ sudo ln -sf /path/to/emp-dir work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```



警告
别对由系统管理的目录（例如“/opt”）使用“链接到目录”，这样的链接在系统升级时可能会被覆盖。

9.7 磁盘映像

我们在这里讨论磁盘影响的操作。

9.7.1 制作磁盘映像文件

一个未挂载设备（例如，第二个 SCSI 或串行 ATA 设备“/dev/sdb”）的磁盘映像文件“disk.img”，可以使用下列方式之一建立。

```
# dd if=/dev/sdb of=disk.img; sync
```

```
# cp /dev/sdb disk.img ; sync
```

```
# cat /dev/sdb > disk.img ; sync
```

传统 PC 中位于主 IDE 硬盘第一扇区的主引导记录（MBR）（参见第 9.6.2 节）的磁盘映像可以使用 dd(1) 通过下列方式建立。

```
# dd if=/dev/sda of=mbr.img bs=512 count=1
# dd if=/dev/sda of=mbr-nopart.img bs=446 count=1
# dd if=/dev/sda of=mbr-part.img skip=446 bs=1 count=66
```

- “mbr.img”：带有分区表的 MBR
- “mbr-nopart.img”：不带分区表的 MBR
- “mbr-part.img”：仅 MBR 的分区表

如果你要建立原始硬盘的一个硬盘分区的映像，你需要使用“/dev/sda1”等替代“/dev/sda”。

9.7.2 直接写入硬盘

磁盘映像文件“disk.img”可以通过下列方式之一写入到一个匹配大小的未挂载设备（例如，第二个 SCSI 设备“/dev/sdb”。

```
# dd if=disk.img of=/dev/sdb ; sync
```

```
# cp disk.img /dev/sdb ; sync
```

```
# cat disk.img >/dev/sdb ; sync
```

相似地，硬盘分区映像文件“partition.img”可以通过下列方式写入到匹配大小的未挂载分区（例如，第二个 SCSI 设备的第一个分区“/dev/sdb1”）。

```
# dd if=partition.img of=/dev/sdb1 ; sync
```

9.7.3 挂载磁盘映像文件

可以使用循环设备通过下列方式挂载和卸载包含单个分区映像的磁盘映像“partition.img”。

```
# losetup --show -f partition.img
/dev/loop0
# mkdir -p /mnt/loop0
# mount -t auto /dev/loop0 /mnt/loop0
...hack...hack...hack
# umount /dev/loop0
# losetup -d /dev/loop0
```

可以简化为如下步骤。

```
# mkdir -p /mnt/loop0
# mount -t auto -o loop partition.img /mnt/loop0
...hack...hack...hack
# umount partition.img
```

可以使用循环设备挂载包含多个分区的磁盘映像“disk.img”的每个分区。

```
# losetup --show -f -P disk.img
/dev/loop0
# ls -l /dev/loop0*
brw-rw---- 1 root disk   7,  0 Apr  2 22:51 /dev/loop0
brw-rw---- 1 root disk 259, 12 Apr  2 22:51 /dev/loop0p1
brw-rw---- 1 root disk 259, 13 Apr  2 22:51 /dev/loop0p14
brw-rw---- 1 root disk 259, 14 Apr  2 22:51 /dev/loop0p15
# fdisk -l /dev/loop0
Disk /dev/loop0: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6A1D9E28-C48C-2144-91F7-968B3CBC9BD1

Device         Start     End Sectors  Size Type
/dev/loop0p1  262144 4192255 3930112  1.9G Linux root (x86-64)
/dev/loop0p14   2048    8191    6144    3M BIOS boot
/dev/loop0p15   8192  262143  253952  124M EFI System

Partition table entries are not in disk order.
# mkdir -p /mnt/loop0p1
# mkdir -p /mnt/loop0p15
# mount -t auto /dev/loop0p1 /mnt/loop0p1
# mount -t auto /dev/loop0p15 /mnt/loop0p15
# mount |grep loop
/dev/loop0p1 on /mnt/loop0p1 type ext4 (rw,relatime)
/dev/loop0p15 on /mnt/loop0p15 type vfat (rw,relatime,fmask=0002,dmask=0002,allow_utime ↩
    =0020,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro)
...hack...hack...hack
# umount /dev/loop0p1
# umount /dev/loop0p15
# losetup -d /dev/loop0
```

或者，你也可以使用 kpartx 软件包中的 kpartx(8) 建立设备映射设备来达到相同的效果。

```
# kpartx -a -v disk.img
add map loop0p1 (253:0): 0 3930112 linear 7:0 262144
add map loop0p14 (253:1): 0 6144 linear 7:0 2048
add map loop0p15 (253:2): 0 253952 linear 7:0 8192
# fdisk -l /dev/loop0
```

```
Disk /dev/loop0: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6A1D9E28-C48C-2144-91F7-968B3CBC9BD1

Device         Start     End Sectors  Size Type
/dev/loop0p1  262144 4192255 3930112  1.9G Linux root (x86-64)
/dev/loop0p14   2048    8191    6144    3M BIOS boot
/dev/loop0p15   8192  262143  253952  124M EFI System

Partition table entries are not in disk order.
# ls -l /dev/mapper/
total 0
crw------- 1 root root 10, 236 Apr  2 22:45 control
lrwxrwxrwx 1 root root       7 Apr  2 23:19 loop0p1 -> ../dm-0
lrwxrwxrwx 1 root root       7 Apr  2 23:19 loop0p14 -> ../dm-1
lrwxrwxrwx 1 root root       7 Apr  2 23:19 loop0p15 -> ../dm-2
# mkdir -p /mnt/loop0p1
# mkdir -p /mnt/loop0p15
# mount -t auto /dev/mapper/loop0p1 /mnt/loop0p1
# mount -t auto /dev/mapper/loop0p15 /mnt/loop0p15
# mount |grep loop
/dev/loop0p1 on /mnt/loop0p1 type ext4 (rw,relatime)
/dev/loop0p15 on /mnt/loop0p15 type vfat (rw,relatime,fmask=0002,dmask=0002,allow_utime ↩
    =0020,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro)
...hack...hack...hack
# umount /dev/mapper/loop0p1
# umount /dev/mapper/loop0p15
# kpartx -d disk.img
```

9.7.4 清理磁盘映像文件

使用下面的方式，一个磁盘映像文件"disk.img" 能够清理掉所有已经删除的文件，成为一个干净的稀疏映像"new.img"。

```
# mkdir old; mkdir new
# mount -t auto -o loop disk.img old
# dd bs=1 count=0 if=/dev/zero of=new.img seek=5G
# mount -t auto -o loop new.img new
# cd old
# cp -a --sparse=always ./ ../new/
# cd ..
# umount new.img
# umount disk.img
```

如果"disk.img"位于 ext2、ext3 或 ext4，你也可以像下面那样使用 zerofree 软件包中的 zerofree(8)。

```
# losetup --show -f disk.img
/dev/loop0
# zerofree /dev/loop0
# cp --sparse=always disk.img new.img
# losetup -d /dev/loop0
```

9.7.5 制作空的磁盘映像文件

按下面的方式使用 dd(1) ，可以制作一个大小为 5GB 的空磁盘映像文件。

```
$ dd bs=1 count=0 if=/dev/zero of=disk.img seek=5G
```

专业的 fallocate(8) 可以在这里被使用，用来替代使用 dd(1)。

按下面的方式使用环回设备，你能够在这个磁盘映像"disk.img" 上创建 ext4 文件系统。

```
# losetup --show -f disk.img
/dev/loop0
# mkfs.ext4 /dev/loop0
...hack...hack...hack
# losetup -d /dev/loop0
$ du  --apparent-size -h disk.img
5.0G  disk.img
$ du -h disk.img
83M disk.img
```

对于"disk.img", 它的文件大小是 5.0 GiB，而它实际磁盘使用仅仅是 83MiB. 这个差距可能是由于 ext4 里面有稀疏文件.

> 提示
> 稀疏文件的实际磁盘使用会随着数据的写入而增加。

回环设备或设备映射设备上使用类似的操作，在这些设备按第 9.7.3 节挂载后, 你能够使用 parted(8) 或 fdisk(8) 对这个磁盘映像"disk.img" 进行分区，能够使用 mkfs.ext4(8), mkswap(8) 在上面创建文件系统等.

9.7.6 制作 ISO9660 镜像文件

> 提示
> cdrkit 提供的 genisoimage(1) 和 Libburnia 提供的 xorrisofs(1) ，这两个命令共享相同的命令语法，仅命令名称不一样。

" 源目录" 下的目录树可以通过如下所示的 cdrkit 提供的 genisoimage(1) 命令来制作 ISO9660 镜像文件，"cd.iso"。

```
#  genisoimage -r -J -T -V volume_id -o cd.iso source_directory
```

类似的，可引导的 ISO9660 镜像文件，"cdboot.iso", 能够从 debian-installer 类似目录树"source_directory" 制作，方式如下。

```
#  genisoimage -r -o cdboot.iso -V volume_id \
   -b isolinux/isolinux.bin -c isolinux/boot.cat \
   -no-emul-boot -boot-load-size 4 -boot-info-table source_directory
```

这里的 Isolinux boot loader (参见第 3.1.2 节) 是用于启动的.

按下面的方式，你可以直接从光驱设备计算 md5sum 值，并制作 ISO9660 镜像。

```
$ isoinfo -d -i /dev/cdrom
CD-ROM is in ISO 9660 format
...
Logical block size is: 2048
Volume size is: 23150592
...
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror | md5sum
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror > cd.iso
```



警告
为了得到正确结果，你必须小心避免 Linux ISO9600 文件系统预读 bug。

9.7.7 直接写入文件到 CD/DVD-R/RW

提示
对 cdrkit 提供的 wodim(1) 和 Libburnia 提供的 xorrecord(1) 而言，DVD 只是一个大的 CD。这些命令共享相同的命令语法，仅命令名称不一样。

你能够通过如下所示的命令找到可用的设备。

```
# wodim --devices
```

然后将空的 CD-R 插入 CD 驱动器并且把 ISO9660 镜像文件，"`cd.iso`" 写入到设备中，例如用如下所示的 wodim(1) 将数据写入到"`/dev/sda`" 设备。

```
# wodim -v -eject dev=/dev/sda cd.iso
```

如果用 CD-RW 代替 CD-R，用如下所示的命令来替代。

```
# wodim -v -eject blank=fast dev=/dev/sda cd.iso
```

提示
如果你的桌面系统自动挂载 CDs，在使用 wodim(1) 之前在终端里面用"`sudo umount /dev/sda`" 卸载它。

9.7.8 挂载 ISO9660 镜像文件

如果"`cd.iso`" 包含一个 ISO9660 镜像, 下面的命令手工挂载这个文件到"`/cdrom`".

```
# mount -t iso9660 -o ro,loop cd.iso /cdrom
```

提示
现代桌面系统能够自动挂载可移动介质，如按 ISO9660 格式化的 CD(参见第 10.1.7 节).

9.8 二进制数据

这里，我们讨论直接操作存储介质上的二进制数据。

软件包	流行度	大小	说明
coreutils	V:914, I:1000	19751	基础软件包，有 od(1) 来导出文件 (HEX, ASCII, OCTAL, …)
bsdmainutils	V:4, I:112	17	工具软件包，有 hd(1) 来导出文件 (HEX, ASCII, OCTAL, …)
hexedit	V:0.7, I:7.4	70	二进制浏览和编辑器 (HEX, ASCII)
ghex	V:0.7, I:7.2	2525	全功能的十六进制编辑器 (GNOME)
okteta	V:1, I:11	1591	全功能的十六进制编辑器 (KDE4)
ncurses-hexedit	V:0.1, I:1.1	130	二进制浏览和编辑器 (HEX, ASCII, EBCDIC)

Table 9.21: 查看和修改二进制数据的软件包列表

9.8.1 查看和编辑二进制数据

最基础的查看二进制数据的方法是使用"`od -t x1`" 命令。

提示
HEX 是十六进制英文 hexadecimal 首字母缩略词，基数 radix 是 16。OCTAL 是八进制英文 octal 首字母缩略词，基数 radix 是 8。ASCII 是美国信息交换标准代码 American Standard Code for Information Interchange 的英文缩写，即正常的英语文本代码。EBCDIC 是扩展二进制编码十进制交换码 Extended Binary Coded Decimal Interchange Code 的英文缩写，在 IBM 大型机操作系统上使用。

9.8.2 不挂载文件系统的情况下操作文件

有工具可以在没有挂载文件系统的情况下读写文件。

软件包	流行度	大小	说明
mtools	V:7, I:51	404	不挂载即可用于 MSDOS 文件系统的工具
hfsplus	V:2, I:20	248	不挂载即可用于 HFS+ 文件系统的工具

Table 9.22: 不挂载文件系统操作文件的软件包列表

9.8.3 数据冗余

Linux 内核所提供的 RAID 软件系统提供内核文件系统级别的数据冗余来实现高水平的存储可靠性。

有在应用程序级别增加数据冗余来实现高水平存储可靠性的工具。

软件包	流行度	大小	说明
par2	V:10, I:115	965	奇偶校验档案卷设置，用于检查和修复文件
dvdisaster	V:0.1, I:1.1	1422	CD/DVD 媒体数据损失/划伤/老化的保护
dvbackup	I:0.04	413	使用 MiniDV 便携式摄像机的备份工具

Table 9.23: 向文件添加数据冗余的工具列表

软件包	流行度	大小	说明
testdisk	V:2, I:25	529	分区扫描和磁盘恢复的实用程序
magicrescue	V:0.2, I:1.6	257	通过查找幻数 magic 字节来恢复文件的工具（译注：请 man file 来了解幻数）
scalpel	V:0.2, I:2.2	89	简洁、高性能的文件提取
myrescue	V:0.2, I:1.9	81	恢复损坏硬盘中的数据
extundelete	V:0.4, I:7.1	152	恢复删除 ext3/4 文件系统上的文件的实用程序
ext4magic	V:0.3, I:3.5	235	恢复删除 ext3/4 文件系统上的文件的实用程序
ext3grep	V:0.1, I:1.7	299	帮助恢复 ext3 文件系统上删除的文件的工具
scrounge-ntfs	V:0.1, I:1.6	49	NTFS 文件系统的数据恢复程序
gzrt	V:0.02, I:0.43	34	gzip 恢复工具包
sleuthkit	V:3, I:24	1130	诊断分析工具 (Sleuthkit)
autopsy	V:0.1, I:1.1	1026	SleuthKit 的图形化界面
foremost	V:0.3, I:3.7	101	恢复数据的诊断程序
guymager	V:0.09, I:0.70	1049	基于 Qt 的诊断图像工具
dcfldd	V:0.2, I:2.4	113	增强版的 dd，用于诊断和安全

Table 9.24: 数据文件恢复和诊断分析软件包列表

9.8.4 数据文件恢复和诊断分析

有用于数据文件恢复和诊断分析的工具。

提示
在 `e2fsprogs` 软件包里有 debugfs(8) 命令，使用该命令里的 `list_deleted_inodes` 和 `undel` 指令，你能够恢复 ext2 文件系统上删除的文件。

9.8.5 把大文件分成多个小文件

当一个文件太大而不能备份的时候，你应该在备份之前先把它分割为多个小于 2000MiB 的小文件，稍后再把这些小文件合并为初始的文件。

```
$ split -b 2000m large_file
$ cat x* >large_file
```



小心
为了防止文件名冲突，请确保没有任何以”x” 开头的文件。

9.8.6 清空文件内容

为了清除诸如日志文件之类的文件的内容，不要用 rm(1) 命令去删除文件然后创建新的空文件，因为这个文件可能在命令执行的期间还在被使用。以下是清除文件内容的正确方法。

```
$ :>file_to_be_cleared
```

9.8.7 样子文件

下面的命令创建样子文件或空文件。

```
$ dd if=/dev/zero    of=5kb.file bs=1k count=5
$ dd if=/dev/urandom of=7mb.file bs=1M count=7
$ touch zero.file
$ : > alwayszero.file
```

你将发现下列文件。

- "`5kb.file`" 是 5KB 的全零数据。
- "`7mb.file`" 是 7MB 随机数据。
- "`zero.file`" 也许是一个 0 字节的文件。如果这个文件之前就存在，则它的 `mtime` 会被更新，而它的内容和长度保持不变。
- "`alwayszero.file`" 一定是一个 0 字节文件。如果这个文件之前存在，则它的 `mtime` 会被更新，而它的内容会被清零。

9.8.8 擦除整块硬盘

有几种方法来完全擦除设备上整个硬盘上数据，比如说，在"`/dev/sda`" 上的 USB 盘。



小心
在执行这里的命令之前，你应该用 mount(8) 命令来查看 USB 盘的挂载位置。"`/dev/sda`" 指向的设备可能是装有整个系统的 SCSI 硬盘或者 serial-ATA 硬盘。

如下所示是通过数据归 0 的方式来擦除硬盘上所有数据的。

```
# dd if=/dev/zero of=/dev/sda
```

如下是用随机数据重写的方式来擦除所有数据的。

```
# dd if=/dev/urandom of=/dev/sda
```

如下是用随机数据重写的方式来高效擦除所有数据。

```
# shred -v -n 1 /dev/sda
```

你或者可以使用 badblocks(8) 加上 `-t random` 选项。

因为 dd(1) 命令在许多可引导的 Linux CDs (例如 Debian 安装光盘) 上的 shell 环境下都是可用的，你能够在装有系统的硬盘上，例如"`/dev/sda`", "`/dev/sda`" 等等设备上运行擦除命令来完全清除已经安装的系统。

9.8.9 擦除硬盘上的未使用的区域

硬盘（或 USB 盘）上未使用的区域，例如"`/dev/sdb1`"可能仍然包含可被擦除的数据，因为他们本身只是解除了从文件系统的链接，这些可以通过重写来清除。

```
# mount -t auto /dev/sdb1 /mnt/foo
# cd /mnt/foo
# dd if=/dev/zero of=junk
dd: writing to `junk': No space left on device
...
# sync
# umount /dev/sdb1
```



警告
这对您的 USB 盘来说通常已经足够好了，但这还不完美。大部分已擦除的文件名和它们的属性可能隐藏并留在文件系统中。

9.8.10 恢复已经删除但仍然被打开的文件

即使你不小心删除了某个文件，只要这个文件仍然被一些应用程序所使用（读或者写），恢复此文件是可能的。

尝试下列例子

```
$ echo foo > bar
$ less bar
$ ps aux | grep ' less[ ]'
bozo    4775  0.0  0.0  92200   884 pts/8    S+   00:18   0:00 less bar
$ rm bar
$ ls -l /proc/4775/fd | grep bar
lr-x------ 1 bozo bozo 64 2008-05-09 00:19 4 -> /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 >bar
$ ls -l
-rw-r--r-- 1 bozo bozo 4 2008-05-09 00:25 bar
$ cat bar
foo
```

当你安装了 `lsof` 软件包的时候，在另外一个终端执行如下命令。

```
$ ls -li bar
2228329 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:02 bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar
$ rm bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 >bar
$ ls -li bar
2228302 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:05 bar
$ cat bar
foo
```

9.8.11 查找所有硬链接

有硬链接的文件，能够使用"`ls -li`" 确认。

```
$ ls -li
total 0
2738405 -rw-r--r-- 1 root root 0 2008-09-15 20:21 bar
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 baz
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 foo
```

"`baz`" 和"`foo`" 的链接数为"2" (>1)，表示他们有硬链接。它们的 inode 号都是"2738404". 这表示它们是同样的硬链接文件。如果你不想偶然碰巧发现硬链接文件，你可以通过 inode 号来查找它。比如说, 按下面的方式查找"2738404" 。

```
# find /path/to/mount/point -xdev -inum 2738404
```

9.8.12 不可见磁盘空间消耗

所有打开的文件被删除后，仍然消耗磁盘空间，尽管他们不能够被普通的 du(1) 所看见。这些被删除的文件和他们的大小，可以通过下面的方式列出。

```
# lsof -s -X / |grep deleted
```

9.9 数据加密提示

在可以物理访问您的 PC 的情况下，任何人都可以轻易获得 root 权限，访问您的 PC 上的所有文件 (见第 4.6.4 节)。这意味着登录密码系统在您的 PC 被偷盗时并不能保证您私人和敏感数据的安全。您必须部署数据加密技术来实现。尽管 GNU 隐私守护 (见第 10.3 节) 可以对文件进行加密，但它需要一些用户端的工作。

Dm-crypt 通过原生的 Linux 内核模块，使用 device-mapper，用很少的用户操作实现本地自动数据加密。

软件包	流行度	大小	说明
cryptsetup	V:35, I:77	476	可用于加密的块设备的实用程序（dm-crypt / 3LUKS）
cryptmount	V:1.8, I:2.2	236	可用于加密的块设备着重于正常用户挂载/卸载的实用程序（dm-crypt / LUKS）
fscrypt	V:0.4, I:1.1	6844	Linux 文件系统加密工具 (fscrypt)
libpam-fscrypt	V:0.32, I:0.35	6941	Linux 文件系统加密 PAM 模块 (fscrypt)

Table 9.25: 数据加密工具列表



小心
数据加密会消耗 CPU 时间等资源，如果它的密码丢失，加密的数据会变成无法访问。请权衡其利弊。

注意
通过 debian-installer (lenny 或更新版)，整个 Debian 系统能够被安装到一个加密的磁盘上，使用 dm-crypt/LUKS 和 initramfs.

