

深度易经

编者：不避风云



Deepin Bible

深度易经

深度操作系统爱好者 编著

尚未出版

版权 © 2018 深度操作系统爱好者

尚未出版

[HTTPS://GITHUB.COM/BUBIFENGYUN/DEEPIN-BIBLE](https://github.com/bubifengyun/deepin-bible)

版权采用 Creative Commons Attribution-NonCommercial 3.0 Unported License，内容见 <http://creativecommons.org/licenses/by-nc/3.0>。共享版权，商业用途请联系我们。如果您采用六角丛书的价格发布图书，可以不用联系我们。

尚未出版于 2019 年 02 月 10 日

前言

为什么要写这本书呢？起源于去武汉参观辛亥革命博物馆。深深的被孙中山的“敢为天下先”所感动。另外也要感谢深度科技公司的 [jingle](#) 提供的帮助和支持，内心总是感觉深度科技就是敢为天下先的代表，俺也不能自甘落后，于是下决心写本书。

另外感谢深度公司还给资助了二百来块钱，太感谢啦，写在[这里的稿费](#)。还有很多默默支持的小伙伴，在此一并表示感谢。

另外还要说一个中国早就存在的二大爷精神。虽然 60 多年过去了，你二大爷还是你二大爷。它代表了一种个性要强不怕输敢于拼搏的精神。也用于纪念我逝去的二大爷。写这本书，也表达对他的敬意和怀念。

本书的封面，是我的好友王一凡帮助设计的，特别表示感谢。相信大家一看到这两个动物就会明白所代表的啥意思。不再过多解释。

本书结构如下。打算模仿《Linux Bible》，采取深入浅出的方式，先介绍一些简单的使用方法，然后后面进行深入。

- 第一部分 Linux 及 Deepin 入门常识
- 第二部分 用户相关
- 第三部分
- 第四部分
- 第五部分
- 附录及其他
 - 本书模板制作说明
 - 常见问题

其中，第一部分主要介绍 Linux 及 Deepin 的哲学以及相关的知识。第二部分，第三部分，

文件名采用如下方式，`xxx-name.Rmd`。其中 `xxx` 表示从 000 到 999，第一位表示第几部分，后两位如果为 00 表示该部分的简介，如果是其他数字则表示章节。`name` 是对应章节的名字。一章一个文件。

本书的代码规范。

- 文件和目录用斜体表示，也就是用 * 括起来。

目 录

前言	I
插图索引	XVII
表格索引	XX
致谢	XXI
作者简介	XXIII
第一部分 Linux 及 Deepin 入门	1
第一章 GNU/Linux 操作系统简介	3
1.1 GNU/Linux 使用范围	3
1.2 GNU/Linux 是什么	4
1.2.1 Richard Stallman	4
1.2.2 Linus Torvalds	6
1.3 Unix 哲学	8
1.3.1 Unix 哲学核心	8
1.3.2 Unix 哲学下的用户	9
1.3.3 Unix 哲学的对手	9
1.4 GNU/Linux 发行版的出现	10
1.4.1 Red Hat 系	10
1.4.2 Debian 系	10
1.5 如何提高自己的 Linux 技术	10
1.5.1 首先安装 GNU/Linux 操作系统	11
1.5.2 平常多用	11
1.5.3 多问多记	11
1.5.4 定个计划	11
1.6 总结	12

第二章 Linux 桌面系统简介	13
2.1 运行 Live CD/DVD 试用系统	13
2.1.1 下载深度系统	13
2.1.2 把系统写到 U 盘里	13
2.1.3 开始试用	14
2.2 X Window 系统简介	14
2.2.1 基本部件	15
2.2.2 用户接口	16
2.2.3 用户界面	16
2.2.4 优缺点	17
2.3 王勇谈 Wayland	18
2.4 常见的桌面环境	21
2.4.1 Cinnamon	21
2.4.2 GNOME	22
2.4.3 KDE	23
2.4.4 LXDE	25
2.5 总结	25
第三章 深度操作系统简介	27
3.1 深度科技简介	27
3.2 深度操作系统的安装	29
3.2.1 安装方案	29
3.2.2 安装环境	30
3.2.3 启动优盘的制作	30
3.2.4 安装过程	30
3.3 桌面使用	32
3.3.1 开机	33
3.3.2 关机	34
3.3.3 控制中心-系统设置	35
3.3.4 开机自启动	36
3.3.5 常用快捷键	37
3.3.6 安装软件	37
3.3.7 卸载软件	38
3.4 深度桌面常用软件	38

3.4.1 搜狗输入法	38
3.4.2 QQ	39
3.4.3 深度截图	39
3.4.4 网易云音乐	40
3.4.5 深度系统监视器	40
3.4.6 深度无线投屏	41
3.5 总结	43
3.6 附注	43
第四章 教学用的 Linux 软件	45
4.1 引言	45
4.2 笔记	45
4.2.1 leanote	45
4.2.2 Xpad 便笺	45
4.3 课堂工具	48
4.3.1 Veyon	48
4.3.2 AContent	50
4.3.3 Moodle	50
4.4 文献阅读写作管理工具	50
4.4.1 bookdown	50
4.4.2 VuFind	51
4.4.3 Calibre	51
4.4.4 CAJViewer	52
4.4.5 搜索神器 Everything	52
第二部分 GNU/Linux 基础知识	53
第五章 shell 用法简介	55
5.1 用 shell 有啥好处?	55
5.2 深度操作系统下 shell 简介	56
5.2.1 牛刀小试	57
5.2.2 命令语法结构	58
5.2.3 选项 (Options)	58
5.2.4 参数 (Arguments)	59

5.2.5	更多例子	59
5.2.6	找找命令文件在哪里	61
5.3	命令快捷编辑	62
5.3.1	方便编辑的快捷键	62
5.3.2	重输之前的命令	64
5.3.3	自动补全	65
5.4	命令连接与扩展	66
5.4.1	匿名管道	66
5.4.2	重定向	67
5.4.3	重定向和管道的区别	70
5.4.4	顺序执行	72
5.4.5	后台命令	72
5.4.6	命令扩展	73
5.4.7	简单数值计算	73
5.4.8	输出变量值	73
5.5	Shell 变量 (Variables)	74
5.5.1	别名的创建和使用	76
5.5.2	退出 shell	77
5.6	定制 shell 环境	77
5.6.1	配置提示 (prompt)	78
5.7	命令帮助	81
5.8	总结	82
第六章	文件系统	83
6.1	树形结构	83
6.2	文件系统常用命令	86
6.3	元字符和操作符	87
6.3.1	元字符	87
6.3.2	操作符	89
6.3.3	花括号 {}	89
6.4	几个常用的目录	90
6.5	文件 (夹) 的权限和归属	90
6.5.1	权限简介	91
6.5.2	权限更改 chmod	92

6.5.3 权限掩码 umask	94
6.5.4 归属	98
6.6 文件（夹）的创建、查看、移动、复制和删除	99
6.6.1 文件（夹）的创建	99
6.6.2 文件（夹）的查看	99
6.6.3 文件（夹）的复制	101
6.6.4 文件（夹）的删除	101
6.7 总结	102
第七章 文件编辑与查找	103
7.1 字符集和字符编码	103
7.1.1 ASCII	103
7.1.2 GB2312	104
7.1.3 GBK 字符集	105
7.1.4 BIG5	105
7.1.5 GB18030	106
7.1.6 Unicode 字符集	107
7.1.7 编码的常用命令	108
7.2 编辑器	110
7.3 文件查找	110
7.3.1 find 查找文件	110
7.3.2 grep 查找文件内部信息	115
7.3.3 其他检索工具	116
7.3.4 正则表达式	116
7.4 总结	124
第八章 进程管理	125
8.1 何谓进程	125
8.2 深度系统监视器管理进程	126
8.3 命令方式查看进程	128
8.3.1 ps 命令基本用法	129
8.3.2 top 命令基本用法	130
8.4 进程的分类	133
8.4.1 怎么生成后台进程呢？	133

8.5	进程的终结与调整优先级 (Killing and Renicing)	134
8.6	通过 cgroups 限制进程	137
8.6.1	cgroups 是什么?	137
8.6.2	cgroups 可以做什么?	137
8.6.3	cgroups 相关概念及其关系	138
8.6.4	cgroups 子系统介绍	139
8.7	进程的状态	139
8.7.1	Linux 进程状态: R (TASK_RUNNING), 可执行状态。 . .	139
8.7.2	Linux 进程状态: S (TASK_INTERRUPTIBLE), 可中断的睡眠状态。	140
8.7.3	Linux 进程状态: D (TASK_UNINTERRUPTIBLE), 不可中断的睡眠状态。	140
8.7.4	Linux 进程状态: T (TASK_STOPPED or TASK_TRACED), 暂停状态或跟踪状态。	140
8.7.5	Linux 进程状态: Z (TASK_DEAD - EXIT_ZOMBIE), 退出状态, 进程成为僵尸进程。	141
8.7.6	Linux 进程状态: X (TASK_DEAD - EXIT_DEAD), 退出状态, 进程即将被销毁。	142
8.7.7	进程的初始状态	142
8.7.8	进程状态变迁	143
8.8	总结	143
第九章	简单 bash 脚本	145
9.1	何谓 shell 脚本	145
9.1.1	执行和调试	145
9.1.2	shell 变量	146
9.1.3	特殊变量	148
9.1.4	执行时输入参数	148
9.1.5	其他需求的参数	149
9.1.6	简单计算	150
9.2	shell 脚本的三大结构	151
9.2.1	分支结构的语法	151
9.3	流编辑器 sed	156
9.4	shell 脚本例子: 转换 UC 缓存视频	156

9.4.1 生成 file.txt 文件	157
9.4.2 生成 MP4 文件	158
9.4.3 做成一个 bash 脚本	158
9.5 总结	160

第三部分 本机管理员 161

第十章 163

第十一章 深度系统安装 165

11.1 UEFI 和 legacy BIOS 区别和联系	165
11.2 优盘安装	165
11.2.1 第一步下载并校验 deepin.iso 文件	165
11.3 安装遇到的问题	166

第十二章 软件安装 167

12.1 npm 软件的安装	167
12.2 veil 软件安装	167
12.3 dpkg 安装	168
12.4 其他安装方式	168
12.4.1 bitnami	168
12.4.2 turnkeylinux	168
12.5 you-get 软件的安装	168
12.6 字体的安装	169
12.7 输入法的安装	170

第十三章 管理账户 171

13.1 引言	171
13.2 创建账户	171
13.2.1 图形方式	171
13.2.2 命令方式	174

第四部分 服务器管理员	175
第十四章 服务器简介	177
第十五章 Web Server 的搭建与运行	179
15.1 面向开发的一键安装类型	179
15.2 关注效率稳定的搭建方法	179
第十六章 与 Windows 共享文件打印机的 samba 服务	181
16.1 Samba 文件传输服务简介	181
16.2 深度自带 samba 的使用	182
16.2.1 共享本地文件	182
16.2.2 访问共享文件	182
16.2.3 我的共享	183
16.2.4 可能存在的小问题	183
16.3 关于 samba 的配置	184
16.3.1 手动安装	184
16.3.2 启动与停止	185
16.3.3 配置文件	185
16.4 例：借助安卓软件 U-File 实现手机电脑互传	185
16.4.1 开启局域网	185
第五部分 安全	187
第十七章 网络安全	189
第六部分 附录	191
附录甲 愚公移山	193
附录乙 如何制作本书	195
乙.1 准备工作	195
乙.1.1 Linux 下使用	195
乙.1.2 Windows 下使用	196
乙.1.3 苹果操作系统下使用	196

乙.2 编译模板	196
乙.2.1 第一种编译方法——命令行编译	196
乙.2.2 第二种编译方法——RStudio 编译	196
乙.2.3 字数统计	196
乙.2.4 本书编译的 R 各包信息	197
乙.3 文件布局	198
乙.4 主要文件介绍	198
乙.4.1 L ^A T _E X 模板文件	198
乙.4.2 各章源文件	199
乙.4.3 配置文件	199
乙.4.4 图片文件夹 images	199
乙.4.5 参考文献数据库 bib	199
乙.4.6 辅助文件	199
附录丙 各章格式说明	201
丙.1 章：前言	201
丙.2 前言后的部分章节	202
丙.3 部分：第一部分简介	202
丙.4 各部分内部的章	203
丙.5 部分：附录及其他	203
丙.6 附录内部各章	203
丙.7 后缀部分	203
附录丁 兼容 L^AT_EX 排版	205
丁.1 列表环境	205
丁.1.1 无序列表	205
丁.1.2 有序列表	205
丁.1.3 描述型列表	205
丁.1.4 自定义列表样式	206
丁.2 数学排版	206
丁.2.1 公式排版	206
丁.2.2 SI 单位	206
丁.2.3 定理环境	207
丁.3 向文档中插入图像	208

丁.3.1 支持的图片格式	208
丁.3.2 长标题的换行	209
丁.3.3 添加图注	209
丁.3.4 绘制流程图	209
丁.4 表格	212
丁.5 参考文献管理	212
丁.6 用 listings 插入源代码	214
丁.7 用 algorithm 和 algorithmicx 宏包插入算法描述	215
附录戊 RMarkdown/Bookdown 排版示例	219
戊.1 Markdown syntax	219
戊.1.1 Inline formatting	219
戊.1.2 Block-level elements	220
戊.1.3 Math expressions	222
戊.2 Markdown extensions by bookdown	223
戊.2.1 Number and reference equations	223
戊.2.2 Theorems and proofs	225
戊.2.3 Special headers	230
戊.2.4 Text references	231
戊.3 R code	232
戊.4 Figures	233
戊.5 Tables	238
戊.6 Cross-references	242
戊.7 Custom blocks	243
戊.8 Citations	246
戊.9 Index	249
戊.10 HTML widgets	249
附录己 常见问题	253
附录庚 操作系统安装延伸阅读	259
庚.1 计算机引导过程	259
庚.1.1 传统 BIOS 引导	259
庚.1.2 UEFI 引导	259
庚.1.3 UEFI 的多重引导	259

庚.2 常见 BIOS 设置	260
庚.2.1 常见启动引导器	260
庚.2.2 NTLDR/BOOTMGR	260
庚.2.3 GNU GRUB 及其使用	261
庚.3 LINUX 启动过程	261
庚.3.1 VMLINUZ	261
庚.3.2 INITRD	262
附录辛 大事记	263

插图索引

1–1 Richard Stallman/理查德·马修·斯托曼	4
1–2 Linus Benedict Torvalds/林纳斯·本纳第克特·托瓦兹	6
2–1 如果只是试用请选择 Live 系统	14
2–2 X 架构图	19
2–3 Wayland 架构图	20
2–4 Cinnamon 桌面环境样图	22
2–5 GNOME 桌面环境样图	23
2–6 KDE 桌面环境样图	24
2–7 LXDE 桌面环境样图	24
3–1 武汉深之度科技有限公司标志	27
3–2 深度操作系统桌面	28
3–3 安装界面	31
3–4 账户界面	31
3–5 磁盘挂载界面	32
3–6 启动项	33
3–7 任务栏打开软件动画	34
3–8 关机	35
3–9 控制中心	36
3–10 开机自启动	37
3–11 深度商店	38
3–12 深度截图中文版	39
3–13 网易云音乐	40
3–14 深度系统监视器	41
3–15 深度无线投屏	42
4–1 leanote 软件安装	46
4–2 leanote 使用界面	46
4–3 小号版软件列表	47
4–4 全屏版软件列表	47

4-5 Xpad 初始界面	48
4-6 Xpad 右键列表	49
4-7 Xpad 配置页面	49
5-1 管道输出示意图	66
6-1 Linux 系统目录结构图	84
8-1 深度系统监视器	126
8-2 标签显示, 快速定位	127
8-3 列表展示, 高效右键	127
8-4 捕捉窗口, 即点即“杀”	128
8-5 top 显示界面	131
8-6 top 显示界面	132
13-1 图形方式创建账户	173
13-2 图形方式配置账户	173
13-3 图形方式配置账户	174
16-1 共享文件夹出现的问题	183
16-2 取消共享出现的问题	184
16-3 开启无线网热点	186
甲-1 徐悲鸿名画《愚公移山》	193
丁-1 这里将出现在插图索引中	208
丁-2 插入 eps 和 pdf 的例子 (使用 subcaptionbox 方式)	208
丁-3 插入 eps 和 pdf 的例子 (使用 subfigure 方式)	209
丁-4 这里将出现在插图索引	209
丁-5 出现在插图索引中	210
丁-6 出现在插图索引中	210
丁-7 绘制流程图效果	211
戊-1 A figure example with the specified aspect ratio, width, and alignment .	235
戊-2 A figure example with a relative width 70%	236
戊-3 Two plots placed side by side	236

戊-4 Three knitr logos included in the document from an external PNG image	
file	237
戊-5 A table widget rendered via the DT package 250
己-1 打开 Windows10 设置	254
己-2 选择系统设置	255
己-3 选择其他电源设置	256
己-4 选择电源和按钮的功能	257
己-5 取消快速启动	258

表格索引

0-1 捐款名录	XXI
3-1 文件挂载说明	32
5-1 光标跳转快捷键说明	63
5-2 编辑快捷键说明	63
5-3 复制粘贴快捷键说明	64
5-4 三个特殊的文件	68
5-5 管道命令与重定向区别	70
5-6 常见环境变量表	74
5-7 Bash 配置文件说明	77
5-8 Shell 提示符中用到的转义字符	78
5-9 用转义序列来设置文本颜色	80
5-10 用转义序列来设置背景颜色	80
5-11 光标移动转义序列	81
5-12 命令帮助信息	81
5-12 命令帮助信息	82
6-1 文件系统常用命令	86
7-1 非打印字符转义字符	117
7-2 特别字符	118
7-3 正则表达式的限定符	119
7-4 正则表达式的定位符	121
8-1 信号对应的数值及意义	135
9-1 常见测试语句含义 (<code>help test</code> 截取)	152
丁-1 指向一个表格的表目录索引	212
丁-2 出现在表目录的标题	212
戊-1 Theorem environments in <code>bookdown</code>	226

戊-2 A table of the first 10 rows of the mtcars data.	239
戊-3 A Tale of Two Tables.	239
戊-4 A table generated by the longtable package.	240

致谢

首先感谢 bookdown 的开发者，谢益辉和他的小伙伴们，提供了这么好的一个做笔记的模板。另外，本书籍模板获得第一届 bookdown 大奖赛的[三等奖](#)，可作为一些技术资料的书籍制作模板。

当然也要感谢《Linux Bible》的作者，以及为 Linux 做出贡献的那些人。

也有深度科技公司，谢谢他们给提供了这么好的一个操作系统。

也要谢谢各位支持我的好友、网友和亲人。给了我很大的动力，让我开启这个活动。能不能坚持下去，相信自己。

当然最主要的是家人的支持。前些日子，深度副总王勇辞职，提到自己深夜加班，家人来公司找的事情，深有同感。写点文字还是很费时间的，在休息时间不能很好的陪陪家人，很容易带来家庭矛盾，而且很需要家人来理解，做好家人的工作。如果写的不好，对别人不说愧疚不愧疚，至少对家人是一种很大的不负责任。

如果您也想感谢这份模板的话，可以捐款支持本人工作。有钱的捧个钱场，没钱的扫扫红包，也算帮忙。谢谢。



表 0-1 捐款名录

捐款人	金额	日期	备注	捐款人	金额	日期	备注
* 小定	6.66	20181007		* 小定	6.66	20181007	

作者简介

这将是一群深度操作系统爱好者的杰作!!!

我一直致力于寻找合作者，但是我感觉现在有点迷茫。这边进度实在是慢，常常一个月写不了多少内容，可能一个月就几百字，都写不到一页内容。经常停下来。我只是想告诉别人，这个事业还在继续中。我却找不到一点点进步的身影。

现在（2017-10-29）写作进行的很困难，主要是零碎的时间很难用于整理大篇大篇的文字。还有其他繁重的任务要做，似乎不太容易写。我觉得可以分析一下困难，研究一下对策。不要再让这本书半途而废了。现在遇到的问题，负责带教，挤占了较多的时间，一时半会，他们也帮不上忙。还有两个网站的内容在整理，不可荒废。包括驾校也要报名。结婚的事情，还有一大堆。也就是有这么几件事，婚前准备，两个网站的维护，驾校考试，带教学习，完成本书第一版草稿，基本就是这么个优先级顺序，当然时刻伴随着要上班这个无法逃脱的事情。鉴于此，我认为接下来可以这样安排。婚前准备虽然优先级高，但是实际占场时间不多，如果可以快速下决定，花不了太长时间的。两个网站的维护，基本上要浪费掉很多晚上的时间。可以每天抽出两个小时的时间，弄一弄。驾校的事情，周末可以去试试，也可以在工作间隙去练练。带教学习，也是很耗费时间的，每次可能要一到两个小时，一周也就两到三次。可以跟网站维护交叉进行。完成本书第一版草稿，更是要耗费几年时间的任务。另外带教学习的就是本书，可以充分利用这个带教的事情，毕竟有徒弟可以帮着写写画的。制定这么一个时间表，有时间里面把婚前准备的事情给结束掉，避免多次设想，带来的巨大时间浪费。每周抽出两到三天晚上带教，可以做好任务安排，每完成一章课程，作业就是帮我整理本草稿。剩下的两到三天去编写网站代码，做好维护，且不再增加新的网站或者其他任务了。驾校的事情，暂且缓一缓，找个时间去把科目一考了。

终于领结婚证了（2017-11-03），我现在觉得写作特别耗费时间，需要“庙算”一下，好好规划一下，要写哪些内容。github的一些功能是非常值得充分利用的，比如 issue，这样可以充分利用零碎时间。半夜我坐在一个孤独的房间，又不看书，怎么可能写好这个笔记。必须边学习边做笔记，也就是写到 issue 上。不要手机整天微信、支付宝了。

（2018-12-04）工作尚可。妻子怀孕，也不能多陪伴，心有内疚。感觉很多事情一团糟，停下来，重整行装再出发。期待明天更美好吧。（2019-01-06）儿子出生，忙着带孩子，期间很忙，告诉自己，一定要坚持下来。

作者简介

当前贡献者：黄煌、贺鹏飞、董春柏、邱鹏飞、高洪亮，还有参考网页的一些好友，抱歉没来得及写在这里，以后慢慢添加，在此表示感谢。

第一部分

Linux 及 Deepin 入门

“合抱之木，生于毫末；九层之台，起于垒土；千里之行，始于足下。”

——老子

“话说天下大势，分久必合，合久必分”。

对于操作系统而言，Linux 有着类似的境地。GNU/Linux 操作系统发行版众多而杂乱，有些发行版之间相互不兼容对方的软件，安装方法也不尽相同，有点军阀割据的感觉，给初学者带来了不少麻烦。

但是，万变不离其宗，总有那么一些核心的东西是不变的，有人说这就是 Linux 哲学。如同“分久必合”能够成功预言一样，那是因为大家骨子里的文化都是一致的，在变化中文化依旧能够保持一致，这样才可以再次“合”。出于对深度 Deepin 操作系统理念的认同，我深信在某个地方存在着一个极强的凝聚力。我以为就是“敢为天下先”，“天道酬勤”，“愚公移山”，“世上本没有路，走的人多了，大家就都往这里走了。”

本部分主要介绍 Linux 常识和 Linux 哲学，以及深度操作系统的理念和潜在用户。

第一章 GNU/Linux 操作系统简介	3
第二章 Linux 桌面系统简介	13
第三章 深度操作系统简介	27
第四章 教学用的 Linux 软件	45

第一章 GNU/Linux 操作系统简介

通过阅读本章，你将会了解到以下几项内容。

- GNU/Linux 使用范围。
- GNU/Linux 发展简史以及 GPL 协议。
- Unix 哲学简介
- 如何学习 GNU/Linux。

1.1 GNU/Linux 使用范围

有人说，GNU/Linux 作为服务器操作系统，在市场上占有绝对优势，在超级计算机前 500 名中更是占有垄断地位。好吧，这些都不是我们关心的事情。我们关心的是 GNU/Linux 好用不，适合我不。

如果你是程序员。你可以用它方便的写代码，并很容易调试。有功能异常强大的终端，供你跟整个世界沟通。方便你写文档，比如[石墨文档](#)等软件。对于公司办公需要的内部邮件系统，以及内部聊天系统，可能不是很好用，但都有相关的替代软件，比如雷鸟邮件、飞鸽等。

如果你是办公室文员。那办公呢，搜狗拼音输入法，赫然在列，金山的 WPS 办公套件也是能满足你基本需求的。那 QQ 呢，这个也是必须有的，能不能告诉你，在深度 Linux 已经实现视频聊天了呢。当然还有很多特殊的办公软件在 GNU/Linux 下没有可用的版本。比如 photo shop，autoCAD 等。根据具体情况来采纳吧。

如果你只是在家里使用。比如趣味性，听听音乐，看看电影，这些都是可以办得到的。那打游戏呢？很不幸的告诉你，Linux 操作系统下能玩的游戏真心不多，但是各位游戏大佬也在逐步把部分游戏移植到 GNU/Linux 系统下，比如 steam 下的很多游戏开始支持 GNU/Linux 系统了。

如果你是求职者。当然为了更好的找工作，提高技术能力，期待更好的薪水，技多不压身，不妨来学学 GNU/Linux，甚至可以考考某些证书，绝对“技多不压身”。

另外 GNU/Linux 桌面操作系统，稳定性不是很高，但也没有 Windows 下全家桶系列乱弹窗的软件。如果你对桌面清洁有特殊爱好，且 GNU/Linux 能满足你日常工作生活需要，不妨来试试它吧。

1.2 GNU/Linux 是什么

什么？其实他就是一个电脑操作系统啦。那操作系统是啥，那我也不知道啦，这个你可以百度一下，简单的说就是方便你操作电脑的一个软件系统。关于 Linux 有啥发展历史，以及跟 GPL 协议有啥关系。这些网上都有铺天盖地的介绍，本书就不介绍啦。不过还是要介绍两个人，一位是 GNU 的老大 [Richard Stallman](#)，一位是 Linux 的老大 [Linus](#)。

1.2.1 Richard Stallman

本节摘自[百度百科](#)，有改动。

个人经历

理查德·马修·斯托曼（Richard Matthew Stallman, RMS）1953 年出生于美国纽约曼哈顿地区，1971 年进入哈佛大学学习，同年受聘于麻省理工学院人工智能实验室（AI Laboratory），成为一名职业黑客。在 AI 实验室工作期间，斯托曼开发了多种今后影响深远的软件，其中最著名的就是 Emacs。斯托曼在 AI 是一名典型的黑客，是整个黑客文化的一份子。

然而进入八十年代后，黑客社群在软件工业商业化的强大压力下日渐土崩瓦解，甚至连 AI 实验室的许多黑客也组成了 Symbolic 公司，试图以专利软件来取代实验室中黑客文化的产物——免费可自由流通的软件。



图 1-1 Richard Stallman/理查德·马修·斯托曼

斯托曼对此感到气愤与无奈。在对 Symbolic 进行了一段时间的抗争后，他于 1985 年发表了著名的 GNU 宣言 (GNU Manifesto)，正式宣布要开始进行一项宏伟的计划：创造一套完全自由免费，兼容于 Unix 的操作系统 GNU (GNU's Not Unix!)。之后他又建立了自由软件基金会来协助该计划。

他于 1989 年与一群律师起草了广为使用的 GNU 通用公共协议证书 (GNU General Public License, GNU GPL)，创造性地提出了“反版权”（或“版权属左”，

或“开权”，copyleft) 的概念。同时，GNU 计划中除了最关键的 Hurd 操作系统内核之外，其他绝大多数软件已经完成。

1991 年芬兰大学生 Linus Torvalds 在 GPL 条例下发布他自己创作的 Linux 操作系统内核，至此 GNU 计划正式完成，操作系统被命名为 GNU/Linux (或简称 Linux)。

斯托曼是一名坚定的自由软件运动倡导者与其他提倡开放源代码的人不同，斯托曼并不是从软件质量的角度而是从道德的角度来看待自由软件。他认为使用专利软件是非常不道德的事，只有附带了源代码的程序才是符合其道德标准的。对此许多人表示异议，并也因此有了自由软件运动与开源软件运动之分。

GNU

斯托曼做了一个与 Unix 兼容的操作系统。这样容易被移植，而且 Unix 用户可以方便地转移过来。这个系统的名字就叫 GNU，这个名字的确定就是遵循黑客传统，是一个递归的缩略词：“GNU IS NOT UNIX。”

但一个操作系统并不仅仅意味着一个内核 (管理磁盘，内存分配等)，而且仅能运行其他程序也是不够的。一个完整的操作系统，还需要有指令处理器、汇编程序、编译器、解释程序、调试器、文本编辑器、邮件软件等等，这样才能形成一个完整的系统。斯托曼决定尽可能采用已有的自由软件，比如一开始他将 Tex 作为主要的文本格式标识符，几年后他又用 X Window 系统作为 GNU 的窗口系统。

1984 年 1 月，斯托曼已启动了 GNU 计划，他担心 MIT 会要求产品的所有权，会给产品强加入他们的销售条件，最终又会成为专有软件，因而他辞去了 MIT 工作。辞职后，他为买不起电脑而发愁时，发现自己原来在人工智能实验室的办公室，还没有分给其他人用时，他就每天晚上溜进去工作。

GNU 工程启动后，斯托曼听说有一个自由大学编译器套件 (VUCK)。他去询问能否用入 GNU。答复是嘲弄式的，说对大学是自由的，但对软件本身不行。于是，他决定为 GNU 编写的第一个软件就是一个多语言、多平台的编译器。他想利用 Pastel 编译器的源代码，但最终放弃。他从头编写了新的编译器，名为 GCC(the GNU Compiler Collection)。

1984 年 9 月，斯托曼开始开发字处理器 GNU Emacs，1985 年初，它开始可以工作。这使它可以在 Unix 系统上进行文本编辑。此时，许多人想使用 Emacs，因此一个现实的问题是：如何传播它？

当然，他将其放到了 MIT 计算机的匿名服务器上。但那时互联网还未普及，人们很难通过 FTP 获得拷贝，而且失业的斯托曼也需要收入。于是，他宣布任何人都可以用 150 美元的价格获得全部程序。当然，所支付的费用是远低于当时的

专有软件的价格的，并且用户可以得到软件的源代码。这样，自由软件的分销商业模式就此诞生。如今，整个基于 Linux 的 GNU 系统都是如此。

1.2.2 Linus Torvalds

本节摘自[百度百科](#)，有改动。

个人经历

林纳斯·本纳第克特·托瓦兹 (Linus Benedict Torvalds)，1969 年 12 月 28 日出生于芬兰赫尔辛基市。父亲尼尔斯·托瓦兹 (Nils Torvalds) 是一名活跃的共产主义者及电台记者，曾当选芬兰共产党中央委员会委员。托瓦兹家族属于在芬兰占 6% 的少数民族芬兰瑞典人。他毕业于赫尔辛基大学计算机系，1997 年至 2003 年在美国加州硅谷任职于全美达公司 (Transmeta Corporation)，现受聘于开放源代码开发实验 (OSDL: Open Source Development Labs, Inc)，全力开发 Linux 内核。与妻子托芙 (Tove, 芬兰前女子空手道冠军) 育有三个女孩。

与很多其他黑客不同，托瓦兹行事低调，一般很少评论商业竞争对手（例如微软）产品的好坏，但坚持开放源代码信念，并对微软等对手的 FUD 战略大为不满。



图 1-2 Linus Benedict Torvalds/林纳斯·本纳第克特·托瓦兹

例如，在一封回应微软资深副总裁 Craig Mundie 有关开放源代码运动的评论 (Mundie 批评开放源代码运动破坏了知识产权) 的电子邮件中，托瓦兹写道：“我不知道 Mundie 是否听说过艾萨克·牛顿 (Isaac Newton) 爵士？他不仅因为创立了经典物理学 (以及他和苹果的故事) 而出名，也还因为说过这样一句话而闻名于世：我之所以能够看得更远，是因为我站在巨人肩膀上的缘故。”托瓦兹又说道：“我宁愿听牛顿的也不愿听 Mundie 的。他 (牛顿) 虽然死了快 300 年了，却也没有让房间这样得臭气熏天。”

现年 49 岁的林纳斯目前受聘于开放源代码开发实验室 (OSDL, Open Source Development Labs)，全身心的开发 Linux 内核。尽管这位年轻人看上去毫不起眼，

比如林纳斯曾在他的自传《乐者为王》(Just for Fun) 中自嘲：“我是一个长相丑陋的孩子，凡是见过我小时候照片的人，都会觉得我的相貌酷似河狸。再想象一下我不修边幅的衣着，以及一个托瓦兹家族祖祖辈辈遗传下来的大鼻子，这样，在你脑海中我的模样就形成了。”但这丝毫也影响不了林纳斯对整个商业社会的巨大价值—Linux 代表着网络时代新形式的开放知识产权形态，这将从根本上颠覆以 Windows 为代表的封闭式软件产权的传统商业模式。更重要的是，这样的颠覆早已悄悄的出现在了商业社会的各个角落。

Linux

颠覆世界的“自由主义教皇”林纳斯。

“有些人生来就具有统率百万人的领袖风范；另一些人则是为写出颠覆世界的软件而生。唯一一个能同时做到这两者的人，就是托瓦兹。”美国《时代》周刊对“Linux 之父”林纳斯·托瓦兹 (Linus Torvalds) 给出了极高的评价。甚至，在《时代》周刊根据读者投票评选出的二十世纪 100 位最重要人物中，林纳斯居然排到了第 15 位，而从 20 世纪的最后几年就开始霸占全球首富称号的盖茨不过才是第 17 位。

林纳斯的出名和“重要”来得并没什么先兆，尽管这个 1969 年出生在芬兰赫尔辛基的天才在年少时就已经颇具黑客神韵---对于电脑的着迷使他很早就能够驱使电脑做事情，对一切细节也都能控制自如。但当林纳斯在 1991 年就读于赫尔辛基大学期间刚刚开始对 Unix 产生浓厚兴趣，尝试着在 Minix (Unix 的变种) 上做一些开发工作的时候，他从来也没想过会构建出一个新操作系统的内核来。

Linux 的诞生显得充满了偶然。林纳斯经常要用他的终端仿真器 (Terminal Emulator) 去访问大学主机上的新闻组和邮件，为了方便读写和下载文件，他自己编写了磁盘驱动程序和文件系统，这些在后来成为了 Linux 第一个内核的雏形。当时，他年仅 21 岁。

在自由软件之父理查德·斯托曼 (Richard Stallman) 某些精神的感召下，林纳斯很快以 Linux 的名字把这款类 Unix 的操作系统加入到了自由软件基金 (FSF) 的 GNU 计划中，并通过 GPL 的通用性授权，允许用户销售、拷贝并且改动程序，但你必须将同样的自由传递下去，而且必须免费公开你修改后的代码。这说明，Linux 并不是被刻意创造的，它完全是日积月累的结果，是经验、创意和一小段一小段代码的集合体。

无疑，正是林纳斯的这一举措带给了 Linux 和他自己巨大的成功和极高的声誉。短短几年间，在 Linux 身边已经聚集了成千上万的狂热分子，大家不计得失的为 Linux 增补、修改，并随之将开源运动的自由主义精神传扬下去，人们几乎

像看待神明一样对林纳斯顶礼膜拜。

然而，在 1996 年底，林纳斯突然离开赫尔辛基，只身奔赴美国硅谷，成为 Transmeta 公司的一员，直到 2003 年才离开。其中的原因曾经扑朔迷离，但现在看来，那只不过是林纳斯对于自身价值的一种现实性追求。虽然当时许多人都怀疑这会给发展中的 Linux 造成致命伤害，不过 Linux 在随后几年内的发展证明，这样的担心是多余的。

林纳斯说，Linux 所取得的许多成功其实可以归结为他的缺点所致。“我很懒散，我喜欢授权给其他人。”就连 Linux 的企鹅形象标识也是林纳斯的妻子，曾获芬兰空手道冠军的托芙（Tove）想到的，因为林纳斯曾经在澳大利亚被一只企鹅咬过。“黑客们，不，程序员们，把在 Linux 和其它开放源代码项目上的工作，放在比睡觉、锻炼身体、娱乐和聚会更优先的地位。因为他们乐于成为一个全球协作努力活动的一部分—Linux 是世界上最大的协作项目。”

1.3 Unix 哲学

参考：

- http://s3.eurecom.fr/~balzarot/softdev/material/0_2_unix_philosophy.pdf

因为 Linux 是类 Unix 操作系统，这里摘抄一些 Unix 哲学相关的内容，方便大家对 Linux 有个浅显的认识。

1.3.1 Unix 哲学核心

英语原文，

This is the UNIX philosophy:

- Write programs that do **one thing** and do **it well**.
- Write programs to **work together**.
- Write programs to handle **text streams**, because that is a universal interface.

— Doug McIlroy

汉语译文，

这是 UNIX 哲学：

- 程序，应当只做一件事，且要做好。

- 程序，应当能够**协作**。
- 程序，应当能够处理**文本流**，因为文本流是通用接口。

— Doug McIlroy

我觉得这种哲学思想都快适合做其他事情了，分而治之，统筹协调，打好基础。对于打算从事 Unix 编程的，还可以继续深入学习，这里就不多介绍了。

1.3.2 Unix 哲学下的用户

Unix 相信用户，并赋予其极多选择极大权力，但也要求用户必须花足够多的时间来学习。

但是，“只要是个人，就会犯错误的”，这种 Unix 绝对信任也有极大的危害。网上仅仅 `rm -rf ./` 误删就害死了不少人。而且要花费非常多的时间来学习一门操作系统，也是一个极大的弊端。正是这种种缺陷，催生了各种发行版，后面[1.4节](#)再说。

1.3.3 Unix 哲学的对手

以当前桌面领域普及最广的 Windows 系统为对照，看看 Windows 和 Unix 的区别。

Unix 是以科学家和程序员为用户群体的，但是类 Unix 的 Linux 的用户群体也开始包含普通吃瓜群众了。而 Windows 是以所有人，当然包括那些普通群众，而且最主要的就是普通群众，为用户群体的。基于不同的目标用户，Unix 要提供各种可能性，让目标用户，“来玩坏自己，实现各种功能，尽情的发挥自己的聪明才智”，相当于给用户提供了大量积木块；Windows 则要提供傻瓜式服务，防备着你玩坏了，对你各种猜测，引导你去完成特定的任务，以免犯错，相当于给用户提供了一个个成品玩具。用参考文章的一句话来说“Unix 的使用依赖用户的聪明才智，Windows 则隐藏这些技巧于操作系统和软件内部”。

下面来分析一下，这两者的优缺点。

- Windows 的优点，降低用户的学习时间成本；缺点，倡导傻瓜式用户，限制用户的灵活发挥。
- Unix 的优点，提供更多的可能性；缺点，增大用户的学习时间成本。

当然，随着 GNU/Linux 发行版的活跃发布，Unix 哲学的一些缺点也在被慢慢修正。也许，未来的 GNU/Linux 发行版既能提供易操作的成品玩具，也附带大量灵活的积木零件，综合两者的优势，更方便也更灵活更高效。

1.4 GNU/Linux 发行版的出现

如果完全从 Unix 哲学起步，用源代码一步步编译为可用软件，对广大普通用户来说就太难了，GNU/Linux 也将举步维艰。以 Linux 为内核，外加 GNU 的一些软件和服务，添上图形界面等，满足用户的基本需求，这样发行版的出现，能够很好的解决上述难题，也就流行开来了。GNU/Linux 发行版的出现，也是对 Unix 哲学的一种实践补充。所以各发行版的目标和口号和 Unix 哲学还是有一定的区别，比如深度公司的口号“免除新手痛苦、节约老手时间”，就不会说，“单一高效”。

发行版的历史就不说了，当前常见的发行版有这两大类，Red Hat 系列和 Debian 系列，当然还有 slackware 以及其他有特殊目的的发行版等等，这里略过他们。

1.4.1 Red Hat 系

Red Hat 系列发行版是一个非常流行的 GNU/Linux 发行版，属于 Red Hat 公司，代码开源，通过售卖服务以及培训等收费，起步非常早。采用非常好用的 rpm 软件包管理器，系统非常稳定，安装简单，有图形化操作界面。该公司对近期比较热门的领域，比如云计算等也有涉猎。旗下主要有企业稳定版的 Red Hat Enterprise Linux 和功能实验版 Fedora 两类操作系统。另外有社区企业版 centos 发行版是完全复制 Red Hat Enterprise Linux 源码，并剔除 Red Hat 公司标志后编译成的发行版，稳定可靠，也值得了解一下。

1.4.2 Debian 系

Debian 也是一个早期的 GNU/Linux 发行版，有非常好用的软件包管理器，软件包为 deb 格式，系统稳定性非常高。有众多的发行版基于或者追溯到 Debian，比如本书介绍的对象 Deepin 深度操作系统，以及比较著名的 Ubuntu，国际排名非常靠前的 Linux Mint，网络安全必备的 Kali Linux，老旧电脑挚爱的 Lubuntu 等发行版。本书介绍的很多技术均适用于 Debian 系列的发行版。

1.5 如何提高自己的 Linux 技术

参考网页：

- <https://www.tecmint.com/free-online-linux-learning-guide-for-beginners/>
- <https://www.cnblogs.com/evilqliang/p/6247496.html>
- <https://blog.csdn.net/erlian1992/article/details/54586156>

最好的办法就是直接上手！古人云“临渊羡鱼，不如退而结网”。下载一个 Linux 发行版，安装在自己的电脑上，平常多用，偶尔折腾一下，就差不多了。

1.5.1 首先安装 GNU/Linux 操作系统

当然这里肯定首先推荐 Deepin 深度操作系统，比较符合中国人的操作习惯。另外也可选择 Ubuntu 的一些分支版本，其中优麒麟就是做得比较好的一个发行版。如果电脑配置不够高，或者比较老旧，也可用选择轻量版的 Lubuntu。电脑配置比较高的，也可用试着在虚拟机，比如 VirtualBox 虚拟机内安装上述这些操作系统。

1.5.2 平常多用

GNU/Linux 的部分发行版在某些情况下，已经能够满足我们的日常工作生活需要。在正常使用不需要太折腾的前提下，我们完全可以多用、经常用。

欧阳修《卖油翁》提到“无他，惟手熟尔。”Linux 技术也是“学而时习之，不亦说乎”的事情，用多了，自然知道咋回事啦。就像“熟读唐诗三百首，不会作诗也会诌”。使用熟练后，我们就会发现 Linux 也是可以提高办公效率的好工具。

1.5.3 多问多记

很喜欢毛泽东的读书方法“不动笔墨不看书”，遇到问题及时去查资料，然后记录下来，很有助于技术水平的提高。笔者曾在某公司实习，不得已整天跟 vim 打交道，刚入门也是效率低下，后来在身边好友帮助下以及网上搜索资料，慢慢写博客，也开始有点爱上 vim 了，半年下来，vim 码字母也能贼溜了。学习就是一个慢慢积累的过程，谁也不能一口吃成大胖子。曾经有一位日本人说“赵先生，中国有五千年历史呢，咱们不急，慢慢来”。慢慢来不等于不来，我们需要时刻督促自己不停的学习，“活到老学到老”。

1.5.4 定个计划

当你有一定认识后，你就会自觉不自觉的制定计划了，也会掌握更多的学习方法，这里我就不废话了。上面参考网页都是一些不错的经验介绍，希望对您有所帮助。

1.6 总结

本文首先根据用户角色，介绍了 GNU/Linux 操作系统的用途范围。通过介绍斯托曼和林纳斯两人的经历，附带讲解了 GNU 和 Linux 的发展历史，以及 GPL 协议的一些内容。接着介绍了 Unix 哲学相关的内容，最后对如何学习 GNU/Linux 提出了一些个人观点。

第二章 Linux 桌面系统简介

通过阅读本章，你将会了解到以下几项内容。

- 从 Live CD/DVD 运行系统
- X Window 系统
- 新秀 Wayland 的评价
- 常见的桌面环境（Desktop Environments）简介

随着技术的发展，电脑桌面操作系统也在不断的完善，虽然不能说是夕阳产业，但也在趋向一个完美的天花板。个人认为很多在 windows 和苹果桌面操作系统下的软件和操作体验，在 Linux 桌面这边一定会慢慢实现并赶上来的。未来软件会趋同，会跨平台。目前为止，Linux 桌面系统下已经有足够多的软件满足一般的日常生活需求了。

2.1 运行 Live CD/DVD 试用系统

本文一直强调“能动手的就不要光动口”，意思就是最好安装 GNU/Linux 操作系统。如果您不喜欢安装，也可用在不安装的前提下试用该系统。下面介绍如何试用系统。当然下载安装镜像还是不可避免的。

2.1.1 下载深度系统

下载深度操作系统系统最新版本的镜像文件（以便您能够体验到最新特性），
下载地址：<https://www.deepin.org/download/> 如图2-1所示，

如果您只是想试用系统，可以选择下载 **Live 系统**。

注意：为了更加专注系统的发展，deepin 15.4 版本将不再提供 32 位官方 iso 镜像，如需获取和技术支持，请发送邮件到 support@deepin.org。

2.1.2 把系统写到 U 盘里

安装说明：<https://www.deepin.org/installation/>

现在流行 U 盘安装，当然也可用光盘安装。下面只介绍 U 盘启动器的制作，
到[这里](#)下载系统启动盘制作工具。根据自己操作系统情况请选择对应版本。另外
也可用选择其他的制作软件，比如 Ultraiso 软件，网上搜索“使用 UltraISO（软碟通）制作 U 盘启动盘完整教程”，可以找到相当多的资料，这里从略。

深度操作系统15.5



图 2-1 如果只是试用请选择 Live 系统

2.1.3 开始试用

试用版有一些功能受限，但是可以感受一下。不过还是强烈建议安装正常版本的软件。

参考：

- <https://bbs.deepin.org/forum.php?mod=viewthread&tid=157091>
- <http://wiki.deepin.org/wiki/%E4%BD%93%E9%AA%8C%E5%AE%89%E8%A3%85>

2.2 X Window 系统简介

当前主要的 linux 桌面发行版基本都是采用 X Window 系统来提供桌面服务。

X Window 是一种以位图方式显示的软件窗口系统，最初是 1984 年麻省理工学院的研究成果，之后变成 UNIX、类 UNIX、以及 OpenVMS 等操作系统所一致适用的标准化软件工具包及显示架构的运作协议。

X Window 通过软件工具及架构协议来建立操作系统所用的图形用户界面，此后则逐渐扩展适用到各形各色的其他操作系统上，几乎所有的操作系统都能支持与使用 X Window，GNOME 和 KDE 也都是以 X Window 为基础建构成的。

X Window 向用户提供基本的窗口功能支持，而显示窗口的内容、模式等可由用户自行定制，在用户定制与管理 X Window 系统时，需要使用窗口管理程序，窗

口管理程序包括 AfterStep、Enlightenment、Fvwm、MWM 和 TWM Window Maker 等，供习惯不同的用户选用。可以定制的窗口环境在给用户带来了个性化与灵活性的同时，要求用户有相对比较高的使用水平，不过这种机制带来的好处也是明显的，它不象 Microsoft Window 那样将窗口元件的风格、桌面、操作方式等千篇一律地规定死，只可以换一下墙纸、图标、调整字体大小等等，在 X Window 系统中可以有多种桌面环境的选择。

2.2.1 基本部件

X 系统由三个相关的部分组成：服务端（Server）、客户端（Client）和通讯通道（communication channel）。

服务端（Server）

Server 是控制显示器和输入设备（键盘和鼠标）的软件。Server 可以创建视窗，在视窗中绘图和文字，回应 Client 程序的“需求”（requests），但它不会自己完成，只有在 Client 程序提出需求后才完成动作。

每一套显示设备只对应唯一的 Server，而 Server 一般由系统供应商提供，通常无法被用户修改。对操作系统而言，Server 只是一个普通的用户程序而已，因此很容易更换新版本，甚至更换成第三方提供的原始程序。

客户端（Client）

Client 是使用系统视窗功能的一些应用程序。在 X 下的应用程序称做 Client，原因是它是 Server 的客户，要求 Server 回应它的请求完成特定动作。

Client 无法直接影响视窗行为或显示效果，它们只能送一个请求（request）给 Server，由 Server 来完成这些的请求。典型的请求通常是“在某个视窗中写‘Hello World’的字符串”，或者从 A 到 B 划一条直线。

Client 的功能大致可分为两部分：向 Server 发出“需求”只是它的一部分功能，其他的功能是为用户执行程序而准备的。例如输入文字信息、作图、计算等等。通常，Client 程序的这一部分是和 X 独立的，它对于 X 几乎不需要知道什么。通常，应用程序（特别是只大型的标准绘图软件、统计软件等）对许多输出设备具有输出的能力，而在 X 视窗中的显示只是 Client 程序许多输出中的一种，所以，Client 程序中和 X 相关的部分只占整个程序中很小的一部分。

用户可以通过不同的途径使用 Client 程序：通过系统提供的程序使用；通过第三方的软件使用；或者用户为了某种特殊应用而自己编写的 Client 程序来使用。

通讯通道（Communication channel）

Client 藉著它送“需求”给 Server，而 Server 藉著它回送状态（status）及一些

其它的资讯 (information)。

只要 Client 和 Server 都知道如何使用通道，通道的本身并不是很重要，在系统或网路上支援通讯型态的需求是内建於系统基本的 X 视窗函数馆 (library)，所有和通讯型态有关的事都从函数馆独立出来，Client 和 Server 之间的通讯只要藉著使用这函数馆 (在标准 X 版为 xlib)。

Server 和 Client 之间的通信

Server 和 Client 通信的方式大致有两类，对应于 X 系统的两种基本操作模式。

第一类，Server 和 Client 在同一台机器上执行，它们可以共同使用机器上任何可用的通信方式做互动式信息处理。在这种模式下，X 可以同其他传统的视窗系统一样高效工作。

第二类，Client 在一台机器上运行，而显示器和 Server 则在另一台机器上运行。因此两者的信息交换就必须通过彼此都遵守的网络协议进行，最常用的协议为 TCP/IP。这种通信方式一般被称为网络透明性，这也几乎是 X 独一无二的特性。

2.2.2 用户接口

X 的设计目标之一就是能创建许多不同形式的用户接口。其他的是视窗系统提供具体的交互方式，而 X 只提供一般的架构，让系统创建者建造所需的交互风格。这种特性使得开发者可以在 X 的基础上建造全新的接口，并且可以在任何时候根据自己的需要选用适当的接口。

一般来说，用户接口可以分为两部分。管理接口也就是视窗管理器，是命令的最高层，它负责在屏幕上创建或重建视窗，改变视窗的大小、位置，或者将视窗改变成图示等。

应用接口确定了用户和应用程序之间的交互风格，即用户如何利用视窗系统的设备程序来控制应用程序并传递输入行为。例如，如何用鼠标来选定一个选项。

2.2.3 用户界面

X 刻意不去规范应用程序在用户界面上的具体细节设计，这些包括按钮、菜单和窗口的标题栏等等，这些都由窗口管理器 (window manager)、GUI 构件工具包、桌面环境 (desktop environment) 或者应用程序指定的 GUI (如 POS) 等等的用户软件来提供，然而因为架构设计上保留了高度的弹性发挥空间，致使多年来 X 在“基础、典型、一般性”的用户界面上，也都有数目惊人的多样性选择。

在 X 的系统架构中，窗口管理器用于控制窗口程序的位置和外观，其界面类似 Microsoft 的 Windows 或者 Macintosh (例如：KDE 的 KWin 或者 GNOME 的

Metacity), 不过在控制机制上却截然不同 (如: X 提供的基本窗口管理器 twm)。窗口管理器可能只是个框架 (如: twm), 但也可能提供了全套的桌面环境功能 (如: Enlightenment)。

虽然不同的 X 用户界面可以有很大的差异, 然而绝大多数的用户在使用 X 时, 多是用已经打包的桌面环境, 这种桌面环境不仅包含窗口管理器, 还具备各种应用程序以及风格一致的界面, 目前最流行的桌面环境是 GNOME 和 KDE, 二者已普遍应用于 Linux 操作系统上, 而 UNIX 所用的标准桌面环境多是通用桌面环境 CDE, 然而有些 UNIX 也开始采用 GNOME。

此外, X 桌面环境及组件虽然很多, 但同时也需要保持兼容性与互通性, 这些由 freedesktop.org 积极与努力地维持各种不同 X 桌面环境的兼容性, 使竞争态势下仍不失 X 的兼容本色。

2.2.4 优缺点

优点

(1) 任务分工明确

客户端可以在远程电脑上执行计算任务, 而 X Server 仅负责复杂的图形显示, 充分发挥 X Server 在显示上的优势。

只有 X Server 服务端与硬件打交道, 所有的客户端都与硬件无关, 这让不同的平台上的移植变得很容易。客户端可以在不同的电脑上运行, 从巨型机到个人电脑, 从而充分发挥网络计算的优越性。

X 系统只负责显示图形, 并不限制显示和操作的风格, 因此不同的 X Window 的风格并不相同, 用户可以根据自己的喜好进行选择。由于 X Window 系统只提供了最基本的系统调用, 而具体的视窗都有很多共性, 因此要开发 X 应用程序, 应该首先使用开发工具包, 而没有必要直接使用最基本的 X Window 的系统调用, 以简化编写程序的工作量。

(2) 独立于操作系统

X 不是内置于操作系统, 它只是比用户层次稍高一些。在系统中也是一个相对独立的元件。这样做有如下优点:

- 易于安装和改版, 甚至去除。这种工作不需要重启系统, 也不会对其他应用程序造成干扰。
- 第三方很容易支持并加强它的功能。
- X 不会制定操作系统, 只是一个标准, 这也是第三方开发软件的原动力。
- 为了开发者。在 Server 上进行工作时, 如果程序异常中断, 只会影响到视

窗系统，不会造成机器的损坏或操作系统内核的破坏。

缺点

(1) 稳定性不强

在 PC 世界里，需要第三方硬件驱动的非主流的个人计算机软件往往不能支持那些专为 Microsoft Windows 出品的设备。X Window 也不例外，它的实现往往缺少较新（或者非常旧）的高效的显卡驱动。X Window 的 C/S 体系（C/S 架构）设计在应用程序和显示硬件之间多加了一层软件，导致绘图效率下降，所以引起了一些批评。X 也被批评提供了过多的对硬件的直接访问，从而影响了系统的稳定性。

(2) 不规范的用户界面

X 刻意不去规范用户界面和程序之间大多数的通信，导致出现了许多非常不同的界面，同时造成程序之间协同的困难；而客户机之间的互操作规范 ICCCM 以难以正确实现而闻名。后来的标准化尝试，也于事无补。长久以来这已经成为用户和程序员的噩梦。直到最近，X 也没有好地解决显示与打印机所打印的内容一致性（所见即所得）的解决方案。

(3) 网络通信能力弱

X 不能像 VNC 那样把客户端的通信从一个服务端卸下然后再附加到另一个上，但现在正在为 X 增加此功能的工作，另外还要实现通过 VNC 实现 X Server 的显示。X Server 和远端客户机之间的网络通信使用明文的缺陷，让攻击者使用封包截取程序就能够截获和阅读它。这一缺陷一般可以通过在 SSH 通信上使用 X 来解决。设备中立和客户端与服务端的分离还带来了一定性能开销。

参考：

- <https://baike.baidu.com/item/X%20Window/7249336?fr=aladdin>

2.3 王勇谈 Wayland

本篇大部分为转载，有改动。

Wayland 是一个简单的“显示服务器”（Display Server），与 X Window 属于同一级的事物，而不是仅仅作为 X Window 下 X Server 的替代（注：X Window 下分 X Server 和 X Client）。也就是说，Wayland 不仅是要完全取代 X Window，而且它将颠覆 Linux 桌面上 X Client/X Server 的概念，以后将没有所谓的“X Client”了，而是“Wayland Client”。更确切的说，Wayland 只是一个协议（Protocol），就像 X Window 当前的协议——X11 一样，它只定义了如何与内核通讯、如何与 Client 通讯，具体的策略，依然是交给开发者自己。所以 Wayland 依然是贯彻“提供机

制，而非策略”的 Unix 程序。

Wayland 相对于古老的 X11 来说最大的提升是，Gtk/Qt 这些图形库进行图形绘制时，不用像 X11 那样发送绘制消息到 XServer 来进行绘制，而是由 Client 自己进行图形绘制，Wayland 只用担任图层混合器的作用。这样不但减少了 X Client 和 X Server 之间不必要的通讯，而且因为由 Client 自己进行渲染，所以很多画面撕裂和闪屏的现象从原理上就避免了。

大家可以看一下下面的两张架构图，图2-2和图2-3，来理解两者的差别：

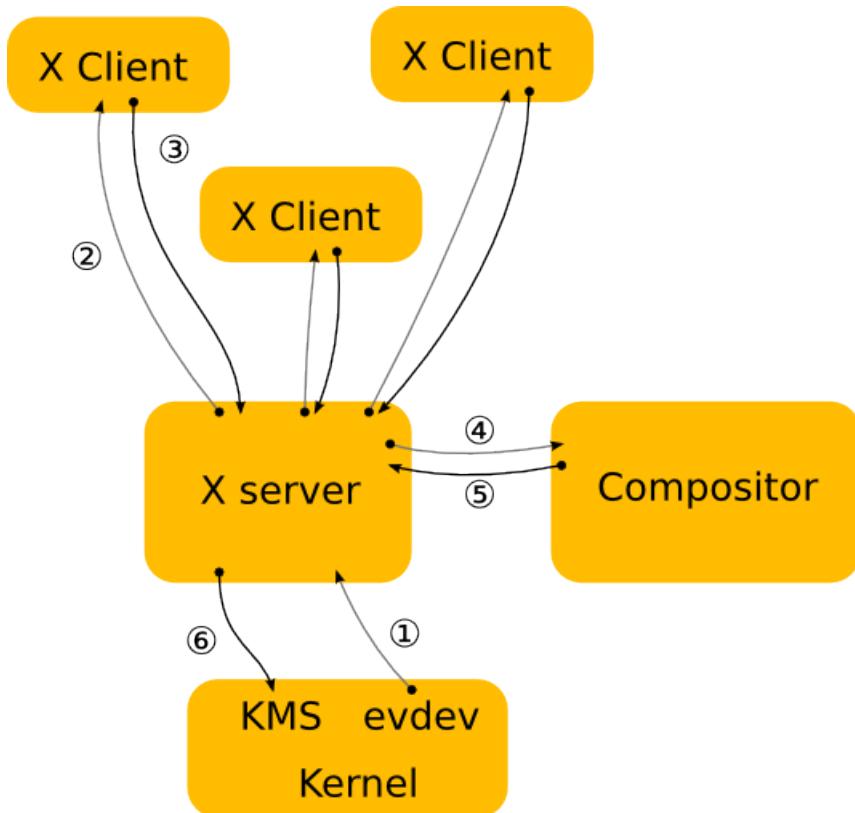


图 2-2 X 架构图

Wayland 因为要彻底从技术架构上颠覆 Linux 几十年 X11 的渲染方式，不论从架构设计还是代码实现上都会非常复杂，不但要开发协议本身，还需要开发适合 Wayland 的混合器和窗口管理器，最后导致 Wayland 1.0 稳定版一再跳票。

而 Ubuntu 在独立开发 Unity 桌面环境的同时，也仿造了 Wayland 的架构开发了自己的 Mir 显示服务器，除了等不及 Wayland 稳定之外，更重要的是 Ubuntu 要为了它的下一步宏伟计划 “Ubuntu Touch”，按照 Mark 的设想，Mir 不仅仅要像 Wayland 那样从原理上提升 Linux 图形渲染效率，而且 Mir 还得担负起手机和电脑融合的使命，可以让 Ubuntu Touch 的手机在插上显示器底座时，手机的应用通

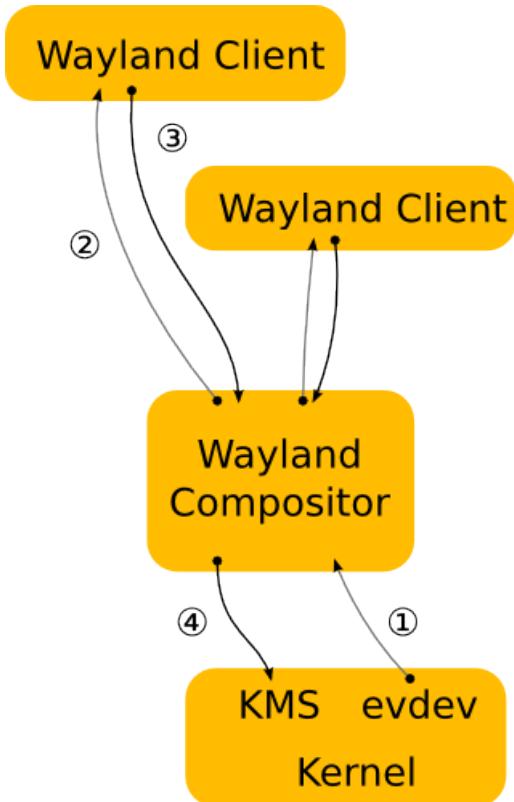


图 2-3 Wayland 架构图

过 Mir 的支持，可以直接在外接显示器上显示手机应用窗口，最终达到“当你手机放到底座时就是电脑，拿走就是手机”的设想。

不论 Wayland 还是 Mir，虽然底层架构都非常先进，但是为了兼容现有的 X11 程序，它们分别开发了 XWayland 和 XMir 用于在新的显示服务器协议上支持现有的大多数 X11 程序（主要是 Gtk2/Qt3/Qt4 开发的大多数应用）。

具有讽刺意味的是，Wayland 和 Mir 本来就是要解决 X11 那种不适合现代 PC 场景繁琐的通讯协议，甚至很多开发者为了技术的洁癖都在大力安利 Wayland/Mir，但是最后 XWayland 和 XMir 本身的兼容实现却比 X11 的实现更加“恶心”和繁琐，包括 Mir 的开发者最后都放弃 XMir 的开发。

正是因为 Wayland/Mir 这样的技术无法彻底解决和大部分原本就基于 X11 协议而开发的应用的兼容性问题，最后导致基于 Wayland/Mir 开发的桌面系统从“解决渲染性能问题”转变到“无法运行很多现有 Linux 应用”这一个更加让用户难以接受的结果。这也是后面基于 Mir 开发的新版 Unity 难产的重要原因。

参考：

- <https://www.jianshu.com/u/E6EbkP>

- <https://www.jianshu.com/p/86dd6e34ce91>
- <https://baike.baidu.com/item/Wayland/7429696?fr=aladdin>

2.4 常见的桌面环境

桌面环境是为了方便用户操作电脑的一系列工具的集合。常常包括以下几个或者全部的组件。

- 窗口管理器 (Window manager)
- 面板 (Panels)
- 菜单 (Menus)
- 小工具 (Widgets)
- 文件管理器 (File Manager)
- 浏览器 (Browser)
- 办公套件 (Office Suite)
- 文本编辑器 (Text Editor)
- 终端 (Terminal)
- 显示管理器 (Display Manager)

Linux 桌面环境有很多种，比如 Cinnamon, Unity, GNOME, KDE, XFCE, LXDE, MATE, Enlightenment, Pantheon, 以及后面详细介绍的深度桌面 DDE(Deepin Desktop Environment) 等。

2.4.1 Cinnamon

图2-4, Cinnamon 桌面环境很现代很时尚，界面类似 Win 7，是 Linux Mate 的默认桌面环境，托 Linux Mate 的福，也非常流行。

此桌面环境，底部有一个面板，在右下角有一个带有快速启动图标和系统托盘的时尚菜单，有一系列的键盘快捷方式方便操作，视觉效果也很棒。同时也有很好的定制功能，比如更改壁纸，添加和位置面板，添加小程序到面板，也可以添加桌面提供新闻，天气和其他关键信息等。

- 内存使用情况: $\approx 175\text{MB}$
- 优点:
 - 仿 Windows 的操作习惯。
 - 好看、特效多、小部件多
 - 键盘快捷键功能强大
 - 可定制

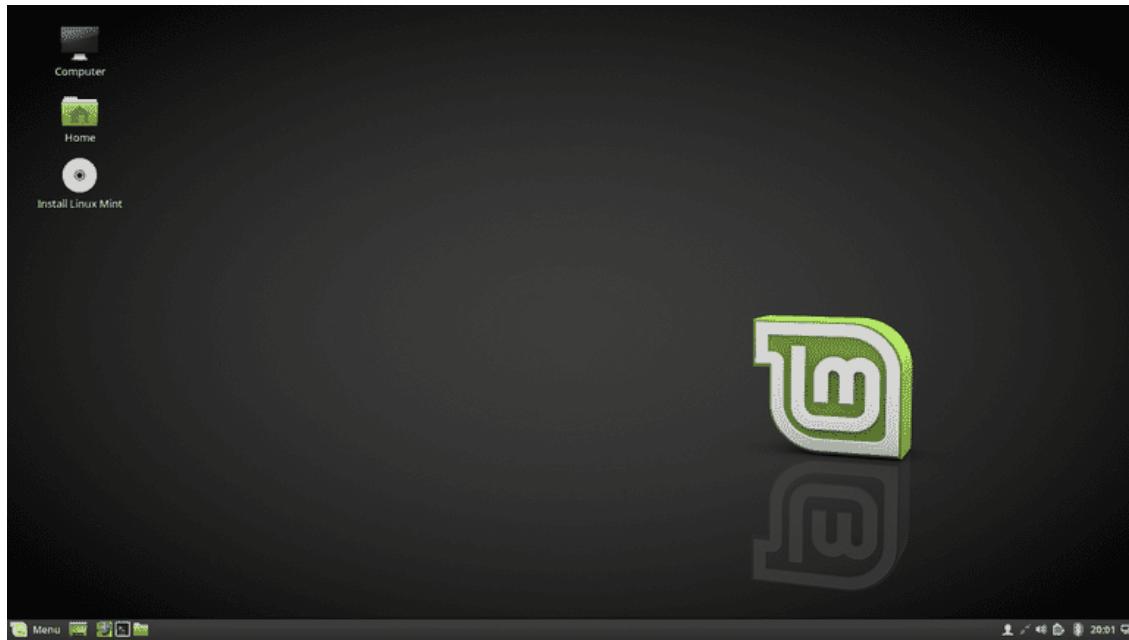


图 2-4 Cinnamon 桌面环境样图

- 缺点：
 - 与其他桌面相比太耗内存
 - 可定制的功能还不够强大

2.4.2 GNOME

图2-5，GNOME 桌面环境，包括 GNOME2 和 GNOME3，GNU 计划的一部分，开放源码运动的一个重要组成部分，是一种让使用者容易操作和设定电脑环境的工具，目标是基于自由软件，为 Unix 或者类 Unix 操作系统构造一个功能完善、操作简单以及界面友好的桌面环境，是 GNU 计划的正式桌面环境。

- 内存使用情况：≈ 250MB
- 优点：
 - 现代的
 - 有大量的核心应用程序和开发工具包，使得开发人员易于创建丰富的应用程序。
 - 功能强大的键盘快捷键
 - 搜索过滤功能强悍
- 缺点：



图 2-5 GNOME 桌面环境样图

- 太耗内存
- 可定制性极差

2.4.3 KDE

图2-6，KDE 桌面环境类似 Cinnamon，仿 Win 7。底部有一个面板，菜单，快速启动栏和系统托盘图标。可以在桌面上添加新闻和天气等小部件。默认安装有大量的应用程序。

- 内存使用情况：≈ 300MB
- 优点：
 - 仿 Windows 的操作习惯
 - 提供大量的默认应用程序，包括 Web 浏览器和邮件客户端。
 - 很多小部件
 - 可定制能力强
- 缺点：
 - 太耗内存

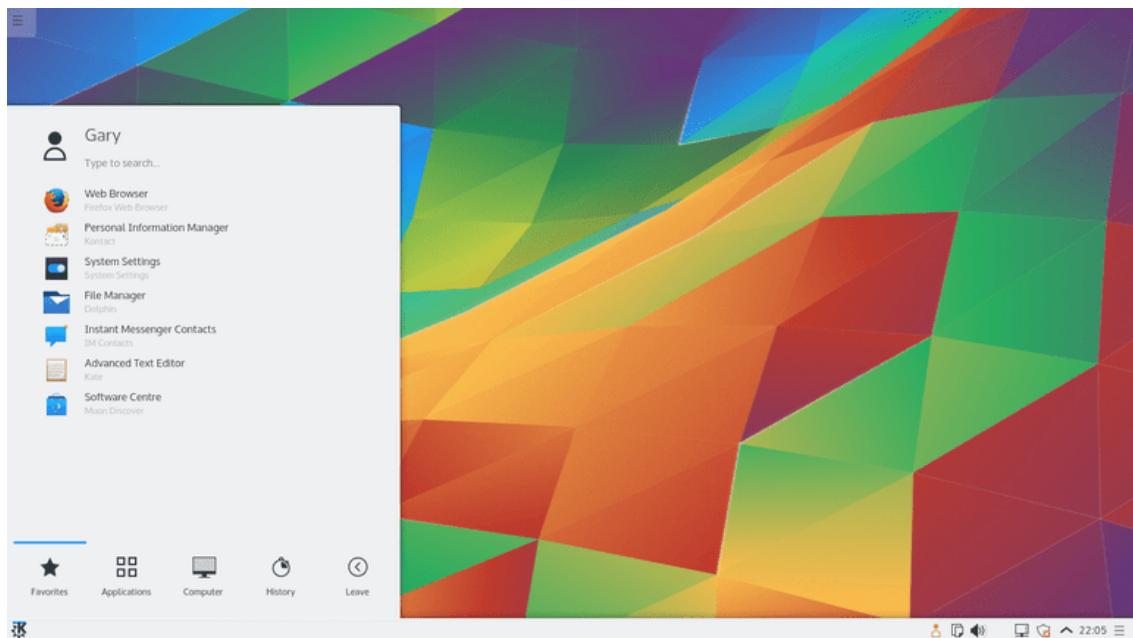


图 2-6 KDE 桌面环境样图

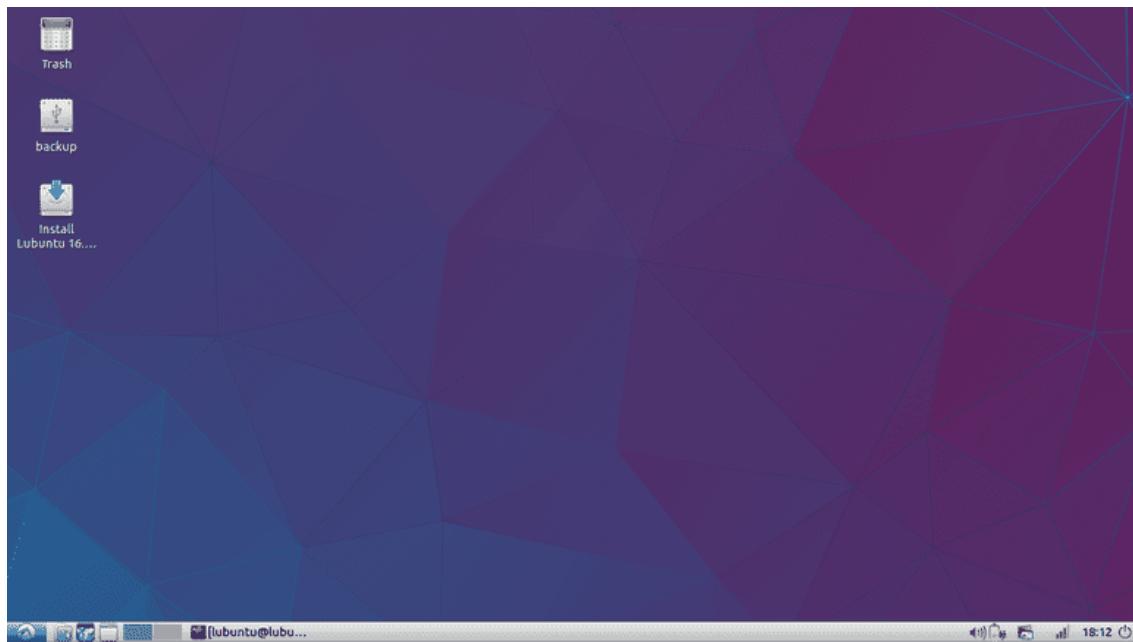


图 2-7 LXDE 桌面环境样图

2.4.4 LXDE

图2-7，LXDE 桌面环境非常适合旧计算机。能提供一些非常基础的功能。

- 内存使用情况：≈ 85MB
- 优点：
 - 非常小巧，适合旧电脑和配置较差的计算机
 - 强大的定制能力
- 缺点：
 - 默认应用程序较少
 - 可用应用程序相比其他桌面环境体验感觉不够好

参考：

- <https://wiki.deepin.org/wiki/Linux%E6%A1%8C%E9%9D%A2%E6%BC%94%E8%BF%9B>
- <https://www.deepin.org/developer-community/architectural-design/>
- <https://itsfoss.com/best-linux-desktop-environments/>
- <https://www.lifewire.com/best-linux-desktop-environments-4120912>
- <https://www.tecmint.com/best-linux-desktop-environments/>

2.5 总结

本章介绍了如何在不安装系统的情况下，试用 GNU/Linux 系统。然后介绍了 X Window 系统的概况，同时附带介绍了当前新技术 Wayland 的状况。最后列举了常见的几种桌面环境。

第三章 深度操作系统简介

本章打算重构，安装操作系统后移，把系统结构转移到这里来。通过阅读本章，你将会了解到以下几项内容。

- 武汉深之度科技有限公司简介
- 深度操作系统的安装
- 深度桌面使用
- 深度桌面常用软件

摘自：[深度商业官网](#)、[深度社区官网](#)及[深度百科](#)。已获许可。

3.1 深度科技简介

武汉深之度科技有限公司

武汉深之度科技有限公司（以下简称深度科技）成立于 2011 年，是专注基于 Linux 的国产操作系统研发与服务的商业公司。

作为国内顶尖的操作系统研发团队，深度科技以提供安全可靠、美观易用的国产操作系统与开源解决方案为目标，拥有操作系统研发、行业定制、国际化、迁移和适配、交互设计、支持服务与培训等多方面专业人才，能够满足不同用户和应用场景对操作系统产品的广泛需求。



图 3-1 武汉深之度科技有限公司标志

深度科技作为国产操作系统生态的打造者，不但与各芯片、整机、中间件、数据库等厂商结成了紧密合作关系，还与 360、金山、网易、搜狗等企业联合开发了

多款符合中国用户需求的应用软件。深度科技的操作系统产品，已通过了公安部安全操作系统认证、工信部国产操作系统适配认证、入围国管局中央集中采购名录，并在国内党政军、金融、运营商、教育等客户中得到了广泛应用。

截止到 2015 年，深度操作系统下载超过 4000 万次，提供 30 种不同的语言版本，以及遍布六大洲的 70 多个镜像站点的升级服务。在全球开源操作系统排行榜上，深度操作系统长期保持前 20 名，也是排名最高的中国操作系统产品。

未来，深度科技将继续秉承开源和创新精神，以操作系统的自主可控替代为契机，争取在三年内成为中国市场主要操作系统供应商，五年后成为具备国际影响力的主要操作系统厂商。

深度操作系统项目

深度操作系统是一个致力于为全球用户提供美观易用、安全可靠体验的 Linux 发行版。它不仅仅对最优秀的开源产品进行集成和配置，还基于 Qt 技术开发了深度桌面环境和深度控制中心，并且开发了一系列面向日常使用的深度特色应用如深度商店、深度截图、深度音乐、深度影院等。深度操作系统非常注重易用的体验和美观的设计，对于大多数用户来说，它易于安装和使用，能够很好的代替 Windows 系统进行工作与娱乐。

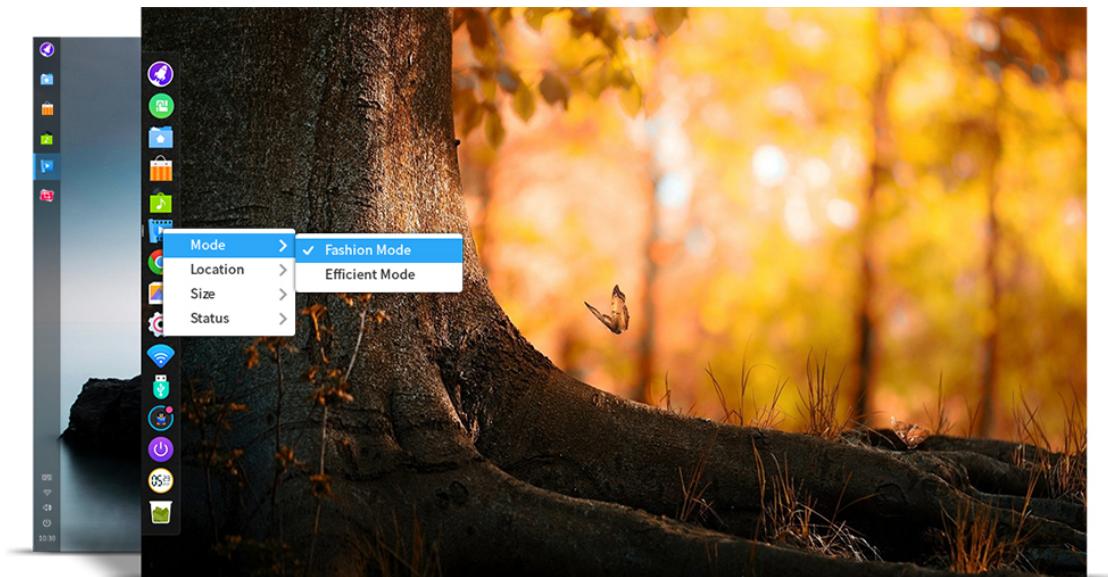


图 3-2 深度操作系统桌面

深度操作系统的历程可以追溯到 2004 年，其前身 Hiweed Linux 是中国第一个基于 Debian 的本地化版本。在 2008 年更名为深度操作系统，2011 年获得商业投

资，已经建立起国内少有的拥有员工数十人专注于桌面 Linux 发行版的团队。深度操作系统与 Sogou, WPS, Maxthon 等合作伙伴进行了多方位合作，共同打造基于 Linux 系统的生态系统。同时，我们还在努力解决迁移 Windows 平台软件带来的各种兼容性问题，以便用户平滑的过渡到开放安全的 Linux 平台上。

对个人用户来说，使用深度操作系统不但完全免除了购买费用，而且不会被流行的各种病毒和木马感染。对于政府和企业用户，由于源代码开放可控，绝无后门隐患，安全性也更有保障，大大降低了维护代价和购买防护软件的费用。

近年来，深度操作系统发展迅速，获得全球 40 多个国家用户的 support，累计下载量数千万次，并成为在 Distrowatch 上排名最高的中国 Linux 操作系统发行版。

非常欢迎您尝试深度操作系统作为您工作和生活的伙伴，作为中国鲜少地专注于桌面操作系统的团队，我们相信它将给您带来完全不同的体验！

注：深度论坛网友 [licardo](#) 建议，“国产操作系统”改为“国内发行版”。深以为然。但是考虑到转帖的别人的文章，就保留不变了，特此说明。

3.2 深度操作系统的安装

3.2.1 安装方案

全新安装 DEEPIN 单系统

所谓安装 deepin 单系统，即计算机上不保留其他操作系统，并且使用单独的分区格式化后安装 deepin。

准备工作：

- 如果是全新的电脑，或者硬盘中的文件数据均已备份无需保留，则直接使用光盘或优盘启动电脑进入安装操作即可。
- 如果电脑中已有文件数据，则可以在现有系统（如 Windows）下将文件移动或备份，留出至少一个 20G 的空白分区；或者使用磁盘工具（推荐分区助手，[下载地址](#)）并选择一个剩余空间合适的分区进行大小调整，使磁盘中有 20G 以上的未分配空间或空白分区。

与 WINDOWS 共存安装双/多系统

在计算机上已经安装后 Windows 操作系统的情况下，如果想要保留已有 Windows 系统，则可安装双/多系统，实现 deepin 与 Windows 的共存。

与全新安装一样的，保证磁盘上有 20G 以上的未分配空间或空白分区即可。

3.2.2 安装环境

请确保您的电脑满足以下的配置要求，如果您的电脑配置低于以下的要求，将无法完美的体验深度操作系统：

- 处理器：Intel Pentium IV 2GHz 或更快的处理器
- 内存：至少 2G 内存 (RAM)，4G 以上是达到更好性能的推荐值
- 硬盘：至少 10 GB 的空闲空间

同时，您还需要一张光盘以及光驱，如果您的电脑无光驱设备，可登录深度科技官方网站下载镜像文件并制作优盘启动盘。

3.2.3 启动优盘的制作

请使用深度科技团队开发的[深度启动盘软件制作工具](#)制作启动优盘，你也可使用压缩软件打开[深度操作系统镜像](#)提取。

请根据自己[操作系统类型](#)的不同，选择[对应操作系统的启动盘制作工具](#)。将优盘插入电脑后，运行深度启动盘制作工具，选择深度操作系统镜像开始制作启动盘，制作期间请不要移除优盘，制作完成请选择重启电脑。

注意：

- 制作前请提前转移优盘中重要数据，制作时可能会清除优盘所有数据；
- 制作前建议将优盘格式化为 FAT32 格式，以提高识别率；
- 部分优盘实则为移动硬盘，因此无法识别，请更换为正规优盘；
- 优盘容量大小不得小于 8G，否则无法成功制作启动盘；
- 制作过程中请不要触碰优盘，以免因为写入不全导致制作失败。

3.2.4 安装过程

一般情况下电脑默认是从硬盘启动，因此，在使用光盘（优盘）安装系统之前，您需要先进入电脑的 BIOS 界面将光盘（优盘）设置为第一启动项。

台式机一般为 Delete 键、笔记本一般为 F2 或 F10 或 F12 键，即可进入 BIOS 设置界面。

您只需在享受一杯咖啡的时间，便可完成系统的安装。

- 1、将深度操作系统光盘（优盘）插入电脑光驱（USB 接口）中。
- 2、启动电脑，将光盘（优盘）设置为第一启动项。
- 3、进入安装界面，选择需要安装的语言。

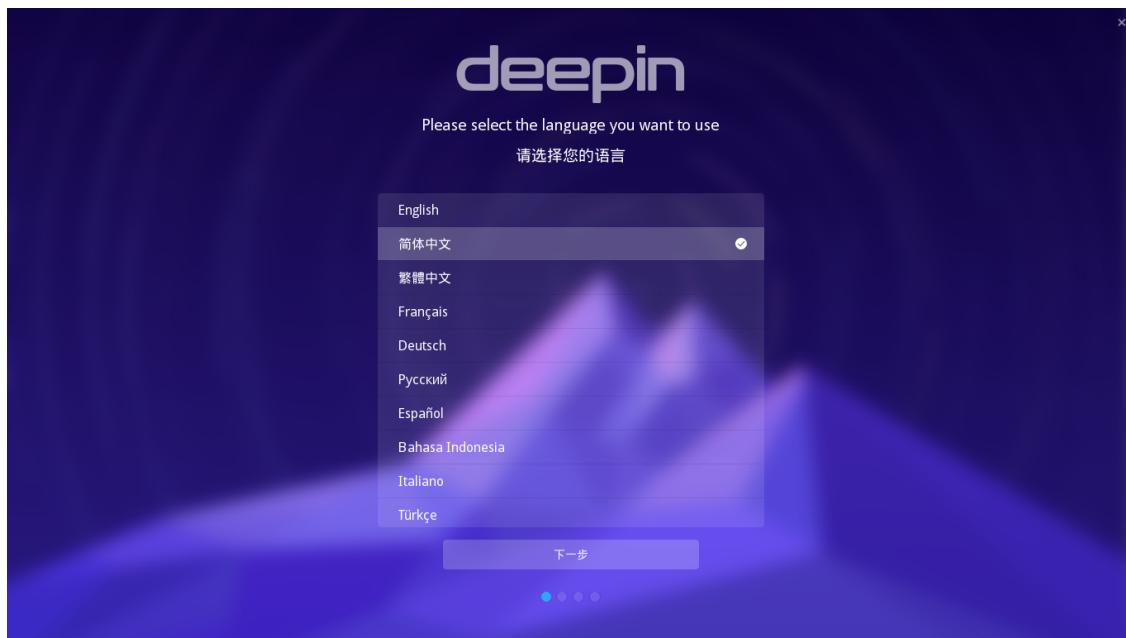


图 3-3 安装界面

4、进入账户界面，输入系统用户名和密码。

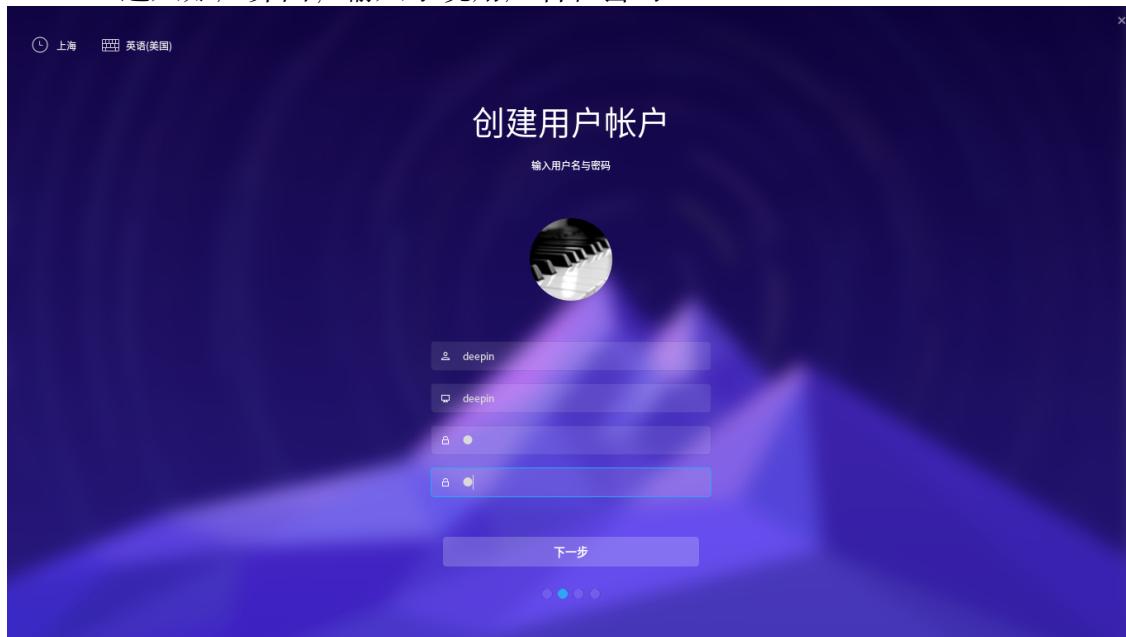


图 3-4 账户界面

5、点击下一步。

6、选择文件格式、挂载点、分配空间等。

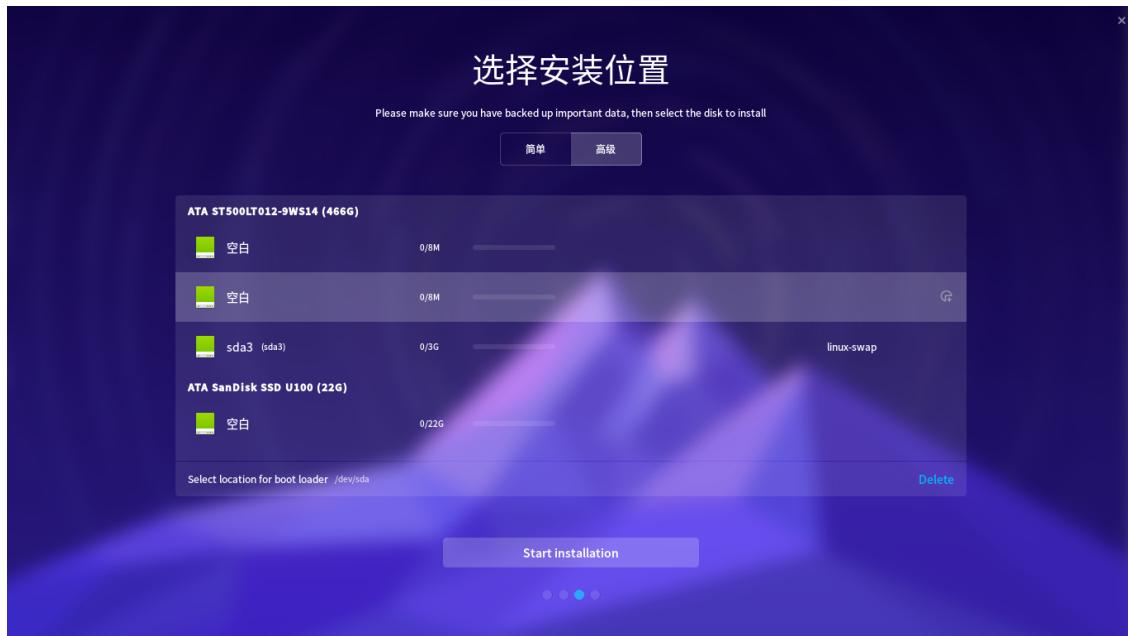


图 3-5 磁盘挂载界面

表 3-1 文件挂载说明

挂载点	挂载点中文名	文件系统	大小
/	根分区（必选）	EXT4（推荐）	最少 10G
/home	家目录（推荐）	EXT4（推荐）	最少 10G
swap	交换分区（可选）	不设置	4G 内存以下分配 2G, 4G 以上可不分配

- 7、点击安装。
 8、在弹出的确认安装窗口中，点击确定。
 9、将开始自动安装深度操作系统。
 安装视频见：<https://www.bilibili.com/video/av16993752/>

想对安装过程有更多理解，可以阅读附录庚，或者网上搜索相关教程。

3.3 桌面使用

其实深度系统自带一套学习指南**深度帮助手册**。可以按下 **Super** 键，也即一般电脑上微软标志的那个键，或者苹果电脑上的花键。在搜索里输入“深度帮助手册”就能够找到。打开后，可以自学很多知识，本书摘录其中部分内容。

还有另外一套帮助工具，**欢迎**，同上，打开后，可以直接设置相关的选项。

3.3.1 开机

如果你使用过 Windows 操作系统，那么使用深度桌面操作系统也不是难事。这里主要讲社区版深度桌面操作系统。开机登录后，点击 **Super** 键会弹出如下界面。

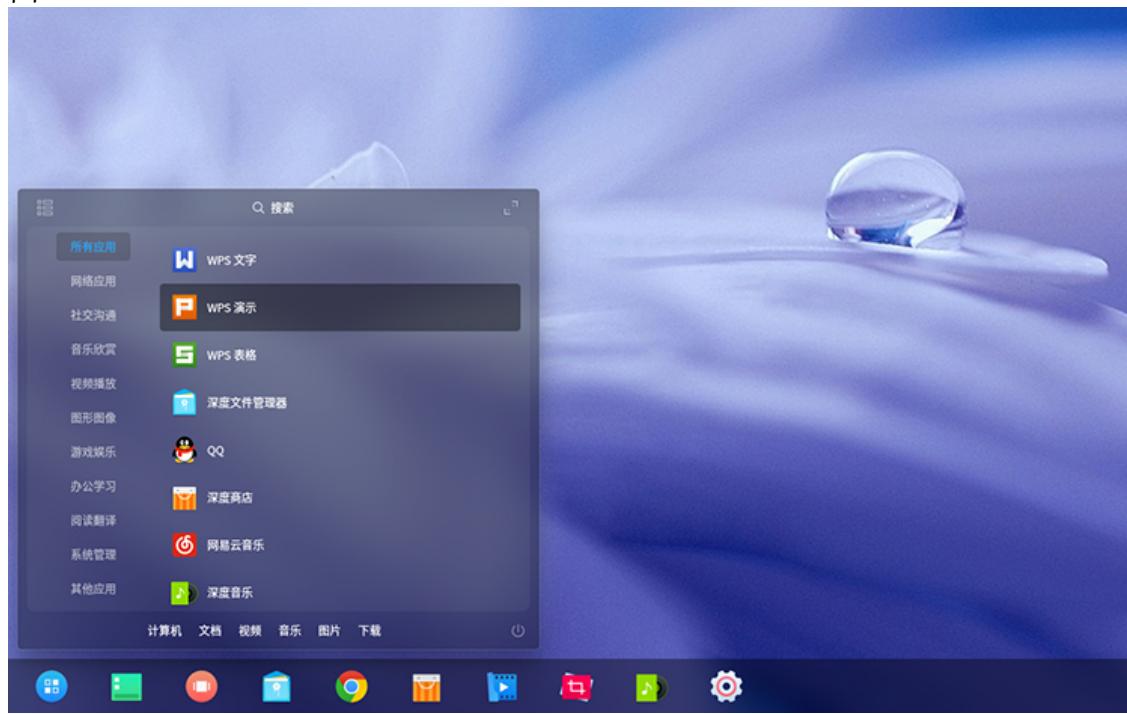


图 3-6 启动项

你可以对所需要的软件进行搜索。直接点击软件图标，就可以打开软件了。分类放置的软件。在任务栏驻留的软件打开时，还会有动画显示。

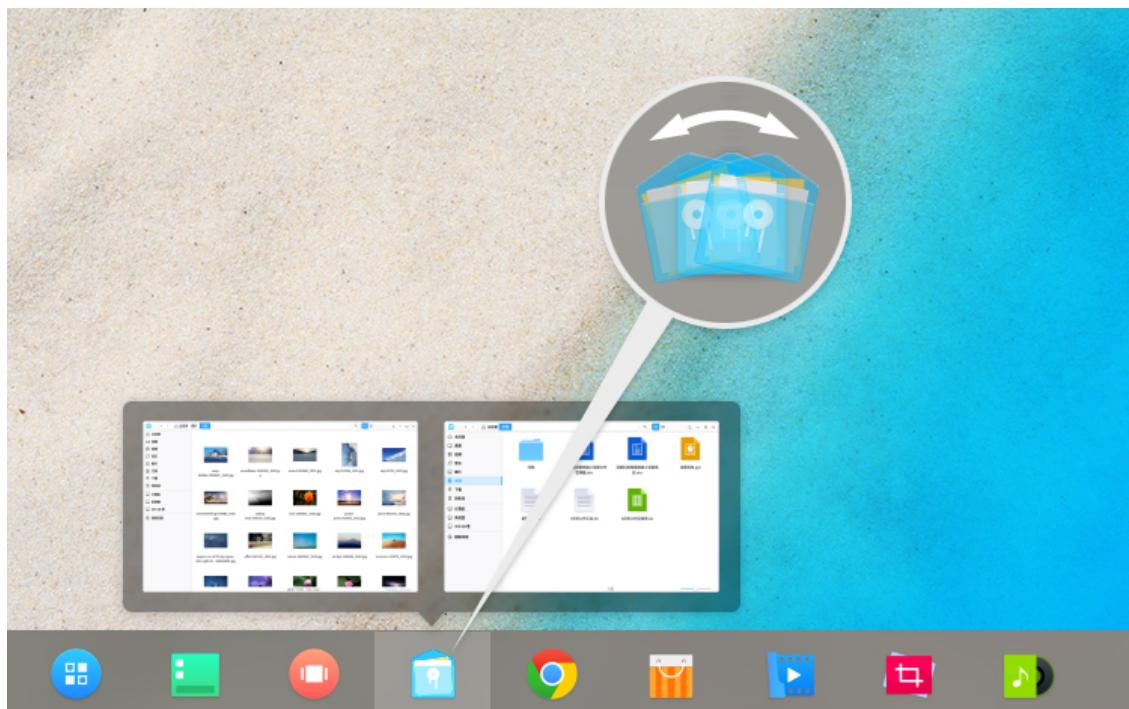


图 3-7 任务栏打开软件动画

3.3.2 关机

同时按 **Ctrl + Alt + Delete**，或者点击右下角关机按钮，就会弹出关机界面，如下图所示。

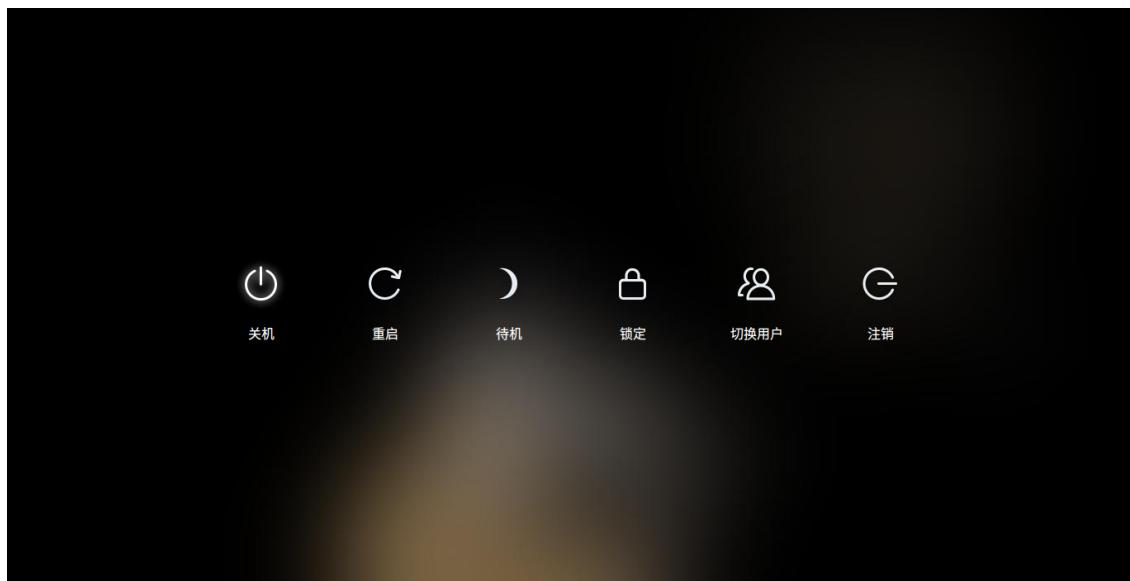


图 3-8 关机

3.3.3 控制中心-系统设置

点击任务栏，或者鼠标滑向桌面右下角几次，会在右侧出现控制中心面板。如下图所示。

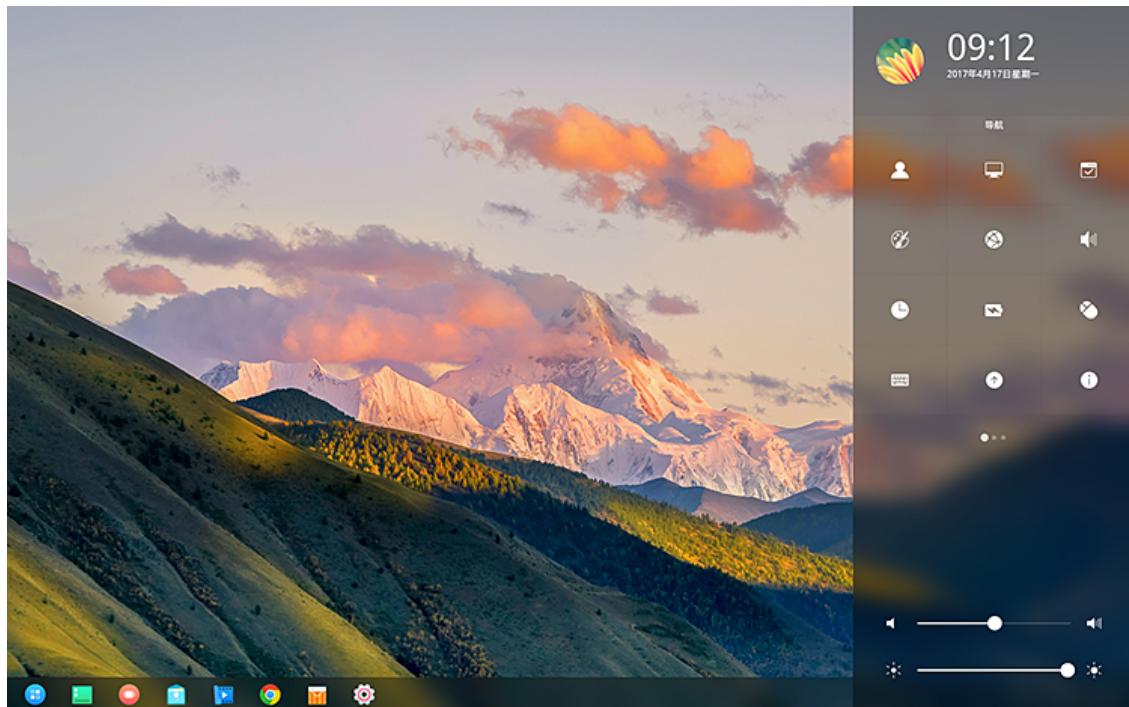


图 3-9 控制中心

首页展示快速入口、常用快捷设置等，让操作更加方便和快捷；天气详情、通知中心通过插件展示，今后会开放插件接口，让您有更多的参与定制。有兴趣的话，可以点开逐步试试。

3.3.4 开机自启动

一些软件需要开机自启动，比如 QQ 还有一些保护眼睛的软件，比如定时休息的 workrave，减少蓝光的 red-shift。点击 **Super** 键，弹出所有的软件图标，看中要开机自启动的，右击选择开机启动。