提示
请参阅第 10.3 节用户空间加密实用程序：GNU Privacy Guard。

9.9.1 使用 dm-crypt/LUKS 加密移动磁盘

您可以用 dm-crypt/LUKS 加密大容量可移动设备上的数据，例如挂载在“/dev/sdx”上的 USB 盘。你只需按如下步骤简单地把它格式化。

```
# fdisk /dev/sdx
... "n" "p" "1" "return" "return" "w"
# cryptsetup luksFormat /dev/sdx1
...
# cryptsetup open /dev/sdx1 secret
...
# ls -l /dev/mapper/
```

```
total 0
crw-rw---- 1 root root  10, 60 2021-10-04 18:44 control
lrwxrwxrwx 1 root root       7 2021-10-04 23:55 secret -> ../dm-0
# mkfs.vfat /dev/mapper/secret
...
# cryptsetup close secret
```

然后，它就可以正常的在现代桌面环境下使用 udisks2 软件包，挂载到"/media/*username/disk_label*"，只不过它会要求输入密码 (参见第 10.1.7 节)。不同的是写入的数据都是加密的。密码条目可以自动使用钥匙环（参见第 10.3.6 节）。

你可以把它格式化成其他格式的文件系统，例如用"mkfs.ext4 /dev/mapper/sdx1" 把它格式化为 ext4。如果使用 btrfs，则需要安装 udisks2-btrfs 软件包。对于这些文件系统，可能需要配置文件的所有者和权限。

9.9.2 使用 dm-crypt/LUKS 挂载加密的磁盘

举个列子，用 dm-crypt/LUKS 在"/dev/sdc5" 上创建的加密磁盘可以用如下步骤挂载到"/mnt"：

```
$ sudo cryptsetup open /dev/sdc5 ninja --type luks
Enter passphrase for /dev/sdc5: ****
$ sudo lvm
lvm> lvscan
  inactive          '/dev/ninja-vg/root' [13.52 GiB] inherit
  inactive          '/dev/ninja-vg/swap_1' [640.00 MiB] inherit
  ACTIVE            '/dev/goofy/root' [180.00 GiB] inherit
  ACTIVE            '/dev/goofy/swap' [9.70 GiB] inherit
lvm> lvchange -a y /dev/ninja-vg/root
lvm> exit
  Exiting.
$ sudo mount /dev/ninja-vg/root /mnt
```

9.10 内核

对于支持的架构，Debian 使用软件包来分发模块化的 Linux 内核.

如果你正在阅读本文档，你可能不需要亲自编译内核。

9.10.1 内核参数

许多 Linux 特性可以按下面的方式，通过内核参数来配置。

- 内核参数通过 bootloader 初始化 (参见第 3.1.2 节)
- 对通过 sysfs 访问的内核参数，在运行时通过 sysctl(8) 修改 (参见第 1.2.12 节)
- 当一个模块被激活时，通过 modprobe(8) 参数来设置模块参数。(参见第 9.7.3 节)

细节参见"The Linux kernel user's and administrator's guide » The kernel's command-line parameters" 。

9.10.2 内核头文件

大部分普通程序编译时不需要内核头文件，如果你直接使用它们来编译，甚至会导致编译中断。在 Debian 系统上，普通程序编译依赖 libc6-dev 软件包 (由 glibc 源代码包创建) 提供的，在"/usr/include/linux" 和"/usr/include/asm" 里的头文件。

注意
对于编译一些内核相关的程序，比如说从外部源代码编译的内核模块和 automounter 后台守护（daemon）程序 (amd)，你必须包含相应的内核头文件到路径里，比如"-I/usr/src/linux-particular-version/include/", 到你的命令行。

9.10.3 编译内核和相关模块

Debian 有它自己的方式来编译内核和相关模块。

软件包	流行度	大小	说明
build-essential	V:20, I:508	17	创建 Debian 软件包所必须的软件包: make, gcc, ···
bzip2	V:163, I:973	113	bz2 文件压缩和解压缩工具
libncurses-dev	V:16, I:148	2380	ncurses 开发者库和文档
git	V:442, I:630	52512	git：Linux 内核使用的分布式版本控制系统
fakeroot	V:34, I:508	225	为非 root 用户创建软件包提供一个伪造的 root 环境
initramfs-tools	V:525, I:989	49	创建 initramfs 的工具（Debian 规范）
dkms	V:81, I:147	235	动态内核模块支持 dynamic kernel module support (DKMS) (通用)
module-assistant	V:1, I:11	395	制作模块软件包的帮助工具（Debian 规范）
devscripts	V:5, I:30	2790	Debian 软件包维护者的帮助脚本（Debian 规范）

Table 9.26: Debian 系统内核编译需要安装的主要软件包列表

如果你在第 3.1.2 节使用 initrd，请一定阅读 initramfs-tools(7), update-initramfs(8), mkinitramfs(8) 和 initramfs.conf(5) 里的相关信息。



警告
在编译 Linux 内核源代码时，请不要放置从"/usr/include/linux" 和"/usr/include/asm" 到源代码树 (比如："/usr/src/linux*") 里目录的符号链接。(一些过期的文档建议这样做.)

注意
当在 Debian stable 版里编译最新的 Linux 内核时，可能需要使用一些从 Debian unstable 版里向后移植（backported）的工具的最新版本。
module-assistant(8) (或者它的短形式 m-a) 帮助用户，从一个或多个定制的内核，容易的构建和安装模块软件包。
dynamic kernel module support (DKMS) 动态内核模块支持是一个新的分布式独立框架，被设计用来允许单个的内核模块在不改变整个内核的情况下升级。这可以用于维护内核代码树外部的模块。这也使你升级内核时，重新编译模块变得非常简单。

9.10.4 编译内核源代码：**Debian** 内核团队推荐

从上游内核源代码编译个性化的内核二进制包，你应当使用由它提供的”`deb-pkg`” 对象。

```
$ sudo apt-get build-dep linux
$ cd /usr/src
$ wget https://mirrors.edge.kernel.org/pub/linux/kernel/v7.x/linux-7.3.5.tar.xz
$ tar --xz -xvf linux-7.3.5.tar.xz
$ cd linux-7.3.5
$ cp /boot/config-7.3.5.config
$ make menuconfig
 ...
$ make deb-pkg
```

提示
linux-source-*version* 软件包使用”`/usr/src/linux-`*`version`*`.tar.bz2`” 提供有 Debian 补丁的 Linux 内核源代码。

从 Debian 内核源代码软件包编译特定的二进制包，你应当使用”`debian/rules.gen`” 里的”`binary-arch_`*`architectur`* 对象。

```
$ sudo apt-get build-dep linux
$ apt-get source linux
$ cd linux-7.*
$ fakeroot make -f debian/rules.gen binary-arch_i386_none_686
```

进阶信息参见：

- Debian Wiki: KernelFAQ
- Debian Wiki: DebianKernel
- Debian Linux 内核手册：https://kernel-handbook.debian.net

9.10.5 硬件驱动和固件

硬件驱动是运行在目标系统上主 CPU 上的代码。大部分硬件驱动现在是自由软件，已经包含在普通的 Debian 内核软件包里，放在 `main` 区域。

- GPU 驱动
  - Intel GPU 驱动 (`main`)
  - AMD/ATI GPU 驱动 (`main`) 和/
  - NVIDIA GPU 驱动 (nouveau 驱动放在 `main`，由厂家支持的二进制驱动，放在 `non-free`.)

固件是加载在连接在目标系统设备上的代码或数据 (比如说，CPU microcode、GPU 运行的渲染代码或 FPGA / CPLD 数据……）部分固件包是作为自由软件存在，但是很多固件包由于包含有没有源代码的数据，二进制不是作为自由软件存在。安装这些固件数据是必需的，这样设备才能按期望运行。

- 固件数据软件包，含有加载到目标设备易失性存储器上的数据。
  - `firmware-linux-free` (`main`)
  - `firmware-linux-nonfree` (`non-free-firmware`)
  - `firmware-linux-*` (`non-free-firmware`)

  - `*-firmware` (`non-free-firmware`)
  - `intel-microcode` (`non-free-firmware`)
  - `amd64-microcode` (`non-free-firmware`)
- 固件更新程序软件包，更新在目标设备非易失性存储器上的数据。
  - fwupd (`main`)：从 Linux Vendor Firmware Service 下载固件数据的固件更新后台守护进程（daemon）。
  - gnome-firmware (`main`): fwupd 的 GTK 前端
  - plasma-discover-backend-fwupd (`main`): fwupd 的 Qt 前端

请注意，从 Debian 12 Bookworm 开始，用户使用由官方安装介质里面提供的 `non-free-firmware` 软件包来提供完善的安装体验。`non-free-firmware` 区域在第 2.1.5 节里面描述。

也请注意到，fwupd 从 Linux Vendor Firmware Service 下载的固件数据并加载到正在运行的 Linux 内核，可能是 `non-free`。

9.11 虚拟化系统

通过使用虚拟系统，我们能在单个机器上同时运行多个系统。

> 提示
> 参见 Debian wiki 的系统虚拟化。

9.11.1 虚拟化和模拟器工具

有几个虚拟化和模拟器工具平台。

- 完全的硬件模拟软件包，比如通过 games-emulator 元软件包安装的软件包
- 大部分 CPU 层的模拟，加上一些 I/O 设备的模拟，比如 QEMU
- 大部分 CPU 层的虚拟化，加上一些 I/O 设备的模拟，比如 Kernel-based Virtual Machine (KVM)
- 操作系统级的容器虚拟化，加上内核级的支持，比如 LXC (Linux Containers), Docker, systemd-nspawn(1), ...
- 操作系统级的文件系统访问虚拟化，加上在文件路径上的系统库调用，比如 chroot
- 操作系统级的文件系统访问虚拟化，加上在文件所有者权限上的系统库调用，比如 fakeroot
- 操作系统 API 模拟器，比如 Wine
- 解释器级的虚拟化，加上它的执行选择和运行时库忽略，比如 Python 的 virtualenv 和 venv

容器虚拟化使用第 4.7.5 节，是第 7.7 节的后端技术。

这里有一些帮你搭建虚拟化系统的软件包。

参见维基百科 Comparison of platform virtual machines 来获得不同平台的虚拟化解决方案的详细比较信息。

软件包	流行度	大小	说明
coreutils	V:914, I:1000	19751	GNU 核心工具包含 chroot(8)
util-linux	V:922, I:1000	4826	其他系统工具包含 unshare(8)
systemd-container	V:74, I:77	2964	systemd container/nspawn 工具包含 systemd-nspawn(1)
schroot	V:7.1, I:8.9	2241	在 chroot 下执行 Debian 二进制包的特异工具
sbuild	V:3.8, I:6.8	160	从 Debian 源码构建 Debian 二进制包的工具
debootstrap	V:4, I:43	332	搭建一个基本的 Debian 系统 (用 sh 写的)
mmdebstrap	V:8, I:12	613	搭建一个 Debian 系统 (用 Perl 写的)
cdebootstrap	V:0.1, I:1.2	113	搭建一个 Debian 系统 (用 C 写的)
cloud-image-utils	V:1, I:14	52	云镜像管理工具
cloud-guest-utils	V:5, I:25	67	云客户机工具
virt-manager	V:12, I:47	2309	虚拟机管理器: 用于管理虚拟机的桌面应用
libvirt-clients	V:49, I:70	1168	libvirt 的库程序
docker.io	V:51, I:55	80705	docker: Linux 容器运行时
podman	V:36, I:40	86285	podman：在 Pods 里面运行基于 OCI 容器的引擎
podman-docker	V:2.9, I:3.4	271	在 Pods 里面运行基于 OCI 容器的引擎 -- docker 封装器
incus	V:1.0, I:3.7	24	Incus：系统容器和虚拟机管理器
games-emulator	I:0.18	21	games-emulator: Debian 的游戏模拟器
bochs	V:0.03, I:0.53	8469	Bochs: IA-32 PC 仿真器
qemu-system	I:19	69	QEMU: 全功能系统的模拟二进制
qemu-user	V:6.3, I:9.8	472221	QEMU: 用户模式的模拟二进制
qemu-utils	V:16, I:117	12651	QEMU: 工具集
qemu-system-x86	V:55, I:101	69265	KVM: x86 硬件上有硬件辅助虚拟化的全虚拟化
virt-manager	V:12, I:47	2309	用于管理虚拟机的桌面应用程序，支持 KVM、Xen、LXC 等类型的虚拟机...
gnome-boxes	V:1.1, I:5.8	7070	Boxes：用于访问虚拟机系统的简单的 GNOME 应用程序
xen-tools	V:0.1, I:1.3	719	用于管理 debian XEN 虚拟服务器的工具
wine	V:13, I:54	209	Wine: Windows 应用程序编程接口实现（标准套件）
dosbox	V:1, I:11	3139	DOSBox: 有 Tandy/Herc/CGA/EGA/VGA/SVGA 显卡, 声音和 DOS 的 x86 模拟器
lxc	V:11, I:13	1680	Linux 容器用户层工具
python3-venv	V:8, I:167	6	venv 创建虚拟的 python 环境（系统库）
python3-virtualenv	V:7, I:35	435	virtualenv 创建隔离的虚拟 python 环境
pipx	V:8, I:52	3261	pipx 在隔离的环境中安装 python 应用程序

Table 9.27: 虚拟化工具列表

9.11.2 虚拟化工作流

注意
自从 lenny 之后，默认的 Debian 内核就是支持 KVM 的。

典型的虚拟化工作流涉及以下几个步骤。

- 创建空文件系统 (目录树或磁盘映像)。
  - 目录树可以通过"mkdir -p /path/to/chroot" 创建。
  - 原始的磁盘映像文件能够使用 dd(1) 创建 (参见第 9.7.1 节和第 9.7.5 节).
  - qemu-img(1) 能够创建和转化 QEMU 支持的磁盘映像文件.
  - 原始的格式和 VMDK 文件格式, 能够作为虚拟化工具的通用格式.
- 使用 mount(8) 挂载磁盘映像到文件系统 (可选).
  - 对于原始磁盘映像文件, 把它作为回环设备或设备映射设备挂载. (参见第 9.7.3 节).
  - 对于 QEMU 支持的磁盘映像, 把它们作为 network block device 网络块设备挂载 (参见第 9.11.3 节).
- 在目标文件系统上部署需要的系统数据。
  - 使用 debootstrap 和 cdebootstrap 之类的程序来协助处理这个过程 (参见第 9.11.4 节).
  - 在全功能系统模拟器下使用操作系统安装器。
- 在虚拟化环境下运行一个程序。
  - chroot 提供基本的虚拟化环境, 足够能在里面编译程序, 运行控制台应用, 运行后台守护程序 daemon.
  - QEMU 提供跨平台的 CPU 模拟器.
  - QEMU 和 KVM 通过 hardware-assisted virtualization 硬件辅助虚拟化来提供全功能系统的模拟.
  - VirtualBox 可以在 i386 和 amd64 上, 使用或者不使用 hardware-assisted virtualization 硬件辅助虚拟化来提供全功能系统模拟.

9.11.3 挂载虚拟磁盘映像文件

对于原始磁盘映像文件, 参见第 9.7 节.

对于其它虚拟磁盘映像文件, 你能够用使用 network block device 网络块设备协议的 qemu-nbd(8) 来导出他们, 并使用内核模块 nbd 来挂载它们.

qemu-nbd(8) 支持 QEMU 所支持的磁盘格式: raw, qcow2, qcow, vmdk, vdi, bochs, cow (user-mode Linux copy-on-write), parallels, dmg, cloop, vpc, vvfat (virtual VFAT) 和主机设备.

网络块设备能够用和回环设备一样的方式支持分区 (参见第 9.7.3 节). 你能够按下面的方式挂载"disk.img" 的第一个分区.

```
# modprobe nbd max_part=16
# qemu-nbd -v -c /dev/nbd0 disk.img
...
# mkdir /mnt/part1
# mount /dev/nbd0p1 /mnt/part1
```

提示
你可以给 qemu-nbd(8) 使用"-P 1" 选项来导出"disk.img" 的第一个分区.

9.11.4 Chroot 系统

如果你希望从终端控制台尝试一个新的 Debian 环境，我推荐你使用 chroot。这使你能够运行 unstable 和 testing 的控制台应用程序，不会有通常的相关风险，并且不需要重启。chroot(8) 是最基础的方法。



小心
下面的列子假设根源系统和 chroot 系统都共享相同的 amd64 CPU 架构。

虽然你能够手工使用 debootstrap(8) 来创建一个 chroot(8) 环境，这要求琐碎的工作。

sbuild 软件包从源代码构建一个 Debian 软件包，使用 schroot 软件包管理的 chroot 环境。它和帮助脚本 sbuild-createchroot(8) 一起。让我们按如下所示的方式运行它，来学会它是怎样工作的。

```
$ sudo mkdir -p /srv/chroot
$ sudo sbuild-createchroot -v --include=eatmydata,ccache unstable /srv/chroot/unstable- ↩
    amd64-sbuild http://deb.debian.org/debian
 ...
```

你能够看到 debootstrap(8) 是如何在”/srv/chroot/unstable-amd64-sbuild” 下部署 unstable 环境的系统数据, 用于一个最小的构建系统。

你可以使用 schroot(1) 来登录到这个环境。

```
$ sudo schroot -v -c chroot:unstable-amd64-sbuild
```

你可以看到一个运行在 unstable 环境的系统 shell 是如何创建的。

注意
”/usr/sbin/policy-rc.d” 文件总是用 101 退出，阻止在 Debian 系统上自动启动后台守护程序。参见”/usr/share/doc/init-system-helpers/README.policy-rc.d.gz”。

注意
一些在 chroot 下的程序，需要访问比上面根源系统上的 sbuild-createchroot 能够提供的文件之外的更多文件才能够运行。例如，”/sys”, ”/etc/passwd”, ”/etc/group”, ”/var/run/utmp”, ”/var/log/wtmp” 等等. 也许需要使用 bind-mounted 或拷贝.

提示
sbuild 软件包帮助建立一个 chroot 系统来构建一个软件包，在 chroot 内使用 schroot 作为它的后端。它是一个检查构建依赖（build-dependencies）的理想系统。更多信息参见 sbuild at Debian wiki 和在”Guide for Debian Maintainers” 中的 sbuild 配置样例 。

提示
systemd-nspawn(1) 命令使用 chroot 类似的方法帮助运行一个命令，或者轻量级容器内的操作系统。它更强大，因为它使用命名空间来完全虚拟化进程树、进程间通讯、主机名、域名，并可选网络和用户数据库。参见 systemd-nspawn。

9.11.5 多桌面系统

如果你希望尝试任一操作系统的一个新的 GUI 桌面环境，我推荐在 Debian 稳定版系统上使用 QEMU 或者 KVM ，这些软件应用虚拟化技术安全的运行多桌面系统。这能让你运行任何桌面应用，包括 Debian 不稳定版和测试版上的桌面应用，并且没有与之相关的通常意义上的风险，并且不需要重启。

因为单纯的 QEMU 工具是非常慢的，当主机系统支持 KVM 的时候，建议使用它来加速。

虚拟机管理器，也被称为 `virt-manager`，它是一个便利的管理 KVM 虚拟机的 GUI（图形用户界面）工具，它调用 libvirt。

按下面的方法，能够创建一个可以用于 QEMU 的包含有 Debian 系统的虚拟磁盘映像"`virtdisk.qcow2`"，这个 Debian 系统使用 debian 安装器: 小 CD 安装。

```
$ wget https://cdimage.debian.org/debian-cd/5.0.3/amd64/iso-cd/debian-503-amd64-netinst.iso
$ qemu-img create -f qcow2 virtdisk.qcow2 5G
$ qemu -hda virtdisk.qcow2 -cdrom debian-503-amd64-netinst.iso -boot d -m 256
...
```

提示
在虚拟化下运行 Ubuntu 和 Fedora 之类的其它 GNU/Linux 发行版，是一个不错的学习其配置技巧的方法。其它专有操作系统也可以在这个 GNU/Linux 虚拟化下很好的运行。

在 Debian wiki: SystemVirtualization 查看更多技巧。

Chapter 10

数据管理

以下是关于在 Debian 系统上管理二进制和文本数据的工具及其相关提示。

10.1 共享，拷贝和存档



警告
为避免 竞争情况，不应当对正在进行写操作的设备和文件，多个进程进行不协调的写操作。采用 flock(1) 的 文件锁定机制可用于避免这种情况。

数据的安全和它的受控共享有如下几个方面。

- 存档文件的建立
- 远程存储访问
- 复制
- 跟踪修改历史
- 促进数据共享
- 防止未经授权的文件访问
- 检测未经授权的文件修改

这些可以通过使用工具集来实现。

- 存档和压缩工具
- 复制和同步工具
- 网络文件系统
- 移动存储媒介
- 安全 shell
- 认证体系
- 版本控制系统工具
- 哈希算法和加密工具

10.1.1 存档和压缩工具

以下是 Debian 系统上可用的存档和压缩工具的预览。

软件包	流行度	大小	扩展名	命令	描述
tar	V:911, I:1000	3101	.tar	tar(1)	标准的归档工具（默认）
cpio	V:388, I:999	1205	.cpio	cpio(1)	Unix System V 风格的归档器，与 find(1) 一起使用
binutils	V:201, I:627	1132	.ar	ar(1)	创建静态库的归档工具
fastjar	V:0.9, I:9.0	182	.jar	fastjar(1)	Java 归档工具（类似 zip）
pax	V:4.9, I:8.5	167	.pax	pax(1)	新的 POSIX 归档工具，介于 tar 和 cpio 之间
gzip	V:904, I:1000	262	.gz	gzip(1), zcat(1), …	GNU LZ77 压缩工具（默认）
bzip2	V:163, I:973	113	.bz2	bzip2(1), bzcat(1), …	Burrows-Wheeler block-sorting 压缩工具有着比 gzip(1) 更高的压缩率 (跟 gzip 有着相似的语法但速度比它慢)
lzma	V:0.7, I:9.4	349	.lzma	lzma(1)	LZMA 压缩工具有着比 gzip(1) 更高的压缩率（不推荐）
xz-utils	V:323, I:981	1551	.xz	xz(1), xzdec(1), …	XZ 压缩工具有着比 bzip2(1) 更高的压缩率（压缩速度慢于 gzip 但是比 bzip2 快; LZMA 压缩工具的替代品）
zstd	V:346, I:850	2308	.zstd	zstd(1), zstdcat(1), …	Zstandard 快速无损压缩工具
7zip	V:86, I:339	7542	.7z	7zr(1), p7zip(1)	有着更高压缩率的 7-Zip 文件归档器 (LZMA 压缩和其他)
lzop	V:13, I:128	164	.lzo	lzop(1)	LZO 压缩工具有着比 gzip(1) 更高的压缩和解压缩速度 (跟 gzip 有着相似的语法但压缩率比它低)
zip	V:52, I:350	635	.zip	zip(1)	InfoZip：DOS 归档器和压缩工具
unzip	V:126, I:738	399	.zip	unzip(1)	InfoZIP：DOS 解档器和解压缩工具