图 3-10 开机自启动

3.3.5 常用快捷键

常用快捷键类似微软 Windows 操作系统，还可以自己定制快捷键。详情请参考控制中心 - > 键盘和语言 - > 快捷键。

3.3.6 安装软件

使用电脑，不可避免的要用到很多软件。深度商店为您解忧愁，方便您查找并安装相关的软件。当然除了深度商店，您还可以使用命令行安装软件，关于命令行安装软件的介绍详见后面章节。

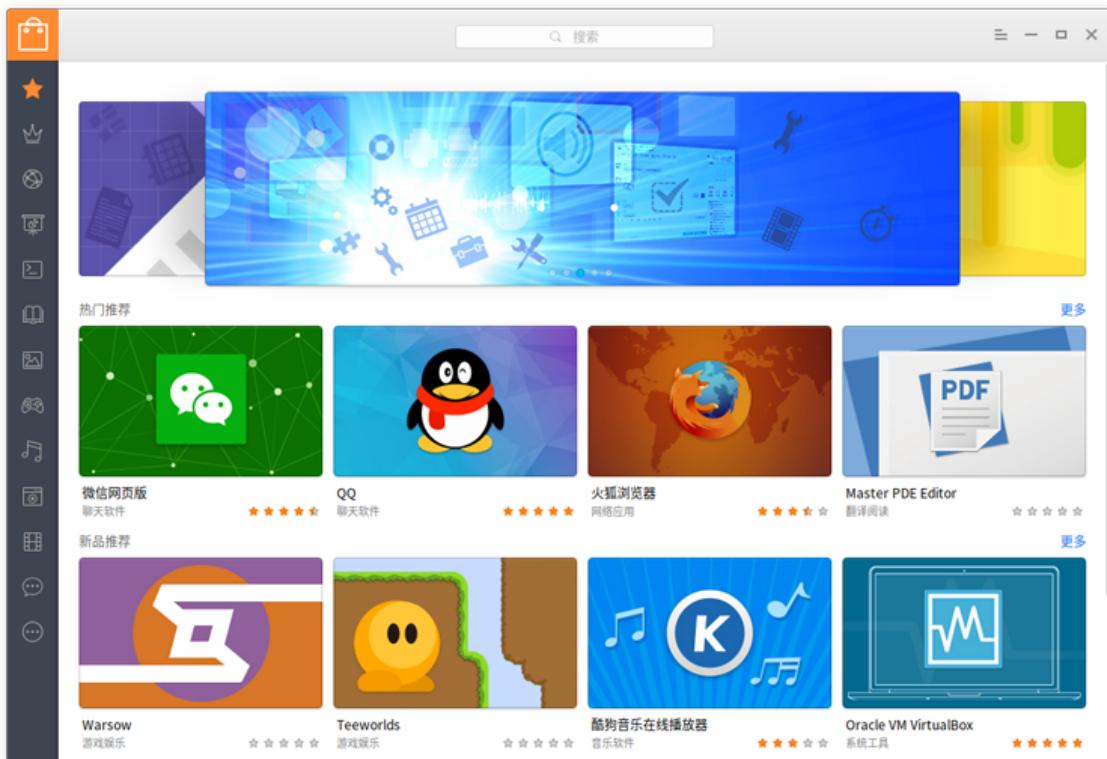


图 3-11 深度商店

3.3.7 卸载软件

按 windows 键，弹出的软件，右击选择卸载。或者命令行，这里就不介绍了，在后面详细说明。

3.4 深度桌面常用软件

打开深度商店，会有很多常用软件推荐，另外在官网[深度原创应用](#)有深度大量原创好用的软件，在官网[深度合作应用](#)等也有几个非常杰出的软件，下面介绍几款。

3.4.1 搜狗输入法

这个是系统默认自带的了，贴它是为了怀念曾经被输入法折磨的岁月。官网有详细介绍：<http://pinyin.sogou.com/linux/?r=pinyin>

不过我最希望有讯飞语音输入法，这个如果可以在 Linux 桌面下生存，是最好不过的。不过当前还没有 Linux 版本。

3.4.2 QQ

国人挚爱。可以视频。就不废话了。系统默认自带，如果没有，命令行安装。

```
1 $ sudo apt-get update  
2 $ sudo apt-get install deepin.com.qq.im
```

3.4.3 深度截图

深度截图是深度科技团队开发的一款精巧截图应用，它具有智能窗口识别、快捷键支持、图片编辑、延迟截图、智能保存、调节图像分辨率等功能。

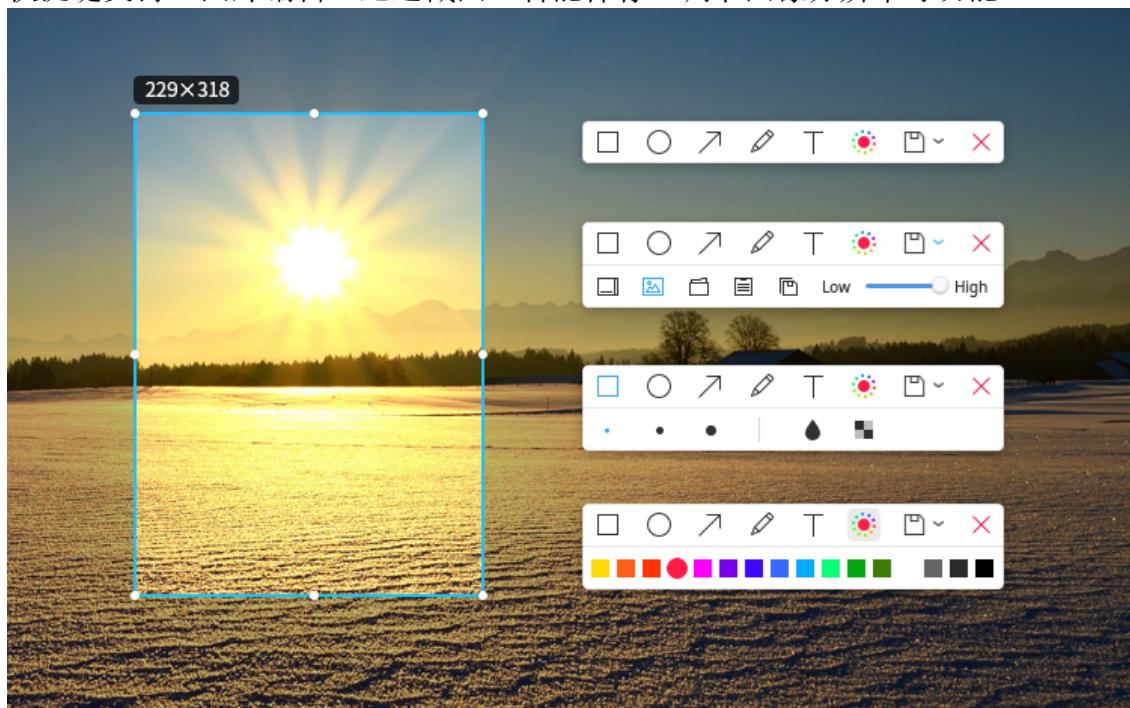


图 3-12 深度截图中文版

深度截图也是系统默认自带的，如果没有，直接在深度商店搜索下载，也可以采用命令行安装。

```
1 $ sudo apt-get install deepin-screenshot
```

3.4.4 网易云音乐

为了带来更好的音乐体验，实现对音乐高品质的追求，经过网易云音乐与深度科技团队长达半年多的联合开发，大家期待已久的网易云音乐正式登陆 Linux 平台！

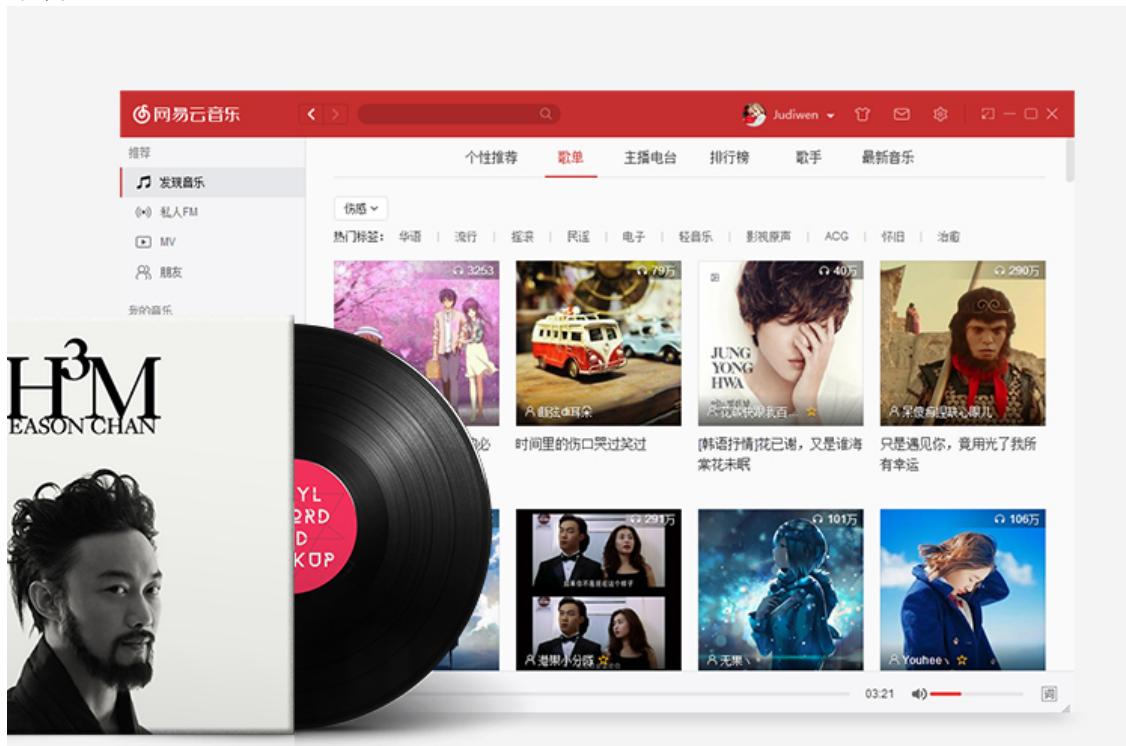


图 3-13 网易云音乐

深度操作系统用户可运行深度商店，搜索“网易云音乐”进行安装体验，非深度操作系统用户可前往网易云音乐官网下载 Linux 客户端安装体验。另外，您也可以命令行安装。

```
$ sudo apt-get install netease-cloud-music
```

3.4.5 深度系统监视器

深度系统监视器是深度科技团队打造一款直观易用的系统监视器应用，它可以实时监控处理器状态、内存占用率、网络上传下载速度；还可以管理您的系统进程和应用进程，支持搜索进程和强制结束进程。

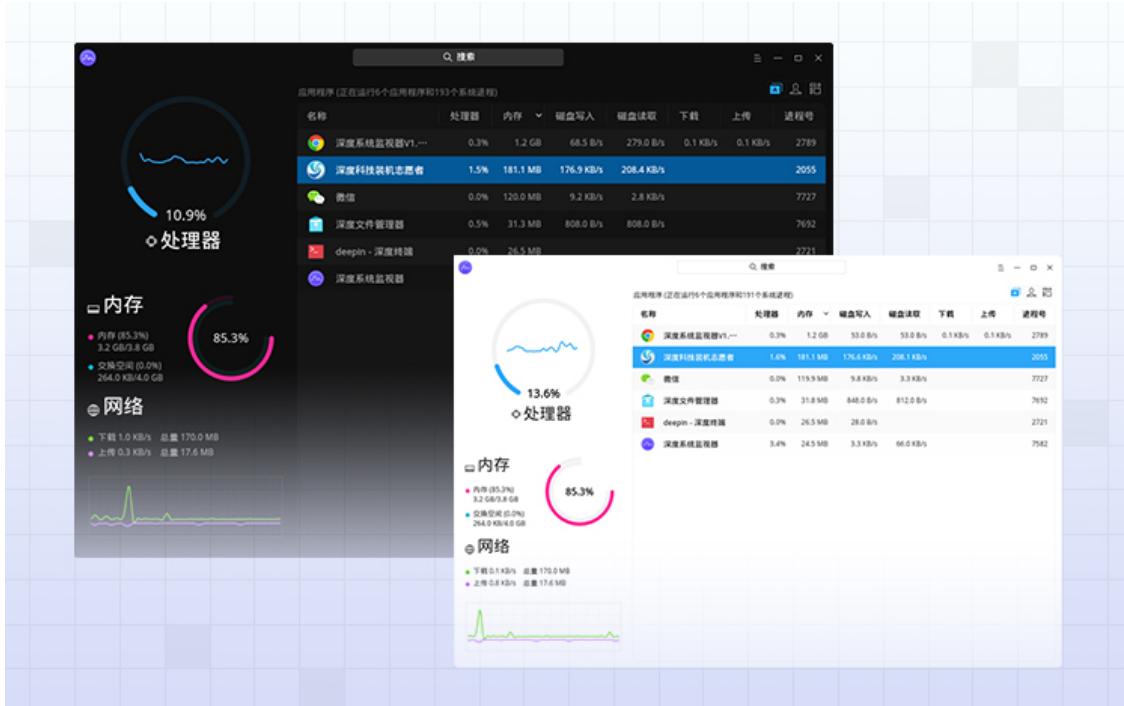


图 3-14 深度系统监视器

系统默认自带，如果没有，可以深度商店搜索下载，也可命令行安装。

```
1 $ sudo apt-get install deepin-system-monitor -y
```

3.4.6 深度无线投屏

深度演示助手是一款用于手机内容分享的演示工具。它支持照片分享功能，分享时可以对照片进行旋转、标记、聚焦等操作。甚至还可以直接将手机屏幕投射到电脑上，视频、游戏都轻松分享。另外还支持幻灯片远程控制，让您在演示幻灯片时无拘无束。



图 3-15 深度无线投屏
—42—

系统默认自带，如果没有，可以深度商店搜索下载，也可命令行安装。

```
1 $ sudo apt-get install deepin-presentation-assistant -y
```

3.5 总结

本章简要介绍了深度科技公司及深度操作系统的部分内容，并对桌面常用的一些操作和常用的若干软件做了简单描述。如果需要对深度操作系统的框架了解更多，可以阅读官网[架构设计](#)，或者更进一步阅读[源代码](#)。

3.6 附注

有建议说要转载王勇的文章，有机会试试。

第四章 教学用的 Linux 软件

参考网页：

- <https://www.linux.com/news/best-linux-tools-teachers-and-students>
- <https://linux.cn/article-9722-1.html>

4.1 引言

邓公当年曾说“计算机的普及要从娃娃做起。”其实深度操作系统的普及也可以从娃娃抓起。电脑的应用，更多的在于顺手的软件，教育方面也不例外。本章参考网上部分资料，介绍一些方便教学的软件。

4.2 笔记

毛主席的一个读书习惯“不动笔墨不读书”，既然要学习肯定是要做笔记的。GNU/Linux 操作系统下的笔记还是有很多种的，比如[为知笔记](#)、[leanote](#)、[QOwnNotes](#)、[BasKet Note Pads](#)、深度官方使用的[石墨文档](#)等，更多软件可以参考深度商店的[办公学习软件列表](#)。当然也有一些便笺软件，比如 Xpad。读者可以根据自己情况选择需要的软件，下面以 leanote 和 Xpad 为例进行介绍。

4.2.1 leanote

leanote 又称蚂蚁笔记，在深度商店里搜索 `leanote`，如下图4-1所示，点击安装，等待安装完成，点击打开即可。

安装完成后，注册登录，就可以使用了。本人试用界面如图4-2。该笔记软件跟其他很多笔记软件类似，可以离线在线使用，甚至自建服务器。能够满足日常需要了。

4.2.2 Xpad 便笺

在深度商店里搜索 `xpad`，找到该软件，点击安装即可。安装成功后，按下 `win` 键或者鼠标点击左下角启动器按钮，弹出软件列表界面。如图4-3 或者图4-4所示

在上述界面中，都有搜索框，可以在搜索框里输入 `xpad` 查找。找到后，点击图标，就会打开类似图4-5的界面，跟 win7 便签很类似。

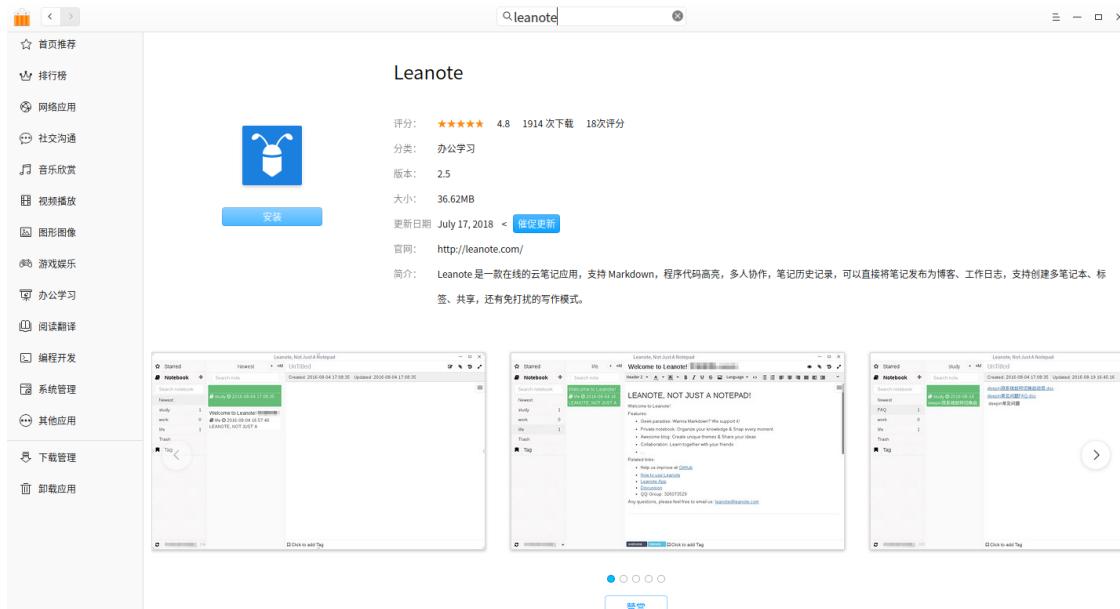


图 4–1 leanote 软件安装

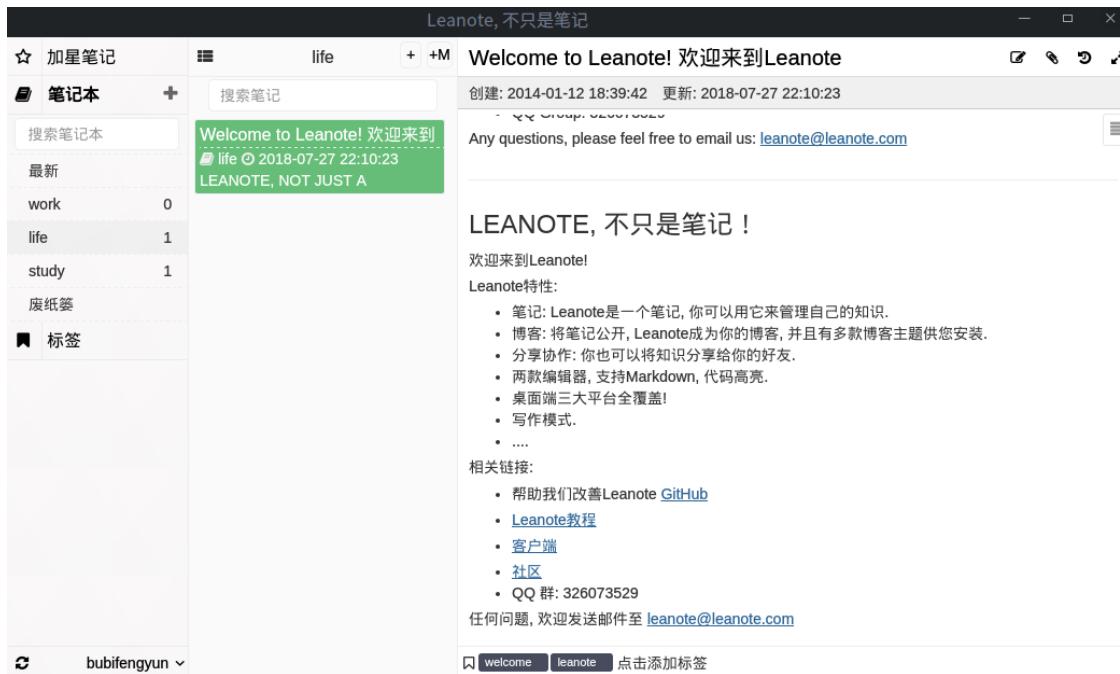


图 4–2 leanote 使用界面



图 4-3 小号版软件列表

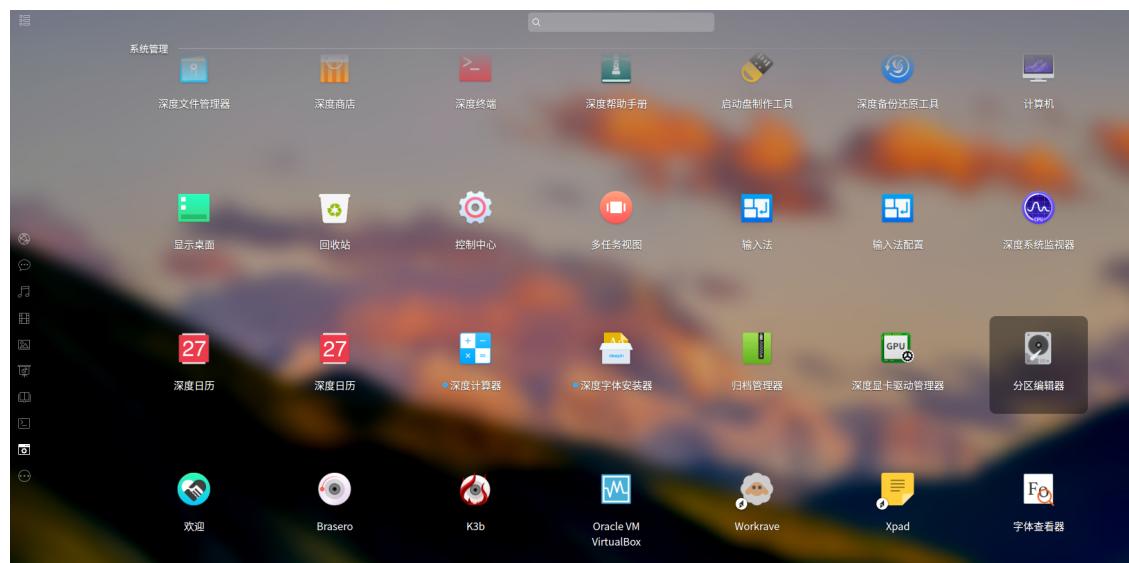


图 4-4 全屏版软件列表

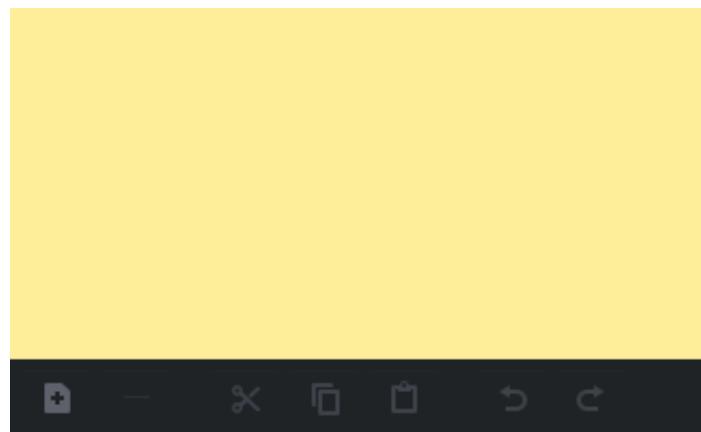


图 4-5 Xpad 初始界面

Xpad 初始界面不一定符合用户习惯，可以自定义配置。在界面上右击，会弹出如图4-6列表，选择 **Preferences**(首选项)，进行配置。

为了方便，对弹出的界面图4-7，我做了如下配置，勾选了 **View >> Show Window Decorations** 用于显示窗口，以及 **Startup >> Start Xpad automatically after login** 开机自启动。

如果需要删除某个便签，选中按下 **Shift + Delete** 键即可。

4.3 课堂工具

对于课堂工具，本章主要介绍师生屏幕广播软件，功能类似 Windows 下的 **红蜘蛛多媒体网络教室** 软件以及。

4.3.1 Veyon

该软件[开源](#)，[官网](#)能够下载 Window 和 Linux 版本的[安装包](#)。主要功能允许教师查看学生桌面上发生了什么，控制他们的桌面，锁定他们的桌面，对桌面演示，打开或关闭桌面，向学生桌面发送文本消息等等。安装方式如下，

```
1 $ sudo apt-get install veyon -y
```

具体使用方法，请参考[官网文档](#)或者网上搜索相关资料。

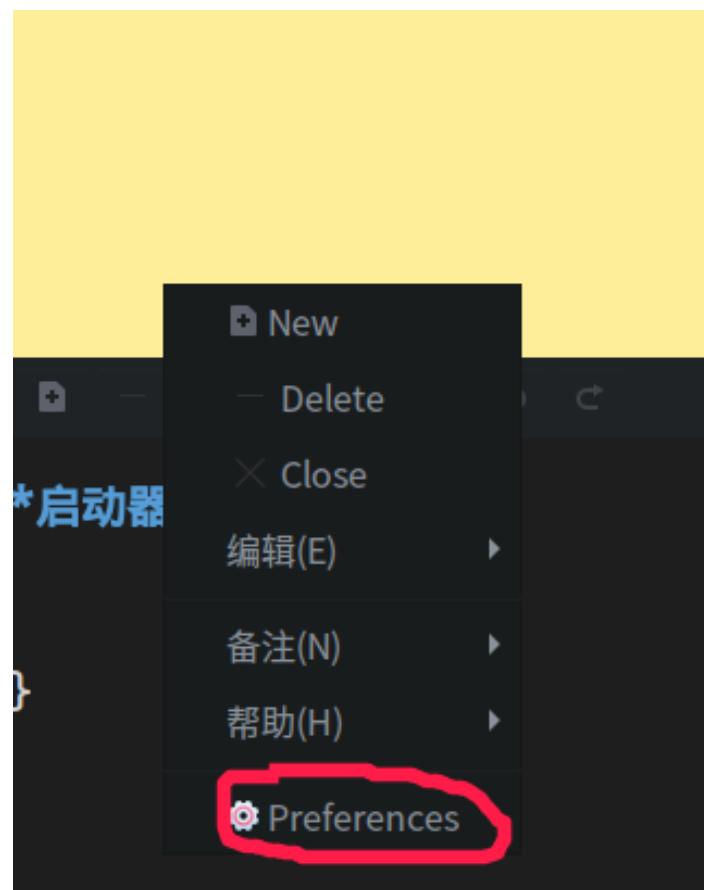


图 4-6 Xpad 右键列表

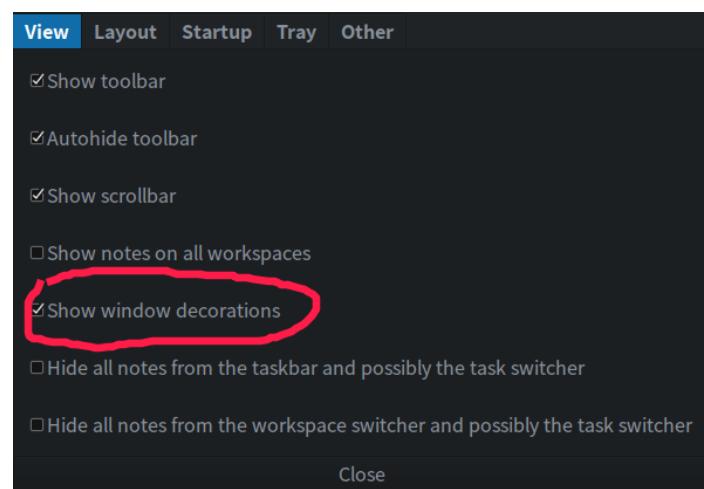


图 4-7 Xpad 配置页面

4.3.2 AContent

AContent 是 [aTutor](#) 的一个加强包，能够创建和管理在线考试和测验，分享教学资料等。

- [下载地址](#)
- [github 源码](#)

安装方法从略。使用有一定难度，不是太推荐。

4.3.3 Moodle

比较推荐这个，这是一个网站，需要预先安装服务器，比如 Apache, MySQL 等，为了方便可以使用一键安装软件 XAMPP 等。

- [官网](#)
- [github 源码](#)
- [百度百科](#)

其官网有相关说明，需要一定的网页技术知识，这里不赘述了。

4.4 文献阅读写作管理工具

对于学生和部分秉承“活到老学到老”精神的人来说，好的文献阅读工具必不可少。同样一些方便写作的编辑器也是很有市场的。对于 word 文档，比如金山 WPS 旗下产品或者永中等已经很多了，这里就不介绍了。对于程序员“编辑器之神”vim 和“神之编辑器”emacs 的争论也未曾停息。这里简要介绍几款论文写作的工具。

如果有更好的软件，或者使用方法，也请读者不吝赐教，感谢。

4.4.1 bookdown

参考网页：

- <https://bookdown.org/yihui/bookdown/>

Bookdown 基于 [R Markdown](#)，具有 Markdown 语法的简单性（您可以在 5 分钟内学习基础知识；参见附录戊），提供多种输出格式（PDF/HTML/Word/...）。还添加了多页 HTML 输出、编号和交叉引用图形/表格/部分/方程、插入部分/附录等特性，支持[GitBook 样式](#)便于创建优雅和吸引人的 HTML 图书页面。

尽管 Bookdown 包含“书”（book）一词，但不限于生成书籍，也可以生成课程讲义、学习笔记、软件手册、论文，甚至日记等。

另外，班门弄斧一下，本书就是采用 [bookdown 模板](#) 编写的，自认为还是有点精美的。另外本人制作了[上海交大论文模板](#)，也有好事者[赵鹏](#)移植了若干模板，有兴趣把各高校论文 [LATEX](#) 模板改为 bookdown 的，可以共同讨论。

4.4.2 VuFind

参考网页：

- <https://www.oschina.net/p/vufind>
- <http://vufind.org/>

VuFind 是图书馆资源门户系统，主要提供图书馆资源的检索和浏览功能，功能模块包括：

- Catalog Records
- Locally Cached Journals
- Digital Library Items
- Institutional Repository
- Institutional Bibliography
- Other Library Collections and Resources

VuFind 是完全模块化的，你可以自由选择模块。全平台支持，当然包括 Linux，二进制文件[官网](#) 提供下载。

4.4.3 Calibre

参考网页：

- <https://calibre-ebook.com/>
- https://calibre-ebook.com/download_linux

Calibre 是电子书管理软件，支持 Amazon、Apple、Bookeen、Ectaco、Endless Ideas、Google/HTC、Hanlin Song 设备及格式，功能十分强大。通常我都是按照[官网](#)提供的方法下载安装

```
$ sudo -v && wget -nv -O- \
https://download.calibre-ebook.com/linux-installer.sh \
| sudo sh /dev/stdin
```

4.4.4 CAJViewer

参考网页：

- <http://cajviewer.cnki.net/introduction.html>

“CAJViewer 7.2”是光盘国家工程研究中心、清华同方知网(北京)技术有限公司 CAJViewer 系列产品截至 2006 年 11 月为止的最新版本，它充分吸取了当前市场上各种同类主流产品和自身上一版本 CAJViewer 6.0 的优点，经过长时间市场调查和系统设计而成，兼容 CAJ 和 PDF 等文件。

对于高校学生，这个阅读器你懂得。深度提供了相应版本。

4.4.5 搜索神器 Everything

参考网页：

- <http://www.voidtools.com/>
- <http://www.voidtools.com/downloads/>
- <https://github.com/DoTheEvo/ANGRYsearch>

Everything 正是当之无愧的最强文件搜索神器！！它可以在闪电般的瞬间从海量的硬盘中找到你需要的文件！速度快到绝对让你难以置信！首次接触到 Everything 可真让我惊讶和兴奋了许久！！而且它还是一款完全免费的软件，界面简洁高效，体积很小巧，但功能却非常丰富！Everything 文件搜索工具最大的优点是近乎变态的速度。其速度不是快，是快到离谱；用户不是满意，而是震惊。你甚至会愤怒，它凭什么这么快？！

— Everything 用户

当然这个 Everything 目前只能用于 Windows 系统。在深度下也有类似的软件，ANGRYsearch。如果感兴趣的不妨试试。源码有安装说明。

第二部分

GNU/Linux 基础知识

“竹外桃花三两枝，春江水暖鸭先知”

—— 苏轼

GNU/Linux 领域也可以说是博大精深，仿佛一江春水，作为初入门庭的小白，暂且从命令行、文件系统、文本处理等方面试试水吧。待轻车熟路后，相信诸位一定能够如龙入海，纵横驰骋在这片快捷高效的领域里。

第五章 shell 用法简介	55
第六章 文件系统	83
第七章 文件编辑与查找	103
第八章 进程管理	125
第九章 简单 bash 脚本	145

第五章 shell 用法简介

通过阅读本章，你将会了解到以下几项内容。

- 了解 Linux Shell
- 打开 Shell 终端的方法
- Bash 语法简介
- Bash 自动补全和输入历史
- 命令的拼接和拓展
- 变量和简写
- 如何使用理解帮助文档

作为普通人，使用电脑除了窗口图形界面外，还有个叫终端的字符界面。Linux 中通过在终端敲击字符跟电脑交互，这玩意就叫做 shell，并且相比图形界面 (Graphic User Interfaces, GUI)，通常 shell 功能更加强大。

深度系统默认采用的 shell 是 bash (Bourne Again Shell)，继承兼容于 UNIX 早期的 Bourne shell (作者：Stephen Bourne，命令名为 sh)，当然两者还是有那么一点点不同的。曾经有一段时间深度默认 shell 是 zsh，经群众投票又改回 bash 了，其实那次投票是我发起的。

除此之外，类 UNIX 系统还有其他的一些 shell，比如 ksh, csh, tcsh, ash，当然还有非常好用的 zsh 等，有兴趣的可以查相关资料。

5.1 用 shell 有啥好处？

为了装吗？不见得。

刚接触 Linux 桌面发行版的时候，按着教材说的，就在学习终端哦。遇到糟糕的 Linux 图形界面卡住了、死机啦，不得不网上搜索解决方案，能不用终端 shell 吗？作为小白，遇到各种各样的问题，查询询问的结果，大神的建议，也多半是在 shell 下解决啊，这叫不得不用。

后来慢慢脱离低级小白，开始喜欢上终端远程连接其他 Linux 电脑，享受这种快捷方便的操作。再后来终端下用 vim 写代码，键盘翻飞，似乎成为了一种享受。

再后来，开始接管一些服务器，开始学习计算机安全知识，似乎更离不开了 shell 了。似乎儿时那种黑客的感觉，只有通过 shell 才可以实现哦。

其实 Windows 下也有 DOS 和 CMD 这类字符操作界面，不过功能弱爆了，powershell 可能功能强大一些，有兴趣的可以查相关资料，这里不说了。

5.2 深度操作系统下 shell 简介

按照《Linux Bible》的说法，有三种方式打开 shell 操作界面，分别叫 shell 提示符 (shell prompt)，终端窗口 (Terminal window)，和虚拟控制台 (virtual console)。我感觉深度操作系统就后面两种，第一种说是登录的时候出现的那种字符界面，不太容易遇到，暂且跳过。

深度操作系统登录后，进入桌面，按下 `Ctrl + Alt + T`（或 `F4` 或 `Alt + F2` 打开终端雷神模式，这个版本不同有所变化）会弹出深度终端窗口，这个软件在第三章已经作了介绍。可用于 shell 练习。对于虚拟控制台，可以通过按下 `Ctrl + Alt + Fi` 其中 `Fi` 表示 `F1, F2, …, F6`，分别弹出六个虚拟控制台，对于不同的电脑，可能 `Ctrl + Alt + F1` 或者 `Ctrl + Alt + F7` 打开的是图形界面 (GUI)。

下面以深度终端窗口为例来说明。

打开深度终端的方式

- 快捷键的方式：按下 `Ctrl + Alt + T` 或者 `F4` 或 `Alt + F2`，弹出深度终端。
- 图形方式：按下 `super` 键，或者点击左下角的图标，会弹出已安装软件图标，找到深度终端的图标，点击即可。或者在最上面搜索框中搜索终端。
- 鼠标右键：在桌面上，右击选择在终端中打开，也可以进入深度终端。

打开终端后，右击窗口内部，选择设置，可以对终端的一些外观属性配置。

深度终端的显示说明

在每行命令行开头会如下显示，

```
litianci@litianci-pc:~$
```

解释

1. `litianci@litianci-pc:~` 表示用户名和电脑名，并用 @ 隔开，: 后是当前工作目录 ~，~ 是主目录的意思，因为古老的键盘中 ~ 和 Home 是同一个按键，所以就用 ~ 代替主目录了。
2. \$ 默认表示普通用户，## 默认表示 root 根用户。
3. 这个显示说明是可以修改的，见后文 @ref() 小节。
4. 在 \$ 或者 ## 后输入相关的命令字符。

\$ 表示正在用普通用户权限运行后面的命令，而 ## 则表示正在使用 root 根用户的权限。通常只有涉及重大核心系统功能的地方才需要用到 root 根用户权限。根据放权的最小权限原则，在普通用户权限可做的事情，建议不要使用 root 根用户权限。

5.2.1 牛刀小试

下面命令行首标志 \$ 和 ## 区分运行权限，其他显示信息略。输入结果行没有行首标志，每行会有序号方便区分。

```
1 $ whoami
2 litianci
3 $ who -H
4 名称 线路 时间 备注
5 litianci tty1 2018-09-20 09:07 (:0)
6 $ grep litianci /etc/passwd
7 litianci:x:1000:1000:::/home/litianci:/bin/bash
```

解释

1. whoami 列出当前用户名
2. who -H 列出信息更详细，-H 表示显示头部标题列，第二行显示当前登录用户名 litianci，当前登录线路终端 tty1，当前用户登录系统时间 2018-09-20 09:07，以及备注 (:0)，该备注啥意思暂时没有找到，是不是图形界面的意思，待后续找资料。
3. 第 6 行，是在 /etc/passwd 文件内查找该用户，在第 7 行最后 /bin/bash，表示该用户默认的 shell 类型。实际操作时请把 litianci 换成你的账户名。

```
1 $ date
2 2018年 09月 21日 星期五 16:45:08 CST
3 $ pwd
4 /home/litianci
5 $ hostname
6 litianci-PC
7 $ hostname -I
8 192.168.43.45
9 $ ls
10 Desktop Documents Downloads Music Pictures Videos
```

解释

1. `date` 命令，无选项无参数时，输出当前日期、星期、时间和时区。其中 CST 表示中国标准时区，也即东八区。不过 CST 这个简写有[歧义](#)，代表如下几个时区，
 - CST Central Standard Time (USA) UT-6:00
 - CST Central Standard Time (Australia) UT 9:30
 - CST China Standard Time UT 8:00
 - CST Cuba Standard Time UT-4:00
2. `pwd` 命令输出当前工作目录。
3. `hostname` 命令，无选项无参数时输出本机名称，如果加选项`-i`，则输出本机 IP 地址。
4. `ls` 命令列出当前目录下所有的可见文件及文件夹。

5.2.2 命令语法结构

为了丰富命令的功能，命令常常带多种选项及参数的，[这里](#)区分两个名称，

- **选项 (Options)**: 是调整命令执行行为的开关，即，选项不同决定了命令的显示结果不同；
- **参数 (Arguments)**: 是指命令的作用对象。

5.2.3 选项 (Options)

选项分为长选项和短选项。下面示例中，`-l -a -t`就是短选项，

```
1 $ ls -l -a -t  
2 $ ls -lat
```

解释

1. `-l`(long listing)，宽列，较长格式列出信息，`-a`(all) 列出所有文件（夹），包括隐藏文件（夹），`-t`(time) 按时间排序。
2. **短选项**一般使用-短横线引导。也有不带-的，比如`ps aux`，这类叫 BSD 风格的选项。
3. 当有多个短选项时，各选项之间使用空格隔开。
4. 有些短选项可以组合，比如 `-l -a -t` 可以组合为`-lat`。

5. 有些短选项需带参数，比如-L 512M，则不便跟其他短选项组合，但如果位于最后一个也是可以组合的。比如tar cvf deepin-bible.tar ~/deepin-bible，各选项说明见后面[5.2.5节](#)解释。

```
1 $ ls --hide=Desktop  
2 Documents Downloads Music Pictures Videos
```

解释

1. --hide=Desktop，不列出 Desktop 这一文件夹。
 2. 长选项一般使用--两个短横线引导，而且后接完整英文单词。
 3. 长选项通常不能组合。
 4. 长选项如果需要参数，一般用=，比如--size=1G。不过也有用空格隔开的，比如 pandoc test.md --to latex 表示使用pandoc命令把 test.md 文件转化为latex格式。
- 以上均为常见格式，具体使用方法，还需查看相关命令的帮助，比如cmd --help，获得正确用法。

5.2.4 参数 (Arguments)

上面讲述选项有所介绍，通过空格或者等号传入参数，也有直接通过空格缀在命令后面的。比如，

```
1 $ cd /media/litianci/data/
```

跳转到`/media/litianci/data/`文件夹。

5.2.5 更多例子

为了方便大家上手，再多介绍几个例子。

```
1 $ tar cvf deepin-bible.tar ~/deepin-bible  
2 $ uname  
3 $ uname -a  
4 $ date  
5 $ date +'%Y-%m-%d'  
6 2018-09-21
```

```

7 $ date +'%A, %B %d, %Y'
8 星期五, 九月 21, 2018
9 $ id
10 uid=1000(litianci) gid=1000(litianci) 组=1000(litianci),7(lp),24(
    cdrom),27(sudo)
11 $ who -aH
12 名称      线路      时间      空闲      进程号 备注      退出
13          系统引导 2018-09-21 15:54
14          运行级别 5 2018-09-21 07:54
15 litianci + tty1      2018-09-21 07:55  18:55      3820 (:0)
16 登录      tty2      2018-09-21 21:48      21868 id=tty2

```

解释

- 第 1 行命令, c(create) 表示创建压缩文件, v(verbose) 表示详细信息, f(file) 表示创建的文件为deepin-bible.tar; 这是典型的组合命令, 而最后一个组合选项f带参数deepin-bible.tar。~/deepin-bible置于最后, 为待压缩文件夹。这样本条命令实现了对文件夹的压缩。
- 第 2、3 行, uname 命令, 显示系统名字, -a(all) 选项, 表示显示系统所有信息。
- 第 4、5、7 行, 输出不同格式的日期时间格式。详细说明可以参考date --help 的说明。有兴趣的可以多试试。注意系统语言对显示结果的影响。
- 第 9 行, id, 常用于核实自己账户编号信息。
- 第 11 行, -a 表示显示较多信息, 具体信息参考who --help 说明。-H已经在前面说明, 显示列标题。

由于采用系统默认的中文, 深度终端自作聪明的翻译, 有些内容乱码。为了方便叙述, 后面把终端显示改为英文的, 也即在 ~/.bashrc 文件加上所用语言选项, 并执行。

```

1 $ echo LANG=en_US >> ~/.bashrc
2 $ source ~/.bashrc

```

或者直接修改所有用户的 bash 信息, sudo gedit /etc/profile, 修改最后一行的LC_ALL=C为LC_ALL=en。

回头再看who -aH命令, 就清爽了。当然需要你对英文有一定理解了。

```

1 $ who -aH

```

NAME	LINE	TIME	IDLE	PID COMMENT
	EXIT			
3	system boot	Sep 21 15:54		
4	run-level 5	Sep 21 07:54		
5	litianci + tty1	Sep 21 07:55	old	3820 (:0)
6	LOGIN tty2	Sep 21 22:02		25289 id=tty2

5.2.6 找找命令文件在哪里

相信大家都听说过一切皆文件的 Unix 哲学，其实很多命令都是可执行文件，是可以查看这部分命令的文件的。比如 bash 位于 /bin/bash，那么命令 bash 和 /bin/bash 就是相同的。为啥我们可以简写为 bash 呢？是因为 shell 有 PATH（路径）这个环境变量。众多已知目录存放在 PATH 变量里，shell 遇到这些没有目录路径的命令，在查找其他默认命令集合无果后，就会去 PATH 路径变量里各目录下去找该文件名，从左到右查找，找到后就执行命令，找不到就会报错。那么你电脑上的 PATH 是啥呢？

```
1 $ echo $PATH
2 /home/litianci/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/games
   :/usr/games:/sbin:/usr/sbin
```

一般命令，都存放在 /bin, /usr/bin, 或 /usr/local/bin 文件夹。一般管理相关的命令放在 /sbin 和 /usr/sbin 文件夹，也有一些 GNU/Linux 发行版放在非常规文件夹下。这些文件夹以英文冒号 : 隔开。最后无冒号。而最开始的 /home/litianci/bin 文件夹是自己所用的目录。你可以对 PATH 修改，添加某些你常用的命令文件夹。配置方法见后面叙述。

不过这里存在安全隐患，如果罪恶黑客侵入你系统，无权修改你系统级别的文件，但是有能力在你的文件夹下创建可执行文件，比如 /home/litianci/bin，跟系统文件夹的命令同名，你以为你用的是系统的命令，并输入密码，却不料执行的是黑客想要的命令。对于系统管理员，建议输入命令的时候，把目录带全，避免这类权限泄露。

下面总结 shell 查找命令位置的优先级别。

1. 别名 (aliases)。别名就是某命令包含或不包含一系列选项的集合，便于输入。后面 @ref() 小节详述。
2. shell 保留字 (Reserved word)，构建 shell 脚本的基础，比如 do, while 等。

见后面 @ref() 章详述。

3. 函数 (function), shell 中创建的命令集合。
4. shell 内置命令 (built-in), 这些命令在文件系统中是找不到的, 比如`cd`改变工作目录, `exit`退出 shell, `echo`输出文本, `history`查看之前命令运行历史, `pwd`显示当前工作目录, 还有`set`和`type`等。
5. 文件系统中可以执行的文件。比如`PATH`变量中那些目录下的文件等。

你输入的命令是哪个呢 ?