Table 10.1: 存档和压缩工具列表



警告
除非你知道将会发生什么，否则不要设置”$TAPE” 变量。它会改变 tar(1) 的行为。

- gzipped tar(1) 归档器用于扩展名是”.tgz” 或者”.tar.gz” 的文件。
- xz-compressed tar(1) 归档器用于扩展名是”.txz” 或者”.tar.xz” 的文件。
- FOSS 工具，例如 tar(1)，中的主流压缩方法已经按如下所示的迁移: gzip → bzip2 → xz
- cp(1),scp(1) 和 tar(1) 工具可能并不适用于一些特殊的文件。cpio(1) 工具的适用范围是最广的。
- cpio(1) 是被设计为与 find(1) 和其它命令一起使用，适合于创建备份脚本的场景，因此，脚本的文件选择部分能够被独立测试。
- Libreoffice 数据文件的内部结构是”.jar” 文件，它也可以使用 unzip 工具来打开。
- 事实上跨平台支持最好的存档工具是 zip。按照“zip -rX”的方式调用可以获得最大的兼容性。如果最大文件大小需要纳入考虑范围，请同时配合“-s”选项使用。

10.1.2 复制和同步工具

以下是 Debian 系统上的可用的简单复制和备份工具的预览。

软件包	流行度	大小	工具	功能
coreutils	V:914, I:1000	19751	GNU cp	复制本地文件和目录 (”-a” 参数实现递归)
openssh-client	V:709, I:997	3611	scp	复制远端文件和目录（客户端，”-r” 参数实现递归)
openssh-server	V:781, I:824	3615	sshd	复制远端文件和目录（远程服务器）
rsync	V:191, I:538	985		单向远程同步和备份
unison	V:3, I:12	14		双向远程同步和备份

Table 10.2: 复制和同步工具列表

在复制文件的时候，rsync(1) 比其他工具提供了更多的特性。

- 差分传输算法只会发送源文件与已存在的目标文件之间的差异部分
- 快速检查算法 (默认) 会查找大小或者最后的修改时间有变化的文件
- ”`--exclude`” 和”`--exclude-from`” 选项类似于 tar(1)
- 在源目录中添加反斜杠的语法能够避免在目标文件中创建额外的目录级别。

> 提示
> 在表 10.14 中的版本控制系统 (VCS) 可以被认为是多路拷贝和同步工具。

10.1.3 归档语法

以下是用不同的工具压缩和解压缩整个”`./source`” 目录中的内容。

GNU tar(1):

```
$ tar -cvJf archive.tar.xz ./source
$ tar -xvJf archive.tar.xz
```

或者，如下所示。

```
$ find ./source -xdev -print0 | tar -cvJf archive.tar.xz --null -T -
```

cpio(1):

```
$ find ./source -xdev -print0 | cpio -ov --null > archive.cpio; xz archive.cpio
$ zcat archive.cpio.xz | cpio -i
```

10.1.4 复制语法

如下是用不同的工具复制整个”`./source`” 目录中的内容。

- 本地复制: ”`./source`” 目录 → ”`/dest`” 目录

- 远程复制：本地主机上的"`./source`" 目录 → "`user@host.dom`" 主机上的"`/dest`" 目录

rsync(8):

```
# cd ./source; rsync -aHAXSv . /dest
# cd ./source; rsync -aHAXSv . user@host.dom:/dest
```

你能够选择使用"源目录上的反斜杠"语法。

```
# rsync -aHAXSv ./source/ /dest
# rsync -aHAXSv ./source/ user@host.dom:/dest
```

或者，如下所示。

```
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . /dest
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . user@host.dom:/dest
```

GNU cp(1) 和 openSSH scp(1):

```
# cd ./source; cp -a . /dest
# cd ./source; scp -pr . user@host.dom:/dest
```

GNU tar(1):

```
# (cd ./source && tar cf - . ) | (cd /dest && tar xvfp - )
# (cd ./source && tar cf - . ) | ssh user@host.dom '(cd /dest && tar xvfp - )'
```

cpio(1):

```
# cd ./source; find . -print0 | cpio -pvdm --null --sparse /dest
```

你能够在所有包含"`.`" 的例子里用"`foo`" 替代"`.`"，这样就可以从"`./source/foo`" 目录复制文件到"`/dest/foo`" 目录。

在所有包含"`.`" 的列子里，你能够使用绝对路径"`/path/to/source/foo`" 来代替"`.`"，这样可以去掉"`cd ./source;`". 如下所示，这些文件会根据工具的不同，拷贝到不同的位置。

- "`/dest/foo`": rsync(8), GNU cp(1), 和 scp(1)
- "`/dest/path/to/source/foo`": GNU tar(1), 和 cpio(1)

提示
rsync(8) 和 GNU cp(1) 可以用"`-u`" 选项来忽略接受端上更新的文件。

10.1.5 查找文件的语法

find(1) 被用作从归档中筛选文件也被用作拷贝命令 (参见第 10.1.3 节和第 10.1.4 节) 或者用于 xargs(1) (参见第 9.4.9 节)。通过 find 的命令行参数能够使其功能得到加强。

以下是 find(1) 基本语法的总结。

- find 条件参数的运算规则是从左到右。
- 一旦输出是确定的，那么运算就会停止。
- "逻辑 **OR**" （由条件之间的"`-o`" 参数指定的）优先级低于" 逻辑 **AND**" （由"`-a`" 参数指定或者条件之间没有任何参数）。
- "逻辑 **NOT**" (由条件前面的"`!`" 指定) 优先级高于"逻辑 **AND**"。

- "`-prune`" 总是返回逻辑 **TRUE** 并且如果这个目录是存在的，将会搜索除这个目录以外的文件。
- "`-name`" 选项匹配带有 shell 通配符 (参见第 1.5.6 节) 的文件名但也匹配带有类似"`*`" 和"`?`" 元字符的"`.`"。(新的 POSIX 特性)
- "`-regex`" 匹配整个文件路径，默认采用 emacs 风格的 **BRE** (参见第 1.6.2 节)。
- "`-size`" 根据文件大小来匹配 (值前面带有"`+`" 号匹配更大的文件，值前面带有"`-`" 号匹配更小的文件)
- "`-newer`" 参数匹配比参数名中指定的文件还要新的文件。
- "`-print0`" 参数总是返回逻辑 **TRUE** 并将完整文件名 (null terminated) 打印到标准输出设备上。

如下是 find(1) 语法格式。

```
# find /path/to \
    -xdev -regextype posix-extended \
    -type f -regex ".*\.cpio|.*~" -prune -o \
    -type d -regex ".*/\.git" -prune -o \
    -type f -size +99M -prune -o \
    -type f -newer /path/to/timestamp -print0
```

这些命令会执行如下动作。

1. 查找"`/path/to`" 下的所有文件
2. 限定全局查找的文件系统并且使用的是 **ERE** (参见第 1.6.2 节)
3. 通过停止处理的方式来排除匹配"`.*\.cpio`" 或"`.*~`" 正则表达式的文件
4. 通过停止处理的方式来排除匹配"`.*/\.git`" 正则表达式的目录
5. 通过停止处理的方式来排除比 99MB (1048576 字节单元) 更大的文件
6. 显示文件名，满足以上搜索条件并且比"`/path/to/timestamp`" 新的文件

请留心以上例子中的"`-prune -o`" 排除文件的习惯用法。

注意
对于非 Debian 系的 Unix-like 系统，有些参数可能不被 find(1) 命令所支持。在这种情况下，应该考虑调整匹配方法并用"`-print`" 替代"`-print0`"。你可能同样需要更改其他相关的命令。

10.1.6 归档媒体

为重要的数据存档寻找存储设备时，你应该注意它们的局限性。对于小型的个人数据备份，我使用品牌公司的 CD-R 和 DVD-R 然后把它放在阴凉、干燥、清洁的地方。(专业的一般使用磁带存档介质)

注意
防火安全是对于纸质文档来说的，大多数的计算机数据存储媒介耐热性比纸差。我经常依赖存储在多个安全地点的加密拷贝。

网上（主要是来源于供应商信息）可以查看存储介质的最大使用寿命。

- 大于 100 年：用墨水的无酸纸
- 100 年：光盘存储（CD/DVD，CD/DVD-R）

- 30 年：磁带存储（磁带，软盘)
- 20 年：相变光盘存储（CD-RW）

这不包括由于人为导致的机械故障等等。

网上（主要来源于供应商信息）可以查看存储介质的最大的写次数。

- 大于 250,000 次：硬盘驱动器
- 大于 10,000 次：闪存
- 1,000 次：CD/DVD-RW
- 1 次：CD/DVD-R，纸



小心
这里的存储寿命和写次数的数据不应该被用来决定任何用于关键数据的存储媒介，请翻阅制造商提供的特定产品的说明。

提示
因为 CD/DVD-R 和纸只能写一次，它们从根本上阻止了因为重写导致的数据意外丢失。这是优点！

提示
如果你需要更快更频繁的进行大数据备份，那么通过高速网络连接的远端主机上的硬盘来实现备份，可能是唯一可行的方法。

提示
如果你在使用一个可重复写入的介质作为你的备份介质，使用支持只读快照的 btrfs 或 zfs 文件系统，也许是一个好注意。

10.1.7 可移动存储设备

可移动存储设备可能是以下的任何一种。

- USB 闪存盘
- 硬盘驱动器
- 光盘驱动器
- 数码相机
- 数字音乐播放器

它们可以通过以下的方式来进行连接。

- USB
- IEEE 1394 / FireWire

像 GNOME 和 KDE 这样的现代桌面环境能够在”/etc/fstab” 文件中没有匹配条目的时候，自动挂载这些可移动设备。

- udisks2 包提供了守护进程和相关的实用程序来挂载和卸载这些设备。
- D-bus 创建事件来触发自动处理。
- PolicyKit 提供了所需的特权。

提示
umount(8) 在自动挂载设备的时候可能会带有”uhelper=” 参数。

提示
只有当这些可移动设备没有在”/etc/fstab” 文件中列出时，桌面环境下才会自动挂载。

现代桌面环境下的挂载点被选为”/media/*username*/*disk_label*”，它可以被如下所示的来定制。

- FAT 格式的文件系统使用 mlabel(1) 命令
- ISO9660 文件系统使用带有”-V” 选项的 genisoimage(1) 命令
- ext2/ext3/ext4 文件系统使用带有”-L” 选项的 tune2fs(8) 命令

提示
挂载时可能需要提供编码选项（参见第 8.1.3 节）。

提示
在图形界面菜单上移除文件系统，可能会移除它的动态设备节点例如”/dev/sdc”。如果你想要保留它的设备节点，你应该在命令行提示符上输入 umount(8) 命令来卸载它。

10.1.8 选择用于分享数据的文件系统

当你通过可移动存储设备与其他系统分享数据的时候，你应该先把它格式化为被两种操作系统都支持的通用的文件系统。下面是文件系统的列表。

注意
对硬盘 (HDD) 的描述也适合用于其它存储设备，比如 SSD / U 盘 / 内存卡 /……。替换例子中 /dev/sda 的设备名称，用其适合的设备名称 /dev/nvme0、/dev/mmcblk0……。

提示
查看第 9.9.1 节来获得关于使用设备级加密的跨平台的数据共享的信息。

FAT 文件系统被绝大多数的现代操作系统支持，它对于通过可移动硬盘进行的数据交换是非常有用的。

当格式化像装有 FAT 文件系统的跨平台数据共享的可移动设备时，以下应该是保险的选择。

- 用 fdisk(8)，cfdisk(8) 或者 parted(8) 命令（参见第 9.6.2 节）把它们格式化为单个的主分区并对把它做如下标记。

文件系统名	典型使用场景
FAT12	软盘（<32MiB）上跨平台的数据分享
FAT16	在小硬盘 (<2GiB) 上的跨平台的数据分享
FAT32	在大硬盘（<8TiB，被 MS Windows95 OSR2 以上的操作系统所支持) 上的跨平台的数据分享
exFAT	在大硬盘类设备上跨平台共享数据（<512TiB，被 WindowsXP, Mac OS X Snow Leopard 10.6.5 和 Linux 内核 5.4 版本以上的操作系统所支持)
NTFS	在大硬盘类设备上的跨平台共享数据 (在 MS Windows NT 和后续版本原生支持；在 Linux 上，通过使用 FUSE 的 NTFS-3G 支持。)
ISO9660	在 CD-R 和 DVD+/-R 上的跨平台的静态数据分享
UDF	CD-R 和 DVD+/-R (新) 上的增量数据写入
MINIX	软盘上磁盘空间高利用率的 unix 文件数据存储
ext2	在装有老旧 linux 系统的硬盘上的数据分享
ext3	在装有老旧 linux 系统的硬盘上的数据分享
ext4	在装有较新的 linux 系统的硬盘上的数据分享
btrfs	使用只读快照在装有较新的 Linux 系统的硬盘上共享数据

Table 10.3: 典型使用场景下可移动存储设备可选择的文件系统列表

  - 标记小于 2GB 的 FAT 设备为字符”6”。
  - 标记更大的 FAT32 设备为字符”c”。
- 如下所示是用 mkfs.vfat(8) 命令格式化主分区的。
  - 它的设备名字，例如”`/dev/sda1`” 用于 FAT16 设备
  - 明确的选项和它的设备名，例如”`-F 32 /dev/sda1`” 用于 FAT32 设备

当使用 FAT 或 ISO9660 文件系统分享数据时，如下是需要注意的安全事项。

- 用 tar(1)，或 cpio(1) 命令压缩文件，目地是为了保留文件名，符号链接，原始的文件权限和文件所有者信息。
- 用 split(1) 命令把压缩文件分解成若干小于 2GiB 的小文件，使其免受文件大小限制。
- 加密压缩文件保护其内容免受未经授权的访问。

注意
因为 FAT 文件系统的设计，最大的文件大小为 `(2^32 - 1) bytes = (4GiB -1 byte)`。对于一些老旧的 32 位系统上的应用程序而言，最大的文件大小甚至更小 `(2^31 -1) bytes = (2GiB -1 byte)`。Debian 没有遇到后者的问题。

注意
微软系统本身并不建议在超过 200MB 的分区或者驱动器上使用 FAT。他们的” Overview of FAT, HPFS, and NTFS File Systems 这篇文章突出显示了微软系统的缺点，例如低效的磁盘空间利用。当然了，我们在 Linux 系统上还是应该使用 ext4 文件系统。

提示
有关文件系统和访问文件系统的更多信息，请参考”Filesystems HOWTO”。

网络服务	典型使用场景描述
SMB/CIFS 用 Samba 挂载网络文件系统	通过“Microsoft Windows 网络”分享文件，参见 smb.conf(5) 和官方 Samba 3.x.x 指导和参考手册（The Official Samba 3.x.x HOWTO and Reference Guide）
NFS 用 Linux 内核挂载网络文件系统	通过“Unix/Linux 网络” 分享文件，参见 exports(5) 和 Linux NFS-HOWTO
HTTP 服务	在 web 服务器/客户端之间分享文件
HTTPS 服务	在有加密的安全套接层 (SSL) 或者安全传输层 (TLS) 的网络服务器/客户端中分享文件
FTP 服务	在 FTP 服务器/客户端之间分享文件

Table 10.4: 典型使用场景下可选择的网络服务列表

10.1.9 网络上的数据分享

当使用网络来分享数据的时候，你应该使用通用的服务。这里有一些提示。

尽管对于文件分享来说，通过网络挂载文件系统和传输文件是相当方便的，但这可能是不安全的。它们的网络连接必须通过如下所示的加强安全性。

- 用 SSL/TLS 加密
- 建立 SSH 通道
- 建立 VPN 通道
- 网络之间需要有安全的防火墙

参见第 6.5 节和第 6.6 节。

10.2 备份和恢复

我们都熟知计算机有时会出问题，或者由于人为的错误导致系统和数据损坏。备份和恢复操作是成功的系统管理中非常重要的一部分。可能有一天你的电脑就会出问题。

提示
保持你的备份系统简洁并且经常备份你的系统，有备份数据比你采用的备份方法的技术先进要重要的多。

10.2.1 备份和恢复策略

有 3 个关键的因素决定实际的备份和恢复策略。

1. 知道要备份和恢复什么。
   - 你自己创建的数据文件：在”~/” 下的数据
   - 你使用的应用程序创建的数据文件：在”/var/” 下的数据(除了”/var/cache/”，”/var/run/” 和”/var/tmp/”)
   - 系统配置文件：在”/etc/”下的数据
   - 本地程序：在”/usr/local/” 或”/opt/” 下的数据
   - 系统安装信息：关键步骤 (分区,...) 的纯文本备忘录
   - 验证数据结果：通过实验性的恢复操作来预先验证

  - 用户进程的 Cron 工作，文件在"`/var/spool/cron/crontabs`"目录，并且重启 cron(8)。参见第 9.4.14 节来获得关于 cron(8) 和 crontab(1) 的信息。
  - 用户进程的 Systemd 计时器工作：文件在"`~/.config/systemd/user`"目录。参见 systemd.timer(5) 和 systemd.service(5)。
  - 用户进程的自动启动工作：文件在"`~/.config/autostart`"目录。参见 Desktop Application Autostart Specification。

2. 知道怎样去备份和恢复。
   - 安全的数据存储：保护其免于覆盖和系统故障
   - 经常备份：有计划的备份
   - 冗余备份：数据镜像
   - 傻瓜式操作：单个简单命令备份
3. 评估涉及的风险和成本。
   - 数据丢失的风险
     - 数据至少是应该在不同的磁盘分区上，最好是在不同的磁盘和机器上，来承受文件系统发生的损坏。重要数据最好存储在一个只读文件系统上。1
   - 数据非法访问的风险
     - 敏感的身份数据，比如"`/etc/ssh/ssh_host_*_key`", "`~/.gnupg/*`", "`~/.ssh/*`", "`~/.local/share/keyr`"`/etc/passwd`", "`/etc/shadow`", "`popularity-contest.conf`", "`/etc/ppp/pap-secrets`", and "`/etc/ex`应当使用加密备份。2 (参见第 9.9 节。)
     - 即使在信任的系统上，也不能够硬编码系统登录密码或者加密密码到任何脚本里面。(参见第 10.3.6 节。)
   - 数据丢失的方式及其可能性
     - 硬件（特别是硬盘）将会损坏
     - 文件系统可能会损坏，里面的数据可能被丢失
     - 对违规安全访问而言，远程存储系统不能够被信任
     - 弱的密码保护能够被轻松的破解
     - 文件权限系统可以被破解
   - 备份所需的资源：人力，硬件，软件，…
     - 使用 cron 任务或者 systemd 计时器任务来自动化调度备份工作

提示
你能够用"`debconf-set-selections debconf-selections`"命令恢复 debconf 配置数据，可以用"`dpkg --set-selection <dpkg-selections.list`"命令恢复 dpkg 筛选数据。

注意
除非你知道自己做的是什么，否则不要备份 `/proc`, `/sys`, `/tmp`, 和 `/run` 目录下的伪文件系统（参见第 1.2.12 节和第 1.2.13 节)。它们是庞大且无用的数据。

注意
当备份数据的时候，你可能希望停止一些应用程序的守护进程例如 MTA（参见第 6.2.4 节）。

软件包	流行度	大小	说明
bacula-common	V:5.7, I:6.4	2501	Bacula: 网络数据备份，恢复和核查-常见的支持文件
bacula-client	V:0.2, I:1.7	199	Bacula: 网络数据备份，恢复和核查-客户端元软件包
bacula-console	V:0.6, I:2.0	112	Bacula: 网络数据备份，恢复和核查-文本终端
bacula-server	I:0.56	199	Bacula: 网络数据备份，恢复和核查-服务器端元软件包
amanda-common	V:0.5, I:2.0	9952	Amanda: 马里兰大学开发的高级自动化网络磁盘归档器（库）
amanda-client	V:0.5, I:2.0	1110	Amanda: 马里兰大学开发的高级自动化网络磁盘归档器（客户端）
amanda-server	V:0.11, I:0.22	1096	Amanda: 马里兰大学开发的高级自动化网络磁盘归档器（服务器端）
backuppc	V:1.3, I:1.5	3088	BackupPC 是用于备份 PC 机数据（基于磁盘）的高性能的企业级工具
duplicity	V:5, I:44	2679	(远程) 增量备份
deja-dup	V:28, I:41	5274	duplicity 的 GUI（图形用户界面）前端
borgbackup	V:13, I:28	3717	(远程) 去重备份
borgmatic	V:3.4, I:4.4	946	borgbackup 备份软件的辅助软件
rdiff-backup	V:2.1, I:6.0	1204	(远程) 增量备份
restic	V:10, I:17	44216	(远程) 增量备份
backupninja	V:2.0, I:2.4	360	轻量的可扩展的 **meta-backup** 系统
slbackup	V:0.10, I:0.12	147	(远程) 增量备份
backup-manager	V:0.39, I:0.68	573	命令行备份工具
backup2l	V:0.34, I:0.46	110	用于可挂载媒介 (基于磁盘的) 的低维护的备份/恢复工具

Table 10.5: 实用备份程序套件列表

10.2.2 实用备份套件

以下是 Debian 系统上值得注意的实用备份程序套件的列表。

备份工具有各自的专用的用途。

- Mondo Rescue 是一个备份系统，它能够方便的从备份 CD/DVD 等设备中快速恢复整个系统，而不需要经过常规的系统安装过程。
- Bacula，Amanda 和 BackupPC 是全功能的备份实用套件，主要用于联网的定期备份。
- Duplicity 和 Borg 是简单的备份工具用于典型的工作站。

10.2.3 备份技巧

对于一个个人工作站，为服务器环境设计的全功能备份套件工具也行不是最合适的。与此同时，已有的用于工作站的备份工具有一些不足。

这里有一些技巧让备份更加容易，只需用户做最小的工作。这些技巧可以同任意备份工具一起使用。

出于演示的目的，让我们假设基本用户和组名为 `penguin`，创建一个备份和快照脚本例子"`/usr/local/bin/bkss.sh`"：

```
#!/bin/sh -e
SRC="$1" # source data path
DSTFS="$2" # backup destination filesystem path
DSTSV="$3" # backup destination subvolume name
```

1一个只能写一次的媒介，例如 CD/DVD-R，能防止覆盖事故。（参见第 9.8 节怎样在 shell 命令行写入存储媒介。GNOME 桌面图形环境可以让你轻松的通过菜单："位置 → CD/DVD 刻录"来实现写入操作。）

2这些数据中的一些，不能够通过在系统里面输入同样的字符串来重新生成。

```
DSTSS="${DSTFS}/${DSTSV}-snapshot" # snapshot destination path
if [ "$(stat -f -c %T "$DSTFS")" != "btrfs" ]; then
  echo "E: $DESTFS needs to be formatted to btrfs" >&2 ; exit 1
fi
MSGID=$(notify-send -p "bkup.sh $DSTSV" "in progress ...")
if [ ! -d "$DSTFS/$DSTSV" ]; then
  btrfs subvolume create "$DSTFS/$DSTSV"
  mkdir -p "$DSTSS"
fi
rsync -aHxS --delete --mkpath "${SRC}/" "${DSTFS}/${DSTSV}"
btrfs subvolume snapshot -r "${DSTFS}/${DSTSV}" ${DSTSS}/$(date -u --iso=min)
notify-send -r "$MSGID" "bkup.sh $DSTSV" "finished!"
```

这里，只使用基本工具 rsync(8) 来帮助备份，存储空间使用 Btrfs 来高效利用。

提示
提示：这个作者在他的工作站上使用他自己的类似 shell 脚本”bss: Btrfs Subvolume Snapshot Utility”。

10.2.3.1 GUI（图形用户界面）备份

这里是一个创建单击 GUI（图形用户界面）图标备份的例子。

- 准备一个 USB 存储设备用来备份。
  - 格式化 USB 存储设备为一个分区，使用 btrfs 文件系统，卷标名为”BKUP”。这个 U 盘也能够加密 (参见第 9.9.1 节)。
  - 把这个插入你的系统。桌面系统将自动挂载它到”/media/penguin/BKUP”。
  - 执行”sudo chown penguin:penguin /media/penguin/BKUP”，让它可以由用户写。
- 创建如下的”~/.local/share/applications/BKUP.desktop” 文件，按照第 9.4.10 节里写的技术：

```
[Desktop Entry]
Name=bkss
Comment=Backup and snapshot of ~/Documents
Exec=/usr/local/bin/bkss.sh /home/penguin/Documents /media/penguin/BKUP Documents
Type=Application
```

对于每一次图标单击，你的数据从”~/Documents” 备份到 USB 存储设备，并创建了一个只读快照。

10.2.3.2 挂载事件触发的备份

这里是一个由挂载事件触发的自动备份例子。

- 按第 10.2.3.1 节的方式准备一个 USB 存储设备用于备份。
- 创建一个如下的 systemd 服务单元文件”~/.config/systemd/user/back-BKUP.service”：

```
[Unit]
Description=USB Disk backup
Requires=media-%u-BKUP.mount
After=media-%u-BKUP.mount

[Service]
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
StandardOutput=append:%h/.cache/systemd-snap.log
```

```
StandardError=append:%h/.cache/systemd-snap.log

[Install]
WantedBy=media-%u-BKUP.mount
```

- 用下面的方式启用这个 systemd 单元配置：

```
 $ systemctl --user enable bkup-BKUP.service
```

对于每一次挂载事件，你的数据从"~/Documents" 备份到 USB 存储设备，并创建了一个只读快照。

这里，当前 systemd 在内存中的 systemd 挂载单元的名字，用户使用"systemctl --user list-units --type=mount
来调用服务管理器来查询。

10.2.3.3 时间事件触发的备份

这里是一个由时间事件触发的自动备份例子。

- 按第 10.2.3.1 节的方式准备一个 USB 存储设备用于备份。
- 创建一个如下的 systemd 时间单元文件"~/.config/systemd/user/snap-Documents.timer"：

```
[Unit]
Description=Run btrfs subvolume snapshot on timer
Documentation=man:btrfs(1)

[Timer]
OnStartupSec=30
OnUnitInactiveSec=900

[Install]
WantedBy=timers.target
```

- 创建一个如下的 systemd 服务单元文件"~/.config/systemd/user/snap-Documents.service"：

```
[Unit]
Description=Run btrfs subvolume snapshot
Documentation=man:btrfs(1)

[Service]
Type=oneshot
Nice=15
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
IOSchedulingClass=idle
CPUSchedulingPolicy=idle
StandardOutput=append:%h/.cache/systemd-snap.log
StandardError=append:%h/.cache/systemd-snap.log
```

- 用下面的方式启用这个 systemd 单元配置：

```
 $ systemctl --user enable snap-Documents.timer
```

对于每一次时间事件，你的数据从"~/Documents" 备份到 USB 存储设备，并创建了一个只读快照。

这里，当前 systemd 在内存中的 systemd 用户时间单元的名字，使用"systemctl --user list-units --type=timer
来调用服务管理器来查询。

对于现在的桌面系统，这个 systemd 方案，比起传统的 Unix at(1)、cron(8) 或 anacron 方式，能够提供更精致的细粒度控制。

10.3 数据安全基础

数据安全基础设施是数据加密，讯息摘要和签名工具的结合。

软件包	流行度	大小	命令	说明
gnupg	V:403, I:853	465	gpg(1)	GNU 隐私卫士 - OpenPGP 加密和签名工具
gpgv	V:206, I:911	576	gpgv(1)	GNU 隐私卫士 - OpenPGP 签名验证工具
sq	V:1, I:37	24191	sq(1)	Sequoia-PGP - OpenPGP 加密和签名工具
sqv	V:517, I:595	1919	sqv(1)	Sequoia-PGP - OpenPGP 签名验证工具
paperkey	V:1, I:12	58	paperkey(1)	从 OpenPGP 私钥里面，仅仅导出私密信息
cryptsetup	V:35, I:77	476	cryptsetup(8), …	dm-crypt 块设备加密支持 LUKS 工具
coreutils	V:914, I:1000	19751	md5sum(1)	计算与校验 MD5 讯息摘要
coreutils	V:914, I:1000	19751	sha1sum(1)	计算与校验 SHA1 讯息摘要
openssl	V:853, I:996	2564	openssl(1ssl)	使用”openssl dgst” (OpenSSL) 计算信息摘要
libsecret-tools	V:2, I:14	44	secret-tool(1)	存储和取回密码 (CLI)
seahorse	V:76, I:258	7999	seahorse(1)	密钥管理工具 (GNOME)

Table 10.6: 数据安全基础工具列表

参见第 9.9 节的 dm-crypt 和 fscrypt，它们通过 Linux 内核模块实现了自动数据加密架构。

10.3.1 GnuPG 密钥管理

如下是 GNU 隐私卫士基本的密钥管理命令。

命令	说明
gpg --gen-key	生成一副新的密钥对
gpg --gen-revoke my_user_ID	生成 my_user_ID 的一份吊销证书
gpg --edit-key user_ID	交互式的编辑密钥，输入”help” 来获得帮助信息
gpg -o file --export	把所有的公共密钥输出到文件
gpg -o file --export-secret-keys	把所有的私有密钥输出到文件
gpg --import file	从文件导入密钥
gpg --send-keys user_ID	发送 user_ID 的公钥到公钥服务器
gpg --recv-keys user_ID	从公钥服务器下载 user_ID 的公钥
gpg --list-keys user_ID	列出 user_ID 的所有密钥
gpg --list-sigs user_ID	列出 user_ID 的签字
gpg --check-sigs user_ID	检查 user_ID 密钥签字
gpg --fingerprint user_ID	检查 user_ID 的指纹
gpg --refresh-keys	更新本地密钥

Table 10.7: GNU 隐私卫士密钥管理命令的列表

信任码含义.

如下命令生成我的公钥”9563FC29932C409F1A77F9C1AB8A1522D8234C6A” 。

代码	信任描述
-	没有所有者信任签名/没有计算
e	信任计算失败
q	没有足够的信息用于计算
n	从不信任这个键
m	最低限度的信任
f	完全信任
u	最终信任

Table 10.8: 信任码含义列表

```
$ gpg --gen-key
gpg (GnuPG) 2.4.7; Copyright (C) 2024 g10 Code GmbH
...
GnuPG needs to construct a user ID to identify your key.

Real name: Osamu Aoki
Email address: osamu@debian.org
You selected this USER-ID:
    "Osamu Aoki <osamu@debian.org>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
...
public and secret key created and signed.

pub   ed25519 2026-02-14 [SC] [expires: 2029-02-13]
      9563FC29932C409F1A77F9C1AB8A1522D8234C6A
uid                      Osamu Aoki <osamu@debian.org>
sub   cv25519 2026-02-14 [E] [expires: 2029-02-13]
```

如下命令上传我的"9563FC29932C409F1A77F9C1AB8A1522D8234C6A"公钥到主流的公钥服务器"hkp://keys.gnupg.

```
$ gpg --keyserver hkp://keys.gnupg.net --send-keys 9563FC29932C409F1A77F9C1AB8A1522D8234C6A
```

默认良好的公钥服务器在"~/.gnupg/gpg.conf"（旧的位置在"~/.gnupg/options"）文件中设置，此文件包含了以下信息。

```
keyserver hkp://keys.gnupg.net
```

从钥匙服务器获取无名钥匙。

```
$ gpg --list-sigs --with-colons | grep '^sig.*\[User ID not found\]' |\
          cut -d ':' -f 5| sort | uniq | xargs gpg --recv-keys
```

有一个错误在 OpenPGP 公钥服务器 (先前的版本 0.9.6)，会将键中断为 2 个以上的子键。新的 gnupg (>1.2.1-2) 软件包能够处理这些中断的子键。参见 gpg(1) 下的"--repair-pks-subkey-bug" 选项.

使用 SHA-1 作为哈希算法已弃用。如果基于 openPGP 的 RSA 使用了 SHA-1 作为哈希算法，可以使用 FixKey-WithSha1 来修复。

> 提示
> sq(1) 和 sqv(1) 命令是 openPGP 命令的替代命令集合。参见 sq 用户文档和使用 sq 的实践介绍，Sequoia PGP 命令行.

10.3.2 在文件上使用 GnuPG

这里有一些在文件上使用 GNU 隐私卫士命令的例子。

命令	说明
`gpg -a -s file`	ASCII 封装的签名文件 file.asc
`gpg --armor --sign file`	,,
`gpg --clearsign file`	生成明文签字信息
`gpg --clearsign file|mail foo@example.org`	发送一份明文签字到 `foo@example.org`
`gpg --clearsign --not-dash-escaped patchfile`	明文签名的补丁文件
`gpg --verify file`	验证明文文件
`gpg -o file.sig -b file`	生成一份分离的签字
`gpg -o file.sig --detach-sign file`	,,
`gpg --verify file.sig file`	使用 file.sig 验证文件
`gpg -o crypt_file.gpg -r name -e file`	公钥加密，从文件里面获取名字，生成二进制的 crypt_file.gpg
`gpg -o crypt_file.gpg --recipient name --encrypt file`	,,
`gpg -o crypt_file.asc -a -r name -e file`	公钥加密，从文件中获取名字，生成 ASCII 封装的 crypt_file.asc
`gpg -o crypt_file.gpg -c file`	将文件对称加密到 crypt_file.gpg
`gpg -o crypt_file.gpg --symmetric file`	,,
`gpg -o crypt_file.asc -a -c file`	对称加密，从文件到 ASCII 封装的 crypt_file.asc
`gpg -o file -d crypt_file.gpg -r name`	解密
`gpg -o file --decrypt crypt_file.gpg`	,,

Table 10.9: 在文件上使用的 GNU 隐私卫士的命令列表

10.3.3 在 Mutt 中使用 GnuPG

增加下面内容到"~/.muttrc"，在自动启动时，避免一个慢的 GnuPG，在索引菜单中按"S" 来允许它使用。

```
macro index S ":toggle pgp_verify_sig\n"
set pgp_verify_sig=no
```

10.3.4 在 Vim 中使用 GnuPG

gnupg 插件可以让你对扩展名为".gpg", ".asc", 和".pgp" 的文件可靠的运行 GnuPG。[3]

```
$ sudo aptitude install vim-scripts
$ echo "packadd! gnupg" >> ~/.vim/vimrc
```

10.3.5 MD5 校验和

md5sum(1) 提供了制作摘要文件的一个工具, 它使用 rfc1321 里的方式制作摘要文件.

```
$ md5sum foo bar >baz.md5
$ cat baz.md5
d3b07384d113edec49eaa6238ad5ff00  foo
c157a79031e1c40f85931829bc5fc552  bar
$ md5sum -c baz.md5
foo: OK
bar: OK
```

注意
MD5 校验和的 CPU 计算强度是比 GNU Privacy Guard (GnuPG) 加密签名要少的. 在通常情况下, 只有顶级的摘要文件才需要加密签名来确保数据完整性.

10.3.6 密码密钥环

在 GNOME 系统，GUI(图形用户界面)工具 seahorse(1) 管理密码，安全的在密钥环 ~/.local/share/keyrings/* 里面保存它们。

secret-tool(1) 能够从命令行存储密码到钥匙环。

让我们存储 LUKS/dm-crypt 加密磁盘镜像用到的密码

```
$ secret-tool store --label='LUKS passphrase for disk.img' LUKS my_disk.img
Password: ********
```

这个存储的密码能够被获取并给到其它程序，比如 cryptsetup(8)。

```
$ secret-tool lookup LUKS my_disk.img | \
  cryptsetup open disk.img disk_img --type luks --keyring -
$ sudo mount /dev/mapper/disk_img /mnt
```

提示
无论何时，你需要在一个脚本里面提供密码时，使用 secret-tool 来避免将密码直接硬编码到脚本里面。

[3] 如果你使用"~/.vimrc" 代替"~/.vim/vimrc"，请进行相应的取代。

10.4 源代码合并工具

这里有许多源代码合并工具。如下的是我感兴趣的工具。

软件包	流行度	大小	命令	说明
patch	V:100, I:729	243	patch(1)	给原文件打补丁
vim	V:73, I:330	4415	vimdiff(1)	在 vim 中并排比较两个文件
imediff	V:0.02, I:0.21	329	imediff(1)	全屏交互式两路/三路合并工具
meld	V:5, I:23	3996	meld(1)	比较和移植文件 (GTK)
wiggle	V:0.01, I:0.12	191	wiggle(1)	应用被拒绝的补丁
diffutils	V:901, I:998	1768	diff(1)	逐行比较两个文件
diffutils	V:901, I:998	1768	diff3(1)	逐行比较和合并三个文件
quilt	V:2, I:16	880	quilt(1)	管理系列补丁
wdiff	V:5, I:37	580	wdiff(1)	在文本文件中，显示单词的不同
diffstat	V:8, I:91	86	diffstat(1)	通过 diff 生成一个改变柱状图
patchutils	V:10, I:91	269	combinediff(1)	从两个增量补丁创建一个积累补丁
patchutils	V:10, I:91	269	dehtmldiff(1)	从一个 HTML 页面提取出一个 diff
patchutils	V:10, I:91	269	filterdiff(1)	从一个 diff 文件里面提取或者排除 diff 文件
patchutils	V:10, I:91	269	fixcvsdiff(1)	修复由 CVS patch(1) 错误创建的 diff 文件
patchutils	V:10, I:91	269	flipdiff(1)	交换两个补丁的顺序
patchutils	V:10, I:91	269	grepdiff(1)	显示哪些文件是由匹配正则表达式的补丁修改
patchutils	V:10, I:91	269	interdiff(1)	显示在两个统一格式 diff 文件（基于同一个文件的两个不同 diff 文件）之间的差异
patchutils	V:10, I:91	269	lsdiff(1)	显示哪些文件由补丁修改
patchutils	V:10, I:91	269	recountdiff(1)	重新计算通用内容 diff 文件的数量和偏移
patchutils	V:10, I:91	269	rediff(1)	修复手工编辑 diff 文件的数量和偏移
patchutils	V:10, I:91	269	splitdiff(1)	隔离出增量补丁
patchutils	V:10, I:91	269	unwrapdiff(1)	识别已经被分词的补丁
dirdiff	V:0.1, I:1.3	167	dirdiff(1)	显示目录树之间的不同并移植改变
docdiff	V:0.02, I:0.15	554	docdiff(1)	逐词逐字地比较两个文件
makepatch	V:0.01, I:0.14	99	makepatch(1)	生成扩展补丁文件
makepatch	V:0.01, I:0.14	99	applypatch(1)	应用扩展补丁文件

Table 10.10: 源代码合并工具列表

10.4.1 从源代码文件导出差异

下面的操作，导出两个源文件的不同，并根据文件的位置，创建通用 diff 文件"`file.patch0`" 或"`file.patch1`".

```
$ diff -u file.old file.new > file.patch0
$ diff -u old/file new/file > file.patch1
```

10.4.2 源代码文件移植更新

diff 文件（通常被叫作 patch 补丁文件），用于发送一个程序更新。通过下面的方式，接收到的部分，应用这个更新到其它文件。

```
$ patch -p0 file < file.patch0
$ patch -p1 file < file.patch1
```

10.4.3 交互式移植

如果一个源代码，你有两个版本，你可以通过下面的方式，使用 imediff(1) 执行两方交互式移植。

```
$ imediff -o file.merged file.old file.new
```

如果一个源代码，你有三个版本，你可以通过下面的方式，使用 imediff(1) 执行交互式三方移植。

```
$ imediff -o file.merged file.yours file.base file.theirs
```

10.5 Git

Git 是这些天选择的用于版本控制系统 version control system (VCS) 的工具，因为 Git 能够同时在本地和远程源代码管理上，做任何事情。

通过 Debian Salsa 服务，Debian 能够提供免费的 Git 服务。在 https://wiki.debian.org/Salsa 能找到它的说明文档。

下面是一些 Git 相关软件包。

软件包	流行度	大小	命令	说明
git	V:442, I:630	52512	git(1)	Git 快速、可扩展、分布式的版本控制系统
gitk	V:3, I:25	2045	gitk(1)	有历史功能的 Git 图形仓库浏览器
qgit	V:0.2, I:1.8	1431	qgit(1)	有历史功能的 Git 图形仓库浏览器
git-cola	V:0.7, I:4.4	5331	git-cola(1)	有历史功能的 Git 图形仓库浏览器
tig	V:2, I:11	1245	tig(1)	浏览 Git 历史仓库的控制台
lazygit	V:1.0, I:4.1	24167	lazygit(1)	浏览 Git 历史仓库的控制台
git-gui	V:1, I:15	2549	git-gui(1)	Git 图形界面（无历史功能）
git-email	V:0.4, I:8.6	1229	git-send-email(1)	从 Git 用电子邮件发送收集到的补丁
git-buildpackage	V:1.2, I:6.5	2027	git-buildpackage(1)	用 Git 自动制作 Debian 包
dgit	V:0.12, I:0.91	758	dgit(1)	Debian 档案库的 git 交互操作
imediff	V:0.02, I:0.21	329	git-ime(1)	交互式的分开 git 提交的辅助工具
stgit	V:0.07, I:0.45	6005	stg(1)	封装的 git (Python)
git-doc	I:9.6	15192	N/A	Git 官方文档
gitmagic	I:0.43	721	N/A	”Git 魔术”，易于理解的 Git 手册

Table 10.11: git 相关包和命令列表

10.5.1 配置 Git 客户端

你可以在”~/.gitconfig” 里面设置几个 Git 接下来需要使用的全局配置，比如说你的名字和电子邮件地址。

```
$ git config --global user.name "Name Surname"
$ git config --global user.email yourname@example.com
```

你也可以按如下所示定制 Git 的默认行为。

```
$ git config --global init.defaultBranch main
$ git config --global pull.rebase true
$ git config --global push.default current
```

如果你习惯使用 CVS 或 Subversion 命令，你也许希望设置如下几个命令别名。

```
$ git config --global alias.ci "commit -a"
$ git config --global alias.co checkout
```

你能够通过如下方式检查你的全局配置：

```
$ git config --global --list
```

10.5.2 基本的 Git 命令

Git 操作涉及几个数据。

- 工作树目录保持面向用户的文件，你可以对这些文件做修改。
  - 需要被记录的改变，必须明确的被选择并暂存到索引。这是 `git add` 和 `git rm` 命令。
- 索引保持暂存文件。
  - 在接下来的请求之前，暂存文件将被提交到本地仓库。这个是 `git commit` 命令。
- 本地仓库保持已经提交的的文件。
  - Git 记录提交数据的链接历史并在仓库里面将它们作为分支组织。
  - 本地仓库通过 `git push` 命令发送数据到远程仓库。
  - 本地仓库能够通过 `git fetch` 和 `git pull` 命令从远程仓库接收数据。
    - `git pull` 命令在 `git fetch` 后执行 `git merge` 或 `git rebase` 命令。
    - 这里，`git merge` 联合两个独立分支的历史结尾到一个点。(在没有定制的 `git pull` ，这个是默认的，同时对上游作者发布分支到许多人时，也是好的)
    - 这里，`git rebase` 创建一个远程分支的序列历史的单个分支，跟着本地分支。(这是定制 `pull.rebase true` 的情况，对我们其余的用途有用。)
- 远程仓库保持已经提交的文件。
  - 到远程仓库的通信，使用安全的通信协议，比如 SSH 或 HTTPS。

工作树是在 `.git/` 目录之外的文件。在 `.git/` 目录里面的文件，包括索引、本地仓库数据和一些 git 配置的文本文件。

这里是主要的 Git 命令概览。

Git 命令	功能
`git init`	创建 (本地) 存储库
`git clone URL`	克隆远程存储库到本地仓库工作目录树
`git pull origin main`	通过远程仓库 `origin` 更新本地 `main` 分支
`git add .`	增加工作树里面的文件仅作为预先存在的索引文件
`git add -A .`	增加工作树里面的所有文件到索引（包括已经删除的）
`git rm filename`	从工作树和索引中删除文件
`git commit`	提交在索引中的暂存改变到本地存储库
`git commit -a`	添加工作树里的所有的改变到索引并提交它们到本地仓库（添加 + 提交）
`git push -u origin branch_name`	使用本地 `branch_name` 分支更新远程仓库 `origin`（初始启用）
`git push origin branch_name`	使用本地 `branch_name` 分支更新远程仓库 `origin`（随后调用）
`git diff treeish1 treeish2`	显示 *treeish1* 提交和 *treeish2* 提交的不同
`gitk`	VCS 存储库分支历史树的图形界面显示

Table 10.12: 主要的 Git 命令

10.5.3 Git 技巧

下面是一些 Git 技巧。



警告
不要使用带空格的标签字符串。即使一些工具，如 gitk(1) 允许你使用它，但会阻碍其它 `git` 命令。



小心
如果一个本地分支推送到一个已经变基或者压缩过的仓库，推送这样的分支有风险，并要求 `--force` 选项。这通常对 `main` 分支来说不可接受，但对于一个移植到 `main` 分支前的特定分支，是可以接受的。



小心
从命令行通过”`git-xyz`” 直接调用 `git` 子命令的方式，从 2006 年早期开始就被取消。

提示
如果有一个可执行文件 `git-foo` 在路径环境变量 `$PATH` 里面, 在命令行输入没有中划线的”`git foo`”，则将调用 `git-foo`. 这是 `git` 命令的一个特性。