在 bash 内, 使用

```
1 $ type ls
2 ls 是 `ls --color=auto` 的别名
```

可以看到你的命令是哪个命令。为了显示全部同一名称的命令, 可以加-a选项。

```
1 $ type -a ls
2 ls 是 `ls --color=auto` 的别名
3 ls 是 /bin/ls
4 $ type -a cd
5 cd 是 shell 内建
```

可以看到有两个`ls`命令。`cd`是内置命令 (翻译有差距)。其他 shell 也可用`which`命令查找当前正在使用的命令是哪一个。此外还有可以找命令文件位置的`which`和`locate`(深度系统未自带), 有兴趣的可以去找找资料。

5.3 命令快捷编辑

5.3.1 方便编辑的快捷键

如果命令不足够快捷, 那还不如鼠标点点点呢。提到编辑器, 在 Linux 世界就跳不过编辑器之神 (`vim`) 和神之编辑器 (`emacs`) 的争论。在这里不探讨孰是孰非。而 bash 默认的编辑方式就是这两者之一的 `emacs` 模式。如果喜欢 `vim` 的编辑方式, 也可用设置为 `vim` 模式。在 `~/.bashrc` 文件末尾加上 `set -o vi`, 下次启动终端的时候, 就是 `vi` 模式了。所谓模式, 也是为了方便编辑, 提供一些快捷键等。我最喜欢用的快捷键, 就是 `Ctrl+A`跳到行头, `Ctrl+E`跳到行尾和`Ctrl+L`清屏了。其他

的都是键盘上下左右键切换，懒得记录这些快捷键。不过还是把这些快捷键列在下面吧。

表 5-1 光标跳转快捷键说明

快捷键	解释
Ctrl+F	向前一个字符
Ctrl+B	后退一个字符
Alt+F	向前一个单词
Alt+B	向后一个单词
Ctrl+A	跳到该行行首
Ctrl+E	跳到该行行尾
Ctrl+L	清空屏幕内容

注：清屏，只是把内容放到屏幕上方，并加入了很大一片空白，滚动鼠标还可以看到刚刚输出的内容。

表 5-2 编辑快捷键说明

快捷键	解释
Ctrl+D	删除当前字符
Backspace	删除前面字符
Ctrl+T	前后字符换位
Alt+T	前后单词换位
Alt+U	当前单词大写
Alt+L	当前单词小写
Alt+C	选中字符大写
Ctrl+V	插入特殊字符

注：每个快捷键具体动作，还需要多加练习记忆。对于插入特殊字符的，ctrl+v比如插入表格键Table，则ctrl+v+Tab，经测试不是那么回事。

表 5-3 复制粘贴快捷键说明

快捷键	解释
Ctrl+K	剪切光标到行尾的全部字符，含光标所在字符
Ctrl+U	剪切光标到行首的全部字符，不含光标所在字符
Ctrl+W	剪切光标前到当前单词词首的字符，不含光标
Alt+D	剪切光标后到当前单词词尾的字符，含光标
Ctrl+Y	粘贴到光标前
Alt+Y	循环显示刚刚剪切的内容，待你确定粘贴对象
Ctrl+C	删除整行

5.3.2 重输之前的命令

参考网页：

- <https://blog.csdn.net/maplesky2017/article/details/78389068>
- <https://www.cnblogs.com/5201351/articles/4208509.html>

bash 是会记录你已经输入命令的。使用history命令，可以查看相关内容。常见用法如下。

- 上下键查看以前的命令（包括Ctrl+N、Ctrl+P快捷键）；
- history 8 查看最近八条命令记录；
- history 查看最近\$HISTSIZE条命令记录
- !n 调用第 n 条命令，n 是一个正整数；
- !! 调用上一条命令；
- !字符串 调用最后一以该字符串开头的命令。

本人感觉以上命令不太实用。除非你知道刚刚是啥命令，才方便操作。Matlab 软件的命令记录就很好查询。输入部分字符就可以把所有以该字符开头的命令记录一次列出。对于 bash，类似的功能非常鸡肋。在空行内按下快捷键Ctrl+R，输入部分字符，会查找相关命令记录，符合你需求，回车，就直接运行了。丝毫不给你重新编辑的机会，也无法上下翻阅，难用至极。另外还有快捷键Ctrl+S在深度操作系统下，不能用。另外两个Alt+P，Alt+N有心情的可以试试。如果哪位有好的用法，欢迎提供。

默认记录 1000 条命令，但是事实上记录文件里记录的命令条数可能超过此数值。可以输入如下命令查看，

```
1 $ echo $HISTSIZE  
2 1000  
3 $ echo $HISTFILE  
4 /home/litianci/.bash_history
```

解释

- 第 1 行命令，查询命令记录条数，显示为 1000。变量\$HISTSIZE用于设置记录条数。

2. 第 3 行命令，查看命令记录文件，本机为`/home/litianci/.bash_history`。

如果我们想修改记录条数，或者记录文件，可以`/etc/profile` 修改相应的变量。

下面以修改记录条数为 200 进行讲解。

```
1 $ sudo sed -i 's/^HISTSIZE=1000/HISTSIZE=200/' /etc/profile  
2 $ source /etc/profile #使其立即生效
```

上方命令使用的是`sed`输入的，您可以以其他方式，在文件`/etc/profile` 加上一行`HISTSIZE=200`。并保存退出。



对于系统管理员来说，可能保存这些操作记录很危险。你可以设置零记录，或者把记录文件设置为垃圾箱。`HISTSIZE=0, HISTFILE=/dev/null`

5.3.3 自动补全

参考网页：

- https://blog.csdn.net/mycwq/article/details/52420330?utm_source=copy
- <http://www.techug.com/post/10-linux-completion-commands.html>

在 Linux 命令行下，输入字符后，按两次`Tab`键，shell 就会列出以这些字符打头的所有可用命令。如果只有一个命令匹配到，按一次`Tab`键就自动将这个命令补全。比如，想更改密码，但只记得这个命令前几个字母是`pass`。这时候，按`Tab`键，shell 就自动输出`passwd`命令，非常方便。

有些发行版，比如centos最小化安装，常常只能补全命令名和文件名，则需要安装补全增强软件包`bash-completion`。而一般补全的内容有如下几种。

- 命令 (Command) 别名 (alias) 和函数名 (function)。常规字符输入，敲一两次`Tab`键即可。

- 变量 (Variable)。如果你输入了美元符号 (\$), shell 会给你补全可能的变量。
- 用户名 (Username)。如果你输入了波浪符号 (~), shell 会给你补全可能的用户名。`~username` 表示该用户的主目录。
- 主机名 (Hostname)。如果你输入了邮箱符号 (@), shell 会给你补全 `/etc/hosts` 文件里的主机名。

5.4 命令连接与扩展

前面 Unix 哲学讲“程序，应当能够协作”。命令协作靠的是元字符 (metacharacter) 功能。元字符有如下 7 个：|,&,;,(),<,>。

5.4.1 匿名管道

参考网页：

- <https://www.cnblogs.com/pengliangcheng/p/5211786.html>
- <https://www.xuebuyuan.com/3234708.html>
- <https://www.jb51.net/article/120741.htm>

管道字符 |，意如其名，类似管道一样将管道入口的数据通过管道传递给管道出口。

管道是为了解决进程间通信问题而存在，它可以让两个进程之间的数据进行传递，将一个进程的输出数据传递给另一个进程作为其输入数据。管道左边是数据给予方，管道右边是数据接收方。

管道仅能处理经由前面一个指令传出的正确输出信息，也就是 standard output 的信息，对于 standard error 信息没有直接处理能力。然后，传递给下一个命令，作为标准的输入 standard input。如下图所示，



图 5-1 管道输出示意图

1. 管道命令只处理前一个命令正确输出，不处理错误输出
2. 管道命令右边命令，必须能够接收标准输入流命令才行。

```
1 $ ps aux | grep "ssh"
2 litianci 11012 0.0 0.0 14660 1124 pts/0 S+ 15:39 0:00
      grep ssh
```

按一般想法，先执行了`ps`，得到输出后将输出数据传递给`grep`，这时候`grep`还没运行而`ps`已经运行完毕了，为什么还能统计到`grep`进程的信息呢？原因是管道实现的是进程间通信，两个进程之间存在交叉，在运行`ps`进程后开始收集进程信息，`grep`也已经开始并处于等待接收数据状态，当`ps`收集到任何数据后都将输出放入内存由管道传递给`grep`进行筛选。

管道其本质是数据传递，管道左边的输出数据放入内存，由管道右边的进程读取。假如内存不足以完全存放输出数据，则管道左边的进程将一直等待，直到管道右边取出内存中一部分的数据以让管道左边的进程继续输出，而管道右边的进程在管道左边的进程启动后也立刻启动了，但是它一直处于等待状态，等待接收管道传递来的数据。

也就是说，管道左右两边的进程运行几乎没有先后顺序的。

上述管道是指匿名管道。使用匿名管道的过程中，可能已经发现管道两边的进程是同属一个进程组的，也就是说管道左方的数据只能传递给管道右方的进程，其他任何进程都没法读取此数据。但除了匿名管道，还有命名管道，命名管道是将一个进程的数据存储到一个管道文件(fifo)中，其他进程可以读取该管道文件来读取其中的数据，也就是说不再限制数据读取方。关于命名管道，请参阅 Linux/Unix 操作系统内核或编程类的书籍，一般都会有详细的介绍。

再来两个简单例子

```
1 $ cat /etc/passwd | sort | less
2 $ gunzip < /usr/share/man/man1/grep.1.gz | nroff -c -man | less
```

解释

- 第 1 行命令，`cat` 命令显示文件`/etc/passwd`的内容，输出到`sort`命令进行排序，再送给`less`命令显示出来。
- 第 2 行命令，`gunzip` 命令解压缩`grep.1.gz` 文件，送给`nroff`命令调整为适合阅读的格式，传递到`less`命令显示出来。

5.4.2 重定向

参考网页：

- <https://blog.csdn.net/u014536527/article/details/51010678>

最常见的标准输入（stdin）、标准输出（stdout）和标准错误输出（stderr）的文件描述符分别是0、1和2，其中0、1、2也可以认为是它们的数字代号。

表 5-4 三个特殊的文件

名称	程序	代号	符号
标准输入	/dev/stdin	0	< 或 <<
标准输出	/dev/stdout	1	> 或 >>
标准错误输出	/dev/stderr	2	2> 或 2>>

<、>、2>实现的是覆盖功能，>>、2>>实现的是追加的功能，但是<<不是追加功能，而是表示此处生成文档(here document)，在后面cat和重定向配合的内容里有说明。此外，还有<<<，它表示此处字符串(here string)，也见下文。

一般-都是表示旧工作文件夹；有时候，-也表示/dev/stdin,。如：

```
1 $ cat /etc/fstab | cat -
```

脚本中常见2>&1和&>的符号，它们都表示将 stdout 和 stderr 都重定向到同一个地方去，即重定向所有输出内容。如最常见的&> /dev/null 表示将 stdout 或 stderr 丢到/dev/null 表示丢弃输出信息，反过来，将/dev/null 重定向到某个文件则表示清空文件。

```
1 $ cat /dev/null > ab.sh
```

除此，还有以下几种方法快速清空文件

```
1 $ > ab.sh
2 $ : > ab.sh      # 或"true >ab.sh"，其实它们都等价于">ab.sh"
3 $ echo '' > ab.sh
4 $ truncate -s 0 ab.sh # truncate 命令用于收缩和扩展文件大小
5 $ dd if=/dev/null of=ab.sh
```

最后最重要的一点：在有重定向符号的语句中，命令执行之前已经将文件截断了。所以如果正在编辑一个文件并将编辑的结果重定向回这个文件将出现异常，

因为截断后就没有合适的内容用于编辑。一个简单的示例如下：

```
1 $ head a.log > a.log
```

有些时候直接使用>覆盖输出是比较危险的。可以使用set -c来设置如果输出重定向文件已经存在则不覆盖。使用set +c来取消set -c的效果。如果在设置了set -c时仍然想强制覆盖，可以使用>|代替>来重定向输出。同理错误输出也有此特性。

```
1 $ set -c
2 $ cat flip >ttt.txt
3 -bash: ttt.txt: cannot overwrite existing file
4 $ cat flip >| ttt.txt
5 $ set +c
```

接下来讲<<一篇字符串。类似 PHP 语言的<<<功能

```
1 $ cat >log1.txt <<EOF
2 > this is stdin character first!
3 > EOF
```

解释

- 第 1 行，<<EOF 表示下面另起一行，开始输入一篇字符，这篇字符直到其中一行为 EOF 截止。其中 EOF 可以换成你喜欢的字符串，一般采用大写，最好不要弄出特殊字符来。
- 第 2 行，就是输入的那篇字符。后面还可以输入很多。
- 第 3 行，输入 EOF，表示这篇字符结束。

在 bash 中，<<和<<<是特殊重定向符号。<<<表示的是其后字符串作为输入数据。

```
1 $ cat <<< PATH
```

等价于

```
1 $ echo PATH | cat
```

一般情况下，重定向要么将信息输入到文件中，要么输出到屏幕上，但是既想输出到屏幕又想输出到文件就比较麻烦。使用 `tee` 的双重定向功能可以实现该想法。

```
| tee [-a] file
```

选项说明：

- `-a`: 默认是将输出覆盖到文件中，使用该选项将变为追加行为。
- `file`: 除了输出到标准输出中，还将输出到 `file` 中。如果 `file` 为“`-`”，则表示再输入一次到标准输出中。

例如下面的代码，将 `a` 开头的文件内容全部保存到 `b.log`，同时把副本交给后面的 `cat`，使用这个 `cat` 又将内容保存到了 `x.log`。其中“`-`”代表前面的 `stdin`。

```
| $ cat a* | tee b.log | cat - >x.log
```

还可以直接输出到屏幕：

```
| $ cat a* | tee b.log | cat
```

`tee` 默认会使用覆盖的方式保存到文件，可以使用`-a` 选项来追加到文件。如：

```
| $ cat a* | tee -a b.log | cat
```

5.4.3 重定向和管道的区别

表 5-5 管道命令与重定向区别

管道 ()	重定向 (>,<)
左边命令应该做到标准输出，右边命令应该接受标准输入	左边的命令应该有标准输出(输入)，右边只能是文件
管道触发两个子进程执行 两边的程序	在一个进程内执行

```
1 $ cat test.sh| grep -n 'echo' # “|” 管道两边都必须是 shell 命令  
2 $ grep -n 'echo' <test.sh # “重定向” 符号，右边只能是文件
```

下面两个也是等同的。

```
1 $ (sed -n '1,$p'|grep -n 'echo')<test.sh
```

这个脚本比较有意思了。由于前面是管道，后面需要把 test.sh 内容重定向到 sed，然后 sed 输出通过管道，输入给 grep. 需要将前面用“()”运算符括起来。在单括号内的命令，可以把它们看作一个象一个命令样。如果不加括号 test.sh 就是 grep 的输入了。

```
1 $ sed -n '1,$p'<test.sh | grep -n 'echo'
```

重定向运算符，在 shell 命令解析前，首先检查的（一个命令，执行前一定检查好它的输入，输出，也就是 0,1,2 设备是否准备好），所以优先级会最高

```
1 $ sed -n '1,10p'<test.sh | grep -n 'echo' <testsh.sh
```

哈哈，这个 grep 又接受管道输入，又有 testsh.sh 输入，那是不是 2 个都接收呢。刚才说了“<”运算符会优先，管道还没有发送数据前，grep 绑定了 testsh.sh 输入，这样 sed 命令输出就被抛弃了。这里一定要小心使用

下面是输出重定向例子。

```
1 $ cat test.sh>test.txt  
2 $ cat test.sh| tee test.txt &>/dev/null
```

通过管道实现将结果存入文件，还需要借助命令 tee，它会把管道过来标准输入写入文件 test.txt，然后将标准输入复制到标准输出 (stdout)，所以重定向到 /dev/null 不显示输出。“>”输出重定向，往往在命令最右边，接收左边命令的，输出结果，重定向到指定文件。也可以用到命令中间。

```
1 $ ls test.sh test1.sh testsh.sh 2>err.txt | grep 'test'
```

目录下面有：test,testsh 文件，test1.sh 不存在，因此将 ls 命令错误输出输入到 err.txt 正确输出，还会通过管道发送到 grep 命令。

```
1 $ ls test.sh test1.sh testsh.sh &>err.txt | grep 'test'
```

这次打印结果是空，& 代表正确与错误输出都输入给 err.txt，通过管道继续往下面传递数据为空，所以没有什么显示的。同样“>”输出重定向符，优先级也是先解析，当一个命令有这个字符，它就会与左边命令标准输出绑定。准备好了这些，就等待命令执行输出数据，它就开始接收

从上面例子可以看，重定向与管道在使用时候很多时候可以通用，其实，在 shell 里面，经常是【条条大路通罗马】的。一般如果是命令间传递参数，还是管道的好，如果处理输出结果需要重定向到文件，还是用重定向输出比较好。

5.4.4 顺序执行

顺序执行，类似 C 语言的；语法。逐条执行命令。

```
1 $ date ; cat <<< LONGLONGJOB | cat - ; date
2 2018年 09月 29日 星期六 17:16:12 CST
3 LONGLONGJOB
4 2018年 09月 29日 星期六 17:16:12 CST
```

解释

1. 本命令实现了对程序耗时的计算。而分号；的优先级是低于管道的。最终送给cat - 中cat的只有LONGLONGJOB。

5.4.5 后台命令

有些命令比较耗时，或者其他原因，我们希望他们运行在后台。就可以用如下方式。

```
1 $ tar cf test.tar test &
2 [1] 9355
```

```
3 $  
4 [1]+ 已完成          tar cf test.tar test
```

解释

1. 第 1 行，命令末尾加&即可让程序运行于后台。shell 会告诉该命令的进程号，示例中为9355；并且会为他编号。
2. 第 4 行，命令运行介绍，shell 会告诉你已经完成了该后台命令。
3. 在运行过程中，如果你不想让他终止，请不要关闭该 shell。

5.4.6 命令扩展

把命令输出结果作为另外一条命令的参数，也是常用的一项功能，可以用美元符号\$和斜点（键盘左上角 ~ 键）`，\$(command) 和 `command`。

```
1 $ vim -p $(find ~ | grep deepin-bible.tex)
```

解释

1. find ~ 查找主目录所有文件，grep deepin-bible.tex 从前者找到文件名包含deepin-bible.tex的所有文件，vim 把找到的所有文件打开。

5.4.7 简单数值计算

使用\$[math expression]的方式计算。

```
1 $ echo "我今年$[`date +%Y` - 1987]岁了."  
2 我今年31岁了。
```

解释

1. `date +%Y` 输出今年年数 2018，\$[2018 - 1987]输出岁数。

5.4.8 输出变量值

一些变量的数值，可以通过\$var输出。

```
1 $ echo $PATH
```

5.5 Shell 变量 (Variables)

参考网页：

- https://blog.csdn.net/apollon_krj/article/details/70148022

变量分为全局变量(环境变量)和局部变量(本地变量)。环境变量可以在定义它们的 shell 及其派生出来的任意子进程的 shell 中使用。局部变量只能在定义它们的函数/脚本中使用。还有一些变量是用户创建的，其他的则是专用的 shell 变量，比如系统变量\$0等。

全局变量(环境变量)

环境变量可用于定义 shell 的运行环境，环境变量可以在配置文件中定义与修改，也可以在命令行中设置，但是命令行中的修改操作在终端重启时就会丢失，因此最好在配置文件中修改（用户家目录的“.bash_profile”文件或者全局配置“/etc/profile”、“/etc/bashrc”文件或者“/etc/profile.d”文件中定义。）将环境变量放在 profile 文件中，每次用户登录时这些变量值将被初始化。比如 HOME、USER、SHELL、UID 等再用户登录之前就已经被/bin/login 程序设置好了。

摘自《Linux Bible》

表 5-6 常见环境变量表

变量	解释
BASH	bash 命令的全路径地址，通常为 /bin/bash。
BASH_VERSION	bash 版本号。
EUID	这是当前用户的有效用户 ID 号，Shell 启动时分配，基于用户在 /etc/passwd 的分配。
FCEDIT	如果设置该变量，表示使用fc命令来编辑命令记录(history commands)；否则改用vi命令。
HISTFILE	命令记录文件，通常为 \$HOME/.bash_history。
HISTFILESIZE	命令记录条数，当超过此数，循环记录，最久的记录被删除。默认值 1000。
HISTCMD	返回当前命令处于命令记录的序号。
HOME	当前用户的主目录，也就是你登录后，或者输入cd回车跳到的文件夹。常用符号~表示主目录。
HOSTTYPE	当前计算机硬件体系，比如 Intel 兼容机，一般是 i386, i486, i586, i686, 或者 i386 等数值。对于 AMD 64 位机，一般是 x86_64。

变量	解释
MAIL	邮箱文件。深度普通用户一般都没有设置该变量，root 用户一般为 <code>/var/mail/root</code> 。对于 centos 系列一般为 <code>/var/spool/mail/</code>
OLDPWD	前一个工作目录。当你频繁在两个目录切换的时候，当前目录的前一个目录就存储在该变量。通常使用减号-表示。使用 <code>cd -</code> 可以跳转到前一个目录。
OSTYPE	本机操作系统类型。比如 <code>linux</code> 或 <code>linux-gnu</code>
PATH	路径变量。一般为一长串字符，用英文冒号:拼接多个文件夹。Windows 系统也有该变量，只不过使用英文分号;拼接的。处于上述文件夹下的可执行文件，只需输入本文件名不用输入完整路径，即可被 shell 找到执行。shell 是从前往后找的，重名可执行文件，需要注意执行的是不是自己的文件。不在 <code>PATH</code> 目录集合下的可执行文件，则需要输入绝对路径或者相对路径才可以执行。例如本机的 root 用户， <code>PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin</code> 。
PPID	当前 shell 父进程 ID 号。
PROMPT_COMMAND	深度操作系统没有设置该变量。通常该变量为一条命令名称。在命令提示符（shell prompt）每次出现之前执行。
PS1	设置命令提示符。有时还会有 <code>PS2</code> , <code>PS3</code> 等。
PWD	当前工作目录。
RANDOM	返回 0 到 99999 的一个随机数。
SECONDS	返回当前 shell 启用了多少秒数。
SHLVL	这是与当前 shell 会话相关联的 shell 级别的数目。当您登录到 shell 时， <code>SHLVL</code> 为 1。每次你新开 bash，比如使用 <code>su</code> 或者键入 <code>bash</code> ，该数字都会加一。
TMOUT	深度操作系统没有设置该变量。设置一定秒数，如果 shell 在设定时间内没有输入，则退出。对于服务器来说，还是跟安全很相关的一个变量。

可以用`echo`来显示查看全局变量 (eg: `echo $HOME`)。`env`(或`printenv`)、`set`也

可以用来查看系统的环境变量，但不是单独查看。而用`unset`临时取消环境变量(eg:`unset USER`)，要永久生效还是要写到配置文件中

自定义环境变量(采用`export`):

- `export 变量名 =value`
- `变量名 =value; export 变量名`
- `declare -x 变量名 =value`

这里依旧是临时生效，在shell终端关闭后就消失了，写到配置文件中则永久生效(注意：写到配置文件中后需要运行一遍配置文件的脚本才可生效，否则等重启时生效)

命令行的三种方式测试如下：

关于环境变量`PATH`与`export`的更详细的内容，可参考：[Linux 环境变量与系统编程学习笔记](#)

2、局部变量(本地变量):

本地变量在用户当前的shell生存期的脚本中使用。在一个函数中将某个变量声明为`local`，则该变量就是一个局部变量，只在本函数中有效。定义：

- `变量名 =value`
- `变量名 ='value'`
- `变量名 ="value"`

shell中变量名的要求：一般遵循字母、数字、下划线组成，不能以数字开头

5.5.1 别名的创建和使用

为了方便，常会配置一些别名。

```
1 $ alias l='ls -CF'
2 $ alias la='ls -A'
3 $ alias rm='rm -i'
4 $ alias
5 $ \ls
6 $ unalias la
7 $ alias
```

解释

1. 第1-3行，定义了三个别名。这些别名相当于后面的那个命令。
2. 第4行，查看当前所有别名。
3. 第5行，使用`\command`表示，使用最原始的命令，非别名。

4. 第 6-7 行，暂时取消别名`la`，然后查看当前所有的别名。这个`unalias`只可以在当前 shell 取消别名。

5.5.2 退出 shell

可以使用快捷键`ctrl+D`，或者输入`exit`，或者直接点击窗口的关闭按钮。

5.6 定制 shell 环境

`bash` 的配置文件，一般如下表所示，

表 5-7 Bash 配置文件说明

文件	解释
<code>/etc/profile</code>	影响所有用户，首次登录时执行。通常在此配置 <code>PATH</code> , <code>MAIL</code> , <code>HISTFILESIZE</code> 等。最后，该文件会读取 <code>/etc/profile.d</code> 文件夹下其他 shell 配置文件。
<code>/etc/bashrc</code>	影响所有 <code>bash</code> 用户，每次打开 <code>bash</code> 都会执行。通常在此配置提示（prompt） <code>PS1</code> 和一些别名。变量设置会被用户自己的 <code>~/.bashrc</code> 覆盖。
<code>~/.bash_profile</code>	只影响该用户，首次登录时执行。一般存放环境变量，并会调用 <code>~/.bashrc</code> 文件，特别适合放置环境变量。
<code>~/.bashrc</code>	只影响该用户，每次打开 <code>bash</code> 都会执行。存放一些自己的配置，特别适合放置别名。
<code>~/.bash_logout</code>	只影响该用户，每次注销（退出最后一个 shell 时）执行。通常只有清屏功能。

一般，修改`/etc/profile`,`/etc/bashrc` 需要 root 权限，而且影响所有用户。后面三个文件，只影响本用户。

编辑好这些文件，需要执行才可以生效。

```
1 $ gedit $HOME/.bashrc #编辑文件
2 $ source $HOME/.bashrc #执行生效
```

5.6.1 配置提示 (prompt)

参考网页：

- <https://www.cnblogs.com/lienhua34/p/5018119.html>
- <http://billie66.github.io/TLCL/book/chap14.html>

Linux 系统终端提示符，就是在前面5.2提到的\$或者##之前的文字。是通过环境变量PS1, PS2, PS3, PS4配置的。一般都是配置系统环境变量PS1（是“prompt string one” 的简写）定义。通过命令echo \$PS1查看当前设置。

```
1 $ echo $PS1
2 \[\e]0;\u@\h: \w\a\]\${debian_chroot:+($debian_chroot)
 }\\\[033[01;32m\]\u@\h\\\[033[00m\]:\\\[033[01;34m\]\w
 \\\[033[00m\]$\\setminus$\$
```

1、基本转义字符

PS1 的值由一系列静态文本或\和转义字符序列组成。示例，

```
1 $ PS1="\u@\H \w\$ "
```

表 5-8 Shell 提示符中用到的转义字符

序列	显示值
\a	以 ASCII 格式编码的铃声。当遇到这个转义序列时，计算机发出嗡嗡的响声。
\d	以日，月，天格式来表示当前日期。例如，“Mon May 26”
\h	本地机的主机名，但不带末尾的域名。
\H	完整的主机名。
\j	运行在当前 shell 会话中的工作数。
\l	当前终端设备名。
\n	一个换行符。
\r	一个回车符。
\s	shell 程序名。
\t	以 24 小时制，hours:minutes:seconds 的格式表示当前时间。

序列	显示值
\T	以 12 小时制表示当前时间。
\@	以 12 小时制, AM/PM 格式来表示当前时间, 例如 “10:51 PM”。
\A	以 24 小时制, hours:minutes 格式表示当前时间。
\u	当前用户名。
\v	shell 程序的版本号, 例如 4.3。
\V	shell 程序的版本号, 例如 4.3.11。
\w	当前工作目录名。
\W	当前工作目录名的最后部分。
\!	当前命令的历史号。
\##	当前 shell 会话中的命令数。
\\$	这会显示一个\$字符, 除非你拥有超级用户权限。在那种情况下, 它会显示一个##字符。
\[标志着一系列一个或多个非打印字符的开始。这被用来嵌入非打印的控制字符, 这些字符以某种方式来操作终端仿真器, 比方说移动光标或者是更改文本颜色。
\]	标志着非打印字符序列结束。

2、字体颜色

上面能够满足我们的效果了, 但是相对于 LinuxMint 原始的提示符, 缺少了颜色, 不太美观。下面我们来学习如何添加颜色。大多数终端仿真器程序支持一定的非打印字符序列来控制, 比方说字符属性 (像颜色, 黑体和可怕的闪烁) 和光标位置。

字体颜色是由一个 ANSI 转义编码来控制的。该控制编码会嵌入字符流中并发送给终端仿真器。但是, 该控制编码不会被“打印”到屏幕上, 而是会被终端解释为一个指令。正如我们在上表看到的字符序列, 这个\[和\]序列被用来封装这些非打印字符。一个 ANSI 转义编码以一个八进制 033 (这个编码是由退出按键产生的) 开头, 其后跟着一个可选的字符属性 (0: 正常、1: 黑体、4: 下划线、5: 闪烁、7: 反向 (前景色和背景色反转)), 在之后是一个指令。

表 5-9 用转义序列来设置文本颜色

序列	文本颜色	序列	文本颜色
\033[0;30m	黑色	\033[1;30m	深灰色
\033[0;31m	红色	\033[1;31m	浅红色
\033[0;32m	绿色	\033[1;32m	浅绿色
\033[0;33m	棕色	\033[1;33m	黄色
\033[0;34m	蓝色	\033[1;34m	浅蓝色
\033[0;35m	粉红	\033[1;35m	浅粉色
\033[0;36m	青色	\033[1;36m	浅青色
\033[0;37m	浅灰色	\033[1;37m	白色

例如我们来设置一个同 LinuxMint 默认的绿色提示符，

```
$ PS1="\[\033[01;32m\]\u@\W\$\$setminus\$ \$\[\033[00m\] "
```

于是，我们便有了下面的效果。

上面跟设置的提示符格式中的最后那个”\[\033[00m\]”是用于将后续的字符颜色还原回原来的颜色。如果没有加最后这个转义码，则会出现下面结果（我们自己手工输入的命令也都将是绿色的）。

3、背景颜色

除了字体颜色，我们也可以设置字体的背景颜色。同样是通过转义的控制编码来实现，下表是背景颜色的控制编码。

表 5-10 用转义序列来设置背景颜色

序列	文本颜色	序列	文本颜色
\033[0;40m	黑色	\033[1;44m	蓝色
\033[0;41m	红色	\033[1;45m	粉红
\033[0;42m	绿色	\033[1;46m	青色
\033[0;43m	棕色	\033[1;47m	浅灰色

4、移动光标

转义编码也可以用来定位光标。这些编码通常被用来，每次当提示符出现的

时候，会在屏幕的不同位置，比如说上面一个角落，显示一个时钟或者其它一些信息。下表是一系列用来定位光标的转义编码：

表 5-11 光标移动转义序列

转义编码	行动
\033[l;cH	把光标移到第 1 行，第 c 列。
\033[nA	把光标向上移动 n 行。
\033[nB	把光标向下移动 n 行。
\033[nC	把光标向前移动 n 个字符。
\033[nD	把光标向后移动 n 个字符。
\033[2J	清空屏幕，把光标移到左上角（第零行，第零列）。
\033[K	清空从光标位置到当前行末的内容。
\033[s	存储当前光标位置。
\033[u	唤醒之前存储的光标位置。

有兴趣的可以试着配置一下，建议先备份，在配置到上面提到的配置文件里。

5.7 命令帮助

<https://www.cnblogs.com/anliven/p/6030074.html>

Manual Page Chapter List

1. 所有用户可以操作的指令或可执行文件
2. 系统核心调用的函数与工具
3. 子调用，常用的函数与函数库
4. 设备，硬件文件说明，通常是 /dev/ 的文件
5. 文件格式，配置文件或者是某些档案的格式
6. 游戏相关
7. 杂项，例如 linux 文件系统、网络协议、ASCIIcode 等说明
8. 系统管理员可用的命令
9. 跟 kernel 有关的文件

表 5-12 命令帮助信息

命令	解释	注意点

表 5-12 命令帮助信息

命令	解释	注意点
type <command>	判断是否是内置命令	如果是外部命令，会给出简要信息
help <command>	显示简洁的帮助信息	适用内置命令
<command> -h/-help	显示简洁的帮助信息	适用外部命令。根据实际情况，使用“-h”或“-help”参数
whatis <command>	显示命令手册的页眉行	等同于 man -f 命令，可确认有哪些章节存在
man <command>	显示命令手册 (manual page)	包含完整的命令帮助，默认只显示第一章节内容。可在命令手册内查询关键字，方法类似 vi/vim，输入“q”退出浏览。 man -a <command> 显示命令手册的所有章节 man <chapter number> <command> 显示指定章节
info <command>	显示 info 文档信息	相比 man 命令，信息可能更新更详细，但使用方法复杂些，按“q”键退出类似命令：pinfo
which <command>	显示命令的完整路径	
whereis <command>	显示命令的路径、源码和手册等信息	
README	README 文件	绝大多数程序自带，保存在/usr/share/doc 文件夹

5.8 总结

本章整理了 shell 的一些基础知识，后面需要继续修改。

第六章 文件系统

通过阅读本章，你将会了解到以下几项内容。

- 了解文件系统
- 查看文件（夹）及属性
- 创建文件（夹）
- 查看设置文件（夹）权限和归属
- 文件（夹）的剪切复制粘贴删除和重命名

6.1 树形结构

参考网页：

- <https://blog.csdn.net/mzl87/article/details/79673012>
- <https://www.cnblogs.com/CoderJYF/p/6092604.html>
- <https://blog.csdn.net/fan0220/article/details/53079618>

常用文件夹说明。

- / —— 根目录，一般根目录下只存放目录，不要存放文件。/etc、/bin、/dev、/lib、/sbin 应该和根目录放置在一个分区中。
- /bin —— 存放系统中最常用的二进制可执行文件（二进制文件）。基础系统所需要的那些命令位于此目录，也是最小系统所需要的命令；例如 ls、cp、mkdir 等命令。功能和/usr/bin 类似，这个目录中的文件都是可执行的，普通用户都可以使用命令
- /boot —— 存放 Linux 内核和系统启动文件，包括 Grub、lilo 启动程序
- /dev —— 存放所有设备文件，包括硬盘、分区、键盘、鼠标、USB 等
- /etc —— 存放系统所有配置文件，例如 passwd 存放用户账户信息，hostname 存放主机名等。/etc/fstab 是开机自动挂载一些分区的，在里面写入一些分区信息，就能实现开机挂载分区
- /home —— 用户目录的默认位置
- /initrd —— 存放启动时挂载 initrd.img 映像文件的目录，以及载入所需设备模块的目录
- /lib —— 存放共享的库文件，包含许多被/bin 和/sbin 中程序使用的库文件
- /lost+found —— 在 ext2 或者 ext3 文件系统中，当系统意外崩溃或者计算

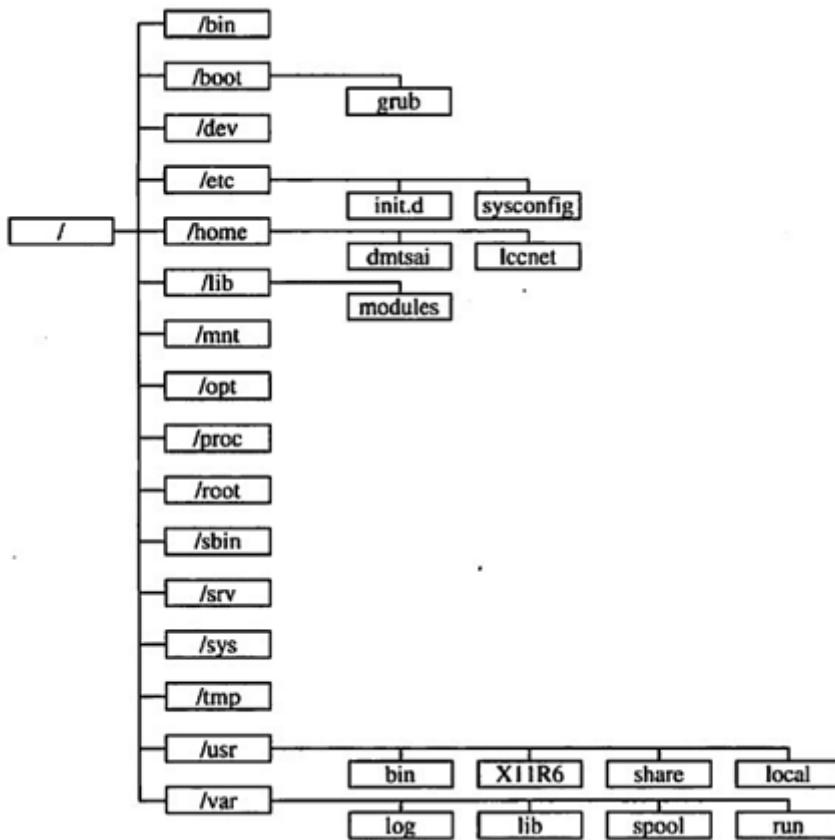


图 6-1 Linux 系统目录结构图

机意外关机，而产生一些文件碎片存放在这里。当系统启动的过程中 fsck 工具会检查这里，并修复已经损坏的文件系统。有时系统发生问题，有很多的文件被移动到这个目录中，可能会用手工的方式来修复或移动到文件的原位置上

- **/media** —— 即插即用型设备的挂载点自动在这个目录下创建。例如 USB 自动挂载后会在这个目录下产生一个目录；CDROM/DVD 自动挂载后，也会在这个目录中创建一个目录，存放临时读入的文件
- **/mnt** —— 此目录通常用于作为被挂载的文件系统的挂载点
- **/proc** —— 存放所有标志为文件的进程，它们是通过进程号或其他的系统动态信息进行标识。例如 CPU、硬盘分区、内存信息等存放在这里
- **/opt** —— 作为可选文件和程序的存放目录。有些软件包也会被安装在这里，也就是自定义软件包；有些用户自己编译的软件包，就可以安装在这个目录中
- **/root** —— 根用户（超级用户）的主目录

- **/sbin** —— 大多是涉及系统管理的命令的存放地，也是超级权限用户 root 的可执行命令存放地。普通用户无权限执行这个目录下的命令，这个目录和/usr/sbin; /usr/X11R6/sbin 或/usr/local/sbin 目录是相似的。注意，凡是目录 sbin 中包含的都是 root 权限才能执行的
- **/srv** —— 存放系统所提供的服务数据
- **/sys** —— 该目录用于将系统设备组织或层次结构，并向用户提供详细的内核数据信息
- **/tmp** —— 临时文件目录，有时用户运行程序的时候，会产生临时文件。
/var/tmp 目录和这个目录相似
- **/usr** —— 用于存放与系统用户直接有关的文件和目录，如应用程序及支出系统的库文件
 - **/usr/X11R6** —— X Window 系统
 - **/usr/bin** —— 用户管理员的标准命令
 - **/usr/include** —— C/C++ 等开发工具语言环境的标准 include 文件
 - **/usr/lib** —— 应用程序及程序报的链接库
 - **/usr/local** —— 系统管理员安装的应用程序
 - **/usr/local/share** —— 系统管理员安装的共享文件
 - **/usr/sbin** —— 用户和管理员的标准命令
 - **/usr/share** —— 存放使用手册等共享文件的地方
 - **/usr/share/dict** —— 存放词表的目录
 - **/usr/share/man** —— 系统使用手册
 - **/usr/share/misc** —— 一般数据
 - **/usr/share/sgml** —— SGML 数据
 - **/usr/share/xml** —— XML 数据
- **/var** —— 通常用于存放长度可变的文件，例如日志文件和打印机文件
 - **/var/cache** —— 应用程序缓存目录
 - **/var/crash** —— 系统错误信息
 - **/var/games** —— 游戏数据
 - **/var/lib** —— 各种状态数据
 - **/var/lock** —— 文件锁定记录
 - **/var/log** —— 日志记录
 - **/var/mail** —— 电子邮件
 - **/var/opt** —— /opt 目录的变量数据

- /var/run —— 进程的标示数据
- /var/spool —— 存放电子邮件，打印任务等的队列目录
- /var/tmp —— 临时文件目录

6.2 文件系统常用命令

表 6-1 文件系统常用命令

命令	解释
cd	改变目录
pwd	显示当前目录
mkdir	创建文件夹
chmod	修改文件（夹）权限
ls	列出文件夹内容

下面给出几个简单例子。

```

1 $ cd
2 $ cd ~
3 $ cd /usr/local/bin
4 $ cd -
5 $ cd ..
6 $ pwd

```

解释

1. cd 输入不带参数，默认是回到自己的主目录，也即~目录，第一行第二行效果是一样的。
2. cd 输入带一个目录，则跳转到该目录，如果你没有权限，则跳转失败。
3. -短横线表示上一个目录，也就是旧目录，方便你在两个目录之间切换。
4. cd 输入的目录支持相对路径和绝对路径。..表示上级目录或者叫父目录，.表示当前目录，对于/根目录，其父目录就是他本身。
5. pwd 打印当前目录。

```

1 $ cd
2 $ mkdir dir1

```

```

3 $ mkdir -p dir2/dir22
4 $ ls -ld dir1
5 drwxr-xr-x 2 litianci litianci 4096 10月 31 22:24 dir1
6 $ chmod o-rx dir1
7 $ touch dir1
8 $ ls -ld dir1
9 drwxr-x--- 2 litianci litianci 4096 10月 31 23:22 dir1

```

解释

1. cd 到主目录。使用mkdir新建一个文件夹。如果你想创建嵌套的文件夹，或者多层文件夹，可以使用mkdir -p选项，这样就可以建立多层文件夹。
2. ls -ld dir1, -d选项，表示只显示文件夹本身，-l是列出详细信息。drwxr-xr-x表示：d该文件是个文件夹，rwxr-xr-x是该文件夹的权限。见后面详述。2表示其内部有2个子文件（夹），从上面的命令可以知道其实dir1是一个空文件夹，怎么会有两个子文件（夹）呢，其实是隐藏的..和.文件夹。你可以使用ls -la dir1看到这两个文件夹。后面依次为本文件（夹）的用户、用户组、文件大小，修改日期和时间。最后是本文件名称。
3. 使用chmod命令，去除其他用户的读和执行权限 (-rx)
4. 使用touch命令，修改dir1的修改日期。
5. 再次使用ls -ld dir1命令，会发现相关的数值发生了变化。

6.3 元字符和操作符

古人云“物以类聚，人以群分”。我们操作文件（夹）的时候，可能就需要批量操作。这个时候我们就可以把这些文件（夹）“归类”，其实就是使用元字符表示某一类文件（夹）。这样操作起来更方便。

如果说元字符是对文件夹“分类”，那么操作符就是对“操作”按照一定的规则拼接。

6.3.1 元字符

常用的元字符有如下三种方式。

- * 匹配任意数目的字符，跟正则表达式的*有所区别。
- ? 匹配一个字符，一个汉字也是作为一个字符。
- [...] 匹配[]内的任意一个字符，支持短横线-表示连续的字符。一个汉字也是作为一个字符。

看下面例子，

```
1 $ mkdir test
2 $ cd test
3 $ touch huawei huashuo 华为 华硕 深度 深不可测
4 $ ls h*
5 huashuo  huawei
6 $ ls 华*
7 华硕  华为
8 $ ls 深*测
9 深不可测
```

解释

- 第 1-3 行，创建空的文件夹 test，并转到该文件夹，创建了 huawei 等六个文件。
- 第 4、6、8 行，使用元字符*表示任意数目的字符，分别显示了匹配相关“分类”的文件。

下面紧接着上面的 shell 环境，继续例子，

```
1 $ ls h?????
2 huawei
3 $ ls 华?
4 华硕  华为
5 $ ls 深?
6 深度
```

解释

- 第 1 行，可以看出每个?代表一个字符。
- 第 3、5 行，可以看出一个汉字算作一个字符。

下面我们看看 [...] 相关的例子，

```
1 $ touch auawei buawei cuawei duawei
2 $ ls [ah]uawei
3 auawei  huawei
4 $ ls [a-dh]uawei
5 auawei  buawei  cuawei  duawei  huawei
6 $ ls [a-dh]u*
7 auawei  buawei  cuawei  duawei  huashuo  huawei
```

解释

1. 第1行，为了方便演示，我新建了四个文件。
2. 第2行，[ah]uawei表示，a和h任选一个，后面紧挨着uawei。匹配的字符见第3行显示结果。
3. 第4行，我们可以看出短横线-的意思，表示一个序列。也支持数字的连续显示，但是只支持个位数，**是否如此请核实**。
4. 第6行，其实这些元字符可以配合使用。