10.5.4 Git 参考

参见下面内容。

- man 手册：git(1) (`/usr/share/doc/git-doc/git.html`)
- Git 用户手册 (`/usr/share/doc/git-doc/user-manual.html`)
- git 介绍教程 (`/usr/share/doc/git-doc/gittutorial.html`)

Git 命令行	功能
`gitk --all`	参看完整的 Git 历史和操作，比如重置 HEAD 到另外一个提交、挑选补丁、创建标签和分支……
`git stash`	得到一个干净的工作树，不会丢失数据
`git remote -v`	检查远程设置
`git branch -vv`	检查分支设置
`git status`	显示工作树状态
`git config -l`	列出 git 设置
`git reset --hard HEAD; git clean -x -d -f`	反转所有工作树的改变并完全清理它们
`git rm --cached filename`	反转由 `git add filename` 改变的暂存索引
`git reflog`	获取参考日志（对从删除的分支中恢复提交有用）
`git branch new_branch_name HEAD@{6}`	从 reflog 信息创建一个新的分支
`git remote add new_remote URL`	增加一个由 URL 指向的远程仓库 `new_remote`
`git remote rename origin upstream`	远程仓库的名字从 `origin` 重命名到 `upstream`
`git branch -u upstream/branch_name`	设置远程跟踪到远程仓库 `upstream` 和它的分支名 `branch_name`。
`git remote set-url origin https://foo/bar.git`	改变 `origin` 的 URL
`git remote set-url --push upstream DISABLED`	禁止推送到 `upstream`（编辑 `.git/config` 来重新启用）
`git remote update upstream`	获取 `upstream` 仓库中所有远程分支更新
`git fetch upstream foo:upstream-foo`	创建本地（可能是孤立的）`upstream-foo` 分支，作为 `upstream` 仓库中 `foo` 分支的一个拷贝
`git checkout -b topic_branch ; git push -u topic_branch origin`	制作一个新的 `topic_branch` 并把它推送到 `origin`
`git branch -m oldname newname`	本地分支改名
`git push -d origin branch_to_be_removed`	删除远程分支（新的方式）
`git push origin :branch_to_be_removed`	删除远程分支（老的方式）
`git checkout --orphan unconnected`	创建一个新的 `unconnected` 分支
`git rebase -i origin/main`	从 `origin/main` 重新排序、删除、压缩提交到一个干净的分支历史
`git reset HEAD^; git commit --amend`	压缩最后两个提交为一个
`git checkout topic_branch ; git merge --squash topic_branch`	压缩整个 `topic_branch` 到一个提交
`git fetch --unshallow --update-head-ok origin '+refs/heads/*:refs/heads/*'`	反转一个浅克隆到一个所有分支的完整克隆
`git ime`	分开最后的提交到一系列单个逐一文件的小提交。（要求 `imediff`）
`git repack -a -d; git prune`	本地仓库重新打包到一个单独的包中（这可能限制从删除分支里面恢复丢失数据等机会）

Table 10.13: Git 技巧

- git 介绍教程：第二部 (`/usr/share/doc/git-doc/gittutorial-2.html`)
- GIT 每一天 20 个左右的命令 (`/usr/share/doc/git-doc/giteveryday.html`)
- Git 魔术 (`/usr/share/doc/gitmagic/html/index.html`)

10.5.5 其它的版本控制系统

版本控制系统（VCS ）有时被认为是修订控制系统 (RCS), 或者是软件配置管理程序 (SCM)。

这里是 Debian 系统上著名的其它非 Git 的 VCS 汇总。

软件包	流行度	大小	工具	**VCS** 类型	描述
`mercurial`	V:3, I:24	2667	Mercurial	分布式	mercurial 主要是用 Python 写的还有一部分是 C 写的
`darcs`	V:0.2, I:2.9	38856	Darcs	分布式	有智能代数补丁的 DVCS（慢）
`tla`	V:0.03, I:0.60	1022	GNU arch	分布式	主要由 Tom Lord 写的 DVCS (成为历史的)
`bzr`	V:0.1, I:3.8	28	GNU Bazaar	分布式	受 `tla` 启发并且是用 Python 写的 DVCS
`subversion`	V:9, I:53	5211	Subversion	远程	”比 CVS 做的好“，远程 VCS 的新标准（历史）
`cvs`	V:3, I:22	4835	CVS	远程	以前的远程 VCS 标准（历史）
`tkcvs`	V:0.10, I:0.87	34	CVS, …	远程	VCS (CVS,Subversion,RCS) 存储库树的图形界面显示
`rcs`	V:1.6, I:8.8	578	RCS	本地	” 比 Unix SCCS 做的好”（历史）
`cssc`	V:0.01, I:0.27	2044	CSSC	本地	Unix SCCS 的克隆（历史）

Table 10.14: 其它版本控制系统工具列表

Chapter 11

数据转换

下面是关于 Debian 系统上可用的格式转化工具及其相关提示的信息。
基于标准的工具，是非常好用的，但支持的专有数据格式有限.

11.1 文本数据转换工具

如下是文本数据转换工具。

软件包	流行度	大小	关键词	说明
libc6	V:924, I:1000	5483	字符集	使用 iconv(1) 的不同语言环境（locale）之间的文本编码转换器 (基础的)
recode	V:2, I:12	534	字符集 + 换行	不同语言环境（locale）之间的文本编码转换器（多功能的，更多别名和特性）
konwert	V:2, I:42	137	字符集	不同语言环境（locale）之间的文本编码转换器（高档的）
nkf	V:0.4, I:8.3	360	字符集	日语字符集翻译
tcs	V:0.01, I:0.12	518	字符集	字符集翻译
unaccent	V:0.03, I:0.26	35	字符集	代替重音字符，使用和它们相当的非重音字符
tofrodos	V:1, I:11	50	换行	在 DOS 和 Unix 之间的文本格式转换: fromdos(1) 和 todos(1)
macutils	V:0.03, I:0.38	319	换行	在 Macintosh 和 Unix 之间的文本格式转换: frommac(1) 和 tomac(1)

Table 11.1: 文本数据转化工具列表

11.1.1 用 iconv 命令来转换文本文件

提示
iconv(1) 是 libc6 软件包的一部分并且它可以在类 Unix 的系统上转换字符的编码。

你能够通过如下的命令用 iconv(1) 来转换文本文件的编码。

```
$ iconv -f encoding1 -t encoding2 input.txt >output.txt
```

编码值是大小写不敏感的，且会在匹配时忽略“-”和“_”。可以使用“iconv -l”命令检查支持的编码。

编码值	用法
ASCII	美国信息交换标准代码，7 位代码不带重音符号
UTF-8	用于所有现代操作系统的多语言标准
ISO-8859-1	旧的西欧语言标准，ASCII + 重音符号
ISO-8859-2	旧的东欧语言标准，ASCII + 重音符号
ISO-8859-15	旧的带有欧元符号的西欧语言标准（ISO-8859-1）
CP850	code page 850，用于西欧语言的微软 DOS 的带有图形的字符，ISO-8859-1 的变体
CP932	code page 932, 日语 Microsoft Windows 的 Shift-JIS 变体
CP936	code page 936，用于简体中文的微软操作系统风格的 GB2312，GBK 或者 GB18030 的变体
CP949	code page 949，用于韩语的微软操作系统风格的 EUC-KR 或者 Unified Hangul Code 的变体
CP950	code page 950，用于繁体中文的微软操作系统风格的 Big5 的变体
CP1251	code page 1251，用于西里尔字母的微软操作系统风格的编码
CP1252	code page 1252，用于西欧语言的微软操作系统风格的 ISO-8859-15 的变体
KOI8-R	用于西里尔字母的旧俄语 UNIX 标准
ISO-2022-JP	日文邮件的标准编码，只使用 7 位字节
eucJP	老的日文 UNIX 标准的 8 位字节，和 Shift-JIS 完全不同
Shift-JIS	日文 JIS X 0208 附录 1 标准 (参见 CP932)

Table 11.2: 编码值和用法的列表

注意
一些编码只支持数据转换，它不能作为语言环境的值 (第 8.1 节)。

像 ASCII 和 ISO-8859 这样适用于单字节的字符集，字符编码和字符集几乎指的是同一件事情。

对于多字符的字符集，比如说，用于日文的 JIS X 0213，或用于差不多所有语言的 Universal Character Set (UCS, Unicode, ISO-10646-1)，有多种编码方案来序列化它们的字节数据。

- 日文的 EUC 和 ISO/IEC 2022 (也被称为 JIS X 0202)
- Unicode 的 UTF-8、UTF-16/UCS-2 和 UTF-32/UCS-4 编码

对于以上这些，字符集和字符编码之间有着明显的区别。

对某些计算机厂家而言，code page 是作为字符编码表的同义词来使用。

注意
请注意，大部分编码系统共享 ASCII 的 7 位字符的同样编码，但也有一些列外。如果你从通常所说的 shift-JIS 编码格式，转化老的日文 C 语言程序和 URL 数据，到 UTF-8 格式，你需要使用”CP932” 作为编码名来代替”shift-JIS” 来得到期望的结果：0x5C → ”\” 和 0x7E → ”~”. 否则，这些将被转化为错误的字符。

提示
recode(1) 也可能被使用并且不仅仅是 iconv(1)，fromdos(1)，todos(1)，frommac(1) 和 tomac(1) 功能的结合。想要获得更多信息，请参见”info recode”。

11.1.2 用 iconv 检查文件是不是 UTF-8 编码

你能够通过如下命令用 iconv(1) 来检查一个文本文件是不是用 UTF-8 编码的。

```
$ iconv -f utf8 -t utf8 input.txt >/dev/null || echo "non-UTF-8 found"
```

提示
在上面的例子中使用"`--verbose`" 参数来找到第一个 non-UTF-8 字符。

11.1.3 使用 iconv 转换文件名

这里是一个示例脚步，在同一目录下，将在老的操作系统系统下创建的文件名编码，转换为现代 UTF-8.

```
#!/bin/sh
ENCDN=iso-8859-1
for x in *;
 do
 mv "$x" "$(echo "$x" | iconv -f $ENCDN -t utf-8)"
done
```

"`$ENCDN`" 变量定义了在老的操作系统下，文件名使用的原始编码，见表 11.2.

对于更加复杂的情况，请使用适当的编码作为 mount(8) 的选项 (参见第 8.1.3 节) 来挂载包含有这样文件名的文件系统（比如说，磁盘上的一个分区），使用"`cp -a`" 命令来拷贝它的整个内容到另外一个使用 UTF-8 挂载的文件系统上。

11.1.4 换行符转换

文本文件的格式，特别是行尾（换行符）编码，有平台独立性。

平台	换行符编码	控制码	十进制	**16** 进制
Debian (unix)	LF	`^J`	10	0A
MSDOS 和 Windows	CR-LF	`^M^J`	13 10	0D 0A
苹果的 Macintosh	CR	`^M`	13	0D

Table 11.3: 不同平台的换行符样式列表

用于转换行尾（EOL）格式的程序 fromdos(1) 或 todos(1)（参见 `tofrodos` 软件包），以及 frommac(1) 或 tomac(1)（参见 `macutils` 软件包）都非常实用。recode(1) 也很有用。

注意
大部分编辑器 (比如：`vim`, `emacs`, `gedit`, ⋯) 能够透明处理 MSDOS 式样的换行符文件。

提示
对于混合 MSDOS 和 Unix 式样的文件，统一到 MSDOS 换行符式样，使用"`sed -e '/\r$/!s/$/\r/'`" 代替 todos(1) 比较好。(例如，在使用 diff3(1) 移植两个 MSDOS 式样的文件后.) 这是因为 `todos` 给所有的行增加 CR.

功能	**bsdmainutils**	**coreutils**
把制表符扩展成空格	"col -x"	expand
将空格转换为制表符（unexpand）	"col -h"	unexpand

Table 11.4: bsdmainutils 和 coreutils 包中的用于转换 TAB 的命令列表

11.1.5 TAB 转换

这里有一些转换 tab 代码的专业工具。

indent 包中的 indent(1) 命令能够重新格式化 C 程序中的空格。

例如 vim 和 emacs 这样的编辑软件可以被用来扩展 TAB。就拿 vim 来说，你能够按顺序输入":set expandtab" 和":%retab" 命令来扩展 TAB。你也可以按顺序输入:%set noexpandtab" 和":%retab" 命令来复原。

11.1.6 带有自动转换功能的编辑器

像 vim 这样的现代智能编辑器软件是相当聪明的并且能够处理任何编码系统以及任何文件格式。你应该在支持 UTF-8 编码的控制台上并在 UTF-8 环境下使用这些编辑器来获得最好的兼容性。

以 latin1（iso-8859-1）编码存储的旧西欧语言的 Unix 文本文件，"u-file.txt"，能通过如下所示的用 vim 轻易的编辑。

```
$ vim u-file.txt
```

这是可能的因为 vim 的文件编码自动检测机制先假定文件是 UTF-8 编码，如果失败了，则假定它是 latin1 编码。

以 latin2（iso-8859-2）编码存储的旧波兰语的 Unix 文本文件，"pu-file.txt"，能通过如下所示的用 vim 编辑。

```
$ vim '+e ++enc=latin2 pu-file.txt'
```

以 eucJP 编码存储的旧日语的 Unix 文本文件，"ju-file.txt"，能通过如下所示的用 vim 编辑。

```
$ vim '+e ++enc=eucJP ju-file.txt'
```

以所谓的 shift-JIS 编码 (更确切的说法是：CP932) 存储的旧日语 MS-Windows 文本文件，"jw-file.txt"，能通过如下所示的用 vim 编辑。

```
$ vim '+e ++enc=CP932 ++ff=dos jw-file.txt'
```

当一个文件用 vim 打开的时候带有"++enc" 和"++ff" 选项，在 Vim 命令行输入":w" 命令会以原格式存储文件并且会覆盖原文件。你也可以在 Vim 命令行指定存储文件名及其格式，例如，":w ++enc=utf8 new.txt"。

请查阅 vim 在线帮助中的 mbyte.txt，" 多字节文本支持" 和表 11.2 来获得"++enc" 使用的本地值的信息。

emacs 家族的程序能够实现同样的功能。

11.1.7 提取纯文本

如下所示读入 web 页面并把它转化成文本文件。当从 Web 中拷贝配置或者是在 web 页面中应用类似 grep(1) 的基础 Unix 文本工具时，以下命令是非常有用的。

```
$ w3m -dump https://www.remote-site.com/help-info.html >textfile
```

同样，你可以使用如下所示的工具从其他格式提取纯文本数据。

软件包	流行度	大小	关键词	功能
w3m	V:11, I:125	2853	html→text	用”w3m -dump” 命令把 HTML 转化为文本的转换器
html2text	V:4, I:66	298	html→text	高级的 HTML 到文本文件的转换器 (ISO8859-1)
lynx	V:30, I:454	2031	html→text	用”lynx -dump” 命令把 HTML 转化为文本的转化器
elinks	V:2, I:14	1839	html→text	用”elinks -dump” 命令把 HTML 转化为文本的转换器
links	V:2, I:19	2344	html→text	用”links -dump” 命令把 HTML 转化为文本的转换器
links2	V:0.8, I:9.2	5497	html→text	用”links2 -dump” 命令把 HTML 转化为文本的转换器
catdoc	V:16, I:169	682	MSWord→tex	转化 MSWord 文件到纯文本或 TeX 文件
antiword	V:0.9, I:6.1	596	MSWord→tex,ps	转化 MSWord 文件到纯文本或 ps 文件
unhtml	V:0.04, I:0.50	40	html→text	从一个 HTML 文件里面删除标记标签
odt2txt	V:1, I:17	60	odt→text	从开放文档格式到文本格式的转化器

Table 11.5: 用于提取纯文本数据的工具列表

软件包	流行度	大小	关键词	说明
vim-runtime	V:18, I:347	39473	高亮	用”:source $VIMRUNTIME/syntax/html.vim” Vim 宏命令转化源代码到 HTML
cxref	V:0.02, I:0.20	1191	c→html	从 C 程序到 latex 和 HTML 的转换器（C 语言）
src2tex	V:0.02, I:0.14	1799	高亮	转换许多源代码到 TeX（C 语言）
source-highlight	V:0.4, I:3.0	2131	高亮	转换源代码到带有高亮显示的 HTML, XHTML, LaTeX, Texinfo, ANSI 颜色转义序列和 DocBook 文件 (C++)
highlight	V:0.3, I:2.7	1452	高亮	转化许多源代码到带有高亮显示的 HTML, XHTML, RTF, LaTeX, TeX or XSL-FO 文件。(C++)
grc	V:0.9, I:5.4	208	text→color	用于任何文本的通用颜色生成器 (Python)
pandoc	V:12, I:48	209559	text→any	通用标记转化器 (Haskell)
python3-docutils	V:13, I:51	2012	text→any	重构文本文档到 XML (Python)
markdown	V:0.4, I:5.5	56	text→html	Markdown 文本文档到 (X)HTML (Perl)
asciidoctor	V:0.4, I:4.4	101	text→any	AsciiDoc 文本文档格式化到 XML/HTML (Ruby)
python3-sphinx	V:7, I:27	3245	text→any	基于文档发布系统 (Python) 重构文本
hugo	V:0.8, I:5.0	66919	text→html	基于 Markdown 的静态站点发布系统 (Go)

Table 11.6: 高亮纯文本数据的工具列表

11.1.8 高亮并格式化纯文本数据

你可以通过如下所示的来高亮并格式化纯文本数据。

11.2 XML 数据

扩展标记语言 Extensible Markup Language (XML) 是一种标记语言，用于含有结构化信息的文档。

在 XML.COM 查看介绍信息.

- ” 什么是 XML?”
- ” 什么是 XSLT?”
- ” 什么是 XSL-FO?”
- ” 什么是 XLink?”

11.2.1 XML 的基本提示

XML 文本看起来有些像 HTML. 它能够使我们管理一个文档的多个格式。一个简单的 XML 系统是 `docbook-xsl` 软件包，在这里使用。

每一个 XML 文件使用下面的标准 XML 声明开始。

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML 元素的基本语法是按下面的方式标记。

```
<name attribute="value">content</name>
```

内容为空的 XML 元素，使用下面的短格式标记。

```
<name attribute="value" />
```

上面列子中的”`attribute="value"`” 是可选的。

XML 里面的注释部分，是按下面的方式标记。

```
<!-- comment -->
```

不同于增加标记，XML 至少要求使用预定义实体里的内容来转化下列字符。

预定义实体	转化的字符
`"`	`"`：引号
`'`	`'`：撇号
`<`	`<`：小于号
`>`	`>`：大于号
`&`	`&`：&号

Table 11.7: XML 预定义实体列表



小心
“<”或“&”不能在属性（attributes）或元素（elements）中使用。

注意
当 SGML 式样的用户定义实体，比如"&some-tag;"，被使用的时候，第一个定义会覆盖其它的。实体定义在"<!ENTITY some-tag "entity value">"里表示。

注意
只要 XML 标记是一致使用某一标签名集合（一些数据作为内容或属性值），使用 Extensible Stylesheet Language Transformations (XSLT) 来转换到另外一个 XML，是一个微不足道的任务。

11.2.2 XML 处理

有许多工具可以用于处理 XML 文件，比如说：可扩展样式表语言 Extensible Stylesheet Language (XSL).

一旦你创建了一个好的成形的 XML 文件，基本上来讲，你就可以使用可扩展样式表语言转换 Extensible Stylesheet Language Transformations (XSLT)，将其转换成任何格式。

格式化对象的可扩展样式表语言 Extensible Stylesheet Language for Formatting Objects (XSL-FO) 是用来作为格式化的解决方案. `fop` 软件包比 Debian `main` 档案库要新，因为它依赖 Java 编程语言. LaTeX 代码通常是从 XML 使用 XSLT 生成，LaTeX 系统是用来创建 DVI, PostScript 和 PDF 这类可打印的文件。

软件包	流行度	大小	关键词	说明
`docbook-xml`	V:15, I:403	2126	xml	DocBook 的 XML 文档类型定义（DTD）
`docbook-xsl`	V:15, I:141	14823	xml/xslt	使用 XSLT 将 DocBook XML 处理成各种输出格式的 XSL 样式表
`xsltproc`	V:15, I:69	83	xslt	XSLT 命令行处理器 (XML → XML, HTML, 纯文本，等等)
`xmlto`	V:0.5, I:7.7	124	xml/xslt	使用 XSLT 将 XML 转换到任意格式的转换器
`fop`	V:0.6, I:7.6	282	xml/xsl-fo	转换 Docbook XML 文件到 PDF
`dblatex`	V:0.9, I:5.3	4635	xml/xslt	使用 XSLT 将 Docbook 文件转换为 DVI, PostScript, PDF 文档
`dbtoepub`	V:0.04, I:0.52	37	xml/xslt	DocBook XML 到.epub 转换

Table 11.8: XML 工具列表

由于 XML 是标准通用标记语言 Standard Generalized Markup Language (SGML) 的一个子集, 用于处理 SGML 的扩展工具，也能够处理 XML，比如说文档式样语言和规范语言 Document Style Semantics and Specification Language (DSSSL).

软件包	流行度	大小	关键词	说明
`openjade`	V:1, I:20	1066	`dsssl`	ISO/IEC 10179:1996 标准 DSSSL 处理器 (最新的)
`docbook-dsssl`	V:0.5, I:7.2	2594	`xml/dsssl`	使用 DSSSL 处理 DocBook XML 到各种输出格式的 DSSSL 样式表
`docbook-utils`	V:0.3, I:5.1	287	`xml/dsssl`	DocBook 文件的工具包，包括使用 DSSSL 的转换成其它格式 (HTML, RTF, PS, man, PDF) 的 `docbook2*` 命令

Table 11.9: DSSSL 工具列表

提示
GNOME 的 `yelp` 往往能够方便的直接读取 DocBook XML 文件，这是因为它可以从 X 获得适当的渲染。

11.2.3 XML 数据提取

使用下面的方法，你能够从其它格式提取 HTML 或 XML 数据。

软件包	流行度	大小	关键词	说明
man2html	V:0.1, I:1.2	142	manpage→html	从 man 手册页到 HTML 的转换器 (支持 CGI)
doclifter	V:0.00, I:0.05	487	troff→xml	troff 到 DocBook XML 的转换器
texi2html	V:0.2, I:2.6	1847	texi→html	从 Texinfo 到 HTML 的转换器
info2www	V:0.8, I:1.4	76	info→html	从 GNU info 到 HTML 的转换器 (支持 CGI)
wv	V:0.2, I:2.2	733	MSWord→any	从微软 Word 格式到 HTML, LaTeX, 等格式的文件转换器。
unrtf	V:0.3, I:2.7	158	rtf→html	从 RTF 到 HTML 等的转换器
wp2x	V:0.00, I:0.07	200	WordPerfect→any	WordPerfect 5.0 和 5.1 文件到 TeX, LaTeX, troff, GML 和 HTML

Table 11.10: XML 数据提取工具列表

11.2.4 XML 数据检查

对于非 XML 的 HTML 文件，你能够转换它们为 XHTML，XHTML 是一个相当成型的 XML 实例。XHTML 能够被 XML 工具处理.

XML 文件的语法和在它们中发现的 URL 的完整性，能够被检查。

软件包	流行度	大小	功能	说明
libxml2-utils	V:64, I:204	214	xml↔html↔xhtml	使用 xmllint(1) 的 XML 命令行工具 (语法检查，重新格式化，梳理, ⋯)
tidy	V:0.8, I:6.7	79	xml↔html↔xhtml	HTML 语法检查和重新格式化
weblint-perl	V:0.05, I:0.85	32	检查	用于 HTML 的小巧的语法检查器
linklint	V:0.05, I:0.45	343	链接检查	快速的网站维护工具及链接检查器

Table 11.11: XML 美化打印工具列表

一旦适当的 XML 生成，基于标记的内容等，你能够使用 XSLT 技术提取数据。

11.3 排版

Unix 上的 troff 程序最初是由 AT&T 公司开发的，可以被用做简单排版。现在被用来创建手册页。

Donald Knuth 发明的 Tex 是非常强大的排版工具也是实际上的标准。最初是由 Leslie Lamport 开发的 LaTex 使得用户可以更为方便的利用 Tex 的强大功能。

11.3.1 roff 排版

传统意义上，roff 是 Unix 上主要的文本处理系统。参见 roff(7), groff(7), groff(1), grotty(1), troff(1), groff_mdoc(7), groff_man(7), groff_ms(7), groff_me(7), groff_mm(7) 和"`info groff`"。

安装好 `groff` 软件包以后，你输入"`-me`"宏指令就能看到一份不错的指导手册，它的位置是"`/usr/share/doc/groff/`"。

软件包	流行度	大小	关键词	说明
texlive	V:1, I:26	55	(La)TeX	用于排版、预览和打印的 TeX 系统
lilypond	V:0.7, I:5.1	18010	(La)TeX	音乐排版
groff	V:2, I:22	16868	troff	GNU troff 文本格式化系统

Table 11.12: 排版工具的列表

提示
"groff -Tascii -me -" 输出带有 ANSI 转义码的纯文本。如果你想要 manpage 的输出带有许多"^H" 和"_"，那么使用替代命令"GROFF_NO_SGR=1 groff -Tascii -me -"。

提示
如果想要移除 groff 生成的文本文件中的"^H" 和"_"，使用"col -b -x" 来过滤它。

11.3.2 TeX/LaTeX

Tex Live 软件提供了全部的 TeX 系统。texlive 元软件包只是 TeX Live 中的一部分，但是它足够应付日常任务。

这里有许多可用的 TeX 和 LaTeX 的参考资料。

- The teTeX HOWTO: The Linux-teTeX Local Guide
- tex(1)
- latex(1)
- texdoc(1)
- texdoctk(1)
- "The TeXbook", 作者 Donald E. Knuth, (Addison-Wesley)
- "LaTeX - A Document Preparation System", 作者 Leslie Lamport, (Addison-Wesley)
- "The LaTeX Companion", 作者 Goossens, Mittelbach, Samarin, (Addison-Wesley)

这是最强大的排版环境。许多 SGML 处理器把它作为其后台字处理工具。lyx 软件包提供的 Lyx 和 texmacs 软件包提供的 GNU TeXmacs 都为 LaTeX 提供了非常不错的所见即所得的编辑环境，然而许多人使用 Emacs 和 Vim 作为其源代码编辑器。

有许多在线资源存在。

- The TEX Live Guide ("/usr/share/doc/texlive-doc/texlive/texlive-en/texlive-en.html")(texlive-ba 包)
- Latex/Lyx 的一个简单指引
- 使用 LaTeX 进行文字处理

当文档变得更大时，TeX 有时会出错。你必须在"/etc/texmf/texmf.cnf" 中增加 pool 的大小 (更确切的说话是编辑是"/etc/texmf/texmf.d/95NonPath" 并且运行 update-texmf(8)) 来修复此问题。

注意
"The TeXbook" 的 TeX 源码可以从 texbook.tex 的 www.ctan.org tex-archive 站点上下载。此文件包含了绝大多数所需的宏指令。我听说把文档中的第 7 到第 10 行注释了并且添加"\input manmac \proofmodefalse"，就可以用 tex(1) 来处理此文档。我强烈建议去购买这本书 (还有 Donald E. Knuth 写的其他书) 而不是使用在线版本，但是在线版本中的源码确实是学习 Tex 输入很好的例子！

11.3.3 漂亮的打印手册页

你能够用如下任意一个命令在打印机上漂亮的打印手册页。

```
$ man -Tps some_manpage | lpr
```

11.3.4 创建手册页

尽管用纯 troff 格式写手册页（manpage）是可能的，这里还是有一些辅助的程序包用于创建手册页。

软件包	流行度	大小	关键词	说明
docbook-to-man	V:0.6, I:5.2	189	SGML → manpage	从 DocBook SGML 到 roff 手册页宏指令的转换器
help2man	V:0.5, I:6.0	542	text → manpage	通过 --help 参数自动生成手册页的工具
info2man	V:0.01, I:0.18	134	info → manpage	转换 GNU info 到 POD 或手册页的转换器
txt2man	V:0.05, I:0.59	112	text → manpage	把纯粹的 ASCII 文本转化为手册页格式

Table 11.13: 创建手册页的工具列表

11.4 可打印的数据

在 Debian 系统中，可打印的数据是 PostScript 格式的。对于非 PostScript 打印机，通用 Unix 打印系统 (CUPS) 使用 Ghostscript 作为其后台光栅（rasterizer）处理程序。

在最近的 Debian 系统中，可打印的数据，也可以用 PDF 格式表示。

PDF 文件能够使用 GUI（图形用户界面）的查看工具显示，它的排版条目也可以被填充到 GUI（图形用户界面）查看工具，比如 Evince 和 Okular (参见第 7.4 节); 以及现代浏览器，比如 Chromium。

PDF 文件能够使用某些图像工具编辑，比如 LibreOffice、Scribus 和 Inkscape (参见第 11.6 节)。

提示
你能够用 GIMP 阅读 PDF 文件，并能够用高于 300 dpi 像素的解析度，将 PDF 转换为 PNG 格式。这个可以作为 LibreOffice 背景图片，用最简单的操作生成一个期望的替代打印输出。

11.4.1 Ghostscript

处理可打印的数据的核心是 Ghostscript PostScript 解释器，它能够生成光栅图像（raster image）。

提示
”gs -h” 能够显示 Ghostscript 的配置信息。

软件包	流行度	大小	说明
ghostscript	V:180, I:554	177	GPL Ghostscript PostScript/PDF 解释器
ghostscript-x	V:0, I:12	88	GPL Ghostscript PostScript/PDF 解释器-X 显示支持
libpoppler162	V:16, I:24	5321	PDF 渲染库 (xpdf PDF 浏览器的分支)
libpoppler-glib8tv4	V:80, I:335	612	PDF 渲染库（基于 Glib 的共享库)
poppler-data	V:170, I:578	13086	用于 PDF 渲染库的 CMaps (CJK 支持：Adobe-*)

Table 11.14: Ghostscript PostScript 解释器列表

11.4.2 合并两个 PS 或 PDF 文件

你能够使用 Ghostscript 中的 gs(1) 来合并两个 PostScript(PS) 或可移植文档格式（PDF）文件。

```
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pswrite -sOutputFile=bla.ps -f foo1.ps foo2.ps
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=bla.pdf -f foo1.pdf foo2.pdf
```

注意
PDF 是用途很广的跨平台可打印的数据格式，它本质上是带有一些额外特性和扩展的压缩了的 PS 格式。

提示
对于命令行来说，psmerge(1) 和 psutils 包中的其他命令在处理 PostScript 文档时是很有用的。pdftk-java 包中的 pdftk(1) 在处理 PDF 文档的时候同样是很好用的。

11.4.3 处理可印刷数据的工具

如下是处理可印刷数据的工具列表。

11.4.4 用 CUPS 打印

Unix 通用打印系统（CUPS）中的 lp(1) 和 lpr(1) 命令都提供了自定义打印数据的选项。

你可以使用下列命令中的一个来打印 3 份有装订页码的文件。

```
$ lp -n 3 -o Collate=True filename
```

```
$ lpr -#3 -o Collate=True filename
```

你能够通过"-o number-up=2","-o page-set=even","-o page-set=odd","-o scaling=200","-o natural-scal
等等打印机选项来进一步定制打印机操作，详细的文档参见命令行打印和选项。

11.5 邮件数据转换

下列邮件数据转换软件包捕获了我的眼球。

提示
如果邮件客户端可以配置使用 IMAP4 服务器，互联网消息访问协议版本 4 (IMAP4) 服务器可以用来把邮件从专有邮件系统里面移出来。

软件包	流行度	大小	关键词	说明
poppler-utils	V:175, I:475	804	pdf→ps, text…	PDF 工具：pdftops, pdfinfo, pdfimages, pdftotext, pdffonts
psutils	V:4, I:47	34	ps→ps	PostScript 文件转换工具
poster	V:0.1, I:1.5	57	ps→ps	用 PostScript 页制作大型海报
enscript	V:1, I:12	2138	text→ps, html, rtf	转化 ASCII 文本到 PostScript, HTML, RTF 或 Pretty-Print
a2ps	V:0.6, I:6.5	4109	text→ps	’ 任何文本到 PostScript’ 的转换器并且也是相当不错的打印程序
pdftk	V:1, I:18	28	pdf→pdf	PDF 文档转换工具：pdftk
html2ps	V:0.1, I:1.7	256	html→ps	从 HTML 到 PostScript 的转换器
gnuhtml2latex	V:0.04, I:0.55	26	html→latex	从 html 到 latex 的转换器
latex2rtf	V:0.1, I:1.9	495	latex→rtf	转换 LaTeX 文档到能被 Microsoft Word 读取的 RTF 格式的文档
ps2eps	V:2, I:30	95	ps→eps	从 PostScript 到 EPS（Encapsulated PostScript）的转换器
e2ps	V:0.01, I:0.10	104	text→ps	带有日文编码支持的文本到 PostScript 转换器
impose+	V:0.1, I:1.5	118	ps→ps	PostScript 工具
trueprint	I:0.07	148	text→ps	漂亮的打印许多源程序（C, C++, Java, Pascal, Perl, Pike, Sh, 和 Verilog）到 PostScript。(C 语言)
pdf2svg	V:0.2, I:2.8	33	pdf→svg	PDF 到 svg 可缩放矢量图形（Scalable Vector Graphics） 格式的转换器
pdftoipe	V:0.02, I:0.44	73	pdf→ipe	从 PDF 到 IPE‘s XML 格式的转换器

Table 11.15: 处理可印刷数据的工具列表

软件包	流行度	大小	关键词	说明
sharutils	V:2, I:26	1436	邮件	shar(1), unshar(1), uuencode(1), uudecode(1)
mpack	V:0.7, I:7.1	109	MIME	编码和解码 MIME 信息: mpack(1) 和 munpack(1)
tnef	V:0.3, I:3.7	103	ms-tnef	解包 MIME 附件类型”application/ms-tnef”，该格式仅由微软使用
uudeview	V:0.2, I:1.6	105	mail	下列格式的编码器和解码器: uuencode, xxencode, BASE64, quoted printable 和 BinHex

Table 11.16: 有助于邮件数据转换的软件包列表

11.5.1 邮件数据基础

邮件 (SMTP) 数据需要被限制为 7 位数据序列。二进制数据和 8 位文本数据使用 Multipurpose Internet Mail Extensions (MIME) 互联网多用途邮件扩展和选择的字符集编码到 7 位格式。(参见表 11.2)。

标准的邮件存储格式是 mbox，它是依据 RFC2822 (由 RFC822 更新) 来的格式. 参见 mbox(5) (由 `mutt` 软件包提供).

对于欧洲语言, 由于没有什么 8 位字符，"`Content-Transfer-Encoding: quoted-printable`" 加 ISO-8859-1 字符集通常被用于邮件。如果欧洲文本是被编码为 UTF-8, 由于几乎全是 7 位数据，使用"`Content-Transfer-Encoding: quoted-printable`" 也是合适的。

对于日语，传统的"`Content-Type: text/plain; charset=ISO-2022-JP`" 通常被用于邮件来保持文本在 7 位。但是老的微软系统会在没有声明的情况下使用 Shift-JIS 来发送邮件。如果日语文本是用 UTF-8 编码, 由于含有许多 8 位数据，使用 Base64 是合适的。其它亚洲语言也是类似情形。

注意
如果你的非 Unix 邮件数据可以通过一个具备和 IMAP4 服务通讯的非 Debian 客户端访问，你可以通过运行你的 IMAP4 服务来将邮件数据移出。

注意
如果你使用其它邮件存储格式，第一步把它们移动到 mbox 格式比较好。像 mutt(1) 这样多功能的客户端程序可以便捷的完成这类操作。

你可以使用 procmail(1) 和 formail(1) 把邮箱内容分开成每一封邮件.

每一封邮件能够使用来自 `mpack` 软件包的 munpack(1) 命令（或其它特异的工具）来获得 MIME 编码内容。

11.6 图形数据工具

虽然像 gimp(1) 这样的图形界面程序是非常强大的，但像 imagemagick(1) 这样的命令行工具在用脚本自动化处理图像时是很有用的。

实际上的数码相机的图像是可交换的图像文件格式 (EXIF)，这种格式是在 JPEG 图像文件格式上添加一些元数据标签。它能够保存诸如日期、时间和相机设置的信息。

Lempel-Ziv-Welch (LZW) 无损数据压缩专利已经过期了。使用 LZW 压缩方式的图形交互格式（GIF）工具现在可以在 Debian 系统上自由使用了。

提示
任何带有可移动记录介质的数码相机或扫描仪都可以在 Linux 上通过 USB 存储读取器来工作，因为它遵循相机文件系统设计规则并且使用 FAT 文件系统，参考第 10.1.7 节。

11.6.1 图形数据工具（元软件包）

下列元软件包是用 aptitude(8) 搜索图像数据工具的一个好的起点。"Debian 图像工具维护团队的软件包概览"能够作为另外一个起点。

提示
在 aptitude(8) （参考第 2.2.6 节）中用正则表达式"`~Gworks-with::image`" 来查找更多的图像工具。

软件包	流行度	大小	关键词	说明
education-graphics	I:0.30	26	svg,jpeg,…	用于图像教学和图片艺术的元软件包。
open-font-design-toolkit	I:0.04	9	ttf,ps,…	用于开放字体设计的元软件包

Table 11.17: 图形数据工具列表（元软件包）

软件包	流行度	大小	关键词	说明
gimp	V:40, I:206	33386	image(bitmap)	GNU 图形处理程序
xsane	V:9, I:127	1512	image(bitmap)	用于 SANE 的基于 GTK 的前端图形界面 (现在访问扫描仪就很简单了)
scribus	V:1, I:12	32942	ps/pdf/SVG/…	Scribus DTP 编辑器
libreoffice-draw	V:78, I:412	11424	image(vector)	LibreOffice 办公套件-绘画
inkscape	V:12, I:73	113357	image(vector)	SVG（可升级矢量图形) 编辑器
dia	V:1, I:16	3846	image(vector)	图表编辑器（Gtk）
xfig	V:0.5, I:8.4	7951	image(vector)	在图形界面下，交互式的生成图像变得方便
gocr	V:0.4, I:3.4	549	image→text	免费 OCR 软件
eog	V:20, I:121	10542	image(Exif)	Eye of GNOME 图像浏览程序
gthumb	V:3, I:11	5131	image(Exif)	图像浏览器（GNOME）
geeqie	V:3, I:10	3854	image(Exif)	基于 GTK 的图像浏览器
shotwell	V:15, I:244	6334	image(Exif)	数码相片管理器（GNOME）
gwenview	V:40, I:113	6051	image(Exif)	图片浏览器（KDE）
kamera	I:112	992	image(Exif)	KDE 上的支持数码相机的应用软件
digikam	V:1.5, I:8.2	333	image(Exif)	用于 KDE 桌面环境的数字照片管理应用
darktable	V:3, I:10	36092	image(Exif)	摄影师的虚拟灯台和暗房
hugin	V:0.5, I:5.2	6885	image(Exif)	全景图片合成器
librecad	V:1, I:13	12361	DXF,…	2D CAD 数据编辑器
freecad	V:1, I:20	112	DXF,…	3D CAD 数据编辑器
blender	V:3, I:20	110904	blend,TIFF,JPEG,…	用于动画的 3D 编辑器
mm3d	V:0.03, I:0.23	4123	ms3d,obj,dxf,…	基于 OpenGL 的 3D 模型编辑器
fontforge	V:0.5, I:5.2	4736	ttf,ps,…	用于 PS，TrueType 和 OpenType 的字体编辑器
xgridfit	V:0.02, I:0.07	878	ttf	用于 TrueType 字体的网格拟合和小字还原技术的程序

Table 11.18: 图形数据工具（GUI 图形用户界面）列表

11.6.2 图形数据工具（GUI 图形用户界面）

下列 GUI（图形用户界面）的图形数据转换、编辑和管理工具的软件包捕获了我的眼球。

11.6.3 图形数据工具 (CLI 命令行)

下列 CLI（命令行）的图形数据转换、编辑和管理工具的软件包捕获了我的眼球。

软件包	流行度	大小	关键词	说明
imagemagick	V:7, I:266	82	image(bitmap)	图形处理程序
graphicsmagick	V:1.0, I:7.8	5988	image(bitmap)	图像处理程序（imagemagick 派生出来的）
netpbm	V:27, I:275	8470	image(bitmap)	图形界面的转换工具
libheif-examples	V:0.4, I:3.7	534	heif→jpeg(bitmap)	转化高性能图像文件格式 (HEIF) 到 JPEG、PNG 或 Y4M 格式，用 heif-convert(1) 命令
icoutils	V:3, I:31	221	png↔ico(bitmap)	MS Windows 符号和光标转化为 PNG 格式，或者从 PNG 格式转化为位图格式 (favicon.ico)
pstoedit	V:2, I:36	1075	ps/pdf→image(vector)	PostScript 和 PDF 文件到可编辑的矢量图形的转换器（SVG）
libwmf-bin	V:5, I:77	149	Windows/image(矢量)	Windows 元文件 (矢量图形数据) 转换工具
fig2sxd	V:0.02, I:0.14	158	fig→sxd(vector)	转换 XFig 文件为 OpenOffice.org 绘画格式
unpaper	V:2, I:16	417	image→image	后处理 OCR 扫描页面的工具
tesseract-ocr	V:9, I:35	2411	image→text	基于惠普的商业 OCR 引擎的免费 OCR 软件
tesseract-ocr-eng	V:9, I:35	4032	image→text	OCR 引擎数据：用于英文文本的 tesseract-ocr 语言文件
ocrad	V:0.3, I:2.2	608	image→text	免费 OCR 软件
exif	V:2, I:50	334	image(Exif)	显示 JPEG 文件中的 EXIF 信息的命令行工具
exiv2	V:1, I:18	436	image(Exif)	EXIF/IPTC 元数据处理工具
exiftran	V:1, I:11	81	image(Exif)	改变数码照相机的 jpeg 图像格式
exiftags	V:0.2, I:2.5	309	image(Exif)	从数码相机的 JPEG 文件读取 Exif 标签的实用工具
exifprobe	V:0.2, I:2.0	506	image(Exif)	从数码图片中读取元数据
dcraw	V:0.7, I:7.0	428	image(Raw)	解码原始的数码相机图片
findimagedupes	V:0.10, I:0.94	76	image→fingerprint	找到相似或重复的图像
ale	V:0.01, I:0.14	850	image→image	合并图像来增加保真度或者用于创建马赛克
imageindex	V:0.1, I:1.1	143	image(Exif)	从图形中创建静态 HTML 图库
outguess	V:0.1, I:1.0	230	jpeg,png	通用的 Steganographic 工具
jpegoptim	V:0.5, I:5.6	59	jpeg	优化 JPEG 文件
optipng	V:2, I:38	187	png	优化 PNG 文件，无损压缩
pngquant	V:1, I:11	62	png	优化 PNG 文件，有损压缩

Table 11.19: 图形数据工具 (CLI 命令行) 列表

11.7 不同种类的数据转换工具

这里有许多其他用于数据转换的工具。在 aptitude(8)（参考第 2.2.6 节）里用正则表达式"~Guse::converting""来查找如下的软件包。

软件包	流行度	大小	关键词	说明
alien	V:1, I:12	150	rpm/tgz→deb	把外来的软件包转换为 Debian 软件包
freepwing	I:0.01	447	EB→EPWING	把" 电子书" (在日本流行) 变成单一的 JIS X 4081 格式 (EPWING V1 的子集)
calibre	V:7, I:24	76289	any→EPUB	电子书转换器和库管理

Table 11.20: 不同种类的数据转换工具列表

你能够通过如下的命令从 RPM 格式的包中提取数据。

```
$ rpm2cpio file.src.rpm | cpio --extract
```

Chapter 12

编程

这里我给出一些 Debian 系统中的信息，帮助学习编程的人找出打包的源代码。下面是值得关注的软件包和与之对应的文档。

安装 `manpages` 和 `manpages-dev` 包之后，可以通过运行“`man` 名称”查看手册页中的参考信息。安装了 GNU 工具的相关文档包之后，可以通过运行“`info` 程序名称”查看参考文档。某些 GFDL 协议的文档与 DFSG 并不兼容，所以你可能需要在 `main` 仓库中包含 `contrib` 和 `non-free` 才能下载并安装它们。