6.3.2 操作符

本节内容见第[5.4.2](#)小节。本节从略。

6.3.3 花括号 {}

第 @ref(filesystem:metachar) (**是否如此请核实**)。如果你想创建一个集合，可以选择花括号{}的方式，列出所有相关的元素。

```
1 $ touch deepin{1,2,3,4}
2 $ ls deepin*
3 deepin1  deepin2  deepin3  deepin4
4 $ rm ./*
5 $ touch {deepin,深度}-{5,6,7,8}
6 $ ls
7 deepin-5  deepin-6  deepin-7  deepin-8  深度-5  深度-6  深度-7  深
     度-8
8 $ rm {deepin,深度}-{5,6,7,8}
9 $ ls
10 $ touch {a..e}{1..3}
11 $ ls
12 a1  a2  a3  b1  b2  b3  c1  c2  c3  d1  d2  d3  e1  e2  e3
13 $ rm ./*
14 $ touch {A..a}
15 touch: 无法创建'': 没有那个文件或目录
16 $ ls
17 ' ` _ ] A C E G I K M O Q S U W Y
18 '^ ' [ ' a B D F H J L N P R T V X Z
19 $ rm ./*
20 $ touch {aa..ac}
21 $ ls
22 {aa..ac}
```

```
23 $ rm ./*
24 $ touch {11..21}
25 $ ls
26 11 12 13 14 15 16 17 18 19 20 21
27 $ rm ./*
28 $ touch {33..22}
29 $ touch {z..b}
```

解释

1. 第1行，创建了`deepin`开头，以后面集合元素结尾的文件。
2. 第5行，集合拼接在一起，类似于笛卡尔积的形式组合起来。
3. 第10行，对于单字母支持序列功能，多字母则不支持。对于数字支持序列功能。按照ASCII码排序，从A到a包含不显示的字符，影响最终的显示。
4. 最后两行，没有显示结果，有兴趣的读者可以使用`ls`看看结果。

6.4 几个常用的目录

- `$HOME` 主目录
- ~ 波浪线，位于键盘左上角第二行开头的那个键（按下的时候需要按`shift`键），也是表示主目录，为了跳转到其他人的主目录，也可用使用`~<username>`的方式。比如跳转到`litianci`的主目录，`cd ~litianci`
- - 或者 `$OLDPWD`, 表示上一个目录，这个是短横线，不是减号。如果你在两个目录之间切换的话，`cd -`方便你跳转，如果你还没有切换目录，跳转到旧目录则会报错。`-`还有其他的功能，比如放权当前环境切换到新用户，`sudo su -`，这个命令就是不管你现在处于什么目录，切换到`root`根用户，且工作目录为`root`根用户的主目录。
- . 或 `$PWD` 当前工作目录。这两者是否有区别，待查。
- .. 上级目录，也即父目录。对于`/`根目录而言，`..`和`.`都表示本级目录。

6.5 文件（夹）的权限和归属

Linux 是多用户的系统，为了在一定程度上保护数据隐私，对文件（夹）进行简单的权限控制。这也是你输入命令或者做其他事情的时候会遇到的没有权限问题的根源，甚至部分用户喜欢用`root`根用户操作，也是因为权限控制造成的某些不方便。为了更好的用权，也为了数据的安全，还请能用普通用户的情况下，尽量少用`root`根用户操作。下面对权限进行简要介绍。

对于一个文件（夹）来说，他一定归属于某个用户，且归属于某个用户组。也一定有其他不属于前两者的用户，简称其他用户，有时候也想访问该文件（夹）。那么这个权限就是针对这三类人来设定的。当然，由于某些用户（组）被删除等，造成文件（夹）归属的用户（组）不存在了，这些就是无主文件（夹），这也是存在安全隐患的。在下章内容会讲述通过find命令查找此类无主文件（夹）的方法，可以采用相关方法，重新赋予其归属用户（组）或者删除等，这里就不叙述了。

6.5.1 权限简介

参考网页：

- <https://blog.csdn.net/zhao12795969/article/details/53448039>

针对这三类用户，权限又可以分为读（Read）、写（Write）和执行（eXecute）三类。

- r(Read, 读取)：对文件而言，具有读取文件内容的权限；对目录来说，具有浏览目录的权限。
- w(Write, 写入)：对文件而言，具有新增，修改，删除文件内容的权限；对目录来说，具有新建，删除，修改，移动目录内文件的权限。
- x(eXecute, 执行)：对文件而言，具有执行文件的权限；对目录来说该用户具有进入目录的权限。

注意

1. 目录的只读访问不允许使用 cd 进入目录，必须要有执行的权限才能进入。
2. 只有执行权限只能进入目录，不能看到目录下的内容，要想看到目录下的文件名和目录名，需要可读权限。
3. 一个文件能不能被删除，主要看该文件所在的目录对用户是否具有写权限，如果目录对用户没有写权限，则该目录下的所有文件都不能被删除，文件所有者除外。
4. 目录的 w 位不设置，即使你拥有目录中某文件的 w 权限也不能写该文件。
5. 以上仅限于 ext3/4 等文件系统的文件。对于安装 windows10 双系统的电脑，还需要注意是否 windows10 开启电源保护模式，哪怕你是 ext4 格式，依旧为只读，无法更改文件。

每个文件属于一个用户和一个组。命令ls -l可以查看用户和组，比如，

```
1 $ ls -l /bin/bash
2 -rwxr-xr-x 1 root root 1111240 5月 13 2018 /bin/bash
```

第二行为输出结果，表明 `/bin/bash` 属于 root 用户，root 组。`ls -l` 命令的第一个字段 `-rwxr-xr-x` 包含 `/bin/bash` 的权限的符号表示。该字段中的首字符（-）指定该文件的类型，本例中它是一个常规文件。可能的首字符如下：

- - 普通文件
- d 目录
- l 符号链接
- c 字符专门设备文件
- b 块专门设备文件
- p 先进先出
- s 套接字

首字符后紧接着三个三元组 `rwx`, `r-x`, `r-x`, 第一个三元字符组代表文件所有者的权限，第二个代表文件的所属组的权限，第三个代表其他用户的权限。但是 root 用户，往往会被跳过这些权限检查（待核实）。

6.5.2 权限更改 `chmod`

`chmod` 用于更改 `rwx` 权限，带有两个或多个参数：`mode`，描述怎样改变权限，后面跟将会受到影响的文件或文件列表：

```
1 $ chmod +x bashscript
```

上例`+x` 表示对 `bashscript` 的三类人员（拥有者、拥有组和其他人）均增加可执行权限。如果我们想要除去一个文件的所有执行权限，

```
1 $ chmod -x bashscript
```

为了更精细的控制权限，比如只修改部分人员权限，需要在`+-` 符号前加上这部分人员的代号。`a` 表示所有人员，`u` 表示拥有者，`g` 表示拥有组，`o` 表示其他人员。这些代号可以合用。比如下面这行命令表示组和其他人员去除读写权限增加执行权限。

```
1 $ chmod go-rw+x bashscript
```

对于权限，除了+-外，还可以使用等号=，表示只保留这些权限，未涉及的权限全部取消，有例外，见下文。

```
1 $ chmod =rx bashscript
```

注意

使用+--改变权限的时候，最好在前面加上相关人员的代号，比如a,u, g,o的组合，否则，可能跟你预期的结果不一样。

```
1 $ chmod a-rwx bashscript
2 $ chmod +rwx bashscript # + 不一定跟你期望的一样
3 $ ls -l bashscript # rwxr-xr-x
4 $ chmod a-rwx bashscript
5 $ chmod a+rwx bashscript
6 $ ls -l bashscript # rwxrwxrwx
7 $ chmod -rwx bashscript # - 不一定跟你期望的一样
8 $ ls -l bashscript # --w-w-
9 $ chmod a-rwx bashscript
10 $ ls -l bashscript # —
11 $ chmod =rwx bashscript
12 $ ls -l bashscript # rwxr-xr-x
```

以上是因为存在权限掩码umask这个参数，在终端输入umask回车，输出本机umask=0022，怎么是数字呢？这是因为权限也可用用数字表示，关于权限掩码umask在后面详述。

数字形式的权限，使用4位八进制数，每一位代表一个权限三元组。例如，在1777中，777设置本章我们所讨论的owner、group和other标志。1用来设置专门的权限位，稍后再讲。这个图表说明了怎样解释第二到

：字符形式权限和数字形式权限对照表

字符	数字	字符	数字	字符	数字	字符	数字
rwx	7	rw-	6	r-x	5	r--	4
-wx	3	-w-	2	--x	1	---	0

采用数字形式的权限管理方法，比如，

```
1 $ chmod 0755 bashscript  
2 $ ls -l bashscript  
3 -rwxr-xr-x 1 litianci litianci 119 11月 17 22:32 bashscript
```

就实现跟字符形式等同的权限设置。

6.5.3 权限掩码 umask

当进程创建了新文件时，它指定新文件应该具有的权限。通常，所请求的模式是 0666（每个人可读和可写），它比我们希望的具有更多的权限。幸运的是，不管什么时候创建了新文件，Linux 将参考叫做权限掩码umask的东西。系统用umask值来将初始指定的权限降低为更合理、更安全的权限。您可以通过在命令行中输入umask来查看您当前的umask设置，

```
1 $ umask  
2 0022
```

Linux 系统上，umask 的缺省值一般为 0022，它允许其他人读您的新文件（如果他们可以得到它们），但是不能进行修改。为了在缺省的情况下使新文件更安全，您可以改变 umask 设置，

```
1 $ umask 0077  
2 $ touch newfile # 新建一个文件  
3 $ ls -l newfile  
4 -rw----- 1 litianci litianci 0 11月 21 18:39 newfile
```

umask 将确保组和其他用户对于新创建的文件绝对没有任何权限。参阅上方的《字符形式权限和数字形式权限对照表》，当umask=0077时，对应字符形式权限为 ---rwxrwx。而新建文件的请求权限为0666（rw-rw-rw-），禁用掉0077后，就剩上方显示的rw-----权限了。

6.5.3.1 suid 和 sgid

当您最初登录时，将启动一个新的 shell 进程。这个新的 shell 进程（通常是 bash）使用您的用户标识运行，可以访问所有属于您的文件（夹）。您启动的程序继承了您的用户标识，因此它们不能访问任何不允许您访问的文件系统对象。例

如，一般用户不能直接修改 */etc/passwd* 文件，因为 write 标志已经对除 root 用户以外的每个用户关闭，

```
1 $ ls -l /etc/passwd  
2 -rw-r--r-- 1 root root 2438 11月 21 15:54 /etc/passwd
```

但是，一般用户确实需要在他们需要改变其密码的任何时候，能够修改 */etc/passwd* (至少间接地)。但是，如果用户不能修改该文件，究竟怎样完成这个工作呢？

幸好，Linux 权限模型有两个专门的位，叫做 *suid* 和 *sgid*。当设置了一个可执行程序的 *suid* 这一位时，它将代表可执行文件的所有者运行，而不是代表启动程序的人运行。现在回到上面这个问题。查看 */usr/bin/passwd* 可执行文件，

```
1 $ ls -l /usr/bin/passwd  
2 -rwsr-xr-x 1 root root 59640 9月 28 2017 /usr/bin/passwd
```

您还将注意到，这里有一个 *s* 取替了用户权限三元组中的一个 *x*。这表明，对于这个特殊程序，设置了 *suid* 和可执行位。由于这个原因，当 */usr/bin/passwd* 运行时，它将代表 root 用户执行 (具有完全超级用户访问权)，而不是代表运行它的用户运行。又因为 */usr/bin/passwd* 以 root 用户访问权运行，所以能够修改 */etc/passwd* 文件，而没有什么问题。

我们看到了 *suid* 怎样工作，*sgid* 以同样的方式工作。它允许程序继承程序的组所有权，而不是当前用户的组所有权。

这里有一些关于 *suid* 和 *sgid* 的其它的但是很重要的信息。

首先，*suid* 和 *sgid* 分别占据与 *ls -l* 清单中用户 *u* 和用户组 *g* 的执行 *x* 位。如果设置了 *x* 位为可执行，则相应的位表示为 *s* (小写) 否则为 *S* (大写)。

其次，在许多环境中，*suid* 和 *sgid* 很管用，但是不恰当地使用这些位可能使系统的安全遭到破坏。最好尽可能地少用 *suid* 程序。*/usr/bin/passwd* 命令是为数不多的必须使用 *suid* 的命令之一。

设置和除去 *suid* 与 *sgid* 位相当简单。这里，我们设置 *suid* 位：

```
1 $ ls -l bashscript  
2 -rwxr-xr-x 1 litianci litianci 119 11月 17 22:32 bashscript  
3 $ chmod u+s bashscript
```

```

4 $ ls -l bashscript
5 -rwsr-xr-x 1 litianci litianci 119 11月 17 22:32 bashscript
6 $ chmod u-x bashscript
7 $ ls -l bashscript
8 -rwSr-xr-x 1 litianci litianci 119 11月 17 22:32 bashscript
9 $ chmod g+s bashscript
10 $ ls -l bashscript
11 -rwSr-sr-x 1 litianci litianci 119 11月 17 22:32 bashscript
12 $ chmod g-s bashscript
13 $ ls -l bashscript
14 -rwSr-xr-x 1 litianci litianci 119 11月 17 22:32 bashscript

```

权限和目到此为止，我们从常规文件的角度来看权限。当从目录的角度看权限时，情况有一点不同。目录使用同样的权限标志，但是它们被解释为表示略微不同的含义。对于一个目录，如果设置了read标志，您可以列出目录的内容；write表示您可以在目录中创建文件，execute表示您可以进入该目录并访问内部的任何子目录。没有execute标志，目录内的文件系统对象是不可访问的。没有read标志，目录内的文件系统对象是不可查看的，但是只要有人知道磁盘上对象的完整路径，就仍然可以访问目录内的对象。

如果启用了目录的sgid标志，在目录内创建的任何文件系统对象将继承目录的组。当您需要创建一个属于同一组的一组人使用的目录树时，这种特殊的功能很管用。只需要这样做，

```

1 $ mkdir -p /media/litianci/data/groupspace
2 $ sudo chgrp mygroup /media/litianci/data/groupspace
3 $ chmod g+s /media/litianci/data/groupspace

```

现在，mygroup 组中的所有用户都可以在 */media/litianci/data/groupspace* 内创建文件或目录，同样，他们也将自动地分配到 mygroup 的组所有权。根据用户的umask 设置，新文件系统对象对于 mygroup 组的其他成员来说，可以或不可以是可读、可写或可执行的。

6.5.3.2 目录和删除

缺省情况下，Linux 目录以一种不是在所有情况下都很理想的方式表现。一般来说，只要对一个目录有写访问权，任何人都可以重命名或删除该目录中的文件。对于个别用户使用的目录，这种行为是很合理的。

但是，对于很多用户使用的目录来说，尤其是 `/tmp` 和 `/var/tmp`，这种行为可能会产生麻烦。因为任何人都可以写这些目录，任何人都可以删除或重命名任何其他人的文件——即使是不属于他们的！显然，当任何其他用户在任何时候都可以输入 `rm -rf /tmp/*` 并损坏每个人的文件时，很难在 `/tmp` 存放任何有意义的文件。

所幸，Linux 提供了“粘滞位”（sticky bit）的东西。当给 `/tmp` 设置了粘滞位（用 `chmod +t`），唯一能够删除或重命名 `/tmp` 中文件的是该目录的所有者（通常是 root 用户）、文件的所有者或 root 用户。事实上，所有 Linux 分发包都缺省地启用了 `/tmp` 的粘滞位，而您还可以发现粘滞位在其它情况下也很管用。很显然，“粘滞位”是属于其他类型人员的权限，跟 `suid,sgid` 一样，占据的也是权限三元组的 x 位，小写是设置了执行权限，大写是禁用了执行权限。所不同的是他们分别占据 u,g, o 的 x 位。

```

1 $ ls -dl /tmp
2 drwxrwxrwt 15 root root 4096 11月 22 10:19 /tmp
3 $ sudo chmod o-x /tmp
4 $ ls -dl /tmp
5 drwxrwxrwT 16 root root 4096 11月 22 10:25 /tmp

```

6.5.3.3 数字形式权限的第一位数字

现在我们回过头来看数字权限的第一位数字的意思，这也是一个八进制数字，0-7，转化为三位的二进制数，从高位到低位，分别表示 `suid,sgid,sticky`。当设置该权限时为 1，禁用该权限时为 0。比如，3 就表示，禁用 `suid`, 启用 `sgid,sticky`。看下面例子。

```

1 $ chmod 1777 bashscript
2 $ ls -l bashscript
3 -rwxrwxrwt 1 litianci litianci 119 11月 17 22:32 bashscript
4 $ chmod 2777 bashscript
5 $ ls -l bashscript
6 -rwxrwsrwx 1 litianci litianci 119 11月 17 22:32 bashscript
7 $ chmod 4777 bashscript
8 $ ls -l bashscript
9 -rwsrwxrwx 1 litianci litianci 119 11月 17 22:32 bashscript

```

6.5.4 归属

参考网页：

- <https://www.cnblogs.com/DawaTech/p/7249734.html>

语法结构如下，

```
1 chgrp 组名 文件（夹） -R
2 chown 用户名 文件（夹） -R
```

其中`chgrp`表示改变组名；`chown`表示改变用户名；`-R`表示递归目录下所有文件，可略。更改所属用户和所属组常常需要根用户权限。

6.5.4.1 修改文件所属组群 `chgrp`

`chgrp`，就是 `change group` 的缩写（我们可以利用这些来记忆命令）。看例子，

```
1 [litianci]$ sudo groupadd groupa # 增加组 groupa
2 [litianci]$ sudo groupadd groupb # 增加组 groupb
3 [litianci]$ sudo useradd -g groupa deepin -m # 新增用户 deepin，默认组 groupa，自带主目录
4 [litianci]$ sudo su - deepin # 切换到 deepin 用户，- 表示不保存当前任何环境变量
5 [deepin]$ touch filea fileb # 创建两个文件
6 [deepin]$ ls -l
7 -rw-r--r-- 1 deepin groupa    0 11月 21 15:55 filea
8 -rw-r--r-- 1 deepin groupa    0 11月 21 15:55 fileb
9 [deepin]$ exit
10 [litianci]$ sudo chgrp groupb ~deepin/filea # 改变 filea 所属群组
11 [litianci]$ sudo ls -l ~deepin/
12 -rw-r--r-- 1 deepin groupb    0 11月 21 15:55 filea
13 -rw-r--r-- 1 deepin groupa    0 11月 21 15:55 fileb
```

由于新增用户权限的问题，后面使用其他用户（`ls -l`）访问文件的属性。执行命令的用户放在每行命令的开头了。

6.5.4.2 修改文件拥有者 `chown`

`chown`，即 `change owner` 改变归属用户。`chown`功能很多，不仅仅能更改文件拥有者，还可以修改文件所属组群。如果需要将某一目录下的所有文件都改变其拥有者，可以使用`-R`参数。

语法结构如下，

```
1 chown [-R] 账号名称    文件/目录  
2 chown [-R] 账号名称:组群  文件/目录
```

下面简单示例，

```
1 $ touch demo  
2 $ ls -l demo  
3 -rw-r--r-- 1 litianci litianci 0 11月 21 16:13 demo  
4 $ sudo chown deepin demo # 修改文件 demo 的拥有者为 deepin  
5 $ ls -l demo  
6 -rw-r--r-- 1 deepin litianci 0 11月 21 16:13 demo  
7 $ sudo chown deepin:groupa demo # 修改 demo 的拥有者为 deepin, 拥有  
     组 groupa  
8 -rw-r--r-- 1 deepin groupa 0 11月 21 16:13 demo
```

6.6 文件（夹）的创建、查看、移动、复制和删除

有点类似数据库的增删改查（CRUD,Create,Read,Update,Delete），这里只提到了文件（夹）的创建、查看、移动（包括重命名）、复制和删除。修改涉及到文件的权限以及日期等等内容，这里略过不提。

6.6.1 文件（夹）的创建

对于文件夹的创建，常用`mkdir`这个命令。具体方法，前面已经讲解了。对于文件的创建，通常使用`touch`创建文件，通过`man touch`可以看出，一般情况，`touch`只是对访问和更改的时间戳重置为当前时间戳，创建文件只是附带功能。另外一般的编辑器，还有其他文件都有新建保存功能，这样也可用创建文件。这里就不详细介绍。

6.6.2 文件（夹）的查看

对于文件的内容查看，可以通过各种阅读器或者编辑器等，根据文件的类型做出不同选择。因为 Unix 哲学奉行“一切皆文件”，编辑器是一种比较常见的查看方式，这个见第七章。

文件（夹）的查看，通常采用**深度文件管理器**。对于其属性，可以通过**深度文件管理器**右击找到属性来查看。当然在命令行下`ls`也是一个非常强大的存在。看下面的例子。

```
1 $ ls
2 $ ls -la
3 $ ls -lR
```

解释

1. `ls`的选项`-a`表示把当前包含隐藏文件（夹）在内的所有文件（夹）显示出来。
2. `-R`选项是递归显示，可以把各个子文件夹的内容也显示出来。`-l`的显示内容，见上面章节说明。更多内容，可以查看`man ls`。**文件（夹）的移动及重命名**

文件（夹）的移动和重命名是通过`mv`命令实现的。

```
1 $ mv litianci deepin
2 $ mv deepin ~
3 $ mv test1 test2 test3 ~
4 $ mv ~/deepin-bible/ ~/test/
5 $ mv ~/Desktop/* ~/test/
```

解释

1. 第1行，对文件`litianci`重命名为`deepin`。`mv`重命名时，不区分文件夹或文件。
2. 第2行，把文件`deepin`移动到`~`文件夹。
3. 第3行，`mv`支持多个文件同时移动到最后一个文件夹。请注意，**当多个文件（夹）同时移动时，最后一个必须是文件夹或指向文件夹的链接**。
4. 第4行，把文件夹`deepin-bible`文件夹，包括文件夹的内容（含隐藏内容），移动到`~/test/`文件夹。
5. 第5行，只是把文件夹`Desktop`内的非隐藏内容，移动到`~/test/`文件夹。

注意，默认的`mv`，对于重名冲突是直接覆盖的。如果想不被覆盖，建议在`~/.bashrc`内加上`alias mv='mv -i'`，遇到这类的问题的时候，就会征询你的意见再决定要不要覆盖。

6.6.3 文件（夹）的复制

```
1 $ cp litianci deepin
2 $ cp deepin ~
3 $ cp test1 test2 test3 ~
4 $ cp -r ~/deepin-bible/ ~/test/
5 $ cp -ra ~/Desktop/* ~/test/
```

解释

1. 第 1-3 行，类似mv命令，实现对文件的复制，不含文件夹的复制。
2. 第 4 行，对于文件夹的复制，需要带-r选项，递归（recursive）的意思。
3. 第 5 行，复制-a选项，表示复制后的文件（夹）各时间戳和权限跟原文件（夹）一样。

6.6.4 文件（夹）的删除

```
1 $ rm deepin
2 $ rm ./*
3 $ rm *
4 $ rmdir /home/joe/nothing/
5 $ rm -r /home/joe/bigdir/
6 $ rm -rf /home/joe/hugedir/
7 $ sudo rm -rf / #千万不要试运行
```

解释

1. 第 1 行，普通文件可以直接删除。不支持文件夹直接删除。如果要删除文件夹见第 4-6 行。
2. 第 2 行，对于特殊文件，比如-，一般写绝对路径。
3. 第 3 行，*一般代表的都是非隐藏文件（夹）。没有-r递归选项，也可用删除当前文件夹下的所有非隐藏内容。
4. 第 4 行，删除文件夹的命令rmdir，删除了nothing文件夹及其内容。
5. 第 5-6 行，-r表示递归，可以使用rm删除文件夹。-f是强制删除，实际是不用询问直接删除，对于没有权限删除的，依旧无法删除。
6. 第 7 行，千万不要试运行。但是有时候为了删除本文件夹下的内容，往往错输。

我的建议是`alias rm='trash-put'`，把删除改为放置到回收站。具体做法见附录己。

6.7 总结

本章介绍了 Linux 操作系统的文件结构，并对文件的权限和归属情况进行了介绍，最后对文件的改动复制等作了简要介绍。

第七章 文件编辑与查找

通过阅读本章，你将会了解到以下几项内容。

- 字符集和字符编码
- 编辑器简介
- 文件查找

7.1 字符集和字符编码

参考网页：

- <https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86/946585?fr=aladdin>
- <https://www.cnblogs.com/happyday56/p/4135845.html>
- <https://linux.cn/article-7959-1.html>

提到文本文件，不得不说文件的字符集，对于英文世界，ASCII 码足够他们玩耍的了，但是对于中文，我们还需要中文字符集，不然就会显示很多框框或者乱码等。

字符 (Character) 是各种文字和符号的总称，包括各国家文字、标点符号、图形符号、数字等。字符集 (Character set) 是多个字符的集合，字符集种类较多，每个字符集包含的字符个数不同，常见字符集名称：ASCII 字符集、GB2312 字符集、BIG5 字符集、GB18030 字符集、Unicode 字符集等。计算机要准确的处理各种字符集文字，需要进行字符编码，以便计算机能够识别和存储各种文字。中文文字数目大，而且还分为简体中文和繁体中文两种不同书写规则的文字，而计算机最初是按英语单字节字符设计的，因此，对中文字符进行编码，是中文信息交流的技术基础。

7.1.1 ASCII

ASCII (American Standard Code for Information Interchange, 美国信息互换标准编码) 是基于罗马字母表的一套电脑编码系统。

- **特点**它主要用于显示现代英语和其他西欧语言。它是现今最通用的单字节编码系统，并等同于国际标准 ISO 646。
- **包含内容**

- 控制字符：回车键、退格、换行键等。
- 可显示字符：英文大小写字符、阿拉伯数字和西文符号。
- **技术特征** 7 位 (bits) 表示一个字符，共 128 字符，字符值从 0 到 127，其中 32 到 126 是可打印字符。
- **扩展字符集** 7 位编码的字符集只能支持 128 个字符，为了表示更多的欧洲常用字符对 ASCII 进行了扩展，ASCII 扩展字符集使用 8 位 (bits) 表示一个字符，共 256 字符，增加了表格符号、计算符号、希腊字母和特殊的拉丁符号。

7.1.2 GB2312

GB2312 又称为 GB2312-80 字符集，全称为《信息交换用汉字编码字符集·基本集》，由原中国国家标准总局发布，1981 年 5 月 1 日实施，是中国国家标准的简体中文字字符集。

它所收录的汉字已经覆盖 99.75% 的使用频率，基本满足了汉字的计算机处理需要。在中国大陆和新加坡获广泛使用。

包含内容

GB2312 收录简化汉字及一般符号、序号、数字、拉丁字母、日文假名、希腊字母、俄文字母、汉语拼音符号、汉语注音字母，共 7445 个图形字符。其中包括 6763 个汉字，其中一级汉字 3755 个，二级汉字 3008 个；包括拉丁字母、希腊字母、日文平假名及片假名字母、俄语西里尔字母在内的 682 个全角字符。

技术特征

(1) 分区表示：

GB2312 中对所收汉字进行了“分区”处理，每区含有 94 个汉字/符号。这种表示方式也称为区位码。各区包含的字符如下：01-09 区为特殊符号；16-55 区为一级汉字，按拼音排序；56-87 区为二级汉字，按部首/笔画排序；10-15 区及 88-94 区则未有编码。

(2) 双字节表示

两个字节中前面的字节为第一字节，后面的字节为第二字节。习惯上称第一字节为“高字节”，而称第二字节为“低字节”。“高位字节”使用了 0xA1-0xF7(把 01-87 区的区号加上 0xA0)，“低位字节”使用了 0xA1-0xFE(把 01-94 加上 0xA0)。

以 GB2312 字符集的第一个汉字“啊”字为例，它的区号 16，位号 01，则区位码是 1601，在大多数计算机程序中，高字节和低字节分别加 0xA0 得到程序的汉字处理编码 0xB0A1。计算公式是： $0xB0=0xA0+16$, $0xA1=0xA0+1$ 。

7.1.3 GBK 字符集

GBK 编码 (Chinese Internal Code Specification) 是中国大陆制订的、等同于 UCS 的新的中文编码扩展国家标准。gbk 编码能够用来同时表示繁体字和简体字，而 gb2312 只能表示简体字，gbk 是兼容 gb2312 编码的。GBK 工作小组于 1995 年 10 月，同年 12 月完成 GBK 规范。该编码标准兼容 GB2312，共收录汉字 21003 个、符号 883 个，并提供 1894 个造字码位，简、繁体字融于一库。Windows95/98 简体中文版的字库表层编码就采用的是 GBK，通过 GBK 与 UCS 之间一一对应的码表与底层字库联系。

- 英文名：Chinese Internal Code Specification
- 中文名：汉字内码扩展规范 1.0 版
- 双字节编码，GB2312-80 的扩充，在码位上和 GB2312-80 兼容
- 范围：8140~FEFE（剔除 xx7F）共 23940 个码位。包含 21003 个汉字，包含了 ISO/IEC 10646-1 中的全部中日韩汉字。
- 作用：它是 GB2312 的扩展，加入对繁体字的支持，兼容 GB2312。
- 位数：使用 2 个字节表示，可表示 21886 个字符。
- 范围：高字节从 81 到 FE，低字节从 40 到 FE。

7.1.4 BIG5

大五码或五大码，1984 年由台湾财团法人信息工业策进会和五家软件公司宏碁 (Acer)、神通 (MiTAC)、佳佳、零壹 (Zero One)、大众 (FIC) 创立，故称大五码。

Big5 码的产生，是因为当时台湾不同厂商各自推出不同的编码，如倚天码、IBM PS55、王安码等，彼此不能兼容；另一方面，台湾政府当时尚未推出官方的汉字编码，而中国大陆的 GB2312 编码亦未有收录繁体中文字。

Big5 字符集共收录 13,053 个中文字，该字符集在中国台湾使用。耐人寻味的是该字符集重复地收录了两个相同的字：“兀”(0xA461 及 0xC94A)、“方”(0xDDC1 及 0xDDFC)。

编码方法

Big5 码使用了双字节储存方法，以两个字节来编码一个字。第一个字节称为“高位字节”，第二个字节称为“低位字节”。高位字节的编码范围 0xA1-0xF9，低位字节的编码范围 0x40-0x7E 及 0xA1-0xFE。

各编码范围对应的字符类型如下：0xA140-0xA3BF 为标点符号、希腊字母及特殊符号，另外于 0xA259-0xA261，存放了双音节度量衡单位用字：方方方方方 方方方方；0xA440-0xC67E 为常用汉字，先按笔划再按部首排序；0xC940-0xF9D5

为次常用汉字，亦是先按笔划再按部首排序。

局限性

尽管 Big5 码内包含一万多个字符，但是没有考虑社会上流通的人名、地名用字、方言用字、化学及生物科等用字，没有包含日文平假名及片假名字母。

例如台湾视“着”为“著”的异体字，故没有收录“着”字。康熙字典中的一些部首用字(如“一”、“广”、“匚”、“匚”等)、常见的人名用字(如“匱”、“煊”、“匱”、“匱”等)也没有收录到 Big5 之中。

7.1.5 GB18030

GB18030 的全称是 GB18030-2000《信息交换用汉字编码字符集基本集的扩充》，是我国政府于 2000 年 3 月 17 日发布的新的汉字编码国家标准，2001 年 8 月 31 日后在中国市场上发布的软件必须符合本标准。

该标准解决汉字、日文假名、朝鲜语和中国少数民族文字组成的大字符集计算机编码问题。该标准的字符总编码空间超过 150 万个编码位，收录了 27484 个汉字，覆盖中文、日文、朝鲜语和中国少数民族文字。满足中国大陆、香港、台湾、日本和韩国等东亚地区信息交换多文种、大字量、多用途、统一编码格式的要求。并且与 Unicode 3.0 版本兼容，填补 Unicode 扩展字符字汇“统一汉字扩展 A”的内容。并且与以前的国家字符编码标准（GB2312，GB13000.1）兼容。

编码方法

GB18030 标准采用单字节、双字节和四字节三种方式对字符编码。单字节部分使用 0xE00 至 0xE7F 码(对应于 ASCII 码的相应码)。双字节部分，首字节码从 0xE81 至 0xEF0，尾字节码位分别是 0xE40 至 0xE7E 和 0xE80 至 0xEF0。四字节部分采用 GB/T 11383 未采用的 0xE30 到 0xE39 作为对双字节编码扩充的后缀，这样扩充的四字节编码，其范围为 0xE81308130 到 0xEF039FE39。其中第一、三个字节编码码位均为 0xE81 至 0xEF0，第二、四个字节编码码位均为 0xE30 至 0xE39。

包含内容

双字节部分收录内容主要包括 GB13000.1 全部 CJK 汉字 20902 个、有关标点符号、表意文字描述符 13 个、增补的汉字和部首/构件 80 个、双字节编码的欧元符号等。四字节部分收录了上述双字节字符之外的，包括 CJK 统一汉字扩充 A 在内的 GB 13000.1 中的全部字符。

7.1.6 Unicode 字符集

Unicode 字符集（简称为 UCS），国际标准组织于 1984 年 4 月成立 ISO/IEC JTC1/SC2/WG2 工作组，针对各国文字、符号进行统一性编码。1991 年美国跨国公司成立 Unicode Consortium，并于 1991 年 10 月与 WG2 达成协议，采用同一编码字集。目前 Unicode 是采用 16 位编码体系，其字符集内容与 ISO10646 的 BMP (Basic Multilingual Plane) 相同。Unicode 于 1992 年 6 月通过 DIS (Draft International Standard)，目前版本 V2.0 于 1996 公布，内容包含符号 6811 个，汉字 20902 个，韩文拼音 11172 个，造字区 6400 个，保留 20249 个，共计 65534 个。Unicode 编码后的大小是一样的。例如一个英文字母“a”和一个汉字“好”，编码后都是占用的空间大小是一样的，都是两个字节！

Unicode 可以用来表示所有语言的字符，而且是定长双字节（也有四字节的）编码，包括英文字母在内。所以可以说它是不兼容 iso8859-1 编码的，也不兼容任何编码。不过，相对于 iso8859-1 编码来说，unicode 编码只是在前面增加了一个 0 字节，比如字母'a'为“00 61”。

需要说明的是，定长编码便于计算机处理（注意 GB2312/GBK 不是定长编码），而 unicode 又可以用来表示所有字符，所以在很多软件内部是使用 unicode 编码来处理的，比如 java。

UNICODE 字符集有多个编码方式，分别是 UTF-8，UTF-16，UTF-32 和 UTF-7 编码。

7.1.6.1 UTF-8

UTF:UCS Transformation Format. 考虑到 unicode 编码不兼容 iso8859-1 编码，而且容易占用更多的空间：因为对于英文字母，unicode 也需要两个字节来表示。所以 unicode 不便于传输和存储。因此而产生了 utf 编码，utf 编码兼容 iso8859-1 编码，同时也可用来表示所有语言的字符，不过，utf 编码是不定长编码，每一个字符的长度从 1-6 个字节不等。另外，utf 编码自带简单的校验功能。一般来讲，英文字母都是用一个字节表示，而汉字使用三个字节。

注意，虽然说 utf 是为了使用更少的空间而使用的，但那只是相对于 unicode 编码来说，如果已经知道是汉字，则使用 GB2312/GBK 无疑是最节省的。不过另一方面，值得说明的是，虽然 utf 编码对汉字使用 3 个字节，但即使对于汉字网页，utf 编码也会比 unicode 编码节省，因为网页中包含了很多的英文字符。

UTF8 编码后的大小是不一定，例如一个英文字母a和一个汉字好，编码后占用的空间大小就不一样了，前者是一个字节，后者是三个字节！编码的方法是从低位

到高位。黄色为标志位其它着色为了显示其，编码后的位置。

7.1.6.2 UTF-16

采用 2 字节，Unicode 中不同部分的字符都同样基于现有的标准。这是为了便于转换。从 0x0000 到 0x007F 是 ASCII 字符，从 0x0080 到 0x00FF 是 ISO-8859-1 对 ASCII 的扩展。希腊字母表使用从 0x0370 到 0x03FF 的代码，斯拉夫语使用从 0x0400 到 0x04FF 的代码，美国使用从 0x0530 到 0x058F 的代码，希伯来语使用从 0x0590 到 0x05FF 的代码。中国、日本和韩国的象形文字（总称为 CJK）占用了从 0x3000 到 0x9FFF 的代码；

由于 0x00 在 c 语言及操作系统文件名等中有特殊意义，故很多情况下需要 UTF-8 编码保存文本，去掉这个 0x00。举例如下：

- UTF-16: 0x0080 = 0000 0000 1000 0000
- UTF-8: 0xC280 = 1100 0010 1000 0000

7.1.6.3 UTF-32

采用 4 字节。

7.1.6.4 UTF-7

A Mail-Safe Transformation Format of Unicode(RFC1642)。这是一种使用 7 位 ASCII 码对 Unicode 码进行转换的编码。它的设计目的仍然是为了在只能传递 7 位编码的邮件网关中传递信息。UTF-7 对英语字母、数字和常见符号直接显示，而对其他符号用修正的 Base64 编码。符号 + 和 - 号控制编码过程的开始和暂停。所以乱码中如果夹有英文单词，并且相伴有 + 号和 - 号，这就有可能是 UTF-7 编码。

- 作用：为世界 650 种语言进行统一编码，兼容 ISO-8859-1。
- 位数：UNICODE 字符集有多个编码方式，分别是 UTF-8，UTF-16 和 UTF-32。

7.1.7 编码的常用命令

在 Linux 下通常使用 UTF-8 编码。如果你需要对用户名排序的话，建议使用 GB18030 编码，以便直接排序。各种编码也是可以相互转换的。

如果我们想查看某个文件的编码方式，可以使用 `file -i` 命令。

```
1 $ file -i test.md
2 test.md: text/plain; charset=utf-8
```

这样就可以看到 **test.md** 文件的属性普通文本文件text/plain以及编码方式为UTF-8, charset=utf-8。

一般的编辑器都是支持编码方式转换的，如果你想改变某个文件的编码方式，可以使用iconv工具。iconv工具的使用方法如下：

```
1 $ iconv option
2 $ iconv options -f from-encoding -t to-encoding inputfile(s) -o
   outputfile
```

在这里，-f或--from-code表明了输入编码，而-t或--to-encoding指定了输出编码。

为了列出所有已有编码的字符集，你可以使用以下命令：

```
1 $ iconv -l
2 以下的列表包含所有已知的编码字符集，但这不代表所有的字符名称组合皆可用于
3 命令行的 "来源" 以及 "目的" 参数。一个编码字符集可以用几个不同的名称
4 来表示（即 "别名"）。
5
6 437, 500, 500V1, 850, 851, 852, 855, 856, 857, 858, 860, 861,
   862, 863, 864,
7 865, 866, 866NAV, 869, 874, 904, 1026, 1046, 1047, 8859_1,
   8859_2, 8859_3,
8 8859_4, 8859_5, 8859_6, 8859_7, 8859_8, 8859_9, 10646-1:1993,
9 10646-1:1993/UCS4, ANSI_X3.4-1968, ANSI_X3.4-1986, ANSI_X3.4,
```

下面举个小例子。其中 *test.md* 文件是包含中英文的。

在运行iconv命令之后，我们可以像下面这样检查输出文件的内容，和它使用的字符编码。

```
1 $ file -i test.md
2 test.md: text/plain; charset=utf-8
3 $ cat test.md
4 : Shell 提示符中用到的转义字符
5 $ iconv -f utf-8 -t GB18030 test.md -o out.md
```

```
6 $ cat out.md  
7 : Shell ┌─'─┐  
8 $ file -i out.md  
9 out.md: text/plain; charset=iso-8859-1
```

你会发现转换后，电脑没有正确识别出 *out.md* 文件的编码，如果你使用文本编辑器gedit打开 *out.md* 文件，还是可以正常打开的。

7.2 编辑器

办公常接触的文档，一般是有格式的文件，比如 xls, doc 等，他们常常需要相应的办公软件，比如金山公司的 WPS。本章提到的文字编辑，是指文本文件的编辑。讲到的编辑器有 vim, gedit, nano, vscode 等。

本人常用 vim 写代码，也自嗨的分享了自己的使用经验教训，但是编写书籍的话，我还是喜欢用 vscode。各有所长吧。大家可以根据需要搜索相关入门教程。

7.3 文件查找

文件查找用得好，绝对是一件利器。有人推荐 locate 命令，本人感觉不太好用，就不介绍了。下面介绍 find 和 grep 这两个命令，其中 find 负责查找文件的信息，grep 负责查找文件内的信息。当然提到搜索不得不提正则表达式，适当了解一些正则表达式，对于我们搜索文件还是很有帮助的，且 find, grep 以及一些编辑器是支持正则表达式搜索的。

7.3.1 find 查找文件

参考网页：

- <http://www.runoob.com/linux/linux-comm-find.html>

Linux find 命令用来在指定目录下查找文件。任何位于参数之前的字符串都将被视为欲查找的目录名。如果使用该命令时，不设置任何参数，则 find 命令将在当前目录下查找子目录与文件。并且将查找到的子目录和文件全部进行显示。前提是你必须对这个文件（夹）有相关权限。

```
1 find path -option [ -print ] [ -exec -ok command ] {} \;
```

解释

1. `find` 根据下列规则判断目录 `path` 和表达式 `expression`, 在命令列上第一个 `-()`, `!` 之前的部份为目录 `path`, 之后的是 `expression`。如果 `path` 是空字串则使用目前路径, 如果 `expression` 是空字串则使用 `-print` 为预设 `expression`。
2. `expression` 中可使用的选项有二三十个之多, 在此只介绍最常用的部份。
 - `-mount, -xdev` : 只检查和指定目录在同一个文件系统下的文件, 避免列出其它文件系统中的文件
 - `-amin n` : 在过去 `n` 分钟内被读取过
 - `-anewer file` : 比文件 `file` 更晚被读取过的文件
 - `-atime n` : 在过去 `n` 天内被读取过的文件
 - `-cmin n` : 在过去 `n` 分钟内被修改过
 - `-cnewer file` : 比文件 `file` 更新的文件
 - `-ctime n` : 在过去 `n` 天内被修改过的文件
 - `-empty` : 空的文件-gid `n` or `-group name` : gid 是 `n` 或是 group 名称是 `name`
 - `-ipath p, -path p` : 路径名称符合 `p` 的文件, `ipath` 会忽略大小写
 - `-name name, -iname name` : 文件名称符合 `name` 的文件。`iname` 会忽略大小写
 - `-size n` : 文件大小是 `n` 单位, `b` 代表 512 位元组的区块, `c` 表示字元数, `k` 表示 kilo bytes, `w` 是二个位元组。
 - `-type c` : 文件类型是 `c` 的文件。其中有如下可选变量。
 - `d`: 目录
 - `c`: 字型装置文件
 - `b`: 区块装置文件
 - `p`: 具名贮列
 - `f`: 一般文件
 - `l`: 符号连结 `()`
 - `s`: socket
 - `-pid n` : process id 是 `n` 的文件

下面介绍几个例子。

7.3.1.1 根据文件名查找文件

```
1 $ find . -name test.md
```

```

2 $ find . -name 'test.md'
3 $ find . -name "*.Rmd"
4 $ find . -name "*.rmd"
5 $ find . -iname "*.rmd"
6 $ find . -iname '????.rmd'
7 $ find . -ls
8 $ find /root -ls

```

解释

- 第 1、2 行，加不加单双引号'、"效果一样的。直接搜索当前目录下，名字是 *test.md* 的文件。
- 第 3 行，`find` 支持星*（任意多字符）和英文问号?（单个字符）的通配搜索的。注意要用单（双）引号括起来。本行就是查找所有以.Rmd结尾的文件。
- 第 4 行，`find` 是区分大小写的，本行命令跟上行命令结果是不一样的。
- 第 5 行，`-iname` 表示不区分大小写的搜索。
- 第 6 行，是?的一个例子，查找所有四个字母后缀不区分大小写.rmd的文件。
- 第 7 行，`-ls` 选项，搜索结果是类似 `ls -l` 命令的显示样式。
- 第 8 行，因为你没有权限，会报错。

7.3.1.2 根据文件大小、用户名、权限、日期等查找文件

将目前目录其其下子目录中所有一般文件列出

```

1 $ find . -type f
2 $ find ./deepin-bible -size +10M
3 $ find /mostlybig -size -1M
4 $ find /bigdata -size +500M -size -5G -exec du -sh {} \;
5 $ find /home -user litianci -ls
6 $ sudo find /home -user litianci -or -user joe -ls
7 $ sudo find /etc -group ntp -ls
8 $ sudo find /var/spool -not -user root -ls
9 $ find /bin -perm 755 -ls
10 $ find /home/chris/ -perm -222 -type d -ls
11 $ find /myreadonly -perm /222 -type f
12 $ find . -perm -002 -type f -ls
13 $ sudo find /etc/ -mmin -10
14 $ sudo find /bin /usr/bin /sbin /usr/sbin -ctime -3
15 $ find /var/ftp /var/www -atime +300

```

解释

1. 第 1 行，`-type f`，查找当前目录下的所有普通文件，`f`代表普通文件的意思，详情见上面说明。
2. 第 2-4 行，根据文件大小查找文件，`+10M`是大于 10MB 的意思，`-1M`是小于 5MB 的意思。单位大小还支持`k,M,G, c,b,w`。其中 `M` 表示 MB 大小，`G` 表示 GB 大小，`b` 代表 512 位元组的区块，`c` 表示字元数，`k` 表示 kB，`w` 是二个位元组。第 2 行的意思就是查找大于 10MB 的文件，读者不难理解第 3 行命令的意思吧。第 4 行，是查找大于 500MB，小于 5G 的文件(夹)，然后对每个文件执行命令`du -sh {找到的文件(夹)}`显示文件大小。方便处理一些过大文件。`-exec`的用法后面讲解。
3. 第 5-8 行，根据文件的用户搜索文件。其中用到了联合查找`-or`关键词，表示或的意思。也可以根据组用户查找文件。第 8 行，查找非 root 用户的文件。
4. 第 9-12 行，根据文件权限搜索文件。前面以及介绍了数字表示权限的方法。第 9 行，实现对权限的精确匹配。第 10-12 行，是对符合一定规则权限的文件进行搜索。其中`-perm +222`也即`-perm +mode`，已经不推荐使用了，深度操作系统自带 bash 已经报错了，建议用`-perm /222`。`-perm -mode`表示所有为 1 的权限都必须匹配才可以，示例中`-perm -222`也即`u,g,o`的写权限都具备才可以。`-perm /mode`表示`u,g,o`只要有一个具备即可。`-perm /mode`对于查找那些非法权限的文件特别有效。
5. 第 13-15, 行，是根据文件修改、创建和访问时间搜索文件的。`mtime,ctime,atime`的单位是天，`mmin,cmin,aamin`的单位是分钟。`mtime,mmin`表示文件内容修改的日期时间，`ctime,cmin`表示文件权限归属等属性信息修改的日期时间，`atime,aamin`表示文件被访问的时间。上述三个例子，分别查找 10 分钟之内修改的/etc 配置文件；3 天内是否有黑客更改我系统可执行文件的属性信息；ftp 服务器上查找 300 天以上都没有访问的文件。

使用`man find`可以看到更多例子。

7.3.1.3 联合若干条件查找文件

操作符(优先级递减；未做任何指定时默认使用 `-and`):

```

1 ( EXPR )
2 ! EXPR
3 -not EXPR
4 EXPR1 -a EXPR2

```

```

5 (EXPR1 -and EXPR2
6 (EXPR1 -o EXPR2
7 (EXPR1 -or EXPR2
8 (EXPR1 , EXPR2

```

下面举几个例子。

```

1 $ find /var/all \(
2   -user litianci -o -user xampp \
3 ) -ls
$ find /var/all/ -user litianci -not -group litianci -ls
$ find /var/all/ -user litianci -and -size +1M -ls

```

解释

1. 这些例子不难理解，需要注意的是，与或非的位置，要位于这些条件的外面。括号要用\表示，不然就歧义了。

7.3.1.4 找到文件后的动作

找到文件后只是列出来，有时不太满足我们的需求。强大的find给我们提供了找到后就操作的功能。语法如下，

```

1 $ find [options] -exec command {} \;
2 $ find [options] -ok command {} \;