请考虑使用版本控制系统工具。参见第 10.5 节。



警告
不要用“`test`”作为可执行的测试文件的名字，因为 shell 中内建有“`test`”命令。



小心
你可以把从源代码编译得到的程序直接放到“`/usr/local`”或“`/opt`”目录，这样可以避免与系统程序撞车。

12.1 Shell 脚本

Shell 脚本是指包含有下面格式的可执行的文本文件。

```
#!/bin/sh
 ... command lines
```

第一行指明了读取并执行这个文件的 shell 解释器。

读懂 shell 脚本的最好办法是先理解类 Unix 系统是如何工作的。这里有一些 shell 编程的提示。看看“Shell 错误”（https://www.greenend.org.uk/rjk/2001/04/shell.html），可以从错误中学习。

不像 shell 交互模式（参见第 1.5 节和第 1.6 节），shell 脚本会频繁使用参数、条件和循环等。

12.1.1 POSIX shell 兼容性

系统中的许多脚本都可以通过任意 POSIX shell（参见表 1.13）来执行。

- 默认的非交互 POSIX shell "`/usr/bin/sh`" 是一个指向到 `/usr/bin/dash` 的符号链接，并被许多系统程序使用。
- 默认的交互式 POSIX shell 是 `/usr/bin/bash`。

避免编写具有 **bashisms**（bash 化）或者 **zshisms**（zsh 化）语法的 shell 脚本，确保脚本在所有 POSIX shell 之间具有可移植性。你可以使用 checkbashisms(1) 对其进行检查。

好的：**POSIX**	应该避免的：**bashism**
`if [ "$foo" = "$bar" ] ; then …`	`if [ "$foo" == "$bar" ] ; then …`
`diff -u file.c.orig file.c`	`diff -u file.c{.orig,}`
`mkdir /foobar /foobaz`	`mkdir /foo{bar,baz}`
`funcname() { …}`	`function funcname() { …}`
八进制格式："`\377`"	十六进制格式："`\xff`"

Table 12.1: 典型 bashism 语法列表

使用"`echo`" 命令的时候需要注意以下几个方面，因为根据内置 shell 和外部命令的不同，它的实现也有差别。

- 避免使用除"`-n`"以外的任何命令行选项。
- 避免在字符串中使用转义序列，因为根据 shell 不同，计算后的结果也不一样。

注意
尽管"`-n`"选项并不是 POSIX 语法，但它已被广泛接受。

提示
如果你想要在输出字符串中嵌入转义序列，用"`printf`" 命令替代"`echo`" 命令。

12.1.2 Shell 参数

特殊的 shell 参数经常在 shell 脚本里面被用到。

shell 参数	值
`$0`	shell 或 shell 脚本的名称
`$1`	第一个 shell 参数
`$9`	第 9 个 shell 参数
`$#`	位置参数数量
`"$*"`	`"$1 $2 $3 $4 …"`
`"$@"`	`"$1" "$2" "$3" "$4" …`
`$?`	最近一次命令的退出状态码
`$$`	这个 shell 脚本的 PID
`$!`	最近开始的后台任务 PID

Table 12.2: shell 参数列表

如下所示是需要记忆的基本的参数展开。

以上这些操作中"`:`" 实际上都是可选的。

- 有"`:`" 等于测试的 var 值是存在且非空
- 没有"`:`" 等于测试的 var 值只是存在的，可以为空

参数表达式形式	如果 **var** 变量已设置那么值为	如果 **var** 变量没有被设置那么值为
`${var:-string}`	"`$var`"	"`string`"
`${var:+string}`	"`string`"	"`null`"
`${var:=string}`	"`$var`"	"`string`" (并运行"`var=string`")
`${var:?string}`	"`$var`"	在 **stderr** 中显示"`string`" (出错退出)

Table 12.3: shell 参数展开列表

参数替换形式	结果
`${var%suffix}`	删除位于 var 结尾的 suffix 最小匹配模式
`${var%%suffix}`	删除位于 var 结尾的 suffix 最大匹配模式
`${var#prefix}`	删除位于 var 开头的 prefix 最小匹配模式
`${var##prefix}`	删除位于 var 开头的 prefix 最大匹配模式

Table 12.4: 重要的 shell 参数替换列表

12.1.3 Shell 条件语句

每个命令都会返回退出状态，这可以被条件语句使用。

- 成功：0 ("True")
- 失败：非 0 ("False")

注意
"0" 在 shell 条件语句中的意思是"True"，然而"0" 在 C 条件语句中的含义为"False"。

注意
"`[`" 跟 `test` 命令是等价的，它评估到"`]`" 之间的参数来作为一个条件表达式.

如下所示是需要记忆的基础条件语法。

- "*`command`* `&&` *`if_success_run_this_command_too`* `||` `true`"
- "*`command`* `||` *`if_not_success_run_this_command_too`* `||` `true`"
- 如下所示是多行脚本片段

```
if [ conditional_expression ]; then
 if_success_run_this_command
else
 if_not_success_run_this_command
fi
```

这里末尾的"`|| true`"是需要的，它可以保证这个 shell 脚本在不小心使用了"`-e`"选项而被调用时不会在该行意外地退出。

算术整数的比较在条件表达式中为"`-eq`"，"`-ne`"，"`-lt`"，"`-le`"，"`-gt`" 和"`-ge`"。

表达式	返回逻辑真所需的条件
`-e` *file*	*file* 存在
`-d` *file*	*file* 存在并且是一个目录
`-f` *file*	*file* 存在并且是一个普通文件
`-w` *file*	*file* 存在并且可写
`-x` *file*	*file* 存在并且可执行
file1 `-nt` *file2*	*file1* 是否比 *file2* 新
file1 `-ot` *file2*	*file1* 是否比 *file2* 旧
file1 `-ef` *file2*	*file1* 和 *file2* 位于相同的设备上并且有相同的 inode 编号

Table 12.5: 在条件表达式中进行文件比较

表达式	返回逻辑真所需的条件
`-z` *str*	*str* 的长度为零
`-n` *str*	*str* 的长度不为零
str1 `=` *str2*	*str1* 和 *str2* 相等
str1 `!=` *str2*	*str1* 和 *str2* 不相等
str1 `<` *str2*	*str1* 排列在 *str2* 之前（取决于语言环境）
str1 `>` *str2*	*str1* 排列在 *str2* 之后（取决于语言环境）

Table 12.6: 在条件表达式中进行字符串比较

12.1.4 shell 循环

这里有几种可用于 POSIX shell 的循环形式。

- "`for x in foo1 foo2 …; do command ; done`"，该循环会将"`foo1 foo2 …`"赋予变量"x"并执行"`command`"。
- "`while condition ; do command ; done`"，当"`condition`"为真时，会重复执行"`command`"。
- "`until condition ; do command ; done`"，当"`condition`"为假时，会重复执行"`command`"。
- "`break`"可以用来退出循环。
- "`continue`"可以用来重新开始下一次循环。

提示
C 语言中的数值迭代可以用 seq(1) 实现来生成"`foo1 foo2 ...`"。

提示
参见第 9.4.9 节。

12.1.5 Shell 环境变量

普通的 shell 命令行提示下的一些常见的环境变量，可能在你的脚本的执行环境中不存在。

- 对于"`$USER`", 使用"`$(id -un)`"
- 对于"`$UID`", 使用"`$(id -u)`"
- 对于"`$HOME`"，使用"`$(getent passwd "$(id -u)"|cut -d ":" -f 6)`" (这个也在第 4.5.2 节下工作)

12.1.6 shell 命令行的处理顺序

shell 大致以下列的顺序来处理一个脚本。

- shell 读取一行。
- 如果该行包含有"…" 或'…'，shell 对该行各部分进行分组作为一个标识（**one token**）（译注：one token 是指 shell 识别的一个结构单元）.
- shell 通过下列方式将行中的其它部分分隔进标识（**tokens**）。
  - 空白字符：空格 *tab* 换行符
  - 元字符：`< > | ; & ( )`
- shell 会检查每一个不位于"…" 或'...' 的 token 中的保留字来调整它的行为。
  - 保留字：`if then elif else fi for in while unless do done case esac`
- shell 展开不位于"…" 或'...' 中的别名。
- shell 展开不位于"…" 或'...' 中的波浪线。
  - ”~” → 当前用户的家目录
  - ”~*user*” → *user* 的家目录
- shell 将不位于'...' 中的变量展开为它的值。
  - 变量：”$PARAMETER” 或”${PARAMETER}”
- shell 展开不位于'...' 中的命令替换。
  - ”$(command)” → ”command” 的输出
  - ”` command `” → ”command” 的输出
- shell 将不位于"…" 或'...' 中的 **glob** 路径展开为匹配的文件名。
  - * → 任何字符
  - ? → 一个字符
  - […] → 任何位于”…” 中的字符
- shell 从下列几方面查找命令并执行。
  - 函数定义
  - 内建命令
  - “$PATH”中的可执行文件
- shell 前往下一行，并按照这个顺序从头再次进行处理。

双引号中的单引号是没有效果的。

在 shell 中执行“`set -x`”或使用“`-x`”选项启动 shell 可以让 shell 显示出所有执行的命令。这对调试来说是非常方便的。

软件包	流行度	大小	说明
dash	V:933, I:999	211	小和快的 POSIX 兼容 shell，用于 sh
coreutils	V:914, I:1000	19751	GNU 核心工具
grep	V:800, I:1000	1297	GNU grep、egrep 和 fgrep
sed	V:789, I:1000	987	GNU sed
mawk	V:517, I:998	299	小和快的 awk
debianutils	V:940, I:997	228	用于 Debian 的各种工具
bsdutils	V:445, I:1000	338	来自 4.4BSD-Lite 的基础工具
bsdextrautils	V:771, I:877	382	来自 4.4BSD-Lite 的额外的工具
moreutils	V:18, I:38	232	额外的 Unix 工具

Table 12.7: 包含用于 shell 脚本的小型应用程序的软件包

12.1.7 用于 shell 脚本的应用程序

为了使你的 shell 程序在 Debian 系统上尽可能地具有可移植性，你应该只使用必要的软件包所提供的应用程序。

- "`aptitude search ~E`"，列出必要的软件包。
- "`dpkg -L` *`package_name`* `|grep '/man/man.*/'`"，列出 *`package_name`* 软件包所提供的 man 手册。

> 提示
> 尽管 `moreutils` 可能不存在 Debian 之外，但它提供了一些有趣的小程序。最值得注意的一个是 sponge(8)，当你想覆盖原来的文件时，它会非常好用。

参见第 1.6 节的例子。

12.2 解释性语言中的脚本

当你希望在 Debian 上自动化执行一个任务，你应当首先使用解释性语言脚本。选择解释性语言的准则是：

- 使用 `dash`，如果任务是简单的，使用 shell 程序联合 CLI 命令行程序。
- 使用 `python3`，如果任务不是简单的，你从零开始写。
- 使用 `perl`、`tcl`、`ruby`……，如果在 Debian 上有用这些语言写的现存代码，需要为完成任务进行调整。

如果最终代码太慢，为提升执行速度，你可以用编译型语言重写关键部分，从解释性语言调用。

12.2.1 调试解释性语言代码

大部分解释器提供基本的语法检查和代码跟踪功能。

- "**dash -n** *script.sh*"- Shell 脚本语法检查
- "**dash -x** *script.sh*"- 跟踪一个 Shell 脚本

软件包	流行度	大小	文档
dash	V:933, I:999	211	sh: 小和快的 POSIX 兼容的 shell，用于 sh
bash	V:900, I:1000	7309	sh: 由 bash-doc 包提供的“info bash”
mawk	V:517, I:998	299	AWK: 小和快的 awk
gawk	V:246, I:288	3289	AWK: 由 gawk-doc 包提供的“info gawk”
perl	V:672, I:991	840	Perl: perl(1) 以及通过 perl-doc 和 perl-doc-html 提供的 html 文档
libterm-readline-gnu-perl	V:2, I:25	435	GNU ReadLine/History 库的 Perl 扩展：perlsh(1)
libreply-perl	I:0.08	171	Perl 的 REPL : reply(1)
libdevel-repl-perl	V:0.02, I:0.47	237	Perl 的 REPL : re.pl(1)
python3	V:748, I:977	81	Python: python3(1) 以及通过 python3-doc 包提供的 html 文档
tcl	V:24, I:171	20	Tcl: tcl(3) 以及通过 tcl-doc 包提供的更详细的手册页文档
tk	V:18, I:165	20	Tk:tk(3) 以及通过 tk-doc 包提供的更详细的手册页文档
ruby	V:61, I:150	32	Ruby: ruby(1), erb(1), irb(1), rdoc(1), ri(1)

Table 12.8: 解释器相关软件包列表

- “**python3 -m py_compile** *script.py*”- Python 脚本语法检查
- “**python3 -mtrace --trace** *script.py*”- 跟踪一个 Python 脚本
- “**perl -I ../libpath -c** *script.pl*”- Perl 脚本语法检查
- “**perl -d:Trace** *script.pl*”- 跟踪一个 Perl 脚本

为测试 dash 代码，尝试下第 9.1.4 节，它提供了和 bash 类似的交互式环境。

为了测试 perl 代码，尝试下 Perl 的 REPL 环境，它为 Perl 提供了 Python 类似的 REPL (=READ + EVAL + PRINT + LOOP) 环境。

12.2.2 使用 shell 脚本的 GUI 程序

shell 脚本能够被改进用来制作一个吸引人的 GUI（图形用户界面）程序。技巧是用一个所谓的对话程序来代替使用 echo 和 read 命令的乏味交互。

软件包	流行度	大小	说明
x11-utils	V:233, I:553	651	xmessage(1)：在一个窗口中显示一条消息或疑问（X）
whiptail	V:344, I:997	61	从 shell 脚本中显示用户友好的对话框（newt）
dialog	V:8, I:75	535	从 shell 脚本中显示用户友好的对话框（ncurses）
zenity	V:69, I:337	193	从 shell 脚本中显示图形对话框（GTK）
ssft	V:0.01, I:0.14	80	Shell 脚本前端工具 (zenity, kdialog, and 带有 gettext 的 dialog 封装)
gettext	V:49, I:206	20549	“/usr/bin/gettext.sh”：翻译信息

Table 12.9: 对话（dialog ）程序列表

这里是一个用来演示的 GUI 程序的例子，仅使用一个 shell 脚本是多么容易。

这个脚本使用 zenity 来选择一个文件 (默认 /etc/motd) 并显示它。

这个脚本的 GUI 启动器能够按第 9.4.10 节创建。

```
#!/bin/sh -e
# Copyright (C) 2021 Osamu Aoki <osamu@debian.org>, Public Domain
# vim:set sw=2 sts=2 et:
DATA_FILE=$(zenity --file-selection --filename="/etc/motd" --title="Select a file to check ↩
    ") || \
  ( echo "E: File selection error" >&2 ; exit 1 )
# Check size of archive
if ( file -ib "$DATA_FILE" | grep -qe '^text/' ) ; then
  zenity --info --title="Check file: $DATA_FILE" --width 640  --height 400 \
    --text="$(head -n 20 "$DATA_FILE")"
else
  zenity --info --title="Check file: $DATA_FILE" --width 640  --height 400 \
    --text="The data is MIME=$(file -ib "$DATA_FILE")"
fi
```

这种使用 shell 脚本的 GUI 程序方案只对简单选择的场景有用。如果你写一个其它任何复杂的程序，请考虑在功能更强的平台上写。

12.2.3 定制 GUI（图形用户界面）文件管理器的行为

GUI（图形用户界面）文件管理器在选定的文件上，能够用外加的扩展软件包来扩展执行一些常见行为。通过增加特定的脚本，它们也能够用来定制执行非常特殊的行为。

- 对于 GNOME，参见 NautilusScriptsHowto。
- 对于 KDE，参见 Creating Dolphin Service Menus。
- 对于 Xfce，参见 Thunar - Custom Actions 和 https://help.ubuntu.com/community/ThunarCustomActions。
- 对于 LXDE，参见 Custom Actions。

12.2.4 Perl 短脚本的疯狂

为了处理数据，sh 需要生成子进程运行 cut、grep、sed 等，是慢的。从另外一个方面，perl 有内部处理数据能力，是快的。所以 Debian 上的许多系统维护脚本使用 perl。

让我们考虑下面一行 AWK 脚本片段和它在 Perl 中的等价物。

```
awk '($2=="1957") { print $3 }' |
```

这等价于下列的任意一行。

```
perl -ne '@f=split; if ($f[1] eq "1957") { print "$f[2]\n"}' |
```

```
perl -ne 'if ((@f=split)[1] eq "1957") { print "$f[2]\n"}' |
```

```
perl -ne '@f=split; print $f[2] if ( $f[1]==1957 )' |
```

```
perl -lane 'print $F[2] if $F[1] eq "1957"' |
```

```
perl -lane 'print$F[2]if$F[1]eq+1957' |
```

最后一个简直就是个迷。它用上了下面列出的这些 Perl 的特性。

- 空格为可选项。

- 存在从数字到字符串的自动转换。
- 通过命令行选项：perlrun(1) 的 Perl 执行技巧
- Perl 特异变量：perlvar(1)

灵活性是 Perl 的强项。与此同时，这允许我们创建令人困惑和繁乱的代码。所以请小心。

12.3 编译型语言代码

软件包	流行度	大小	说明
gcc	V:167, I:557	36	GNU C 编译器
libc6-dev	V:259, I:577	12863	GNU C 库：开发库和头文件
g++	V:64, I:527	13	GNU C++ 编译器
libstdc++-16-dev	V:4, I:21	24416	GNU 标准 C++ 库版本 3（开发文件）
cpp	V:346, I:717	18	GNU C 预处理
gettext	V:49, I:206	20549	GNU 国际化工具
glade	V:0.2, I:2.5	1613	GTK 用户界面构建器
valac	V:0.2, I:2.9	533	使用 GObject 系统类似 C# 的语言
flex	V:7, I:66	1247	LEX 兼容的 fast lexical analyzer generator
bison	V:7, I:70	3122	YACC 兼容的解析器生成器
susv2	I:0.02	16	通过“单一 UNIX 规范（版本 2）”获取（英语文档）
susv3	I:0.04	16	通过“单一 UNIX 规范（版本 3）”获取（英语文档）
susv4	I:0.03	16	通过“单一 UNIX 规范（版本 4）”获取（英语文档）
golang	I:21	11	Go 编程语言编译器
rustc	V:6, I:20	12745	Rust 系统编程语言
gfortran	V:4, I:47	15	GNU Fortran 95 编译器
fpc	I:2.3	101	自由 Pascal

Table 12.10: 编译相关软件包列表

这里，包括了第 12.3.3 节和第 12.3.4 节，用来说明类似编译器的程序怎样用 C 语言来编写，是通过编译高级描述到 C 语言。

12.3.1 C

你可以通过下列方法设置适当的环境来编译使用 C 编程语言编写的程序。

```
# apt-get install glibc-doc manpages-dev libc6-dev gcc build-essential
```

libc6-dev 软件包，即 GNU C 库，提供了 C 标准库，它包含了 C 编程语言所使用的头文件和库例程。

参考信息如下。

- “info libc”（C 库函数参考）
- gcc(1) 和“info gcc”
- each_C_library_function_name(3)
- Kernighan & Ritchie，“C 程序设计语言”，第二版（Prentice Hall）

12.3.2 简单的 C 程序（gcc）

一个简单的例子“example.c”可以通过如下方式和“libm”库一起编译为可执行程序“run_example”。

```
$ cat > example.c << EOF
#include <stdio.h>
#include <math.h>
#include <string.h>

int main(int argc, char **argv, char **envp){
        double x;
        char y[11];
        x=sqrt(argc+7.5);
        strncpy(y, argv[0], 10); /* prevent buffer overflow */
        y[10] = '\0'; /* fill to make sure string ends with '\0' */
        printf("%5i, %5.3f, %10s, %10s\n", argc, x, y, argv[1]);
        return 0;
}
EOF
$ gcc -Wall -g -o run_example example.c -lm
$ ./run_example
        1, 2.915, ./run_exam,     (null)
$ ./run_example 1234567890qwerty
        2, 3.082, ./run_exam, 1234567890qwerty
```

为了使用 sqrt(3)，必须使用“-lm”链接来自 libc6 软件包的库“/usr/lib/libm.so”。实际的库文件位于“/lib/”，文件名为“libm.so.6”，它是指向“libm-2.7.so”的一个链接。

请看一下输出文本的最后一段。即使指定了“%10s”，它依旧超出了 10 个字符。

使用没有边界检查的指针内存操作函数，比如 sprintf(3) 和 strcpy(3), 是不建议使用，是为防止缓存溢出泄露而导致上面的溢出问题。请使用 snprintf(3) 和 strncpy(3) 来替代.

12.3.3 Flex —一个更好的 Lex

Flex 是兼容 Lex 的快速语法分析程序生成器。

可以使用“info flex”查看 flex(1) 的教程。

很多简单的例子能够在“/usr/share/doc/flex/examples/”下发现。1

12.3.4 Bison —一个更好的 Yacc

在 Debian 里，有几个软件包提供 Yacc 兼容的前瞻性的 LR 解析或 LALR 解析的生成器。

软件包	流行度	大小	说明
bison	V:7, I:70	3122	GNU LALR 解析器生成器
byacc	V:0.1, I:2.6	263	伯克利（Berkeley）LALR 解析器生成器
btyacc	V:0.01, I:0.04	247	基于 byacc 的回溯解析生成器

Table 12.11: 兼容 Yacc 的 LALR 解析器生成器列表

可以使用“info bison”查看 bison(1) 的教程。

你需要提供你自己的的”main()”和”yyerror()”. 通常，Flex 创建的”main()”调用”yyparse()”，它又调用了”yylex()”.