```

解释

1. find命令对匹配的文件执行该参数所给出的 shell 命令。相应命令的形式为command' {} \;，注意{}和\;之间的空格。-ok和-exec的作用相同，只不过以一种更为安全的模式来执行该参数所给出的 shell 命令，在执行每一个命令之前，都会给出提示，让用户来确定是否执行。

下面给出几个例子。

```

1 $ find deepin-bible/ -iname "*.Rmd" -exec echo "我找到了{}" \;
2 $ find /usr/share -size +5M -exec du {} \; | sort -nr
3 $ sudo find /opt/lampp/ -user litianci -ok mv {} /tmp/litianci/

```

解释

1. 第1行，找所有的文件，然后换种样式显示出来。

2. 第 2 行，搜索`/usr/share` 文件夹下所有大于 5MB 的文件，显示他们的大小，并排序。对于命令`du`和`sort -nr`，读者可以试着用前文提到的方法，查找他们的功能。
3. 第 3 行，搜`/opt/lampp/` 下我的文件，每个文件均需我同意，才可以放到`/tmp/litianci/`目录下。大家可以理解{}就是代表搜索到文件（夹）了吧。

7.3.2 grep 查找文件内部信息

有时候你想找某个文件或者某个目录下的一段话，`grep`就能过来帮忙啦。但是对于中文，还是存在编码问题，`grep`默认文件为utf8编码（这句话对吗？需要核实。至少 ASCII 的，或者其他编码的文件，搜英文字母还是都可以搜得到的）。对于非 UTF8 文件，建议先转化为 UFT8 格式的临时文件，再查找。

```
1 $ cat test.md
2 : Shell 提示符中用到的转义字符
3 $ grep 转义字符 test.md
4 : Shell 提示符中用到的转义字符
5 $ grep 转义字符 -rn .
6 grep 转义字符 -rn .
7 ./test.md:1::: Shell 提示符中用到的转义字符
8 $ grep 转义 -rln .
9 ./test.md
10 $ grep s test.md
11 $ grep s -i test.md
12 : Shell 提示符中用到的转义字符
13 $ grep s -v test.md
14 : Shell 提示符中用到的转义字符
15 $ grep -i --color shell test.md
16 : Shell 提示符中用到的转义字符
17 $ grep Shell -rn {test.md,xxx.md}
```

解释

1. 第 1 行，用于显示 `test.md` 文件的内容。
2. 第 3 行，在 `test.md` 文件内搜索“转义字符”。`grep`的命令格式，就是先写待搜索字段，后写搜索的范围，比如某个文件或者文件夹。
3. 第 4 行，显示搜索结果，显示含有搜索词的那些行数。
4. 第 5 行，`grep`的选项`-rn`分别表示递归以及搜索结果显示文件所在行数。从第 6 行，我们可以看到，搜索当前目录`.`，`./test.md` 文件的第 1 行，包含该

搜索词，该行内容紧接着显示出来了。

5. 第 7 行，`grep`的选项`-l`表示，只显示包含搜索词的文件，不显示文件夹。
6. 第 9、10 行，`grep`搜索是区分大小写的。选项`-i`也即`--ignore-case`，是忽略大小写的意思，表示在忽略大小写的情况下搜索。
7. 第 12 行，选项`-v`，`--invert-match`表示显示不匹配的行数，也即搜索不包含搜索词的那些文件或者行数等。
8. 第 14 行，`--color`表示搜索结果中高亮匹配词。
9. 第 16 行，`{}`表示集合，该语句实现了对集合内所有文件进行内部查找Shell的功能。

如果有兴趣，可以使用`grep --help`或者其他命令，查看更多`grep`相关的用法。另外还有类似的`rgrep`等命令，这里就不介绍了。

7.3.3 其他检索工具

除了这些命令行工具外，深度系统的深度文件管理器支持在相关的文件夹内搜索对应的文件，一些编辑器，比如某些配置的`vim`，支持文件及文件夹内搜索词汇，比如`VSCODE`也支持。前面介绍的`ANGRYsearch`，大家也可看看。

7.3.4 正则表达式

说到正则表达式，可能需要一本书来细细的讲，这里从略吧。简要介绍几个常用的内容。正则表达式 (regular expression) 描述了一种字符串匹配的模式 (pattern)，可以用来检查一个串是否含有某种子串、将匹配的子串替换或者从某个串中取出符合某个条件的子串等。

参考网页

- <http://www.runoob.com/regexp/regexp-syntax.html>
- <https://deerchao.net/tutorials/regex/regex.htm>

例如：

- `runoo+b`，可以匹配`runoob`、`runooob`、`runooooob`等，`+`号代表前面的字符必须至少出现一次（1次或多次）。
- `runoo*b`，可以匹配`runob`、`runoob`、`runooooob`等，`*`号代表字符可以不出现，也可以出现一次或者多次（0次、或1次、或多次）。
- `colo?r` 可以匹配`color`或者`colour`，`?`问号代表前面的字符最多只可以出现一次（0次、或1次）（存疑，难道可以0次？）。

构造正则表达式的方法和创建数学表达式的方法一样。也就是用多种元字符

与运算符可以将小的表达式结合在一起创建更大的表达式。正则表达式的组件可以是单个的字符、字符集合、字符范围、字符间的选择或者所有这些组件的任意组合。

正则表达式是由普通字符（例如字符 a 到 z）以及特殊字符（称为“元字符”）组成的名字模式。模式描述在搜索文本时要匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。

7.3.4.1 普通字符

普通字符包括没有显式指定为元字符的所有可打印和不可打印字符。这包括所有大写和小写字母、所有数字、所有标点符号和一些其他符号。

7.3.4.2 非打印字符

非打印字符也可以是正则表达式的组成部分。下表列出了表示非打印字符的转义序列：

表 7-1 非打印字符转义字符

变量	解释
\cx	匹配由 x 指明的控制字符。例如，\cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 ‘c’ 字符。
\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。注意 Unicode 正则表达式会匹配全角空格符。
\S	匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。

7.3.4.3 特殊字符

所谓特殊字符，就是一些有特殊含义的字符，如上面说的 `runoob` 中的 *，简单的说就是表示任何字符串的意思。如果要查找字符串中的 * 符号，则需要对 * 进行转义，即在其前加一个 \： `runo*ob` 匹配 `runo*ob`。

许多元字符要求在试图匹配它们时特别对待。若要匹配这些特殊字符，必须首先使字符“转义”，即，将反斜杠字符 \ 放在它们前面。下表列出了正则表达式中的特殊字符：

表 7-2 特别字符

变量	解释
\$	匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 \$ 也匹配 '\n' 或 '\r'。要匹配 \$ 字符本身，请使用 \\$。
()	标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 \() 和 \)。
*	匹配前面的子表达式零次或多次。要匹配 * 字符，请使用 *。
+	匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+。
.	匹配除换行符 \n 之外的任何单字符。要匹配 .，请使用 \.。
[]	标记一个中括号表达式的开始。要匹配 [，请使用 \[。
?	匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如，'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\\"，而 '\\(' 则匹配 "("。
^	匹配输入字符串的开始位置，除非在方括号表达式中使用，此时它表示不接受该字符集合。要匹配 ^ 字符本身，请使用 \^。
{ }	标记限定符表达式的开始。要匹配 {，请使用 \{。
	指明两项之间的一个选择。要匹配 ，请使用 \ 。

7.3.4.4 限定符

限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。有 * 或 + 或 ? 或 {n} 或 {n,} 或 {n,m} 共 6 种。

表 7-3 正则表达式的限定符

变量	解释
*	匹配前面的子表达式零次或多次。例如，'zo*' 能匹配 "z" 以及 "zoo"。* 等价于 {0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do"、"does" 中的 "does"、"doxy" 中的 "do"。? 等价于 {0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配 n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "fooooo" 中的所有 o。 'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{n,m}	m 和 n 均为非负整数，其中 n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooo" 中的前三个 o。 'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

由于章节编号在大的输入文档中会很可能超过九，所以您需要一种方式来处理两位或三位章节编号。限定符给您这种能力。下面的正则表达式匹配编号为任何位数的章节标题：

```
/Chapter [1-9][0-9]*/
```

请注意，限定符出现在范围表达式之后。因此，它应用于整个范围表达式，在本例中，只指定从 0 到 9 的数字（包括 0 和 9）。

这里不使用 + 限定符，因为在第二个位置或后面的位置不一定需要有一个数字。也不使用 ? 字符，因为使用 ? 会将章节编号限制到只有两位数。您需要至少匹配 Chapter 和空格字符后面的一个数字。

如果您知道章节编号被限制为只有 99 章，可以使用下面的表达式来至少指定

一位但至多两位数字。

```
/Chapter [0-9]{1,2}/
```

上面的表达式的缺点是，大于 99 的章节编号仍只匹配开头两位数字。另一个缺点是 Chapter 0 也将匹配。只匹配两位数字的更好的表达式如下：

```
/Chapter [1-9][0-9]?>/pre>
```

或

```
/Chapter [1-9][0-9]{0,1}/
```

*、+限定符都是贪婪的，因为它们会尽可能多的匹配文字，只有在它们的后面加上一个?就可以实现非贪婪或最小匹配。

例如，您可能搜索 HTML 文档，以查找括在 H1 标记内的章节标题。该文本在您的文档中如下：

```
<H1>Chapter 1 - 介绍正则表达式</H1>
```

贪婪：下面的表达式匹配从开始小于符号(<)到关闭 H1 标记的大于符号(>)之间的所有内容。

```
/<.*>/
```

非贪婪：如果您只需要匹配开始和结束 H1 标签，下面的非贪婪表达式只匹配

。

```
/<.*?>/
```

如果只想匹配开始的 H1 标签，表达式则是：

```
/<\w+?>/
```

通过在 *、+ 或 ? 限定符之后放置?，该表达式从“贪心”表达式转换为“非贪心”表达式或者最小匹配。

7.3.4.5 定位符

定位符使您能够将正则表达式固定到行首或行尾。它们还使您能够创建这样的正则表达式，这些正则表达式出现在一个单词内、在一个单词的开头或者一个单词的结尾。

定位符用来描述字符串或单词的边界，^ 和 \$ 分别指字符串的开始与结束，\b 描述单词的前或后边界，\B 表示非单词边界。

表 7-4 正则表达式的定位符

变量	解释
^	匹配输入字符串开始的位置。如果设置了 RegExp 对象的 Multiline 属性，^ 还会与 \n 或 \r 之后的位置匹配。
\$	匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性，\$ 还会与 \n 或 \r 之前的位置匹配。
\b	匹配一个单词边界，即字与空格间的位置。
\B	非单词边界匹配。

注意：不能将限定符与定位符一起使用。由于在紧靠换行或者单词边界的前面或后面不能有一个以上位置，因此不允许诸如 ^* 之类的表达式。

若要匹配一行文本开始处的文本，请在正则表达式的开始使用 ^ 字符。不要将 ^ 的这种用法与中括号表达式内的用法混淆。

若要匹配一行文本的结束处的文本，请在正则表达式的结束处使用 \$ 字符。

若要在搜索章节标题时使用定位点，下面的正则表达式匹配一个章节标题，该标题只包含两个尾随数字，并且出现在行首：

```
/^Chapter [1-9][0-9]{0,1}/
```

真正的章节标题不仅出现行的开始处，而且它还是该行中仅有的文本。它即出现在行首又出现在同一行的结尾。下面的表达式能确保指定的匹配只匹配章节而不匹配交叉引用。通过创建只匹配一行文本的开始和结尾的正则表达式，就可做到这一点。

```
/^Chapter [1-9][0-9]{0,1}\$/
```

匹配单词边界稍有不同，但向正则表达式添加了很重要的能力。单词边界是单词和空格之间的位置。非单词边界是任何其他位置。下面的表达式匹配单词 Chapter 的开头三个字符，因为这三个字符出现在单词边界后面：

```
/\bCha/
```

\b 字符的位置是非常重要的。如果它位于要匹配的字符串的开始，它在单词的开始处查找匹配项。如果它位于字符串的结尾，它在单词的结尾处查找匹配项。例如，下面的表达式匹配单词 Chapter 中的字符串 ter，因为它出现在单词边界的前面：

```
/ter\b/
```

下面的表达式匹配 Chapter 中的字符串 apt，但不匹配 aptitude 中的字符串 apt：

```
/\Bapt/
```

字符串 apt 出现在单词 Chapter 中的非单词边界处，但出现在单词 aptitude 中的单词边界处。对于 \B 非单词边界运算符，位置并不重要，因为匹配不关心究竟是单词的开头还是结尾。

7.3.4.6 选择

用圆括号将所有选择项括起来，相邻的选择项之间用 | 分隔。但用圆括号会有一个副作用，使相关的匹配会被缓存，此时可用 ?: 放在第一个选项前来消除这种副作用。

其中 ?: 是非捕获元之一，还有两个非捕获元是 ?: 和 ?:，这两个还有更多的含义，前者为正向预查，在任何开始匹配圆括号内的正则表达式模式的位置来匹配搜索字符串，后者为负向预查，在任何开始不匹配该正则表达式模式的位置来匹配搜索字符串。

7.3.4.7 反向引用

对一个正则表达式模式或部分模式两边添加圆括号将导致相关匹配存储到一个临时缓冲区中，所捕获的每个子匹配都按照在正则表达式模式中从左到右出现的顺序存储。缓冲区编号从 1 开始，最多可存储 99 个捕获的子表达式。每个缓冲区都可以使用 \n 访问，其中 n 为一个标识特定缓冲区的一位或两位十进制数。

可以使用非捕获元字符 ?:、?: 或 ?: 来重写捕获，忽略对相关匹配的保存。

反向引用的最简单的、最有用的应用之一，是提供查找文本中两个相同的相邻单词的匹配项的能力。以下面的句子为例：

```
Is is the cost of of gasoline going up up?
```

上面的句子很显然有多个重复的单词。如果能设计一种方法定位该句子，而不必查找每个单词的重复出现，那该有多好。下面的正则表达式使用单个子表达式来实现这一点：实例

查找重复的单词：

```
1 var str = "Is is the cost of of gasoline going up up";
2 var patt1 = /\b([a-z]+) \1\b/ig;
3 document.write(str.match(patt1));
```

捕获的表达式，正如 [a-z]+ 指定的，包括一个或多个字母。正则表达式的第二部分是对以前捕获的子匹配项的引用，即，单词的第二个匹配项正好由括号表达式匹配。\\1 指定第一个子匹配项。

单词边界元字符确保只检测整个单词。否则，诸如 "is issued" 或 "this is" 之类的词组将不能正确地被此表达式识别。

正则表达式后面的全局标记 `g` 指定将该表达式应用到输入字符串中能够查找到的尽可能多的匹配。

表达式的结尾处的不区分大小写 `i` 标记指定不区分大小写。

多行标记指定换行符的两边可能出现潜在的匹配。

反向引用还可以将通用资源指示符 (URI) 分解为其组件。假定您想将下面的 URI 分解为协议（`ftp`、`http` 等等）、域地址和页/路径：

```
http://www.runoob.com:80/html/html-tutorial.html
```

下面的正则表达式提供该功能：

输出所有匹配的数据：

```
1 var str = "http://www.runoob.com:80/html/html-tutorial.html";
2 var patt1 = /(\w+):\/\/([^\:/]+)(:\d*)?([^\# ]*)/;
3 arr = str.match(patt1);
4 for (var i = 0; i < arr.length ; i++) {
5     document.write(arr[i]);
6     document.write("<br>");
7 }
```

第三行代码 `str.match(patt1)` 返回一个数组，实例中的数组包含 5 个元素，索引 0 对应的是整个字符串，索引 1 对应第一个匹配符（括号内），以此类推。

第一个括号子表达式捕获 Web 地址的协议部分。该子表达式匹配在冒号和两个正斜杠前面的任何单词。

第二个括号子表达式捕获地址的域地址部分。子表达式匹配 `:` 和 `/` 之后的一个或多个字符。

第三个括号子表达式捕获端口号（如果指定了的话）。该子表达式匹配冒号后面的零个或多个数字。只能重复一次该子表达式。

最后，第四个括号子表达式捕获 Web 地址指定的路径和 `/` 或页信息。该子表达式能匹配不包括## 或空格字符的任何字符序列。

将正则表达式应用到上面的 URI，各子匹配项包含下面的内容：

```
1 第一个括号子表达式包含 http
2 第二个括号子表达式包含 www.runoob.com
3 第三个括号子表达式包含 :80
```

4 第四个括号子表达式包含 /html/html-tutorial.html

- 特殊字符在中括号表达式时如 [.] 只会匹配 . 字符，等价于 \.，而非匹配除换行符 \n 外的所有字符。

```
1 var str = "runoob.com";
2 var patt1 = /[.]/;
3 document.write(str.match(patt1));
```

^ 和 [^指定字符串] 之间的区别:

^ 指的是匹配字符串开始的位置

[^指定字符串] 指的是除指定字符串以外的其他字符串

```
1 (^[0-9])+      //匹配有一至多个数字的字符串组合
2 [^0-9]+        // 匹配有一至多个不含数字的字符串组合
```

7.4 总结

本章介绍了字符集编码知识，以及文件的查找文件内的查找命令。

第八章 进程管理

通过阅读本章，你将会了解到以下几项内容。

- 显示进程
- 进程前后台切换
- 对进程的生杀予夺

8.1 何谓进程

参考网页：

- <http://www.cnblogs.com/xiaojiang1025/p/6021049.html>
- <https://www.cnblogs.com/vinotly/p/5585683.html>
- https://blog.csdn.net/hq_buddhist/article/details/51233808
- <https://linux.cn/article-8451-1.html>

Linux 是一个多用户多任务的操作系统。多用户是指多个用户可以在同一时间使用计算机系统；多任务是指 Linux 可以同时执行几个任务，它可以在还未执行完一个任务时又执行另一项任务。

进程是指正在执行的程序；是程序正在运行的一个实例。它由程序指令，和从文件、其它程序中读取的数据或系统用户的输入组成。进程的一个比较正式的定义是在自身的虚拟地址空间运行的一个单独的程序。进程与程序是有区别的，进程不是程序，虽然它由程序产生。程序只是一个静态的指令集合，不占系统的运行资源；而进程是一个随时都可能发生变化的、动态的、使用系统运行资源的程序。而且一个程序可以启动多个进程。

系统运行时，会为每个进程分配当前唯一的数字标识，PID（Process ID），其他进程在该进程存活期间是无法抢占该 PID 的，当然，该进程结束了，系统会回收该 PID 的。

除了 PID 号外，进程还会与特定的用户帐户和组帐户相关联。这些账户信息有助于确定进程可以访问的系统资源。例如，进程运行时，root 根用户进程比普通用户进程在访问系统文件和资源方面有更多的权限。

有效管理进程，有助于我们更好的控制电脑。进程的信息一般存储与 /proc 文件夹，每个进程将其信息存储在 /proc 的子目录中，以进程 PID 命名。你可以使用 cat, less 以及其他编辑器查看这些文件。

8.2 深度系统监视器管理进程



图 8-1 深度系统监视器

深度系统监视器是深度科技团队打造一款直观易用的系统监视器应用，它可以实时监控处理器状态、内存占用率、网络上传下载速度；还可以管理您的系统进程和应用进程，支持搜索进程和强制结束进程。

标签显示，快速定位

通过标签分类的方式显示：应用程序进程、我的进程和所有进程，快速切换自己想要的进程显示；同时应用程序进程名称国际化处理，进程名称一目了然。当需要快速定位到某个进程时，可以快速搜索定位。

列表展示，高效右键

系统的进程采用列表方式展示，可以自定义处理器、内存、磁盘写入、磁盘读取、下载、上传、进程号是否显示，还可以根据列表排序显示；同时对进程还可以右键菜单快速操作。

捕捉窗口，即点即“杀”

当在使用系统的过 程，不知道进程的 ID 或者想直接结束某个应用进程，只需点击菜单中强制结束进程选项，自动采用红色透明遮框捕捉窗口，点击即可结束进程。

大家可以通过更新系统以获取深度系统监视器 V1.0，或者直接在深度商店搜索下载。也可以采用命令行安装。

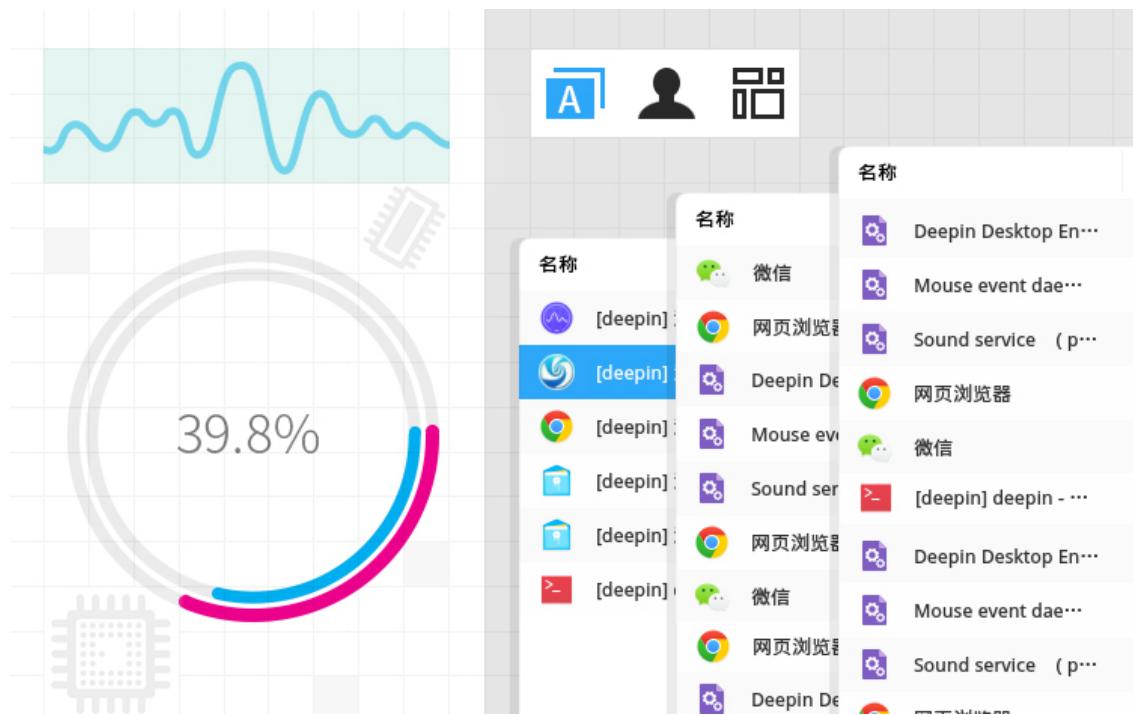


图 8-2 标签显示，快速定位



图 8-3 列表展示，高效右键



图 8-4 捕捉窗口，即点即“杀”

```
1 | $ sudo apt-get install deepin-system-monitor -y
```

8.3 命令方式查看进程

参考网页：

- <https://www.cnblogs.com/w10234/p/5642552.html>

要对进程进行监测和控制，首先必须要了解当前进程的情况，也就是需要查看当前进程，而ps命令（Process Status）就是最基本同时也是非常强大的进程查看命令。使用该命令，可以确定有哪些进程正在运行和运行的状态、进程是否结束、进程有没有僵尸、哪些进程占用了过多的资源等等。

ps 为我们提供了进程的一次性的查看，它所提供的查看结果并不动态连续的；如果想对进程时间监控并进行操作，应该用 top 工具。

如果直接用ps命令，会显示所有进程的状态，通常结合grep命令查看某进程的状态。grep命令已经在前面第七章介绍过了。

8.3.1 ps 命令基本用法

按照`man ps`的说法，

This version of ps accepts several kinds of options:

1. UNIX options, which may be grouped and must be preceded by a dash.
2. BSD options, which may be grouped and must not be used with a dash.
3. GNU long options, which are preceded by two dashes.

`ps`的选项支持三种方式，

1. UNIX 风格选项，可以组合且必须以短横线-开头，类似第五章介绍的。
2. BSD 风格选项，可以组合且不可使用短横线。
3. GNU 长风格选项，也即第五章提到的双短横线--开头的选项。

不同风格的选项可以混用，但是部分选项可能会相互冲突，比如`ps aux`跟`ps -aux`并不完全一样，详情请参考`man ps`，这里不转载他们的内容了。下面介绍常用的几个例子。下面例子未经上机测试。特殊字符\$,&在代码中由于是`mathtype=true`，需要特别关注。

```
1 $ # To see every process on the system using standard syntax:  
2 $ ps -e  
3 $ ps -ef  
4 $ ps -eF  
5 $ ps -ely  
6 $ # To see every process on the system using BSD syntax:  
7 $ ps ax  
8 $ ps axu  
9 $ # 查找包含 php 的进程  
10 $ ps aux|grep php  
11 $ # To print a process tree:  
12 $ ps -ejH  
13 $ ps axjf  
14 $ # To get info about threads:  
15 $ ps -eLf  
16 $ ps axms  
17 $ # To get security info:  
18 $ ps -eo euser,ruser,suser,fuser,f,comm,label  
19 $ ps axZ  
20 $ ps -eM  
21 $ # To see every process running as root (real & effective ID) in  
     user format:
```

```

22 $ ps -U root -u root u
23 $ # To see every process with a user-defined format:
24 $ ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
25 $ ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
26 $ ps -Ao pid,tt,user, fname,tmout,f,wchan
27 $ # -sort=-rss 是输出结果安装 rss 从大到小排序, rss 前正负号是否倒序。
28 $ ps -eo pid,user,group,gid,vsz,rss,comm --sort=-rss | less
29 $ # Print only the process IDs of syslogd:
30 $ ps -C syslogd -o pid=
31 $ # Print only the name of PID 42:
32 $ ps -q 42 -o comm=

```

对各个选项的简单解释如下，摘自：<https://www.cnblogs.com/w10234/p/5642552.html>

1. ps a 显示现行终端机下的所有程序，包括其他用户的程序。
2. ps -A 显示所有程序。
3. ps c 列出程序时，显示每个程序真正的指令名称，而不包含路径，参数或常驻服务的标示。
4. ps -e 此参数的效果和指定“A”参数相同。
5. ps e 列出程序时，显示每个程序所使用的环境变量。
6. ps f 用 ASCII 字符显示树状结构，表达程序间的相互关系。
7. ps -H 显示树状结构，表示程序间的相互关系。
8. ps -N 显示所有的程序，除了执行 ps 指令终端机下的程序之外。
9. ps s 采用程序信号的格式显示程序状况。
10. ps S 列出程序时，包括已中断的子程序资料。
11. ps -t<终端机编号> 指定终端机编号，并列出属于该终端机的程序的状况。
12. ps u 以用户为主的格式来显示程序状况。
13. ps x 显示所有程序，不以终端机来区分。

8.3.2 top 命令基本用法

top 动态实时的显示当前资源占用情况，如图8-5，并允许你对进程操作。如果你需要对所有的进程都能够生杀予夺，可能你需要root根权限。

解释

1. 图中红色矩形框圈出的，表示当前时间22:19:51，已经开机运行了多长时间13min(分钟)，当前有1个用户，当前CPU负载。
2. 图中手划线的分别表示进程统计信息Tasks、CPU 及 Ni 等的信息%cpu(s)、

top - [22:19:51] up 13 min, 1 user load average: 0.22, 0.23, 0.18										
Tasks: 196 total, 1 running, 195 sleeping, 0 stopped, 0 zombie										
%Cpu(s): 0.3 us, 0.2 sy, 0.0 hi, 99.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st										
MiB Mem : 7857.7 total, 4658.2 free, 1054.1 used, 2145.3 buff/cache										
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 6444.8 avail Mem										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
3070	root	20	0	1320724	137844	54808	S	0.7	1.7	0:26.93 Xorg
905	root	20	0	981056	10396	6912	S	0.3	0.1	0:03.01 warm-daemon
3717	litianci	20	0	50648	4220	2844	S	0.3	0.1	0:00.39 dbus-daemon
3911	litianci	20	0	1765276	144304	96020	S	0.3	1.8	0:02.14 dde-desktop
4286	litianci	20	0	3240256	105968	55908	S	0.3	1.3	0:01.57 sogou-qimpanel
6995	litianci	20	0	635264	63672	30724	S	0.3	0.8	0:00.83 deepin-terminal
9117	litianci	20	0	44288	3668	3048	R	0.3	0.0	0:00.06 top
1	root	20	0	225216	8984	6612	S	0.0	0.1	0:01.32 systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00 kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.02 ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:00.27 rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00 rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00 migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00 watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00 cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00 cpuhp/1
14	root	rt	0	0	0	0	S	0.0	0.0	0:00.00 watchdog/1
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.00 migration/1
16	root	20	0	0	0	0	S	0.0	0.0	0:00.02 ksoftirqd/1
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/1:0H

图 8-5 top 显示界面

内存占用情况(单位:MiB)MiB Mem、交换空间占用情况(单位:MiB)MiB Swap。

对于 MiB 以及 MB 单位的区别,

3. 按下h键, 会弹出帮助信息。如图8-6所示。按下q或者esc键, 返回到 top 命令界面。想看更多信息, 可以man top或者必应百度一下。
4. 按下M或P, 分别表示安装内存和CPU排序, 数字1显示更多CPU, R表示正倒序切换。
5. 按下u, 紧接着输入用户名回车, 查看特定用户的进程。直接回车查看所有用户的进程。

调整进程

通常我们使用top是电脑太卡了, 想要看看哪些进程占用 CPU 或者内存过多。上面提到的按键M和P分别实现内存和CPU 占用的排序, 就非常有用, 按R实现正倒序的切换。接下来就需要对相关进程“动手”啦。

1. 降低权重, 减小优先级。其实就是图8-5中NI和PR那两栏。关于优先级的内容, 详情见下文。在top下可以按r (表示 renice 调整好感值) 键, 接着输入你想提高好感 (NI) 的进程 PID, 回车后再给他一个好感值。一般情况,

```

Help for Interactive Commands - procps-ng 3.3.15
Window 1:Def: Cumulative mode Off. System: Delay 3.0 secs; Secure mode Off.

Z,B,E,e  Global: 'Z' colors; 'B' bold; 'E'/'e' summary/task memory scale
l,t,m    Toggle Summary: 'l' load avg; 't' task/cpu stats; 'm' memory info
0,1,2,3,I Toggle: '0' zeros; '1/2/3' cpus or numa node views; 'I' Irix mode
f,F,X    Fields: 'f'/'F' add/remove/order/sort; 'X' increase fixed-width

L,&,<,> . Locate: 'L'/'&' find/again; Move sort column: '<'/'>' left/right
R,H,V,J . Toggle: 'R' Sort; 'H' Threads; 'V' Forest view; 'J' Num justify
c,i,S,j . Toggle: 'c' Cmd name/line; 'i' Idle; 'S' Time; 'j' Str justify
x,y    . Toggle highlights: 'x' sort field; 'y' running tasks
z,b    . Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
u,U,o,O . Filter by: 'u'/'U' effective/any user; 'o'/'O' other criteria
n,#,^0 . Set: 'n'/'#' max tasks displayed; Show: Ctrl+'0' other filter(s)
C,...   . Toggle scroll coordinates msg for: up,down,left,right,home,end

k,r    Manipulate tasks: 'k' kill; 'r' renice
d or s Set update interval
W,Y    Write configuration file 'W'; Inspect other output 'Y'
q      Quit
( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
Type 'q' or <Esc> to continue █

```

图 8-6 top 显示界面



众所周知，在计算机中是采用二进制，在电脑世界里，以 2 的次方数为“批量”处理 Byte 会方便一些，整齐一些。每 1024Byte 为 1KB，每 1024KB 为 1MB，每 1024MB 为 1GB，每 1024GB 为 1TB，而在国际单位制中 TB、GB、MB、KB 是“1000 进制”的数，为此国际电工协会（IEC）拟定了“KiB”、“MiB”、“GiB”的二进制单位，专用来标示“1024 进位”的数据大小；而硬盘厂商在计算容量方面是以每 1000 为一进制的，每 1000 字节为 1KB，每 1000KB 为 1MB，每 1000MB 为 1GB，每 1000GB 为 1TB

参考网页：<https://blog.csdn.net/u012256258/article/details/52565500>

对于普通用户只能提高进程的好感值，无权降低。好感值（NI）越高，占用资源的权限就越低，详细内容后面再说。

2. 干掉进程。当你知道某个进程 PID，就可以按下k（表示 kill 干掉键，接着输入该 PID，最后输入kill命令的选项值，比如15清屏或者绝情些用9。

8.4 进程的分类

当你使用 Putty 远程连接服务器时，只有一个终端，你想后台跑一个程序，但是在此同时还想干其他事情，怎么办呢？那就用上后台程序啦。

在 Linux 中主要有两种类型的进程：

- 前台进程（也称为交互式进程）- 这些进程由终端会话初始化和控制。换句话说，需要有一个连接到系统中的用户来启动这样的进程；它们不是作为系统功能/服务的一部分自动启动。
- 后台进程（也称为非交互式/自动进程）- 这些进程没有连接到终端；它们不需要任何用户输入。

什么是守护进程

这是后台进程的特殊类型，它们在系统启动时启动，并作为服务一直运行；它们不会死亡。它们自发地作为系统任务启动（作为服务运行）。但是，它们能被用户通过 init 进程控制。

8.4.1 怎么生成后台进程呢？

我感觉我的理解可能有误，比如我知道evince PDF 阅读器不锁定文件的修改，当你修改了 PDF 文件，可以从evince上直接看到变化。这对于编写本书反复编译太有帮助了。但是编译本书要运行make命令，我习惯只打开一个终端，我就可以让evince程序运行于后台。有这么几种方法，

- 在命令行末尾加&。
- 使用at命令（没有例子）。

```
1 $ evince _book/deepin-bible.pdf &
2 [1] 19589
```

解释

1. [1]表示，后台进程序号。19589是进程号 PID。

如果想查看当前有多少后台进程。使用命令jobs

```

1 $ jobs
2 [1]  运行中          evince _book/deepin-bible.pdf &
3 [2]- 已停止          man top
4 [3]+ 已停止          vim ~/test.md

```

解释

1. 使用`jobs`会弹出当前正在运行的后台程序。还会显示他们的状态，比如运行中、已停止等。
2. [3]+，的+，-号，表示最新和次新加入后台的进程。就是最后两个加入后台的进程。但是这个貌似不是非常准确。`man top`不是最新加入后台的进程，但是却被标注为最新加入的。好在这个不怎么影响，后面再去找资料核实。有时还需要把这些后台再拉回前台来，`fg`表示`foreground`的意思，就可以派上用场了。先看看`fg`的用法。貌似，不带%也不影响结果。需要核实。

```

1 $ fg % # 把最新的进程，也就是带 ‘+’ 的进程放到前台。
2 $ fg %n # n 表示 ‘jobs’ 命令中 [n]
3 $ fg %string # string 表示命令，比如 ‘vim’，前提是不混淆。
4 $ fg %?string # ?string 表示 string 在任意位置。
5 $ fg %-- # 倒数第二个已停止的进程拉回前台。

```

当然，我试过`fg n`直接输入`jobs`命令出来的那些后台序号[n]，可以直接把它们拉回前台。另外还有`bg`命令，用于唤醒已停止的后台命令。

8.5 进程的终结与调整优先级 (Killing and Renicing)

有些进程太耗资源，我们需要关闭它，可以使用`kill`和`killall`命令，分别是通过进程号 PID 终结某个进程和通过进程名称终结某类进程。在终结进程的时候，还需要传递系统信号。具体用法可以通过`man kill`等查看。先来常用的几个例子吧，

```

1 $ kill -l # 列出所有可用信号，见下面信号表
2 $ kill 999 # 使用默认系统信号杀死 PID=999 的进程
3 $ kill -15 999 # 默认系统信号也就是 15 号
4 $ kill -SIGKILL 999 # 默认系统信号常量为 SIGKILL，跟上面一致。
5 $ kill -9 999 # 对于顽固分子，电脑卡住的，往往用 -9 KILL 信号

```

```
6 | $ killall -9 workrave # killall 使用的信号类似 kill, 只不过用的是命令  
名称进行终结
```

参考网页：

- <http://www.cnblogs.com/taobataoma/archive/2007/08/30/875743.html>

表 8-1 信号对应的数值及意义

信号	数值	意义
SIGHUP	1	Hang-up detected on controlling terminal or death of controlling process.
SIGINT	2	Interrupt from keyboard.
SIGQUIT	3	Quit from keyboard.
SIGABRT	6	Abort signal from abort(3).
SIGKILL	9	Kill signal.
SIGTERM	15	Termination signal.
SIGCONT	19,18,25	Continue if stopped.
SIGSTOP	17,19,23	Stop process.

参考网页：

- <https://blog.csdn.net/longdel/article/details/7317511>

用top或者ps命令会输出PRI/PR、NI、%ni/%nice这三种指标值，这些到底是什么东西？先给出大概的解释如下：

- PRI：进程优先权，代表这个进程可被执行的优先级，其值越小，优先级就越高，越早被执行。
- NI：进程nice值，代表这个进程的优先值。
- %nice：改变过优先级的进程占用CPU的百分比。

PRI是比较容易理解的，即进程的优先级，或者通俗点说就是程序被CPU执行的先后顺序，此值越小进程的优先级别越高。那NI呢？就是我们所要说的nice值了，其表示进程可被执行的优先级的修正数值。如前面所说，PRI值越小越快被执行，那么加入nice值后，将会使得PRI变为：PRI(new)=PRI(old)+nice。由此看出，PRI是根据nice排序的，规则是nice越小PRI越前（小，优先权更大），即其优先级会变高，则其越快被执行。如果nice相同则进程uid是root的优先权更大。

在LINUX系统中，nice值的范围从-20到+19（不同系统的值范围是不一样的），正值表示低优先级，负值表示高优先级，值为零则表示不会调整该进程的优先级。

具有最高优先级的程序，其nice值最低，所以在 LINUX 系统中，值-20使得一项任务变得非常重要；与之相反，如果任务的nice为+19，则表示它是一个高尚的、无私的任务，允许所有其他任务比自己享有宝贵的 CPU 时间的更大使用份额，这也就是nice的名称的来意。

进程在创建时被赋予不同的优先级值，而如前面所说，nice的值是表示进程优先级值可被修正数据值，因此，每个进程都在其计划执行时被赋予一个nice值，这样系统就可以根据系统的资源以及具体进程的各类资源消耗情况，主动干预进程的优先级值。在通常情况下，子进程会继承父进程的nice值，比如在系统启动的过程中，init进程会被赋予0，其他所有进程继承了这个nice值（因为其他进程都是init的子进程）。

对nice值一个形象比喻，假设在一个CPU轮转中，有2个Runnable的进程A和B，如果他们的nice值都为0，假设内核会给他们每人分配1k个cpu时间片。但是假设进程A的为0，但是B的值为-10，那么此时CPU可能分别给A和B分配1k和1.5k的时间片。故可以形象的理解为，nice的值影响了内核分配给进程的cpu时间片的多少，时间片越多的进程，其优先级越高，其优先级值（PRI）越低。`%nice`，就是改变过优先级的进程的占用CPU的百分比，如上例中就是 $0.5k/2.5k=1/5=20\%$ 。

由此可见，进程nice值和进程优先级不是一个概念，但是进程nice值会影响到进程的优先级变化。

进程的 nice 值是可以被修改的，修改命令分别是nice和renice。

1. nice命令就是设置一个要执行 command 进程的nice值，其命令格式是 `nice -n adjustment command command_option`。如果这里不指定adjustment，则默认认为10，非root用户，adjustment不可为负值。
2. renice命令就是设置一个已经在运行的进程的nice值，假设一运行进程本来nice值为0，renice为3后，则这个运行进程的nice值就为3了。
3. 如果用户设置的nice值超过了nice的边界值（LINUX为-20到+19），系统就取nice的边界值作为进程的nice值。
4. 对非root用户，只能将其底下的进程的nice值变大而不能变小。若想变小，得要有相应的权限。

```
1 $ nice
2 0
3 $ nice -n 3 ls -l bashscript
4 -rwsrwxrwx 1 litianci litianci 119 11月 17 22:32 bashscript
5 $ nice -n -2 ls -l bashscript
```

```

6 nice: 无法设置优先级: 权限不够
7 -rwsrwxrwx 1 litianci litianci 119 11月 17 22:32 bashscript
8 $ sudo nice -n -2 ls -l bashscript
9 -rwsrwxrwx 1 litianci litianci 119 11月 17 22:32 bashscript

```

同样，renice与nice命令类似，但是需要输入已经运行的进程号 PID。

```

1 $ evince _book/deepin-bible.pdf & # 启动一个后台程序，显示了进程号
2 [1] 26159
3 $ renice -n 5 26159 # renice 该进程
4 26159 (process ID) old priority 0, new priority 5

```

8.6 通过 cgroups 限制进程

参考网页：

- <http://www.cnblogs.com/lisperl/archive/2012/04/17/2453838.html>

8.6.1 cgroups 是什么？

cgroups是 control groups 的缩写，是 Linux 内核提供的一种可以限制、记录、隔离进程组（process groups）所使用的物理资源（如：cpu,memory,IO 等等）的机制。最初由 google 的工程师提出，后来被整合进 Linux 内核。cgroups 也是 LXC 为实现虚拟化所使用的资源管理手段，可以说没有 cgroups 就没有 LXC。

8.6.2 cgroups 可以做什么？

cgroups最初的目标是为资源管理提供的一个统一的框架，既整合现有的cpuset等子系统，也为未来开发新的子系统提供接口。现在的cgroups适用于多种应用场景，从单个进程的资源控制，到实现操作系统层次的虚拟化（OS Level Virtualization）。

cgroups提供了以下功能：

1. 限制进程组可以使用的资源数量（Resource limiting）。比如：memory 子系统可以为进程组设定一个 memory 使用上限，一旦进程组使用的内存达到限额再申请内存，就会出发 OOM（out of memory）。
2. 进程组的优先级控制（Prioritization）。比如：可以使用 cpu 子系统为某个进程组分配特定 cpu share。

3. 记录进程组使用的资源数量 (Accounting)。比如：可以使用 cpuacct 子系统记录某个进程组使用的 cpu 时间
4. 进程组隔离 (Isolation)。比如：使用 ns 子系统可以使不同的进程组使用不同的 namespace，以达到隔离的目的，不同的进程组有各自的进程、网络、文件系统挂载空间。
5. 进程组控制 (Control)。比如：使用 freezer 子系统可以将进程组挂起和恢复。

8.6.3 cgroups 相关概念及其关系

相关概念：

1. 任务 (task)。在 cgroups 中，任务就是系统的一个进程。
2. 控制族群 (control group)。控制族群就是一组按照某种标准划分的进程。cgroups 中的资源控制都是以控制族群为单位实现。一个进程可以加入到某个控制族群，也从一个进程组迁移到另一个控制族群。一个进程组的进程可以使用 cgroups 以控制族群为单位分配的资源，同时受到 cgroups 以控制族群为单位设定的限制。
3. 层级 (hierarchy)。控制族群可以组织成 hierarchical 的形式，即一颗控制族群树。控制族群树上的子节点控制族群是父节点控制族群的孩子，继承父控制族群的特定的属性。
4. 子系统 (subsystem)。一个子系统就是一个资源控制器，比如 cpu 子系统就是控制 cpu 时间分配的一个控制器。子系统必须附加 (attach) 到一个层级上才能起作用，一个子系统附加到某个层级以后，这个层级上的所有控制族群都受到这个子系统的控制。

相互关系：

1. 每次在系统中创建新层级时，该系统中的所有任务都是那个层级的默认 cgroup（我们称之为 root cgroup，此 cgroup 在创建层级时自动创建，后面在该层级中创建的 cgroup 都是此 cgroup 的后代）的初始成员。
2. 一个子系统最多只能附加到一个层级。
3. 一个层级可以附加多个子系统。
4. 一个任务可以是多个 cgroup 的成员，但是这些 cgroup 必须在不同的层级。
5. 系统中的进程（任务）创建子进程（任务）时，该子任务自动成为其父进程所在 cgroup 的成员。然后可根据需要将该子任务移动到不同的 cgroup 中，但开始时它总是继承其父任务的 cgroup。

8.6.4 cgroups 子系统介绍

- `blkio` – 这个子系统为块设备设定输入/输出限制，比如物理设备（磁盘，固态硬盘，USB 等等）。
- `cpu` – 这个子系统使用调度程序提供对 CPU 的 `cgroup` 任务访问。
- `cpuacct` – 这个子系统自动生成 `cgroup` 中任务所使用的 CPU 报告。
- `cpuset` – 这个子系统为 `cgroup` 中的任务分配独立 CPU（在多核系统）和内存节点。
- `devices` – 这个子系统可允许或者拒绝 `cgroup` 中的任务访问设备。
- `freezer` – 这个子系统挂起或者恢复 `cgroup` 中的任务。
- `memory` – 这个子系统设定 `cgroup` 中任务使用的内存限制，并自动生成由那些任务使用的内存资源报告。
- `net_cls` – 这个子系统使用等级识别符（`classid`）标记网络数据包，可允许 Linux 流量控制程序（`tc`）识别从具体 `cgroup` 中生成的数据包。
- `ns` – 名称空间子系统。

关于 `cgroups` 的更多介绍，请参考相关书籍。

8.7 进程的状态

参考网页：

- <https://blog.csdn.net/shenwansangz/article/details/51981459>

Linux 是一个多用户，多任务的系统，可以同时运行多个用户的多个程序，就必然会产生很多的进程，而每个进程会有不同的状态。

8.7.1 Linux 进程状态：R (TASK_RUNNING)，可执行状态。

只有在该状态的进程才可能在 CPU 上运行。而同一时刻可能有多个进程处于可执行状态，这些进程的 `task_struct` 结构（进程控制块）被放入对应 CPU 的可执行队列中（一个进程最多只能出现在一个 CPU 的可执行队列中）。进程调度器的任务就是从各个 CPU 的可执行队列中分别选择一个进程在该 CPU 上运行。

很多操作系统教科书将正在 CPU 上执行的进程定义为 RUNNING 状态、而将可执行但是尚未被调度执行的进程定义为 READY 状态，这两种状态在 linux 下统一为 TASK_RUNNING 状态。

8.7.2 Linux 进程状态: S (TASK_INTERRUPTIBLE), 可中断的睡眠状态。

处于这个状态的进程因为等待某某事件的发生（比如等待 socket 连接、等待信号量），而被挂起。这些进程的 task_struct 结构被放入对应事件的等待队列中。当这些事件发生时（由外部中断触发、或由其他进程触发），对应的等待队列中的一个或多个进程将被唤醒。

通过 ps 命令我们会看到，一般情况下，进程列表中的绝大多数进程都处于 TASK_INTERRUPTIBLE 状态（除非机器的负载很高）。毕竟 CPU 就这么一两个，进程动辄几十上百个，如果不是绝大多数进程都在睡眠，CPU 又怎么响应得过来。

8.7.3 Linux 进程状态: D (TASK_UNINTERRUPTIBLE), 不可中断的睡眠状态。

与 TASK_INTERRUPTIBLE 状态类似，进程处于睡眠状态，但是此刻进程是不可中断的。不可中断，指的并不是 CPU 不响应外部硬件的中断，而是指进程不响应异步信号。

绝大多数情况下，进程处在睡眠状态时，总是应该能够响应异步信号的。否则你将惊奇的发现，kill -9 竟然杀不死一个正在睡眠的进程了！于是我们也很好理解，为什么 ps 命令看到的进程几乎不会出现 TASK_UNINTERRUPTIBLE 状态，而总是 TASK_INTERRUPTIBLE 状态。

而 TASK_UNINTERRUPTIBLE 状态存在的意义就在于，内核的某些处理流程是不能被打断的。如果响应异步信号，程序的执行流程中就会被插入一段用于处理异步信号的流程（这个插入的流程可能只存在于内核态，也可能延伸到用户态），于是原有的流程就被中断了。（参见《linux 内核异步中断浅析》）

在进程对某些硬件进行操作时（比如进程调用 read 系统调用对某个设备文件进行读操作，而 read 系统调用最终执行到对应设备驱动的代码，并与对应的物理设备进行交互），可能需要使用 TASK_UNINTERRUPTIBLE 状态对进程进行保护，以避免进程与设备交互的过程被打断，造成设备陷入不可控的状态。这种情况下的 TASK_UNINTERRUPTIBLE 状态总是非常短暂的，通过 ps 命令基本上不可能捕捉到。

8.7.4 Linux 进程状态: T (TASK_STOPPED or TASK_TRACED), 暂停状态或跟踪状态。

向进程发送一个 SIGSTOP 信号，它就会因响应该信号而进入 TASK_STOPPED 状态（除非该进程本身处于 TASK_UNINTERRUPTIBLE 状态而不响应信号）。（SIGSTOP

与 SIGKILL 信号一样，是非常强制的。不允许用户进程通过 signal 系列的系统调用重新设置对应的信号处理函数。)

向进程发送一个 SIGCONT 信号，可以让其从 TASK_STOPPED 状态恢复到 TASK_RUNNING 状态。

当进程正在被跟踪时，它处于 TASK_TRACED 这个特殊的状态。“正在被跟踪”指的是进程暂停下来，等待跟踪它的进程对它进行操作。比如在 gdb 中对被跟踪的进程下一个断点，进程在断点处停下来的时候就处于 TASK_TRACED 状态。而在其他时候，被跟踪的进程还是处于前面提到的那些状态。

对于进程本身来说，TASK_STOPPED 和 TASK_TRACED 状态很类似，都是表示进程暂停下来。而 TASK_TRACED 状态相当于在 TASK_STOPPED 之上多了一层保护，处于 TASK_TRACED 状态的进程不能响应 SIGCONT 信号而被唤醒。只能等到调试进程通过 ptrace 系统调用执行 PTRACE_CONT、PTRACE_DETACH 等操作（通过 ptrace 系统调用的参数指定操作），或调试进程退出，被调试的进程才能恢复 TASK_RUNNING 状态。

8.7.5 Linux 进程状态：Z (TASK_DEAD - EXIT_ZOMBIE)，退出状态，进程成为僵尸进程。

进程在退出的过程中，处于 TASK_DEAD 状态。

在这个退出过程中，进程占有的所有资源将被回收，除了 task_struct 结构（以及少数资源）以外。于是进程就只剩下 task_struct 这么个空壳，故称为僵尸。

之所以保留 task_struct，是因为 task_struct 里面保存了进程的退出码、以及一些统计信息。而其父进程很可能会关心这些信息。比如在 shell 中，\$? 变量就保存了最后一个退出的前台进程的退出码，而这个退出码往往被作为 if 语句的判断条件。

当然，内核也可以将这些信息保存在别的地方，而将 task_struct 结构释放掉，以节省一些空间。但是使用 task_struct 结构更为方便，因为在内核中已经建立了从 pid 到 task_struct 查找关系，还有进程间的父子关系。释放掉 task_struct，则需要建立一些新的数据结构，以便让父进程找到它的子进程的退出信息。

父进程可以通过 wait 系列的系统调用（如 wait4、waitid）来等待某个或某些子进程的退出，并获取它的退出信息。然后 wait 系列的系统调用会顺便将子进程的尸体（task_struct）也释放掉。

子进程在退出的过程中，内核会给其父进程发送一个信号，通知父进程来“收尸”。这个信号默认是 SIGCHLD，但是在通过 clone 系统调用创建子进程时，可

以设置这个信号。

只要父进程不退出，这个僵尸状态的子进程就一直存在。那么如果父进程退出了呢，谁又来给子进程“收尸”？当进程退出的时候，会将它的所有子进程都托管给别的进程（使之成为别的进程的子进程）。托管给谁呢？可能是退出进程所在进程组的下一个进程（如果存在的话），或者是1号进程。所以每个进程、每时每刻都有父进程存在。除非它是1号进程。

1号进程，pid为1的进程，又称init进程。linux系统启动后，第一个被创建的用户态进程就是init进程。它有两项使命：

1. 执行系统初始化脚本，创建一系列的进程（它们都是init进程的子孙）；
2. 在一个死循环中等待其子进程的退出事件，并调用waitid系统调用来完成“收尸”工作；

init进程不会被暂停、也不会被杀死（这是由内核来保证的）。它在等待子进程退出的过程中处于TASK_INTERRUPTIBLE状态，“收尸”过程中则处于TASK_RUNNING状态。

8.7.6 Linux 进程状态：X (TASK_DEAD - EXIT_DEAD)，退出状态，进程即将被销毁。

而进程在退出过程中也可能不会保留它的task_struct。比如这个进程是多线程程序中被detach过的进程（进程？线程？参见《linux线程浅析》）。或者父进程通过设置SIGCHLD信号的handler为SIG_IGN，显式的忽略了SIGCHLD信号。（这是posix的规定，尽管子进程的退出信号可以被设置为SIGCHLD以外的其他信号。）

此时，进程将被置于EXIT_DEAD退出状态，这意味着接下来的代码立即就会将该进程彻底释放。所以EXIT_DEAD状态是非常短暂的，几乎不可能通过ps命令捕捉到。

8.7.7 进程的初始状态

进程是通过fork系列的系统调用（fork、clone、vfork）来创建的，内核（或内核模块）也可以通过kernel_thread函数创建内核进程。这些创建子进程的函数本质上都完成了相同的功能——将调用进程复制一份，得到子进程。（可以通过选项参数来决定各种资源是共享、还是私有。）

那么既然调用进程处于TASK_RUNNING状态（否则，它若不是正在运行，又怎么进行调用？），则子进程默认也处于TASK_RUNNING状态。

另外，在系统调用调用 clone 和内核函数 kernel_thread 也接受 CLONE_STOPPED 选项，从而将子进程的初始状态置为 TASK_STOPPED。

8.7.8 进程状态变迁

进程自创建以后，状态可能发生一系列的变化，直到进程退出。而尽管进程状态有好几种，但是进程状态的变迁却只有两个方向——从 TASK_RUNNING 状态变为非 TASK_RUNNING 状态、或者从非 TASK_RUNNING 状态变为 TASK_RUNNING 状态。

也就是说，如果给一个 TASK_INTERRUPTIBLE 状态的进程发送 SIGKILL 信号，这个进程将先被唤醒（进入 TASK_RUNNING 状态），然后再响应 SIGKILL 信号而退出（变为 TASK_DEAD 状态）。并不会从 TASK_INTERRUPTIBLE 状态直接退出。

进程从非 TASK_RUNNING 状态变为 TASK_RUNNING 状态，是由别的进程（也可能是中断处理程序）执行唤醒操作来实现的。执行唤醒的进程设置被唤醒进程的状态为 TASK_RUNNING，然后将其 task_struct 结构加入到某个 CPU 的可执行队列中。于是被唤醒的进程将有机会被调度执行。

而进程从 TASK_RUNNING 状态变为非 TASK_RUNNING 状态，则有两种途径：

1. 响应信号而进入 TASK_STOPPED 状态、或 TASK_DEAD 状态；
2. 执行系统调用主动进入 TASK_INTERRUPTIBLE 状态（如 nanosleep 系统调用）、或 TASK_DEAD 状态（如 exit 系统调用）；或由于执行系统调用需要的资源得不到满足，而进入 TASK_INTERRUPTIBLE 状态或 TASK_UNINTERRUPTIBLE 状态（如 select 系统调用）。

显然，这两种情况都只能发生在进程正在 CPU 上执行的情况下。

8.8 总结

本章简要介绍了进程概念，如何查看进程，并对进程进行操作控制。本书不会太详细的介绍某个命令，如果需要深入了解，请用 man <command> 查找，或者网上搜索相关资料。

第九章 简单 bash 脚本

通过阅读本章，你将会了解到以下几项内容。

- 理解脚本的概念
- bash 进行判断和简单数学计算
- bash 的流程结构
- 实现把 UC 视频缓存变为普通视频文件的脚本

9.1 何谓 shell 脚本

我认为就是最初的程序员一条条命令写得太累了，能不能把每行命令都放在一个文本文件里，让 shell 自己来读取呢，这样脚本就诞生了。囊括了命令、函数、变量等内容，实现一条命令完成众多工作的功能，可以复杂到启动系统，也可用简单到只有一条命令，类似 Windows 下的批处理文件。

9.1.1 执行和调试

由于 shell 脚本都是文本，可以用任意编辑器打开，可当做 bash 或 zsh 等 shell 的参数来逐行直接执行，比如我们新建一个文本文件，内写上，`uname -a`，保存为 myscript。可以通过如下方式执行，

```
1 $ echo 'uname -a' > myscript
2 $ bash myscript
3 Linux litianci-PC 4.15.0-29deepin-generic #31 SMP Fri Jul 27
   07:12:08 UTC 2018 x86_64 GNU/Linux
```

解释

1. 第 1 行，创建 myscript 文件。
2. 第 2 行，执行该脚本文件
3. 第 3 行，输出结果。

另外，也可用给脚本加上可执行权限，直接运行。通常在脚本第一行`#!/bin/bash`告诉 shell 使用 /bin/bash 执行该脚本。对于使用 Python 或者 R 语言等执行的脚本，相应的把 /bin/bash 换成相应的脚本解释器 Python 或者 RScript 等。

在 shell 脚本中，使用 ## 表示单行注释，也就是从 ## 到行尾的内容为注释内容。当然，有些##属于字符串的内容或者其他语法格式，不表示单行注释。如果你使用 vim 等编辑器打开脚本的时候，会发现注释的颜色是跟其他部分不一样的。通常在脚本第二行开始该脚本的功能注释，也可以添加作者、编辑信息等，然后另起一行注释该脚本的名称。空一行，开始脚本正文内容。如下面所示，

```
1 #!/bin/bash
2 # 本脚本实现 UC 浏览器视频缓存内容转换为一个完整的 MP4 文件
3 # ucvideo
4
5 if [ ! -e $2 ]
6   echo "请按照如下格式调用该脚本"
7 fi
```

写完脚本，保存后，一般使用 chmod u+x ucvideo 的方式，给该脚本添加可执行权限。这样，可以像普通命令那样直接调用该脚本了。

```
1 $ ./ucvideo
```

当然，跟其他程序类似，脚本不可避免的要调试纠错，下面几种方法可能有用，

- 注释掉某些内容，方法就是在行首加##；
- 使用 echo 输出相关参数信息或者其他需要显示的内容。
- 使用 bash -x myscript，会输出每行命令及执行结果，对于循环或者分支判断语句，可以告诉你具体执行了那些内容。

当然，最主要的还是要做到代码整洁，及时给自己的代码注释，避免后面自己都忘记咋回事了。

9.1.2 shell 变量

为了存储一些输出结果，或者一些参数等，需要用到变量存储，方便脚本编写。对于内容偏大的临时结果，也可用使用文件存储。通常采用如下方式，

```
1 NAME=value
```

变量名NAME类似C语言的变量名规则，只可数字字母下划线，数字不可开头，区分大小写，中文不能出现在变量名中。对于变量值value则没有太多要求。通常是字符串、数字等，可以包含中文。比如，

```
1 HOME="中国"
2 e=2.7
```

对于命令的输出结果，通常采用\$(command)和`command`的方式实现。比如，

```
1 MACHINE=`uname -n`
2 TODAY=$(date)
```

如果想获取变量的数值，可以使用\$NAME的方法。

```
1 echo $MACHINE
```

第五章，简要介绍了\$,*,!,等特殊字符。在脚本中有时需要他们的特殊功能，有时候需要他们保持原样，该怎么做呢，通常使用双引号""，单引号' '，以及反斜杠\，看下面例子。

```
1 $ echo $HOME
2 /home/litianci
3 $ echo "$HOME , today is `date`"
4 /home/litianci,today is 2018年 11月 16日 星期五 21:25:46 CST
5 $ echo '$HOME , today is `date`'
6 $HOME , today is `date`
7 $ echo \$HOME \`date\``
8 $HOME `date`
```

特殊字符可以使用\转义为本来样子，直接输出即可。如果作为字符串，使用单引号'则保持原样不变，使用双引号"则实现转义。这在其他语言中也有类似做法。

9.1.3 特殊变量

作为脚本，作为命令来用时，不可避免的要传入参数，在脚本中，通常使用 $\$0$, $\$1, \dots, \n 的方式来获取这些参数值。其中 $\$0$ ，表示本脚本； $\$n$ ，表示第 n 个输入参数。不管是`bash myscript`还是`./myscript`方式执行脚本，上述 $\$n (n! = 0)$ 都是一样的。另外 $\$\#\#$ 表示共有多少个参数， $\$@$ 保存着正行的输入。 $\$?$ 显示上一个命令的返回状态，一般返回 0 表示正常，其他数值表示异常或错误。看下面例子，脚本`myscript`的内容为。

```
1 #!/bin/bash
2 # 测试这些特使变量
3 # myscript
4
5 echo "第一个参数为： \$1, 第二个参数为 \$2."
6 echo "共有 \$# 个参数"
7 echo "这些参数是 \$@"
8 echo "该脚本名称为： \$0."
```

分别执行`bash myscript first second`和`./myscript first second`，结果如下，

```
1 $ bash myscript first second
2 第一个参数为： first, 第二个参数为 second.
3 共有 2 个参数
4 这些参数是 first second
5 该脚本名称为： myscript.
6 $ chmod u+x myscript # 设置文件执行属性
7 $ ./myscript first second
8 第一个参数为： first, 第二个参数为 second.
9 共有 2 个参数
10 这些参数是 first second
11 该脚本名称为： ./myscript.
12 $ echo $?
13 0
```

9.1.4 执行时输入参数

$\$n (n > 0)$ 一般都是执行前输入的参数，有时候还需要在执行中跟用户交互，那就用到`read`命令了。新建脚本`readscript`内容如下，

```
1 #!/bin/bash
2 # 测试交互信息
3 # readscript
4
5 read -p "你叫啥名字，几岁啦？（两个答案请用空格隔开）" name age
6 echo "我知道啦，你叫 $name, $age 岁了。"
```

执行该脚本，

```
1 $ bash readscript
2 你叫啥名字，几岁啦？（两个答案请用空格隔开）深度易经 3
3 我知道啦，你叫 深度易经，3 岁了。
```

关于**read**命令的更多内容，比如输入密码，或者其他信息，请**read --help**查看。

9.1.5 其他需求的参数

有些脚本，输入的参数可能有默认值，使用**var1=\${var2:-defaultValue}**的方式，意思是变量**var2**如果存在赋值给**var1**，否则，把**defaultValue**赋值给**var1**。

```
1 $ Birthday="2018-11-11"
2 $ Birthday=${Birthday:-`date`}
3 $ echo $Birthday
4 2018-11-11
5 $ Birthday=${DefaultDate:-`date`}
6 $ echo $Birthday
7 $ echo $Birthday
8 2018年 11月 19日 星期一 21:48:35 CST
9 $ Name=${Name:-'尚未设置'}
10 $ echo $Name
11 尚未设置
```

有时，我们需要对参数再处理，比如对于路径的不同取舍，对某些合并参数等的提取，可能使用其他正则表达式截取部分字符串更合适，不过 Shell 还是提供了一些简单的截取功能。

- **\${var##pattern}**: 从头删除满足匹配模式**pattern**的最短子字符串。
- **\${var###pattern}**: 从头删除满足匹配模式**pattern**的最长子字符串。
- **\${var%pattern}**: 从尾删除满足匹配模式**pattern**的最短子字符串。

- \${var%%pattern}：从尾删除满足匹配模式pattern的最长子字符串。

```

1 $ readme=/home/litianci/deepin-bible/Readme.md
2 $ file=${readme##*/}
3 $ echo $file
4 Readme.md
5 $ dir=${readme%/*}
6 $ echo $dir
7 /home/litianci/deepin-bible
8 $ stringchar="--folder=ucvideo"
9 $ option=${stringchar%*=}
10 $ echo $option
11 --folder
12 $ value=${stringchar#*=}
13 $ echo $value
14 ucvideo

```

9.1.6 简单计算

bash 认为所有的输入都是字符串或者文本，如果你想让 bash 理解为数字，貌似只可以整数，就得声明，比如使用let,expr,bc等方式。

```

1 Total=1024
2 let div=$Total/8 # let 表达式中间不可以有空格，所有参数必须为整数
3 div=`echo "$Total / 8" | bc` # bc 对空格不敏感，可以有小数，但是结果还是取整
4 div=`echo "$Total      /      9.8      " | bc`
5 div=`echo "$Total/9.8" | bc`
6 div=`expr $Total / 8` # expr 必须有空格，所有参数必须为整数
7 echo $RANDOM # random 生成随机数

```

同时使用((statement))也可用实现简单的数学语句，

```

1 i=0
2 ((i++))
3 echo $i
4 echo $((i++))
5 echo $((++i))
6 ((i=i+10))

```

```
7 echo $i
```

其中`i++`跟`++i`实现`i`数值加一，这两者区别类似 C 语言的规定，`i++`是先使用后递增一，`++i`是先递增一再使用。

9.2 shell 脚本的三大结构

学过编程语言的，应当多多少少都知道结构化编程语言的三大结构：[顺序](#)、[分支](#)和[循环](#)。shell 脚本支持这三种结构的。顺序执行就是逐行执行的命令，这里略过，下面介绍分支结构的语法。

9.2.1 分支结构的语法

9.2.1.1 if then 语句

语法结构如下，

```
1 if [ condition1 ] ; then
2   statement1
3 elif [ condition2 ] ; then
4   statement2
5 else
6   statement3
7 fi
```

解释

- [`condition1`] 这里是测试语句，用于测试条件是否满足，满足则执行`statement1`语句。
- `elif` 是`else if`的意思，用于多个测试条件。
- 其中`elif`和`else`部分可以不要。
- `fi`其实是`if`的倒序写法。后面的`case`的结尾语句`esac`是同样的处理方式。

```
1 touch emptyfile
2 if [ ! -s emptyfile ] ; then
3   echo "emptyfile 是一个空文件"
4 fi
```

解释

1. [! -s emptyfile] 中 -s 判断一个文件存在且非空是为真, ! 是逻辑运算取反的意思。
2. 测试条件, 需要注意[]以及各个选项、运算符、参数均用空格隔开。
3. 关于测试条件的更多信息, 可以使用 help test 命令查看。

表 9-1 常见测试语句含义 (help test 截取)

测试语句	解释
-a FILE	True if file exists.
-b FILE	True if file is block special.
-c FILE	True if file is character special.
-d FILE	True if file is a directory.
-e FILE	True if file exists.
-f FILE	True if file exists and is a regular file.
-g FILE	True if file is set-group-id.
-h FILE	True if file is a symbolic link.
-L FILE	True if file is a symbolic link.
-k FILE	True if file has its ‘sticky’ bit set.
-p FILE	True if file is a named pipe.
-r FILE	True if file is readable by you.
-s FILE	True if file exists and is not empty.
-S FILE	True if file is a socket.
-t FD	True if FD is opened on a terminal.
-u FILE	True if the file is set-user-id.
-w FILE	True if the file is writable by you.
-x FILE	True if the file is executable by you.
-o FILE	True if the file is effectively owned by you.
-G FILE	True if the file is effectively owned by your group.
-N FILE	True if the file has been modified since it was last read.
FILE1 -nt FILE2	True if file1 is newer than file2 (according to modification date).
FILE1 -ot FILE2	True if file1 is older than file2.
FILE1 -ef FILE2	True if file1 is a hard link to file2.
-z STRING	True if string is empty.
-n STRING	True if string is not empty.

测试语句	解释
STRING	True if string is not empty.
STRING1 = STRING2	True if the strings are equal.
STRING1 != STRING2	True if the strings are not equal.
STRING1 < STRING2	True if STRING1 sorts before STRING2 lexicographically.
STRING1 > STRING2	True if STRING1 sorts after STRING2 lexicographically.
-o OPTION	True if the shell option OPTION is enabled.
-v VAR	True if the shell variable VAR is set.
-R VAR	True if the shell variable VAR is set and is a name reference.
! EXPR	True if expr is false.
EXPR1 -a EXPR2	True if both expr1 AND expr2 are true.
EXPR1 -o EXPR2	True if either expr1 OR expr2 is true.
arg1 OP arg2	Arithmetic tests. OP is one of -eq, -ne, -lt, -le, -gt, or -ge.

9.2.1.2 && || 条件语句

该条件语句，是if..then单条语句的简写，语法结构如下，

```
1 [ condition ] && statement1 || statement2
```

其中[condition]跟if后的判断条件一致，&&后的语句 statement1是条件满足的执行语句，||后的语句statement2是条件不满足执行的语句。且&&和||可以根据需要略掉。

```
1 [ -s emptyfile ] || echo "emptyfile 不存在或者是空文件"
2 [ -e umdir ] && echo "可以继续执行" || echo "不存在该文件夹，请核实后继续"
```

9.2.1.3 case 条件语句

类似 C 语言的switch分支语句，语法结构如下，

```

1  case $var in
2      'value1')
3          { statement1 }
4          ;;
5      'value2')
6          { statement2 }
7          ;;
8  *)
9      { statement3 }
10    ;;
11 esac

```

解释

1. \$var表示输入的某个变量，或者一个运行结果，
2. *表示其他都不满足的执行语句。

```

1  case $destination in
2      'Shanghai')
3          echo "你想去上海啊！那边风景不错。"
4          ;;
5      '北京')
6          echo "北京天气不好，雾霾多！"
7          ;;
8  *)
9      echo "不允许去其他地方！"
10    ;;
11 esac

```

上面语句实现了对目的地的简单判断，支持汉字输入。

9.2.1.4 for...do 循环语句

类似 C 语言的for循环，语法结构如下，

```

1  for i in array ; do
2      statement

```

3 **done**

实现使用参数*i*遍历数组array的所有子元素，并对每个*i*执行statement语句。看下面例子，

```
1 for i in {1..5} ; do  
2   echo $i  
3 done
```

会输出如下结果，

```
1 1  
2 2  
3 3  
4 4  
5 5
```

再看几个例子，

```
1 for file in `ls` ; do  
2   echo $file  
3 done  
4  
5 for destination in 成都 上海 北京 广州 徐州 ; do  
6   echo 我很想去$destination  
7 done
```

对于循环，除了使用{m..n}列出从m到n的数字外，还可以用下面这种方式，关于(())作为数学计算的介绍，上面已经提及，这里不再赘述。

```
1 for ((i=1; i <= 5 ; i++)) ; do  
2   echo $i  
3 done
```

9.2.1.5 while..do 和 until..do 循环语句

可能我们更熟悉while语句，跟其他C系列的语言相似，语法结构如下，

```
1 while condition ; do  
2     statement  
3 done
```

比如下面这段代码，实现了按序输出i

```
1 i=0  
2 while ((i<5)) ; do  
3     echo $((++i))  
4 done
```

until的语法跟while类似，语法结构如下，

```
1 until condition ; do  
2     statement  
3 done
```

只不过until条件是不满足才循环，相当于while的条件condition取反。不再举例子了。

9.3 流编辑器 sed

流编辑器sed还算常用，操作方式跟vim有相似之处，下面摘抄几个例子，详细内容请参考man sed或者网上搜索vim sed相关资料。

```
1 sed 's/Windows/Linux/g' deepin.txt > ok\_deepin.txt
```

如果你常用vim会发现，当光标在某一行，在正常模式下输入:s/Windows/Linux/g实现该行所有的Windows被Linux替代。而sed命令会逐行执行，也就实现了对全文的替换。

9.4 shell 脚本例子：转换 UC 缓存视频

阿里宝卡正当时，UC 看视频免流量是个多么大的诱惑。当然一些缓存视频，也是相当不错的。一般都会存在于./UCDownloads/videodata 文件夹下。文件一般

是从0开始排序，顺序增加，可达几百个文件。另外还有一个index.m3u8文件。但是，因为网络协议对视频文件的第一要求是及时传达，而不是完整传达，导致部分视频文件丢失，甚至为空，所以需要妥善处理这些文件。

首先，我们需要知道如何把若干个视频文件转化为一个完整的视频文件，参阅[网页](#)，我们得知，

如果存在文件file.txt，其内容为

```
1 file '1'  
2 file '2'
```

则，

```
1 ffmpeg -f concat -i input.txt -c copy output.mp4
```

即可把这些文件给转换为 MP4 格式的单个文件。

9.4.1 生成 file.txt 文件

因为缓存文件都是数字，且文件夹内还有其他文件，包括index.m3u8的文件。

```
1 $ ls -1v +([0-9]) > file.txt
```

解释

- ls -1v 中v表示按照把文件按照数字的大小排序，1表示按行显示。
- +([0-9])只选择纯数字的文件。

对file.txt文件再处理，生成每行类似file '1'的样式。

```
1 $ sed "s/.*/file '&/'" file.txt > file2.txt  
2 $ mv file2.txt file.txt
```

解释

- 第一行实现对每一行行首行尾分别添加file '和'内容。
- 第二行，重命名，替换掉中间文件。

参考网页：

- <https://unix.stackexchange.com/questions/33909/list-files-sorted-numerically>
- <https://superuser.com/questions/716001/how-can-i-get-files-with-numeric-names-using-ls-command>
- <https://www.shellhacks.com/sed-awk-add-end-beginning-line/>

9.4.2 生成 MP4 文件

前提需要你安装 `ffmpeg` 软件，如果没有，命令行 `sudo apt-get install ffmpeg` 安装。

```
1 $ ffmpeg -f concat -i file.txt -c copy film.mp4
```

9.4.3 做成一个 bash 脚本

下面是完整的代码

```
1 #!/bin/bash
2 # 本脚本实现 UC 浏览器视频缓存内容转换为一个完整的 MP4 文件
3 # ucvideo
4
5 echo "语法: $0 <UC浏览器视频缓存文件夹> <输出文件>"
6
7 output='ucvideo.mp4'
8 output=${2:-$output} # 读第二个参数作为输出文件
9
10 if [ -e $output ]
11 then
12     echo "已经存在 $output 文件, 请更改输出文件名字!"
13     exit 1
14 fi
15
16 umdir=${1:-'.'} # 第一个参数作为缓存文件夹, 默认为当前文件夹
17 if [ ! -d $umdir ]
18 then
19     echo "$umdir 文件夹不存在!"
20     exit 2
21 fi
22
23 cd $umdir
```

```

24
25 find ./ -size -1b -exec rm {} \; # 删除空文件
26 ls -1v | grep -E '^([0-9]+)$' | sed "s/.*/file '&'/" > file.txt # 把
    数字文件按序，并加上行首行尾，写入 file.txt
27
28 ffmpeg -f concat -i file.txt -c copy $output
29
30 if [ ! $ucdir -ef $OLDPWD ];then
31     mv $output $OLDPWD
32 fi

```

当然为了方便直接使用，把他放在`/usr/local/bin/ucvideo`。并为之添加执行权限

```
1 $ chmod u+x /usr/local/bin/ucvideo
```

当你从手机上复制缓存文件夹过来，就可以

```
1 $ ucvideo /<指向缓存文件夹>/ <电影名称>.mp4
```

期间发生了一个小问题，bash 命令在脚本和终端下运行不一致。比如`ls -1v +([0-9])`可以在终端下运行，但是在脚本里就会报错。`bash <<<'ls -1v +([0-9])'`是无法运行的。所以上面解决方案就出现了部分调整。

也就是说一般在 deepin 下，sh 或者 /bin/sh 指向 /bin/dash，可以通过 `sudo dpkg -reconfigure dash` 关闭 dash，改为默认的 /bin/bash。其中 dash 跟 bash 还是存在一些小差异。

不过，通过 `echo $SHELL` 和查看 `/etc/passwd` 文件，看当前用户的 shell，均为 bash。但是在终端默认的 shell 可以执行 `ls -1v +([0-9])`，在脚本里使用 /bin/bash 执行 `/bin/bash <<<'ls -1v +([0-9])'` 执行就是报错。原因待查。

后来无意间在 [unixstackexchange](#) 发现，是没有启用 glob 扩展，在该行代码前一行加上 `shopt -s extglob` 即可。

关于 glob 的更多介绍，

- https://en.wikipedia.org/wiki/Glob_%28programming%29
- <http://www.tldp.org/LDP/abs/html/globbingref.html>
- <http://www.mamicode.com/info-detail-1227028.html>

9.5 总结

本章简要介绍了 bash 的若干语法结构，并给出了一个小例子。

第三部分

本机管理员

“一屋不扫，何以扫天下”

—— 出处《习惯说》

本部分介绍软件安装、用户管理、磁盘管理等基础知识，为后续工作打牢基础。

第十章	163
第十一章 深度系统安装	165
第十二章 软件安装	167
第十三章 管理账户	171

第十章

workrave

第十一章 深度系统安装

本章讲介绍如何从优盘、光盘和硬盘安装深度操作系统，以及 LiveCD 模式简介。

参考网页：

- <https://bbs.deepin.org/forum.php?mod=viewthread&tid=135051>
- <https://wiki.deepin.org/index.php?title=%E5%8E%9F%E7%94%9F%E5%AE%89%E8%A3%85>

对以上作者表示感谢。

11.1 UEFI 和 legacy BIOS 区别和联系

参考网页：

- <https://baike.baidu.com/item/UEFI/3556240?fr=aladdin>

有些电脑过于老旧，不支持 UEFI 模式，或者没能设置启动（boot）为 UEFI 模式，导致安装失败。

11.2 优盘安装

一般电脑都是可以安装的，当然也存在安装失败的情况，对于遇到困难的用户，可以百度或者谷歌怎么解决。这里只叙述需要注意的问题。现在的电脑一般都带 USB 接口，支持优盘安装的。USB 接口，现有 1.0,2.0,3.0 等各种版本，更多了解可以参考[百度百科](#)。在采用优盘安装的时候，需要事先检查你电脑的 USB 接口是否工作正常，之前帮一同事安装系统，折腾了一上午才发现 USB 接口是坏的。更有甚者，部分电脑的 USB 3.0 的接口，可能不支持优盘安装，这个都是需要特别注意的。

接下来是制作优盘。你可以采用 windows 或者已经安装的深度操作系统制作优盘，当然对优盘大小等还是有限制的，一般大于 8G 为宜。

11.2.1 第一步下载并校验 deepin.iso 文件

本节转载自[深度维基百科](#)，有改动。对前人的辛勤工作，表示感谢。

官方镜像

访问 deepin 社区[下载页面](#)，下载深度操作系统系统最新版本的镜像文件（以便您能够体验到最新特性）。（据我本人的经验，如果你没有百度 VIP 账号，不建议从百度网盘下载。国内从官网或者[sourceforge](#)相对快一点，也不是绝对的，可以根据实际情况选用不同的下载点）注意：为了更加专注系统的发展，deepin 15.4 及后续版本将不再提供 32 位官方 iso 镜像，如需获取和技术支持，support@deepin.org。

MD5 校验

下载深度操作系统镜像完成后，需要对其进行校验，非官方或不完整的镜像将不能用于深度操作系统的安装：

- Windows 系统：下载 Hash 软件，校验您下载的镜像的 MD5 值与下载页面提供的 MD5 值是否一致。（MD5 值在立即下载按钮下方）
- Linux 系统：在对应的镜像文件下，打开深度终端，执行 `md5sum deepin-xxx.iso` 命令，请确认下载的镜像的 MD5 值与下载页面提供的 MD5 值是否一致。（MD5 值在立即下载按钮下方）。说明：deepin-xxx.iso 即为下载的系统镜像文件名，可使用 Tab 键自动补全文件名。

11.3 安装遇到的问题

参考网页：

- <https://bbs.deepin.org/forum.php?mod=viewthread&tid=146222>
- <https://bbs.deepin.org/forum.php?mod=viewthread&tid=146224>

基本全文转载，有改动。已获许可。

第十二章 软件安装

这里会介绍如下几种安装方式，

- 直接双击安装
- apt 安装
- dpkg 安装
- .run 类型的文件安装
- 其他软件安装，比如 you-get 的安装
- 字体的安装
- 源码安装
- 输入法的安装
- iso 文件安装
- pandoc 安装

12.1 npm 软件的安装

这里以 you-get 软件为例，介绍 npm 软件的安装方式。

12.2 veil 软件安装

其他发行版可以直接用 apt-get 直接安装的，比如 veil，

```
1 $ sudo apt-get install veil
```

在 Kali Linux 就可以直接运行，但是在深度操作系统下就无法下载，这个时候不妨网上搜索一下，加入私有源。当然最好的是搜到这个软件的官网，比如 veil 的官网：<https://github.com/Veil-Framework/Veil>，按照原作者的方法安装，也是不错的。

```
1 $ sudo apt-get -y install git
2 $ git clone https://github.com/Veil-Framework/Veil.git
3 $ cd Veil/
4 $ cd setup
```

```
s $ sudo ./setup.sh -c
```

这样一路点击确认下来，慢慢等待，就可以成功安装 `veil` 了，当然还需要对 `setup.sh` 文件检测操作系统的代码改一改，让他支持 `deepin`。详情见[pull request](#)。用方法有所变样。

也可以到 `debian` [软件包库](#)去搜索，相关的软件包，选择合适的版本安装。不过 `veil` 没有对应的 `deb` 包。这是一个比较庞大的软件，估计也不会有一个小的包。

12.3 dpkg 安装

12.4 其他安装方式

很多软件，由于依赖或者其他原因，安装起来特别费劲，对于初学者，不啻于一场灾难。有好事者，就提供了各种一键安装包。下面介绍两个，限于本人水平，也就只知道这俩，有同仁知道更多的谢谢发表评论。

12.4.1 bitnami

官网：<https://bitnami.com/stacks>

提供一键安装软件包，或者虚拟机级别文件等。有需要的可以去查询，肯定有你所需要的。比如 `gitlab`, `OwnCloud` 等等

12.4.2 turnkeylinux

官网：<https://www.turnkeylinux.org/>

对于常见的 `gitlab`,`lamp` 等提供虚拟机安装包，可以尽情享受快捷。

另外，我常用的 `xampp` 一键安装包，却是从其他网站上下载的。如果你安装一个软件特别费劲，(不包含无法破解软件的哦) 不妨试试搜索一下“`xxx` 一键安装包”，看看有没有其他人已经帮你解决这个问题了。

12.5 you-get 软件的安装

参考网页：

- <https://github.com/soimort/you-get>
-

根据 `you-get` [源码网站](#)的提示，在 `deepin` 下，用如下命令安装，较为合适。

```
1 $ sudo pip3 install you-get
```

但是，你可能会发现，自己的电脑没有安装 pip3 软件。试着 sudo apt-get install pip3 也不存在这个软件包啊。接着百度 how to install pip3，找到一大堆页码。参考[这个](#)，如下命令安装。

```
1 $ sudo apt-get install python3-pip python3-dev build-essential
```

这样就安装好了 pip3，我的理解，pip3 就是 python3-pip 的意思，不知此解对不对。那么接下来安装 you-get 就容易多啦。

```
1 $ sudo pip3 install you-get
```

如果需要更新 you-get，可以如下。

```
1 $ sudo pip3 install --upgrade you-get
```

对于 you-get 如何使用，这里就不赘述啦。详见[官网说明](#)。

12.6 字体的安装

有些书籍的制作需要特殊的字体，在 Windows、Mac 下安装字体比较简单，在 Deepin 下安装字体同样很简单。参考[这里](#)，

下载需要的字体，下面以安装交大论文需要的字体为例，网上下好需要的字体，解压缩后，双击安装即可，也可以仿照上面[网页](#)提供的方法安装。

```
1 $ mkdir ~/.fonts
2 $ cp *.ttf ~/.fonts          # 当前用户可用新字体
3 $ sudo cp *.ttf /usr/share/fonts/local/ # 所有用户可以使用新字体
4 $ sudo fc-cache -f
```

解释 - cp *.ttf ~/.fonts 的 *.ttf 表示你下载的那些字体，需要你修改为对应字体的路径。

12.7 输入法的安装

这里以搜狗输入法为例，讲解怎么在没有安装搜狗输入法的 Linux 桌面发行版上安装搜狗输入法，其实深度操作系统自带搜狗输入法的。当然第一步百度搜索了，搜索到搜狗输入法的 Linux 专用版[网站](#)，上面网址如果不对，请继续搜索，毕竟某搜索网址广告比较厉害，不一定就第一时间提供有用的答案。进入官网，下载对应 *.deb 文件，使用命令

```
1 $ sudo dpkg -i /path/to/***.deb
```

就可以安装了。本来这个输入法安装应该放在上面的，但是考虑到很多人找不到怎么安装，特来说一下。深度操作系统下，可以双击下载的 *.deb 安装包的，一会就会自动安装完。不过汉语简体版的深度操作系统已经自带了该输入法。

第十三章 管理账户

13.1 引言

本章会涉及到

- 账户 (user account) 管理
- 账户组管理
- 其他方式的账户管理

深度操作系统安装过程中，会引导你设置一个普通账户，非根账户，也就是没有最高权限的普通账户。当有家人、朋友或者同事要使用我们的电脑的时候，可能要涉及到账户的添加和权限管理。这就是本章要讨论的内容。为了更方便的管理多个具有相同权限的账户，还会涉及到账户组的设置等。接下来会讲到怎么使用 `useradd`,`usermod` 等命令，配置自家目录、默认的 shell、分组、以及账户和组编号等。

13.2 创建账户

为了更好的分配资源权限，保护信息安全，为某些使用者创建新的账户还是很有必要的。下面介绍，在深度操作系统下，两种创建账户的方式：图像方式 (GUI)，命令方式 (Shell)。

13.2.1 图形方式

如下图所示，创建账户是非常简单的，打开设置侧栏，找到账户相关的那一部分，点击创建账户，进入到下图，填上用户名和密码等需要的数据，就可以了。



如果需要更改头像、设置全名、更改密码、设置是否自动登录免密码登录，可以选中需要修改的账户，点击进去就可以啦。

图形方式默认创建的账户是普通账户，比如是 `phptester` 账户，在 `/etc/`-



图 13-1 图形方式创建账户



图 13-2 图形方式配置账户



图 13-3 图形方式配置账户

passwd, */etc/group* 会出现 phptester 相关的信息。有兴趣的读者，可以先去看看上述俩文件，后面还会对这俩文件作进一步的讲解。

13.2.2 命令方式

第四部分

服务器管理员

“运筹帷幄之中，决胜千里之外”

—— 刘邦

在服务器领域，Linux 无愧于操作系统中的王者，本部分介绍几个常见服务器软件的安装使用维护的方法。

第十四章 服务器简介 177

第十五章 Web Server 的搭建与运行 179

第十六章 与 Windows 共享文件打印机的 samba 服务 181

第十四章 服务器简介

通过阅读本章，你将会了解到以下几项内容。

- 管理 Linux 服务器
- 访问远程或局域网服务器
- 服务器的本地或远程日志记录
- 监控服务器
- 服务器的开启终止与状态查询

深度操作系统不仅仅是桌面办公的操作系统，还可以作为服务器使用。当然，也可根据自己的需要，用把深度桌面操作系统作为办公环境，另外安装个 redhat/-centos 或者 debian 的服务器，以求更稳定的服务器。

第十五章 Web Server 的搭建与运行

随着互联网技术的发展，网络开发越来越普及，越来越多的内容以网页的形式呈现。掌握一些网站服务器知识还是挺实用的。

15.1 面向开发的一键安装类型

前面我们提到了几个教学软件，

15.2 关注效率稳定的搭建方法

第十六章 与 Windows 共享文件打印机的 samba 服务

通过阅读本章，你将会了解到以下几项内容。

- 深度系统如何跟 Windows 系统共享文件、打印机；
- bash 进行判断和简单数学计算
- bash 的流程结构
- 实现把 UC 视频缓存变为普通视频文件的脚本

Windows 局域网中，为了方便互传文件或者打印，常常开启文件夹共享。在网上邻居里，能够看到其他人共享的文件夹，方便下载甚至上传。当前 Windows 局域网的通信工具也有一些，基本都支持文件互传，比如[飞鸽传书](#)（跨平台支持苹果、Linux 和安卓）、[imo 内网通](#)、[RTX 腾讯通](#)（貌似停更在[2015 版](#)）、[飞秋](#)（貌似停更在[2013 版](#)）、[微软 LYNC](#)（貌似停更在[2013 版](#)）等。对于互联网，除了 FTP、SCP 和 SSH 外，[QQ](#)（深度系统自带），[安司密信](#)、[钉钉](#)（暂时不支持 Linux 平台）等也是不错的选择。

但是，对于局域网的 Deepin 深度用户来讲，除了[iptux 信使](#)（只支持 Linux、Mac，号称 GNU/Linux 版飞鸽传书）、[飞鸽传书](#)（最新版跨平台支持 Linux）外，还可以安装 samba 文件传输服务器。

16.1 Samba 文件传输服务简介

参考网页：

- <https://docs.microsoft.com/en-us/windows/desktop/FileIO/microsoft-smb-protocol-and-cifs-protocol-overview>
- <https://www.cnblogs.com/LittleHann/p/6916326.html>

Samba 是一种网络文件共享的应用程序，基于微软制定的 SMB（Server Message Block）通信协议，被很多种操作系统，比如 Windows、OS2、Linux 等，作为 CS（Client-Server 客户端服务器端）的网络架构。通过基于 SMB 的 samba，Linux 系统就可以愉快的跟 Windows 系统共享文件和打印机。

在 Samba 的帮助下，Windows 看 Linux 电脑的文件和打印机，仿佛就是自家的，反过来亦然。

16.2 深度自带 samba 的使用

参考网页：

- <https://wiki.deepin.org/wiki/%E6%B7%B1%E5%BA%A6%E6%96%87%E4%BB%B6%E7%AE%A1%E7%90%86%E5%99%A8>

其实 samba 已经被深度文件管理器集成了。通过搜索深度文件管理的源码，会发现很多 samba 相关的代码。也就是说，作为用户，我们不需要再单独安装 samba 服务了。

16.2.1 共享本地文件

1. 在文件管理器界面上，右键单击文件夹。
2. 选择共享文件夹。
3. 勾选共享此文件夹。
4. 根据需要设置共享名、权限、匿名访问后关闭标签。
5. 在文件管理器界面上，点击 Icon_menu。
6. 选择设置共享密码。
7. 输入共享密码。
8. 点击确定。

注意：

1. 取消勾选 **共享此文件夹** 可以取消文件共享，也可以右键单击文件，选择 **取消共享**。
2. 有可能部分文件夹由于缺乏相应权限无法共享，请安装前面章节介绍更改权限的方法重新进行共享设置。

16.2.2 访问共享文件

局域网中其他用户共享的文件一般都可以在网络邻居中找到，您也可以通过网络邻居访问共享文件。

1. 输入局域网用户的共享地址，按下键盘上的 Enter 键。（如：smb://xxx.xxx.xxx/share）
2. 输入用户名密码或者匿名访问。
 - 1) 未加密的网络文件可以匿名访问，不需要输入用户名和密码。
 - 2) 加密的网络文件会弹出登陆框，输入账号和密码之后才能访问。如果在用户名密码提示框中勾选 **记住密码**，再次访问不再需要密码。
3. 点击 **连接**。

16.2.3 我的共享

当您设置了共享文件时，我的共享图标将会出现在导航栏上，当所有共享文件都取消共享后，我的共享图标自动从侧边栏中移除。

16.2.4 可能存在的小问题

16.1 共享失败，显示缺乏权限。

答. 如果文件夹不是自己的，设置为共享会弹出如下的问题框，采用管理员权限打开文件夹，成功共享。当然也可以根据异常提示，修改或添加/etc/samba/smb.conf 的usershare owner only=false。



图 16-1 共享文件夹出现的问题

由于权限的问题，取消共享也会出现类似问题，也请采用管理员权限或者修改配置文件的方法取消共享。



图 16-2 取消共享出现的问题

16.3 关于 samba 的配置

16.3.1 手动安装

如果你的操作系统不是 deepin，或者尚未自带 samba 服务，则需要先安装 samba 才可以使用其功能。安装方法很简单，

```
1 # 如果您的操作系统是 centos 或者 Fedora, redhat 等。  
2 $ sudo yum install samba -y  
3 # 如果你的操作系统是 debian 系列，比如 ubuntu 等。  
4 $ sudo apt-get install samba -y  
5 # 其他版本的操作系统略。
```

16.3.2 启动与停止

因为深度采用的 `systemctl` 控制服务，所以可以采用如下命令，

```
1 $ sudo systemctl start smbd.service # 开启  
2 $ sudo systemctl status smbd.service # 查看状态  
3 $ systemctl status smbd # 查看状态不需要根权限  
4 $ sudo systemctl stop smbd.service # 关闭  
5 $ sudo systemctl restart smbd.service # 重启
```

其实 `smbd.service` 可以省略为 `smbd`。当然，还可以用另外一种方式，直接调命令文件，启动这些命令。比如，

```
1 $ sudo /etc/init.d/smbd start
```

这里是直接调用的 `smbd` 程序，最后的 `start` 可以换成 `stop`, `status` 和 `restart`，效果跟上面 `systemctl` 的方式一致。对于 `status` 状态的查询，同样不需要 root 根权限。对于 redhat 系列没有实测，查部分资料，把命令中 `smbd` 改为 `smb` 即可。

16.3.3 配置文件

Samba 的配置文件为 `/etc/samba/smb.conf`。

vim

16.4 例：借助安卓软件 U-File 实现手机电脑互传

本例是建立在局域网上的手机和电脑的互传，先介绍局域网的两种建立方法。如果您已经配置好了局域网，请跳过这一步。

16.4.1 开启局域网

可以使用手机开热点或者电脑开热点的方法。

由于本人所处工作环境电脑接触互联网不太方便，常用手机开热点的方法蹭网。手机开好热点后，电脑连接到相关无线网，在电脑命令行输入 `hostname -I` 会显示电脑的 IP 地址，本机为 `192.168.43.45`。也可用通过手机端查看已经连接到热点设备的 IP 地址。

深度操作系统支持开启无线网热点。控制中心-> 网络-> 热点-> 热点设置，即可进入无线网设置页面，如图16-3所示，



图 16-3 开启无线网热点

第五部分

安全

“三军之事，莫亲于间，赏莫厚于间，事莫密于间。非圣智不能用间，非仁义不能使间，非微妙不能得间之实。微哉！微哉！无所不用间也。”

—— 孙武《孙子兵法用间》

古人云“千里之堤，溃于蚁穴”。害人之心不可有，防人之心不可无。作为系统使用者和管理者，必须牢牢绷紧安全这根弦。本部分简要的介绍一些安全常识，也只是避免一些小的安全隐患。更多知识，还需查阅更专业的安全书籍和资料。

第十七章 网络安全

本章您可以了解到，

- 网络安全是啥

第六部分

附录

“这只是万里长征走完了第一步，以后的路程更长，工作更伟大，更艰苦。务必使同志们继续地保持谦虚、谨慎、不骄不躁的作风，务必使同志们继续地保持艰苦奋斗的作风。”

—— 毛泽东

附录，首先引用了中国的一篇古文《愚公移山》，目的是告诉自己要坚持坚持再坚持；其次讲述了本书的制作步骤，方便后来人对模板进行改编和再创作；再次介绍了本书的大致框架设计，提醒需要注意的问题；再次列举了写作中常见的 L^AT_EX 和 Bookdown 编写的示例；最后是常见问题解答以及本书的大事记。

附录甲 愚公移山	193
附录乙 如何制作本书	195
附录丙 各章格式说明	201
附录丁 兼容 L ^A T _E X 排版	205
附录戊 RMarkdown/Bookdown 排版示例	219
附录己 常见问题	253
附录庚 操作系统安装延伸阅读	259
附录辛 大事记	263

附录甲 愚公移山

——先秦 列御寇



图甲-1 徐悲鸿名画《愚公移山》

太行、王屋二山，方七百里，高万仞，本在冀州之南，河阳之北。

北山愚公者，年且九十，面山而居。惩山北之塞，出入之迂也，聚室而谋曰：“吾与汝毕力平险，指通豫南，达于汉阴，可乎？”杂然相许。其妻献疑曰：“以君之力，曾不能损魁父之丘，如太行、王屋何？且焉置土石？”杂曰：“投诸渤海之尾，隐土之北。”遂率子孙荷担者三夫，叩石垦壤，箕畚运于渤海之尾。邻人京城氏之孀妻有遗男，始龀，跳往助之。寒暑易节，始一返焉。