1在当前系统下，为了让它们工作，需要做一些调整。

这里是一个创建简单终端计算程序的例子。

让我们创建 example.y:

```
/* calculator source for bison */
%{
#include <stdio.h>
extern int yylex(void);
extern int yyerror(char *);
%}

/* declare tokens */
%token NUMBER
%token OP_ADD OP_SUB OP_MUL OP_RGT OP_LFT OP_EQU

%%
calc:
 | calc exp OP_EQU    { printf("Y: RESULT = %d\n", $2); }
 ;

exp: factor
 | exp OP_ADD factor  { $$ = $1 + $3; }
 | exp OP_SUB factor  { $$ = $1 - $3; }
 ;

factor: term
 | factor OP_MUL term { $$ = $1 * $3; }
 ;

term: NUMBER
 | OP_LFT exp OP_RGT  { $$ = $2; }
  ;
%%

int main(int argc, char **argv)
{
  yyparse();
}

int yyerror(char *s)
{
  fprintf(stderr, "error: '%s'\n", s);
}
```

让我们创建 example.l:

```
/* calculator source for flex */
%{
#include "example.tab.h"
%}

%%
[0-9]+ { printf("L: NUMBER = %s\n", yytext); yylval = atoi(yytext); return NUMBER; }
"+"    { printf("L: OP_ADD\n"); return OP_ADD; }
"-"    { printf("L: OP_SUB\n"); return OP_SUB; }
"*"    { printf("L: OP_MUL\n"); return OP_MUL; }
"("    { printf("L: OP_LFT\n"); return OP_LFT; }
")"    { printf("L: OP_RGT\n"); return OP_RGT; }
"="    { printf("L: OP_EQU\n"); return OP_EQU; }
"exit" { printf("L: exit\n");   return YYEOF; } /* YYEOF = 0 */
.      { /* ignore all other */ }
%%
```

按下面的方法来从 shell 提示符执行来尝试这个：

```
$ bison -d example.y
$ flex example.l
$ gcc -lfl example.tab.c lex.yy.c -o example
$ ./example
1 + 2 * ( 3 + 1 ) =
L: NUMBER = 1
L: OP_ADD
L: NUMBER = 2
L: OP_MUL
L: OP_LFT
L: NUMBER = 3
L: OP_ADD
L: NUMBER = 1
L: OP_RGT
L: OP_EQU
Y: RESULT = 9

exit
L: exit
```

12.4 静态代码分析工具

类似 lint 的工具能够帮助进行自动化静态代码分析。

类似 Indent 的工具能够帮助人进行代码检查，通过一致性的重新格式化源代码。

类似 Ctags 的工具能够帮助人进行代码检查，通过利用源代码中发现的名字生成索引（或标签）文件。

提示
配置你喜欢的编辑器 (`emacs` 或 `vim`) 使用异步 lint 引擎插件帮助你的代码写作。这些插件通过充分利用 Language Server Protocol 的优点，会变得非常强大。因它们在快速开发，使用它们上游的代码代替 Debian 软件包，是一个好的选择。

12.5 调试

调试是程序中很重要的一部分。知道怎样去调试程序，能够让你成为一个好的 Debian 使用者, 能够做出有意义的错误报告。

12.5.1 基本的 **gdb** 使用命令

Debian 上原始的调试器是 gdb(1), 它能让你在程序执行的时候检查程序。

让我们通过如下所示的命令来安装 `gdb` 及其相关程序。

```
# apt-get install gdb gdb-doc build-essential devscripts
```

好的 `gdb` 教程能够被发现：

- “`info gdb`”

软件包	流行度	大小	说明
vim-ale	I:0.72	2833	用于 Vim 8 和 NeoVim 的异步 Lint 引擎
vim-syntastic	I:1.8	1379	vim 语法检查利器
elpa-flycheck	V:0.1, I:1.5	1130	Emacs 现代实时语法检查
elpa-relint	I:0.05	150	Emacs Lisp 正则错误发现器
cppcheck-gui	V:0.09, I:0.93	7973	静态 C/C++ 代码分析工具 (GUI)
shellcheck	V:4, I:17	22860	shell 脚本的 lint 工具
pyflakes3	V:2, I:14	20	Python 3 程序被动检查器
pylint	V:4, I:19	2109	Python 代码静态检查器
perl	V:672, I:991	840	带有内部静态代码检测的解释器：B::Lint(3perl)
rubocop	V:0.08, I:1.00	4708	Ruby 静态代码分析器
clang-tidy	V:2, I:11	20	基于 clang 的 C++ 规则格式检查工具
splint	V:0.05, I:0.75	2325	静态检查 C 程序 bug 的工具
flawfinder	V:0.05, I:0.41	187	检查 C/C++ 源代码和查找安全漏洞的工具
black	V:3, I:15	11130	强硬的 Python 代码格式化器
perltidy	V:0.4, I:2.9	3086	Perl 脚本缩进和重新格式化
indent	V:0.3, I:4.7	438	C 语言源代码格式化程序
astyle	V:0.2, I:2.4	770	C、C++、Objective-C、C# 和 Java 的源代码缩进器
bcpp	V:0.02, I:0.21	114	美化 C(++)
xmlindent	V:0.06, I:0.70	52	XML 流重新格式化
global	V:0.1, I:1.4	1923	源代码检索和浏览工具
exuberant-ctags	V:1, I:12	341	构建源代码定义的标签文件索引
universal-ctags	V:1, I:12	4238	构建源代码定义的标签文件索引

Table 12.12: 静态代码分析工具的列表

软件包	流行度	大小	文档
gdb	V:94, I:163	13472	由 gdb-doc 包提供的“info gdb”
ddd	V:0.2, I:4.8	4519	由 ddd-doc 包提供的“info ddd”

Table 12.13: 调试软件包列表

- 在 `/usr/share/doc/gdb-doc/html/gdb/index.html` 的“Debugging with GDB”
- “tutorial on the web”

这里是一个简单的列子，用 gdb(1) 在” 程序” 带有”`-g`” 选项编译的时候来产生调试信息。

```
$ gdb program
(gdb) b 1               # set break point at line 1
(gdb) run args          # run program with args
(gdb) next              # next line
...
(gdb) step              # step forward
...
(gdb) p parm            # print parm
...
(gdb) p parm=12         # set value to 12
...
(gdb) quit
```

提示
许多 gdb(1) 命令都能被缩写。Tab 扩展跟在 shell 一样都能工作。

12.5.2 调试 Debian 软件包

Debian 系统在默认情况下，所有安装的二进制程序会被 stripped，因此大部分调试符号（debugging symbols）在通常的软件包里面会被移除。为了使用 gdb(1) 调试 Debian 软件包，`*-dbgsym` 软件包需要被安装。（例如，安装 `coreutils-dbgsym`，用于调试 `coreutils`）源代码软件包和普通的二进制软件包一起自动生成 `*-dbgsym` 软件包。那些调试软件包将被独立放在 debian-debug 档案库。更多信息请参阅 Debian Wiki 文档 。

如果一个需要被调试的软件包没有提供其 `*-dbgsym` 软件包，你需要按如下所示的从源代码中重构并且安装它。

```
$ mkdir /path/new ; cd /path/new
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install fakeroot devscripts build-essential
$ apt-get source package_name
$ cd package_name*
$ sudo apt-get build-dep ./
```

按需修改 bug。

软件包调试版本跟它的官方 Debian 版本不冲突，例如当重新编译已存在的软件包版本产生的”`+debug1`” 后缀，如下所示是编译未发行的软件包版本产生的”`~pre1`” 后缀。

```
$ dch -i
```

如下所示编译并安装带有调试符号的软件包。

```
$ export DEB_BUILD_OPTIONS="nostrip noopt"
$ debuild
$ cd ..
$ sudo debi package_name*.changes
```

你需要检查软件包的构建脚本并确保编译二进制的时候使用了”`CFLAGS=-g -Wall`” 选项。

12.5.3 获得栈帧

当你碰到程序崩溃的时候，报告 bug 时附上栈帧信息是个不错的注意。

使用如下方案之一，可以通过 gdb(1) 取得栈帧信息：

- 在 GDB 中崩溃的方案：
  - 从 GDB 运行程序。
  - 崩溃程序。
  - 在 GDB 提示符输入"`bt`"。
- 先奔溃的方案：
  - 更新"**/etc/security/limits.conf**"文件，包括下面内容：

```
* soft core unlimited
```

  - shell 提示符下输入"`ulimit -c unlimited`"。
  - 从这个 shell 提示符运行程序。
  - 崩溃的程序产生一个 core dump 文件。
  - 加载 core dump 文件到 GDB，用"`gdb gdb ./program_binary core`" 。
  - 在 GDB 提示符输入"`bt`"。

对于无限循环或者键盘冻结的情况，你可以通过按 `Ctrl-\` 或 `Ctrl-C` 或者执行"`kill -ABRT` *`PID`*"强制奔溃程序。(参见第 9.4.12 节)

提示
通常，你会看到堆栈顶部有一行或者多行有"`malloc()`" 或"`g_malloc()`". 当这个出现的时候，你的堆栈不是非常有用的。找到一些有用信息的一个简单方法是设置环境变量"`$MALLOC_CHECK_`" 的值为 2 (malloc(3)). 你可以通过下面的方式在运行 `gdb` 时设置。

```
$ MALLOC_CHECK_=2 gdb hello
```

12.5.4 高级 **gdb** 命令

命令	命令用途的描述
`(gdb) thread apply all bt`	得到多线程程序的所有线程栈帧
`(gdb) bt full`	查看函数调用栈中的参数信息
`(gdb) thread apply all bt full`	和前面的选项一起得到堆栈和参数
`(gdb) thread apply all bt full 10`	得到前 10 个调用的栈帧和参数信息，以此来去除不相关的输出
`(gdb) set logging on`	把 `gdb` 的日志输出到文件 (默认的是"`gdb.txt`")

Table 12.14: 高级 gdb 命令列表

12.5.5 检查库依赖性

按如下所示使用 ldd(1) 来找出程序的库依赖性。

```
$ ldd /usr/bin/ls
        librt.so.1 => /lib/librt.so.1 (0x4001e000)
        libc.so.6 => /lib/libc.so.6 (0x40030000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x40153000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

因为 ls(1) 运行在 `chroot`ed 环境，以上的库在 `chroot`ed 环境也必须是可用的。

参见第 9.4.6 节。

12.5.6 动态调用跟踪工具

在 Debian 中，有几个动态调用跟踪工具存在。参见第 9.4 节。

12.5.7 调试与 X 相关的错误

如果一个 GNOME 程序 `preview1` 收到了一个 X 错误，您应当看见一条下面这样的信息。

```
The program 'preview1' received an X Window System error.
```

如果就是这种情况，你可以尝试在运行程序的时候加上"`--sync`" 选项，并且在"`gdk_x_error`" 函数处设置中断来获得栈帧信息。

12.5.8 内存泄漏检测工具

Debian 上有一些可用的内存泄漏检测工具。

软件包	流行度	大小	说明
`libc6-dev`	V:259, I:577	12863	mtrace(1)：调试 glibc 中的 malloc
`valgrind`	V:4, I:31	89318	内存调试器和分析器
`electric-fence`	V:0.2, I:2.2	69	`malloc`(e) 调试器
`libdmalloc5`	I:0.54	373	内存分配库调试
`duma`	I:0.06	295	在 C 和 C++ 程序中检测缓存溢出和缓存欠载（buffer under-runs）的库

Table 12.15: 内存泄漏检测工具的列表

12.5.9 反汇编二进制程序

你可以使用下面的方式通过 objdump(1) 反编译二进制代码。

```
$  objdump -m i386 -b binary -D /usr/lib/grub/x86_64-pc/stage1
```

> 注意
> gdb(1) 可以用来交互式反汇编代码。

12.6 编译工具

软件包	流行度	大小	文档
make	V:155, I:558	1762	通过 make-doc 包提供“info make”
autoconf	V:28, I:189	2244	由 autoconf-doc 包提供“info autoconf”
automake	V:27, I:188	1959	由 automake 包提供“info automake”
libtool	V:23, I:172	1276	由 libtool-doc 包提供”info libtool”
cmake	V:23, I:124	63999	cmake(1) 跨平台、开源的编译系统
ninja-build	V:9, I:55	456	ninja(1) 接近 Make 精髓的小编译系统
meson	V:6, I:29	4402	meson(1) 在 ninja 之上的高生产力的构建系统
xutils-dev	V:0.6, I:6.6	1495	imake(1)，xmkmf(1) 等。

Table 12.16: 编译工具软件包列表

12.6.1 make

Make 是一个维护程序组的工具。一旦执行 make(1)，make 会读取规则文件 Makefile，自从上次目标文件被修改后，如果目标文件依赖的相关文件发生了改变，那么就会更新目标文件，或者目标文件不存在，那么这些文件更新可能会同时发生。

规则文件的语法如下所示。

```
target: [ prerequisites ... ]
 [TAB]  command1
 [TAB]  -command2 # ignore errors
 [TAB]  @command3 # suppress echoing
```

这里面的”[TAB]” 是一个 TAB 代码。每一行在进行变量替换以后会被 shell 解释。在行末使用”\” 来继续此脚本。使用”$$” 输入”$” 来获得 shell 脚本中的环境变量值。

目标跟相关文件也可以通过隐式规则给出，例如，如下所示。

```
%.o: %.c header.h
```

在这里，目标包含了”%” 字符 (只是它们中确切的某一个)。”%” 字符能够匹配实际的目标文件中任意一个非空的子串。相关文件同样使用”%” 来表明它们是怎样与目标文件建立联系的。

自动变量	值
$@	当前目标
$<	首个相关文件
$?	所有较新的相关文件
$^	所有相关文件
$*	目标模式中，$* 指代匹配符”%” 匹配的部分

Table 12.17: 自动变量的列表

变量扩展	说明
foo1 := bar	一次性扩展
foo2 = bar	递归扩展
foo3 += bar	增加

Table 12.18: 变量扩展的列表

运行”make -p -f/dev/null” 命令来查看内部自动化的规则。

12.6.2 Autotools（自动化工具）

Autotools 是一套编程工具，被设计作为协助将源代码软件包移植到许多类 Unix 系统。

- Autoconf 是一个从"`configure.ac`" 生成 shell 脚本"`configure`" 的工具。
  - "`configure`" 随后用于从"`Makefile.in`" 模板生成"`Makefile`"。
- Automake 是一个从"`Makefile.am`" 生成"`Makefile.in`" 的工具。
- Libtool 是一个 shell 脚本，当从源代码编译共享库时，用来定位软件的移植性问题。

12.6.2.1 编译并安装程序



警告
当你安装编译好的程序的时候，注意不要覆盖系统文件。

Debian 不会在"`/usr/local`" 或"`/opt`" 目录下创建文件。如果你想要源码编译程序，把它安装到"`/usr/local/`" 目录下，因为这并不会影响到 Debian。

```
$ cd src
$ ./configure --prefix=/usr/local
$ make # this compiles program
$ sudo make install # this installs the files in the system
```

12.6.2.2 卸载程序

如果你有源码并且它使用 autoconf(1)/automake(1)，如果你能记得你是怎样配置它的话，执行如下的命令来卸载程序。

```
$ ./configure all-of-the-options-you-gave-it
$ sudo make uninstall
```

或者，如果你十分确信安装进程把文件都放在了"`/usr/local/`" 下并且这里没什么重要的东西，你可以通过如下的命令来清除它所有的内容。

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

如果你不确定文件被安装到了哪里，你可以考虑使用 `checkinstall` 软件包中的 checkinstall(8)，它将会提供一个清晰的卸载路径。现在，它支持创建带有"`-D`"选项的 Debian 软件包。

12.6.3 Meson

软件构建系统也在演进：

- Autotools 位于 Make 之上，从 90 年代开始，便是可移植构建架构的事实标准。它是相当慢的。
- CMake 在 2000 年初始发布，显著地改善了速度，但是它源于建立在本质上慢的 Make 之上。(目前 Ninja 能够作为它的后端。)
- Ninja 在 2012 年初始发布，是为了取代 Make，进一步改善构建速度，在设计上，它的输入文件由上层的构建系统来生成。
- Meson 在 2013 年初始发布，是新的和流行的，并且是快速的和上层的构建系统，它使用 Ninja 作为它的后端。

参见在"The Meson Build system" 和"The Ninja build system" 里发现的文档。

12.7 Web

基本的动态交互网页可由如下方法制作。

- 呈现给浏览器用户的是 HTML 形式。
- 填充并点击表单条目将会从浏览器向 web 服务器发送带有编码参数的下列 URL 字符串之一。
  - "`https://www.foo.dom/cgi-bin/program.pl?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`"
  - "`https://www.foo.dom/cgi-bin/program.py?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`"
  - "`https://www.foo.dom/program.php?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`"
- 在 URL 里面"%nn" 是使用一个 16 进制字符的 nn 值代替。
- 环境变量设置为: "`QUERY_STRING="VAR1=VAL1 VAR2=VAL2 VAR3=VAL3"`".
- Web 服务器上的 CGI 程序 (任何一个"`program.*`") 在执行时，都会使用"`$QUERY_STRING`" 环境变量.
- CGI 程序的 `stdout` 发送到浏览器，作为交互式的动态 web 页面展示。

出于安全考虑，最好不要自己从头编写解析 CGI 参数的手艺. 在 Per l 和 Python 中有现有的模块可以使用. PHP 中包含这些功能. 当需要客户端数据存储时, 可使用 HTTP cookies . 当需要处理客户端数据时, 通常使用 Javascript.

更多信息，参见通用网关接口, Apache 软件基金会, 和 JavaScript.

直接在浏览器地址中输入 https://www.google.com/search?hl=en&ie=UTF-8&q=CGI+tutorial 就可以在 Google 上搜索 "CGI tutorial"。这是在 Google 服务器上查看 CGI 脚本运行的好方法。

12.8 源代码转换

源代码转换程序。

软件包	流行度	大小	关键词	说明
`f2c`	V:0.1, I:1.9	443	FORTRAN → C	把源代码从 FORTRAN 77 转换成 C/C++: f2c(1)
`intel2gas`	V:0.01, I:0.17	178	NASM → GAS	从 NASM (Intel 格式) 转换成 GNU 汇编程序 (GAS)

Table 12.19: 源代码转换工具列表

12.9 制作 Debian 包

如果你想制作一个 Debian 包，阅读下面内容。

- 第 2 章理解基本的包管理系统
- 第 2.7.13 节理解基本的移植过程
- 第 9.11.4 节理解基本的 chroot 技术
- debuild(1) 和 sbuild(1)
- 第 12.5.2 节编译和除错
- Debian 维护者指南 (`debmake-doc` 包)

- Debian 开发者参考手册 (`developers-reference` 包)
- Debian 策略手册 (`debian-policy` 包)

`debmake`, `dh-make`, `dh-make-perl` 等软件包，对软件包打包过程，也有帮助。

Appendix A

附录

下列为本文档背景。

A.1 Debian 迷宫

Linux 系统是一个面向网络计算机的功能强大的计算平台。然而，学习使用它的全部功能并非易事。使用非 PostScript 的打印机配置 LPR 打印机队列，就是个好例子。（自从使用新 CUPS 系统的新的安装系统出现后，就不再会有问题。）

有一张完整而详尽的地图叫做“SOURCE CODE”，它非常准确但极难理解。还有一些参考书叫 HOWTO 和 mini-HOWTO，它们易于理解，但给出了太多细节反而让人忘记了大方向。为了使用某个命令，我有时得在长长的 HOWTO 中找上半天。

我希望这个“Debian 参考手册（版本 2.153）”（2026-11-29 05:38:00 UTC）能帮助在 Debian 迷宫里面徘徊的人们，为他们提供一个好的出发方向。

A.2 版权历史

Debian 参考手册由我（Osamu Aoki<osamu at debian dot org>）发起，将其作为一个个人系统管理笔记。其中的许多内容都是我从 Debian 用户邮件列表和其他 Debian 相关资源获得的积累。

在采纳了来自 Josip Rodin 的建议之后（Josip Rodin 在 Debian 文档项目（DDP）中非常活跃），“Debian 参考手册（第一版，2001-2007）”成为了 DDP 文档的一员。

6 年后，我意识到原来的“Debian 参考手册（第一版）”内容陈旧，便开始重新很多内容。新的“Debian 参考手册（第二版）”在 2008 年发布。

我已经更新了”Debian 参考手册 (版本 2)” 来处理新的话题 (Systemd, Wayland, IMAP, PipeWire, Linux 内核 5.10)，移除过期话题 (SysV init, CVS, Subversion, SSH 1 协议, 2.5 版本之前的 Linux 内核). Jessie 8 (2015-2020) 版本的情况，或者更老的内容也被大部分移除。

这个”Debian 参考手册 (version 2.153)” (2026-11-29 05:38:00 UTC) 覆盖了大部分 Trixie (=`stable`) 和 Forky (=`testing`) Debian 版本。

教程的起源和灵感，可以通过下面的内容来追溯。

- ”Linux 用户手册” Larry Greenfield (1996 年 12 月)
  - 该文档被后来的《Debian 教程》取代
- ”Debian 教程” Havoc Pennington (1998 年 12 月 11 日)

  - 部分由 Oliver Elphick, Ole Tetlie, James Treacy, Craig Sawyer 和 Ivan E. Moore II 书写
  - 该文档被后来的《Debian GNU/Linux: 安装和使用手册》取代
- ”Debian GNU/Linux: 安装和使用手册” John Goerzen 和 Ossama Othman (1999)
  - 该文档被《Debian 参考手册（第一版）》取代

软件包和文档描述的一些起源和灵感，能够通过下面的内容来追溯。

- ”Debian FAQ” (2002 年 3 月版本，当时是由 Josip Rodin 维护)

其它内容的一些起源和灵感，能够通过下面的内容来追溯。

- ”Debian 参考手册 (第一版)” Osamu Aoki (2001–2007)
  - 于 2008 年被这个新的”Debian 参考手册 (第二版)” 取代。

先前的“Debian 参考手册 (第一版)”由许多贡献者创建。

- Thomas Hood 是网络配置主题的主要内容贡献者
- Brian Nelson 突出贡献了关于 X 和 VCS 的相关主题
- Jens Seidel 对构建脚本和许多内容的更正提供了帮助
- David Sewell 进行了大量的校对
- 来自翻译者、贡献者和 bug 报告者的许多贡献

Debian 系统中许多的手册页面和 info 信息页面，和上游网站页面，Wikipedia 维基百科文档，被用来作为这个文档的主要参考。在一定范围内，青木修（Osamu Aoki）也考虑了公平使用，它们中的许多地方，尤其是命令的定义，在细心的编辑以适应样式和本文档的目标后，作为了本文档的短语部分。

gdb 调试器的描述使用了扩展 Debian 维基内容的回溯系统, 这是被 Ari Pollak, Loïc Minier, 和 Dafydd Harries 同意的.

除了上面提到的部分之外，“Debian 参考手册（版本 2.153）（2026-11-29 05:38:00 UTC）”的大部分内容是我自己的工作。一些贡献者也会对内容进行更新。

作者 Osamu Aoki 在此感谢所有在文档写作过程中曾给予帮助的人。

A.3 文档格式

目前，英文原始文档使用 DocBook XML 文件写作。此源文件可被转换成 HTML、纯文本、PostScript 和 PDF。(发布时会省略部分格式。)