河曲智叟笑而止之曰：“甚矣，汝之不惠！以残年余力，曾不能毁山之一毛，其如土石何？”北山愚公长息曰：“汝心之固，固不可彻，曾不若孀妻弱子。虽我之死，有子存焉；子又生孙，孙又生子；子又有子，子又有孙；子子孙孙无穷匮也，而山不加增，何苦而不平？”河曲智叟亡以应。

操蛇之神闻之，惧其不已也，告之于帝。帝感其诚，命夸娥氏二子负二山，一厝朔东，一厝雍南。自此，冀之南，汉之阴，无陇断焉。

附录乙 如何制作本书

乙.1 准备工作

本模板使用 bookdown 实现了由 R Markdown (Bookdown) -> Markdown (Pandoc 标准) -> L^AT_EX -> PDF 的一系列转换，其中用到的依赖有：

- R
- pandoc
- xetex

需要安装这些软件，才可以使用。

乙.1.1 Linux 下使用

下面以深度操作系统 15.5 版本为例说明。安装过程中，请确保当前 rstudio 的版本高于 1.0.0, texlive 为 2015 年后的版本，pandoc 的版本高于 2.0.0。

```
$ sudo apt-get install r-base r-base-dev \
rstudio texlive-full pandoc make
```

如果不能满足，请参考相关软件官网，下载最新版本。我的[博客](#)记录了如何安装最新版 texlive，有需要的可以去看看。[pandoc 官网](#)有其安装教程。R 语言的安装方法[官网镜像](#)也可找到，这里就不赘述了，有问题可以邮件联系。

曾经在 Lubuntu 下试着安装过。一般先安装 make，运行命令，仍旧出错，提示没有 RScript，接着安装 r-base r-base-dev。再接着，sudo make 还是出错。最后是 pandoc 版本问题了。由于 Lubuntu 版本太老，没能更新好。总结一下，

```
$ sudo apt-get install make r-base r-base-dev -y
$ sudo make
```

第一次编译，可能需要安装很多软件，请静静等待。后面，直接运行下面这条语句，实现编译。

```
$ make
```

乙.1.2 Windows 下使用

略

乙.1.3 苹果操作系统下使用

略

乙.2 编译模板

乙.2.1 第一种编译方法 —— 命令行编译

模板默认使用 GNUMake 构建，后续如无特殊说明，默认执行命令的文件夹以及当前文件夹均为本模板的根目录。对于部分用户，第一次编译可能需要下载很多文件，最好联网编译，简言之**首次编译请联网**。如果第一次编译成功，后面再次编译则不需要联网。

```
$ make
```

乙.2.2 第二种编译方法 —— RStudio 编译

在已经安装 RStudio 的前提下，也可手动编译。打开 RStudio 软件，在弹出的界面，左上角点击 File->Open Project，选中 *./deepin-bible.Rproj* 文件打开，点击右上角靠下的一栏，有个 Build 格，会出现 Build Book 等按钮。点击 Build Book 就会生成 PDF 书籍了，位于 *./_book/deepin-bible.pdf*。本质上讲，第二种编译方法是调用的第一种编译方法，只不过方便懒得写命令行的用户。

乙.2.3 字数统计

如果需要统计字数，先生成文件，然后执行如下命令，特别提醒**先编译书籍才可以统计字数**。

```
$ make wordcount
```

乙.2.4 本书编译的 R 各包信息

```
1 ## R version 3.4.4 (2018-03-15)
2 ## Platform: x86_64-pc-linux-gnu (64-bit)
3 ## Running under: Deepin 15
4 ##
5 ## Matrix products: default
6 ## BLAS: /usr/lib/x86_64-linux-gnublas/libblas.so.3.8.0
7 ## LAPACK: /usr/lib/x86_64-linux-gnulapack/liblapack.so.3.8.0
8 ##
9 ## locale:
10 ## [1] LC_CTYPE=zh_CN.UTF-8
11 ## [2] LC_NUMERIC=C
12 ## [3] LC_TIME=en_US.UTF-8
13 ## [4] LC_COLLATE=zh_CN.UTF-8
14 ## [5] LC_MONETARY=zh_CN.UTF-8
15 ## [6] LC_MESSAGES=zh_CN.UTF-8
16 ## [7] LC_PAPER=zh_CN.UTF-8
17 ## [8] LC_NAME=C
18 ## [9] LC_ADDRESS=C
19 ## [10] LC_TELEPHONE=C
20 ## [11] LC_MEASUREMENT=zh_CN.UTF-8
21 ## [12] LC_IDENTIFICATION=C
22 ##
23 ## attached base packages:
24 ## [1] stats      graphics   grDevices utils      datasets
25 ## [6] base
26 ##
27 ## other attached packages:
28 ## [1] knitr_1.20
29 ##
30 ## loaded via a namespace (and not attached):
31 ## [1] compiler_3.4.4 backports_1.1.2 magrittr_1.5
32 ## [4] bookdown_0.7.13 rprojroot_1.3-2 htmltools_0.3.6
33 ## [7] tools_3.4.4     rstudioapi_0.7  yaml_2.2.0
34 ## [10] Rcpp_0.12.18   stringi_1.2.4   rmarkdown_1.10
35 ## [13] methods_3.4.4  stringr_1.3.1   digest_0.6.17
36 ## [16] xfun_0.3       evaluate_0.11
```

乙.3 文件布局

使用 `tree` 命令所得。为了篇幅，删除了部分文件名。

代码 乙-1 模板文件布局

```
.  
├── bib  
│   ├── book.bib  
│   └── packages.bib  
├── bin  
│   ├── linux_x64  
│   └── linux_x86  
├── _bookdown.yml  
├── css  
│   └── style.css  
├── deepin-bible.Rproj  
├── images  
│   ├── about-us_img-2.jpg  
│   └── zhifubaozhifu.png  
├── imakeidx.ist  
├── index.Rmd  
├── latex  
│   ├── content.tex  
│   └── template.tex  
├── LICENSE  
├── Makefile  
├── _output.yml  
├── README.md  
├── _render.R  
└── rmd  
    ├── 000-intro.Rmd  
    └── 899-appendix-history.Rmd  
└── sjtuthesis.cfg  
└── sjtuthesis.cls
```

乙.4 主要文件介绍

乙.4.1 L^AT_EX 模板文件

格式控制文件控制着论文的表现形式，包括 `./sjtuthesis.cfg` 和 `./sjtuthesis.cls`。其中，“cls”控制论文主体格式，“cfg”为配置文件。上述文件全部来自最新（2018-

12-12) 上海交通大学学位论文 L^AT_EX 模板

乙.4.2 各章源文件

主要位于 *./rmd/* 文件夹。另外目录和模板的内容以 L^AT_EX 形式存放在 *./latex/* 文件夹下。各文件的详细说明见第丙章。

乙.4.3 配置文件

主要为 *./index.Rmd* 文件，另外两个配置文件 (*./_bookdown.yml*、*./_output.yml*) 没有特殊需求不需要更改。

对于 *./index.Rmd* 文件，书名、作者等信息直接更改自己的即可。由于本文采用的上海交大论文模板，一些配置不建议修改。如果想修改书籍字体，比如 `classoption: [doctor, openright, twoside, fontset=adobe]`，就表示中文字体修改为 adobe 系列的字体。可供选择的中文字体：fandol(Fandol 开源字体)、windows(Windows 系统下的中文字体)、mac(macOS 系统下的华文字体)、ubuntu(Ubuntu 系统下的文泉驿和文鼎字体)、adobe(Adobe 公司的中文字体)、founder(方正公司的中文字体)，默认根据操作系统自动配置。

如果打算使用本文的英文模板，可采用如下方式，在 `classoption: [doctor, openright, twoside, fontset=adobe]` 添加 “english”，比如 `classoption: [doctor, openright, twoside, fontset=adobe, english]`。

乙.4.4 图片文件夹 images

`images` 文件夹放置了需要插入文档中的图片文件(支持 PNG/JPG/PDF/EPS 格式的图片)，可以在按照章节划分子目录。模板文件中使用 `\graphicspath` 命令定义了图片存储的顶层目录，在插入图片时，顶层目录名“`images`”可省略。

乙.4.5 参考文献数据库 bib

目前参考文件数据库目录只存放一个参考文件数据库 *./bib/book.bib*，而 *./bib/packages.bib* 是书籍编译参考的包信息。关于参考文献引用，可参考附录丁中的例子。

乙.4.6 辅助文件

- *./deepin-bible.Rproj* 项目文件，方便使用 rstudio 打开，如果仅使用 `make`，可以不要。

- `./_render.R` 渲染文件，复制自[谢益辉 Bookdown 中文模板](#)，致谢。
- `./Makefile make` 的文件，综合[上海交通大学学位论文 L^AT_EX 模板](#)和[谢益辉 Bookdown 中文模板](#)。
- `./_book/` 最终生成文件所在文件夹。
- `./bookdown_files/` 过程中生成的文件所在文件夹。
- `./bin/`, `./makeidx.ist` 是索引引擎及索引模板文件。
- `./css/` 用于生成非 PDF 书籍的模板，编译本书暂时用不到。

附录丙 各章格式说明

Bookdown 编译各 `Rmd` 文件时，是按照文件名的字母排序拼接在一起的，故而我们不需要专门写一个文件来组织各章节，但是为了更好的利用这一排序规则，需要对文件名稍作调整。本模板的文件名采用如下命名方式，`xxx-name.Rmd`。其中 `xxx` 表示从 `000` 到 `999`，第一位表示第几部分，后两位如果为 `00` 表示该部分的简介，如果是其他数字则表示章节。`name` 是对应章节的名字。一章一个文件。

由于自身水平所限，部分格式实现起来费劲，就采用直接书写 `LATEX` 代码的形式实现了。

丙.1 章：前言

前言作为一章，不属于书主体内容，位于本书开头，需要一些特殊设定。命名为`./rmd/000-intro.Rmd`。样式如下，

代码丙-1 前言示例

```
1 \frontmatter
2 \pagestyle{main}
3
4 # 前言
5
6 为什么要写这本书呢？起源于去武汉参观辛亥革命博物馆。
7
8 \include{latex/content}
```

解释

1. 第 1、2 行，表示下面属于前言部分，采用罗马数字对前言编号，页面格式属于正文格式。
2. 第 4、6 行，表示前言内容。
3. 第 8 行，表示接下来为目录部分。
4. 为了减少文件个数，把文件格式以及目录也放在该文件了，但也造成了文件不专用的问题。请根据爱好，自行斟酌。

丙.2 前言后的部分章节

前言后的部分章节，属于标准的 Markdown 文件，框架示例如代码丙-2所示。

代码 丙-2 前言后文示例

```

1 # 作者简介 {#author}
2
3 这将是是一群深度操作系统爱好者的杰作！！！

```

解释

- 第 1 行，标准 Markdown 语句，表示这是一章。
- 第 3 行，该章的具体内容，可以使用 L^AT_EX 或 Markdown 格式书写。

丙.3 部分：第一部分简介

代码 丙-3 第一部分简介示例

```

1 \mainmatter
2 \pagestyle{main}
3
4 \partquote{合抱之木，生于毫末；九层之台，起于垒土；千里之行，始于足下。}{老
    \quad 子}
5
6 \partintro{
7 \quad\quad“话说天下大势，分久必合，合久必分”。
8 }
9
10 # (PART) Linux 及 Deepin 入门 {#part:intro -}

```

解释

- 第 1、2 行，表示下面属于本文正文部分，采用阿拉伯数字对正文编号，页面格式属于正文格式。
- 第 4 行，用于添加第一部分简介的名言警句。
- 第 6-8 行，表示第一部分简介的具体内容。
- 第 10 行，表示第一部分名称为 **Linux 及 Deepin 入门**，标签为 part:intro。

丙.4 各部分内部的章

框架格式类似前言后的文件，如代码丙-2所示，略。

丙.5 部分：附录及其他

代码丙-4 附录及其他部分框架示例

```
1 \partquote{这只是万里长征走完了第一步，以后的路程更长，工作更伟大，更艰苦。  
务必使同志们继续地保持谦虚、谨慎、不骄不躁的作风，务必使同志们继续地  
保持艰苦奋斗的作风。}{毛泽东}  
2 \partintro{  
3 \quad\quad\quad 附录及其他部分，首先引用了中国的一篇古文《愚公移山》，目的是告诉  
自己要坚持坚持再坚持；其次讲述了本书的制作步骤，方便后来人对模板进行  
改编和再创作；再次介绍了本书的大致框架设计，提醒需要注意的问题；再次  
列举了写作中常见的 \LaTeX 和 Bookdown 编写的示例；最后是常见问题解  
答以及本书的大事记。  
4 }  
5 # (PART) 附录及其他 {#part:others -}  
6  
7 # (APPENDIX) 附录 {#chap:appendix -}
```

解释

- 第1-5行，同框架示例丙-3的介绍。
- 第7行，表示下面属于论文附录部分，采用天干顺序对附录编号。

丙.6 附录内部各章

框架格式类似前言后的文件，如代码丙-2所示，略。

丙.7 后缀部分

相关代码为 LATEX 形式，包括索引和参考文献等，由模板自动生成，不用关心。

附录丁 兼容 LATEX 排版

本章完全照抄交大论文模板的 LATEX 排版例子，除了动了标题和多了这行话，运行良好，但是部分地方会多出括号来，请对照原文删除。

丁.1 列表环境

丁.1.1 无序列表

以下是一个无序列表的例子，列表的每个条目单独分段。

- 这是一个无序列表。
- 这是一个无序列表。
- 这是一个无序列表。

使用 `itemize*` 环境可以创建行内无序列表。

- 这是一个无序列表。• 这是一个无序列表。• 这是一个无序列表。

行内无序列表条目不单独分段，所有内容直接插入在原文的段落中。

丁.1.2 有序列表

使用环境 `enumerate` 和 `enumerate*` 创建有序列表，使用方法无序列表类似。

1. 这是一个有序列表。
2. 这是一个有序列表。
3. 这是一个有序列表。

使用 `enumerate*` 环境可以创建行内有序列表。

1. 这是一个默认有序列表。2. 这是一个默认有序列表。3. 这是一个默认有序列表。

行内有序列表条目不单独分段，所有内容直接插入在原文的段落中。

丁.1.3 描述型列表

使用环境 `description` 可创建带有主题词的列表，条目语法是 `\item[主题]` 内容。

主题一 详细内容

主题二 详细内容

主题三 详细内容 ...

丁.1.4 自定义列表样式

可以使用`label` 参数控制列表的样式，详细可以参考 WikiBooks¹。比如一个自定义样式的行内有序列表

a) 这是一个自定义样式有序列表。*b)* 这是一个自定义样式有序列表。*c)* 这是一个自定义样式有序列表。

丁.2 数学排版

丁.2.1 公式排版

这里有举一个长公式排版的例子，来自 《[Math mode](#)》：

$$\frac{1}{2}\Delta(f_{ij}f^{ij}) = 2 \left(\sum_{i < j} \chi_{ij}(\sigma_i - \sigma_j)^2 + f^{ij}\nabla_j\nabla_i(\Delta f) + \right. \\ \left. + \nabla_k f_{ij} \nabla^k f^{ij} + f^{ij} f^k [2\nabla_i R_{jk} - \nabla_k R_{ij}] \right) \quad (\text{丁}-1)$$

丁.2.2 SI 单位

使用`siunitx` 宏包可以方便地输入 SI 单位制单位，例如`\SI{5}{\mu m}` 可以得到 $5\mu\text{m}$ 。

丁.2.2.1 一个四级标题

这是全文唯一的一个四级标题。在这部分中将演示了`mathtools` 宏包中可伸长符号（箭头、等号的例子）的例子。

$$A \xleftarrow[n=0]{Long Long Long Long} B \xrightarrow[n>0]{n>0} C$$

$$f(x) \xleftrightarrow[A=B]{above} B \quad (\text{丁}-2)$$

$$\begin{array}{c} \xleftrightarrow[below]{above} B \\ \xrightleftharpoons[below]{above} B \end{array} \quad (\text{丁}-3)$$

¹https://en.wikibooks.org/wiki/LaTeX/List_Structures#Customizing_lists

又如：

$$\begin{aligned} & I(X_3; X_4) - I(X_3; X_4 | X_1) - I(X_3; X_4 | X_2) \\ &= [I(X_3; X_4) - I(X_3; X_4 | X_1)] - I(X_3; X_4 | \tilde{X}_2) \end{aligned} \quad (\text{丁}-4)$$

$$= I(X_1; X_3; X_4) - I(X_3; X_4 | \tilde{X}_2) \quad (\text{丁}-5)$$

丁.2.3 定理环境

模板中定义了丰富的定理环境 algo(算法), thm(定理), lem(引理), prop(命题), cor(推论), defn(定义), conj(猜想), exmp(例), rem(注), case(情形), bthm(断言定理), blem(断言引理), bprop(断言命题), bcpr(断言推论)。amsmath 还提供了一个 proof(证明) 的环境。这里举一个“定理”和“证明”的例子。

定理 丁.1(留数定理). 假设 U 是复平面上的一个单连通开子集, a_1, \dots, a_n 是复平面上有限个点, f 是定义在 $U \setminus \{a_1, \dots, a_n\}$ 上的全纯函数, 如果 γ 是一条把 a_1, \dots, a_n 包围起来的可求长曲线, 但不经过任何一个 a_k , 并且其起点与终点重合, 那么:

$$\oint_{\gamma} f(z) dz = 2\pi i \sum_{k=1}^n I(\gamma, a_k) \operatorname{Res}(f, a_k) \quad (\text{丁}-6)$$

如果 γ 是若尔当曲线, 那么 $I(\gamma, a_k) = 1$, 因此:

$$\oint_{\gamma} f(z) dz = 2\pi i \sum_{k=1}^n \operatorname{Res}(f, a_k) \quad (\text{丁}-7)$$

在这里, $\operatorname{Res}(f, a_k)$ 表示 f 在点 a_k 的留数, $I(\gamma, a_k)$ 表示 γ 关于点 a_k 的卷绕数。卷绕数是一个整数, 它描述了曲线 γ 绕过点 a_k 的次数。如果 γ 依逆时针方向绕着 a_k 移动, 卷绕数就是一个正数, 如果 γ 根本不绕过 a_k , 卷绕数就是零。

定理 丁.1 的证明。

证明. 首先, 由……

其次, ……

所以……

□

上面的公式例子中, 有一些细节希望大家注意。微分号 d 应该使用“直立体”也就是用 `mathrm` 包围起来。并且, 微分号和被积函数之间应该有一段小间隔, 可以插入 `\,`, 得到。斜体的 d 通常只作为一般变量。 i, j 作为虚数单位时, 也应该使用

“直立体”为了明显，还加上了粗体，例如`\mathbf{i}`。斜体*i,j*通常用作表示“序号”。其他字母在表示常量时，也推荐使用“直立体”譬如，圆周率 π （需要`upgreek`宏包），自然对数的底 e 。不过，我个人觉得斜体的 e 和 π 很潇洒，在不至于引起混淆的情况下，我也用这两个字母的斜体表示对应的常量。

丁.3 向文档中插入图像

丁.3.1 支持的图片格式

X_ET_EX 可以很方便地插入 PDF、PNG、JPG 格式的图片。

插入 PNG/JPG 的例子如图丁-1 所示。这两个水平并列放置的图共享一个“图标题”(table caption)，没有各自的小标题。



图 丁-1 中文题图

Figure 丁-1 English caption

这里还有插入 EPS 图像和 PDF 图像的例子，如图丁-2a 和 图丁-2b。这里将 EPS 和 PDF 图片作为子图插入，每个子图有自己的小标题。子图标题使用 `subcaption` 宏包添加。



(a) EPS 图像



(b) PDF 图像，注意这个图略矮些。如果标题很长的话，它会自动换行

图 丁-2 插入 eps 和 pdf 的例子（使用 `subcaptionbox` 方式）

Figure 丁-2 An EPS and PDF demo with `subcaptionbox`

更多关于 L^AT_EX 插图的例子可以参考《L^AT_EX 插图指南》。



(a) EPS 图像



(b) PDF 图像, 注意这个图略矮些。

subfigure 中同一行的子图在顶端对齐。

图 丁-3 插入 eps 和 pdf 的例子 (使用 subfigure 方式)

Figure 丁-3 An EPS and PDF demo with subfigure

丁.3.2 长标题的换行

图丁-4和图丁-5都有比较长图标题，通过对比发现，图丁-5的换行效果更好一些。其中使用了 minipage 环境来限制整个浮动体的宽度。



图 丁-4 上海交通大学是我国历史最悠久的高等学府之一，是教育部直属、教育部与上海市共建的全国重点大学。

Figure 丁-4 Where there is a will, there is a way.

丁.3.3 添加图注

当插图中组成部件由数字或字母等编号表示时，可在插图下方添加图注进行说明，如图丁-6所示。

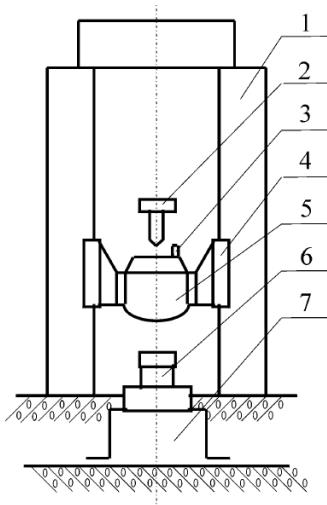
丁.3.4 绘制流程图

图丁-7是一张流程图示意。使用 tikz 环境，搭配四种预定义节点 (`startstop`、`process`、`decision` 和 `io`)，可以容易地绘制出流程图。



图 丁-5 上海交通大学是我国历史最悠久的高等学府之一，是教育部直属、教育部与上海市共建的全国重点大学。

Figure 丁-5 Where there is a will, there is a way.



1. 立柱 2. 提升释放机构 3. 标准冲击加速度计
4. 导轨 5. 重锤 6. 被校力传感器 7. 底座

图 丁-6 示例图片来源于 [1]

Figure 丁-6 Stay hungry, stay foolish.

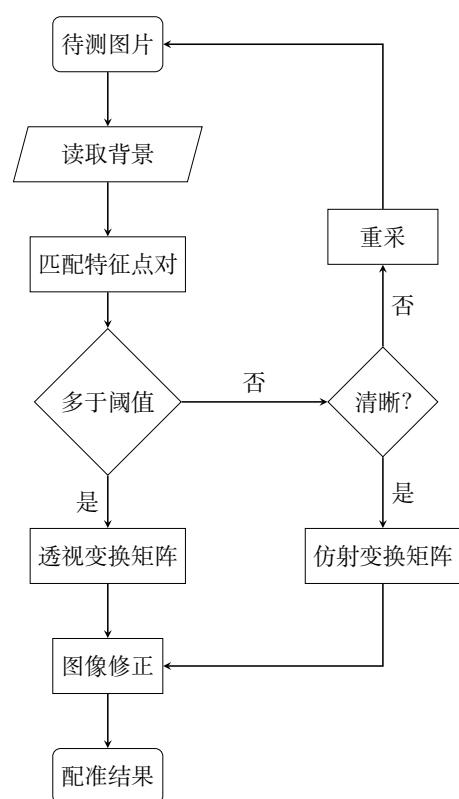


图 丁-7 绘制流程图效果

Figure 丁-7 Flow chart

丁.4 表格

这一节给出的是一些表格的例子，如表丁-1所示。

表 丁-1 一个颇为标准的三线表格¹

Table 丁-1 A Table

Item		
Animal	Description	Price (\$)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

下面一个是一个更复杂的表格，用 `threeparttable` 实现带有脚注的表格，如表丁-2。

表 丁-2 一个带有脚注的表格的例子

Table 丁-2 A Table with footnotes

total	20 ¹		40		60	
	www	k	www	k	www	k
4.22 (2.12)	120.0140 ²	333.15	0.0411	444.99	0.1387	
168.6123	10.86	255.37	0.0353	376.14	0.1058	
6.761	0.007	235.37	0.0267	348.66	0.1010	

¹ the first note.

² the second note.

丁.5 参考文献管理

L^AT_EX 具有将参考文献内容和表现形式分开管理的能力，涉及三个要素：参考文献数据库、参考文献引用格式、在正文中引用参考文献。这样的流程需要多次

¹这个例子来自 [《Publication quality tables in LATEX》](#) (booktabs 宏包的文档)。这也是一个在表格中使用脚注的例子，请留意与 `threeparttable` 实现的效果有何不同。

编译：

1. 用户将论文中需要引用的参考文献条目，录入纯文本数据库文件 (bib 文件)。
2. 调用 xelatex 对论文模板做第一次编译，扫描文中引用的参考文献，生成参考文献入口文件 (aux) 文件。
3. 调用 bibtex，以参考文献格式和入口文件为输入，生成格式化以后的参考文献条目文件 (bib)。
4. 再次调用 xelatex 编译模板，将格式化以后的参考文献条目插入正文。

参考文献数据库 (thesis.bib) 的条目，可以从 Google Scholar 搜索引擎¹、CiteseerX 搜索引擎²中查找，文献管理软件 Papers³、Mendeley⁴、JabRef⁵也能够输出条目信息。

下面是在 Google Scholar 上搜索到的一条文献信息，格式是纯文本：

代码丁-1 从 Google Scholar 找到的参考文献条目

```
@phdthesis{bai_2008 信用风险传染模型和信用衍生品的定价,
    title={信用风险传染模型和信用衍生品的定价},
    author={白云芬},
    year={2008},
    school={上海交通大学}
}
```

推荐修改后在 bib 文件中的内容为：

代码丁-2 修改后的参考文献条目

```
@phdthesis{bai2008,
    title={信用风险传染模型和信用衍生品的定价},
    author={白云芬},
    date={2008},
    address={上海},
    school={上海交通大学}
}
```

¹<https://scholar.google.com>

²<http://citeseerx.ist.psu.edu>

³<http://papersapp.com>

⁴<http://www.mendeley.com>

⁵<http://jabref.sourceforge.net>

按照教务处的要求，参考文献外观应符合国标 GBT7714 的要求¹。在模板中，表现形式的控制逻辑通过 biblatex-gb7714-2015 包实现²，基于 {BibL^AT_EX} 管理文献。在目前的多数 TeX 发行版中，可能都没有默认包含 biblatex-gb7714-2015，需要手动安装。

正文中引用参考文献时，用 \cite{key1, key2, key3...} 可以产生“上标引用的参考文献”，如^[2-4]。使用 \parencite{key1, key2, key3...} 则可以产生水平引用的参考文献，例如^[5-7]。请看下面的例子，将会穿插使用水平的和上标的参考文献：关于书的^[2, 5, 7]，关于期刊的^[3, 8]，会议论文^[4, 9, 10]，硕士学位论文^[6, 11]，博士学位论文^[12-14]，标准文件^[7]，技术报告^[15]，电子文献^[16, 17]，用户手册^[18]。

总结一些注意事项：

- 参考文献只有在正文中被引用了，才会在最后的参考文献列表中出现；
- 参考文献“数据库文件” bib 是纯文本文件，请使用 UTF-8 编码，不要使用 GBK 编码；
- 参考文献条目中默认通过 date 域输入时间。兼容使用 year 域时会产生编译 warning，可忽略。

丁.6 用 listings 插入源代码

原先 ctexbook 文档类和 listings 宏包配合使用时，代码在换页时会出现莫名其妙的错误，后来经高人指点，顺利解决了。感兴趣的话，可以看看[这里](#)。这里给使用 listings 宏包插入源代码的例子，这里是一段 C 代码。另外，listings 宏包真可谓博大精深，可以实现各种复杂、漂亮的效果，想要进一步学习的同学，可以参考 [listings 宏包手册](#)。

代码 丁-3 一段 C 源代码

```

1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4 #include <sys/wait.h>
5
6 int main() {
7     pid_t pid;
```

¹http://www.cces.net.cn/guild/sites/tmxb/Files/19798_2.pdf

²<https://www.ctan.org/pkg/biblatex-gb7714-2015>

```

8
9   switch ((pid = fork()) ) {
10
11     case -1:
12       printf("fork failed\n");
13       break;
14
15     case 0:
16       /* child calls exec */
17       execl("/bin/ls", "ls", "-l", (char*)0);
18       printf("execl failed\n");
19       break;
20
21     default:
22       /* parent uses wait to suspend execution until child
23          finishes */
24       wait((int*)0);
25       printf("is completed\n");
26       break;
27
28
29   return 0;
30 }
```

丁.7 用 algorithm 和 algorithmicx 宏包插入算法描述

algorithmicx 比 algorithm 增加了一些命令。示例如算法丁-1和算法丁-2，后者的代码来自 [xhSong 的博客](#)。algorithmicx 的详细使用方法见 [官方 README](#)。使用算法宏包时，算法出现的位置很多时候不按照 tex 文件里的书写顺序，需要强制定位时可以使用 \begin{algorithm}[H]¹

这是写在算法丁-1前面的一段话，在生成的文件里它会出现在算法丁-1前面。

算法 丁-1 求 100 以内的整数和

输出：100 以内的整数和

```

1: sum ← 0
2: for i = 0 → 100 do
3:   sum ← sum + i
4: end for
```

这是写在两个算法中间的一段话，当算法丁-1不使用 \begin{algorithm}[H]

¹<http://tex.stackexchange.com/questions/165021/fixing-the-location-of-the-appearance-in-algorithmicx-environment>

时它也会出现在算法丁-1前面。

对于很长的算法,单一的算法块\begin{algorithm}... \end{algorithm}是不能自动跨页的¹,会出现的情况有:

- 该页放不下当前的算法,留下大片空白,算法在下一页显示
- 单一页面放不下当前的算法,显示时超过页码的位置直到超出整个页面范围

解决方法有:

- (推荐) 使用\algstore{algnname} 和\algrestore{algnname} 来讲算法分为两个部分²,如算法丁-2。
- 人工拆分算法为多个小的部分。

这是写在算法丁-2后面的一段话,但是当算法丁-2不使用\begin{algorithm}[H]时它会出现在算法丁-2甚至算法丁-1前面。

对于算法的索引要注意\caption 和\label 的位置,必须是先\caption 再\label³,否则会出现\ref{algo:sum_100}生成的编号跟对应算法上显示不一致的问题。

根据 Werner 的回答⁴增加了Switch 和Case 的支持,见算法丁-3。

¹<http://tex.stackexchange.com/questions/70733/latex-algorithm-not-display-under-correct-section>

²<http://tex.stackexchange.com/questions/29816/algorithm-over-2-pages>

³<http://tex.stackexchange.com/questions/65993/algorithm-numbering>

⁴<http://tex.stackexchange.com/questions/53357/switch-cases-in-algorithmic>

算法丁-2 用归并排序求逆序数

输入: *Array* 数组, *n* 数组大小

输出: 逆序数

```
1: function MERGERSORT(Array,left,right)
2:   result  $\leftarrow$  0
3:   if left  $<$  right then
4:     middle  $\leftarrow$  (left + right) / 2
5:     result  $\leftarrow$  result + MERGERSORT(Array,left,middle)
6:     result  $\leftarrow$  result + MERGERSORT(Array,middle,right)
7:     result  $\leftarrow$  result + MERGER(Array,left,middle,right)
8:   end if
9:   return result
10: end function
11:
12: function MERGER(Array,left,middle,right)
13:   i  $\leftarrow$  left
14:   j  $\leftarrow$  middle
15:   k  $\leftarrow$  0
16:   result  $\leftarrow$  0
17:   while i  $<$  middle and j  $<$  right do
18:     if Array[i]  $<$  Array[j] then
19:       B[k  $\leftarrow$ ]  $\leftarrow$  Array[i  $\leftarrow$ ]
20:     else
21:       B[k  $\leftarrow$ ]  $\leftarrow$  Array[j  $\leftarrow$ ]
22:       result  $\leftarrow$  result + (middle - i)
23:     end if
24:   end while
```

```
25: while  $i < middle$  do
26:    $B[k + 1] \leftarrow Array[i + 1]$ 
27: end while
28: while  $j < right$  do
29:    $B[k + 1] \leftarrow Array[j + 1]$ 
30: end while
31: for  $i = 0 \rightarrow k - 1$  do
32:    $Array[left + i] \leftarrow B[i]$ 
33: end for
34: return  $result$ 
35: end function
```

算法 丁-3 Switch 示例

```
1: switch ( $s$ )
2:   case  $a$ :
3:     assert(0)
4:   case  $b$ :
5:     assert(1)
6:   default :
7:     assert(2)
8: end switch
```

附录戊 RMarkdown/Bookdown 排版示例

本文无意全文翻译 RMarkdown/Bookdown 的使用方法，只是简单的复制，详细内容请参考页面：

- <https://rmarkdown.rstudio.com/lesson-1.html>
- <https://bookdown.org/yihui/bookdown/>
- <https://bookdown.org/yihui/bookdown/components.html>
- <http://pandoc.org/>

以下为全文复制 <https://bookdown.org/yihui/bookdown/components.html>，有改动。

This chapter demonstrates the syntax of common components of a book written in **bookdown**, including code chunks, figures, tables, citations, math theorems, and equations. The approach is based on Pandoc, so we start with the syntax of Pandoc’s flavor of Markdown.

戊.1 Markdown syntax

In this section, we give a very brief introduction to Pandoc’s Markdown. Readers who are familiar with Markdown can skip this section. The comprehensive syntax of Pandoc’s Markdown can be found on the Pandoc website <http://pandoc.org>.

戊.1.1 Inline formatting

You can make text *italic* by surrounding it with underscores or asterisks, e.g., `_text_` or `*text*`. For **bold** text, use two underscores (`__text__`) or asterisks (`**text**`). Text surrounded by `~` will be converted to a subscript (e.g., `H~2~S0~4~` renders H_2SO_4), and similarly, two carets (`^`) produce a superscript (e.g., `Fe^2+^` renders Fe^{2+}). To mark text as `inline code`, use a pair of backticks, e.g., ``code``.¹ Small caps can be produced by the HTML tag `span`, e.g., `Small Caps` renders **Small Caps**. Links are created using `[text](link)`, e.g., `[RStudio](https://www.rstudio.com)`, and the syntax for images is similar: just add an exclamation mark,

¹To include literal backticks, use more backticks outside, e.g., you can use two backticks to preserve one backtick inside: ``` `code` ```.

e.g., `![alt text or image title](path/to/image)`. Footnotes are put inside the square brackets after a caret `^[]`, e.g., `^[This is a footnote.]`. We will talk about citations in Section 戊.8.

戊.1.2 Block-level elements

Section headers can be written after a number of pound signs, e.g.,

```
1 # First-level header
2
3 ## Second-level header
4
5 ### Third-level header
```

If you do not want a certain heading to be numbered, you can add `{-}` after the heading, e.g.,

```
1 # Preface {-}
```

Unordered list items start with `*`, `-`, or `+`, and you can nest one list within another list by indenting the sub-list by four spaces, e.g.,

```
1 - one item
2 - one item
3 - one item
4   - one item
5   - one item
```

The output is:

- one item
- one item
- one item
 - one item
 - one item

Ordered list items start with numbers (the rule for nested lists is the same as above), e.g.,

```
1 1. the first item  
2 2. the second item  
3 3. the third item
```

The output does not look too much different with the Markdown source:

1. the first item
2. the second item
3. the third item

Blockquotes are written after >, e.g.,

```
1 > "I thoroughly disapprove of duels. If a man should challenge  
2     me,  
3     I would take him kindly and forgivingly by the hand and lead  
4         him  
5     to a quiet place and kill him."  
>  
> --- Mark Twain
```

The actual output (we customized the style for blockquotes in this book):

“I thoroughly disapprove of duels. If a man should challenge me, I would take him kindly and forgivingly by the hand and lead him to a quiet place and kill him.”

— Mark Twain

Plain code blocks can be written after three or more backticks, and you can also indent the blocks by four spaces, e.g.,

```
1 ````  
2 This text is displayed verbatim / preformatted  
3 ````  
4  
5 Or indent by four spaces:  
6  
7     This text is displayed verbatim / preformatted
```

戊.1.3 Math expressions

Inline LaTeX equations can be written in a pair of dollar signs using the LaTeX syntax, e.g., `$f(k) = {n \choose k} p^k (1-p)^{n-k}$` (actual output: $f(k) = \binom{n}{k} p^k (1-p)^{n-k}$); math expressions of the display style can be written in a pair of double dollar signs, e.g., `$$f(k) = {n \choose k} p^k (1-p)^{n-k}$$`, and the output looks like this:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

You can also use math environments inside `$ $` or `$$ $$`, e.g.,

```

1 $$\begin{array}{ccc}
2 x_{11} & x_{12} & x_{13} \\
3 x_{21} & x_{22} & x_{23} \\
4 \end{array}$$

```

$$\begin{matrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{matrix}$$

```

1 $$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}$$
2
3
4

```

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}$$

```

1 $$\Theta = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$
2
3

```

$$\Theta = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

```

1 $$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$
2
3 \end{vmatrix}=ad-bc$$

```

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

戊.2 Markdown extensions by bookdown

Although Pandoc's Markdown is much richer than the original Markdown syntax, it still lacks a number of things that we may need for academic writing. For example, it supports math equations, but you cannot number and reference equations in multi-page HTML or EPUB output. We have provided a few Markdown extensions in **bookdown** to fill the gaps.

戊.2.1 Number and reference equations

To number and refer to equations, put them in the equation environments and assign labels to them using the syntax (\#\#eq:label), e.g.,

```

1 \begin{equation}
2   f\left(k\right) = \binom{n}{k} p^k (1-p)^{n-k}
3   \label{eq:binom}
4 \end{equation}

```

It renders the equation below:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (\text{戊}-1)$$

You may refer to it using \eqref {eq:binom}, e.g., see Equation (戊-1).

We demonstrate a few more math equation environments below. Here is an unnumbered equation using the `equation*` environment:

```

1 \begin{equation*}
2 \frac{d}{dx} \left( \int_a^x f(u) du \right) = f(x)

```



Equation labels must start with the prefix `eq:` in **bookdown**. All labels in **bookdown** must only contain alphanumeric characters, `:`, `-`, and/or `/`. Equation references work best for LaTeX/PDF output, and they are not well supported in Word output or e-books. For HTML output, **bookdown** can only number the equations with labels. Please make sure equations without labels are not numbered by either using the `equation*` environment or adding `\nonumber` or `\notag` to your equations. The same rules apply to other math environments, such as `eqnarray`, `gather`, `align`, and so on (e.g., you can use the `align*` environment).

3 | \end{equation*}

$$\frac{d}{dx} \left(\int_a^x f(u) du \right) = f(x)$$

Below is an `align` environment (戊-2):

```
1 \begin{align}
2 g(X_n) &= g(\theta) + g'(\tilde{\theta})(X_n - \theta) \notag \\
3 &\quad \\
4 \sqrt{n}[g(X_n) - g(\theta)] &= g'(\tilde{\theta})\sqrt{n}(X_n - \theta) \label{eq:align} \\
5 \end{align}
```

$$\begin{aligned} g(X_n) &= g(\theta) + g'(\tilde{\theta})(X_n - \theta) \\ \sqrt{n}[g(X_n) - g(\theta)] &= g'(\tilde{\theta})\sqrt{n}(X_n - \theta) \end{aligned} \quad (\text{戊-2})$$

You can use the `split` environment inside `equation` so that all lines share the same number (戊-3). By default, each line in the `align` environment will be assigned an equation number. We suppressed the number of the first line in the previous example using `\notag`. In this example, the whole `split` environment was assigned a single number.

```

1 \begin{equation}
2 \begin{split}
3 \mathrm{Var}(\hat{\beta}) &= \mathrm{Var}((X'X)^{-1}X'y) \\
4 &= (X'X)^{-1}X'\mathrm{Var}(y)((X'X)^{-1}X')' \\
5 &= (X'X)^{-1}X'\mathrm{Var}(y)X(X'X)^{-1} \\
6 &= (X'X)^{-1}X'\sigma^2IX(X'X)^{-1} \\
7 &= (X'X)^{-1}\sigma^2 \\
8 \end{split}
9 \label{eq:var-beta}
10 \end{equation}

```

$$\begin{aligned}
\mathrm{Var}(\hat{\beta}) &= \mathrm{Var}((X'X)^{-1}X'y) \\
&= (X'X)^{-1}X'\mathrm{Var}(y)((X'X)^{-1}X')' \\
&= (X'X)^{-1}X'\mathrm{Var}(y)X(X'X)^{-1} \\
&= (X'X)^{-1}X'\sigma^2IX(X'X)^{-1} \\
&= (X'X)^{-1}\sigma^2
\end{aligned} \tag{戊-3}$$

戊.2.2 Theorems and proofs

Theorems and proofs are commonly used in articles and books in mathematics. However, please do not be misled by the names: a “theorem” is just a numbered/labeled environment, and it does not have to be a mathematical theorem (e.g., it can be an example irrelevant to mathematics). Similarly, a “proof” is an unnumbered environment. In this section, we always use the *general* meanings of a “theorem” and “proof” unless explicitly stated.

In **bookdown**, the types of theorem environments supported are in Table 戊-1. To write a theorem, you can use the syntax below:

```

1 ```\{theorem}
2 Here is my theorem.
3 ```

```

To write other theorem environments, replace ````{theorem}` with other environment names in Table 戊-1, e.g., ````{lemma}`.

A theorem can have a `name` option so its name will be printed, e.g.,

表 戊-1 Theorem environments in **bookdown**.

Environment	Printed Name	Label Prefix
theorem	Theorem	thm
lemma	Lemma	lem
corollary	Corollary	cor
proposition	Proposition	prp
conjecture	Conjecture	cnj
definition	Definition	def
example	Example	exm
exercise	Exercise	exr

```

1 ````{theorem, name="Pythagorean theorem"}
2 For a right triangle, if $c$ denotes the length of the
   hypotenuse
3 and $a$ and $b$ denote the lengths of the other two sides, we
   have
4 $$a^2 + b^2 = c^2$$
5 ````
```

If you want to refer to a theorem, you should label it. The label can be written after

```
```{theorem, e.g.,
```

```

1 ````{theorem, label="foo"}
2 A labeled theorem here.
3 ````
```

The `label` option can be implicit, e.g., the following theorem has the label `bar`:

```

1 ````{theorem, bar}
2 A labeled theorem here.
3 ````
```

After you label a theorem, you can refer to it using the syntax `\ref {prefix:label}`.

See the column `Label Prefix` in Table 戊-1 for the value of `prefix` for each environment. For example, we have a labeled and named theorem below, and `\ref {thm:pyth}` gives us its theorem number 戊.1:

```

1 ````{theorem, pyth, name="Pythagorean theorem"}
2 For a right triangle, if c denotes the length of the
 hypotenuse
3 and a and b denote the lengths of the other two sides, we
 have
4
5 $$a^2 + b^2 = c^2$$
6 ````
```

**定理 戊.1** (Pythagorean theorem). *For a right triangle, if  $c$  denotes the length of the hypotenuse and  $a$  and  $b$  denote the lengths of the other two sides, we have*

$$a^2 + b^2 = c^2$$

The proof environments currently supported are `proof`, `remark`, and `solution`. The syntax is similar to theorem environments, and proof environments can also be named. The only difference is that since they are unnumbered, you cannot reference them.

We have tried to make all these theorem and proof environments work out of the box, no matter if your output is PDF, HTML, or EPUB. If you are a LaTeX or HTML expert, you may want to customize the style of these environments anyway (see Chapter ??). Customization in HTML is easy with CSS, and each environment is enclosed in `<div></div>` with the CSS class being the environment name, e.g., `<div class="lemma"></div>`. For LaTeX output, we have predefined the style to be `definition` for environments `definition`, `example`, and `exercise`, and `remark` for environments `proof` and `remark`. All other environments use the `plain` style. The style definition is done through the `\theoremstyle{}` command of the **amsthm** package.

Theorems are numbered by chapters by default. If there are no chapters in your document, they are numbered by sections instead. If the whole document is unnumbered (the output format option `number_sections = FALSE`), all theorems are numbered sequentially from 1, 2, ..., N. LaTeX supports numbering one theorem environment after another, e.g., let theorems and lemmas share the same counter. This is not supported

for HTML/EPUB output in **bookdown**. You can change the numbering scheme in the LaTeX preamble by defining your own theorem environments, e.g.,

```
1 \newtheorem{theorem}{Theorem}
2 \newtheorem{lemma}[theorem]{Lemma}
```

When **bookdown** detects `\newtheorem{theorem}` in your LaTeX preamble, it will not write out its default theorem definitions, which means you have to define all theorem environments by yourself. For the sake of simplicity and consistency, we do not recommend that you do this. It can be confusing when your Theorem 18 in PDF becomes Theorem 2.4 in HTML.

Theorem and proof environments will be hidden if the chunk option `echo` is set to `FALSE`. To make sure they are always shown, you may add the chunk option `echo=TRUE`, e.g.,

```
1 ```{theorem, echo=TRUE}
2 Here is my theorem.
3 ````
```

Below we show more examples<sup>1</sup> of the theorem and proof environments, so you can see the default styles in **bookdown**.

**定义 戊.1.** The characteristic function of a random variable  $X$  is defined by

$$\varphi_X(t) = \mathbb{E} [e^{itX}], t \in \mathcal{R}$$

**例 戊.1.** We derive the characteristic function of  $X \sim U(0, 1)$  with the probability density function  $f(x) = \mathbf{1}_{x \in [0,1]}$ .

---

<sup>1</sup>Some examples are adapted from the Wikipedia page [https://en.wikipedia.org/wiki/Characteristic\\_function\\_\(probability\\_theory\)](https://en.wikipedia.org/wiki/Characteristic_function_(probability_theory))

$$\begin{aligned}
\varphi_X(t) &= \mathbb{E}[e^{itX}] \\
&= \int e^{itx} f(x) dx \\
&= \int_0^1 e^{itx} dx \\
&= \int_0^1 (\cos(tx) + i \sin(tx)) dx \\
&= \left( \frac{\sin(tx)}{t} - i \frac{\cos(tx)}{t} \right) \Big|_0^1 \\
&= \frac{\sin(t)}{t} - i \left( \frac{\cos(t) - 1}{t} \right) \\
&= \frac{i \sin(t)}{it} + \frac{\cos(t) - 1}{it} \\
&= \frac{e^{it} - 1}{it}
\end{aligned}$$

Note that we used the fact  $e^{ix} = \cos(x) + i \sin(x)$  twice.

**引理 戊.2.** *For any two random variables  $X_1, X_2$ , they both have the same probability distribution if and only if*

$$\varphi_{X_1}(t) = \varphi_{X_2}(t)$$

**定理 戊.3.** *If  $X_1, \dots, X_n$  are independent random variables, and  $a_1, \dots, a_n$  are some constants, then the characteristic function of the linear combination  $S_n = \sum_{i=1}^n a_i X_i$  is*

$$\varphi_{S_n}(t) = \prod_{i=1}^n \varphi_{X_i}(a_i t) = \varphi_{X_1}(a_1 t) \cdots \varphi_{X_n}(a_n t)$$

**命题 戊.4.** *The distribution of the sum of independent Poisson random variables  $X_i \sim \text{Pois}(\lambda_i)$ ,  $i = 1, 2, \dots, n$  is  $\text{Pois}(\sum_{i=1}^n \lambda_i)$ .*

**证明.** The characteristic function of  $X \sim \text{Pois}(\lambda)$  is  $\varphi_X(t) = e^{\lambda(e^{it}-1)}$ . Let  $P_n = \sum_{i=1}^n X_i$ . We know from Theorem 戊.3 that

$$\begin{aligned}\varphi_{P_n}(t) &= \prod_{i=1}^n \varphi_{X_i}(t) \\ &= \prod_{i=1}^n e^{\lambda_i(e^{it}-1)} \\ &= e^{\sum_{i=1}^n \lambda_i(e^{it}-1)}\end{aligned}$$

This is the characteristic function of a Poisson random variable with the parameter  $\lambda = \sum_{i=1}^n \lambda_i$ . From Lemma 戊.2, we know the distribution of  $P_n$  is  $\text{Pois}(\sum_{i=1}^n \lambda_i)$ .  $\square$

**注** 1. In some cases, it is very convenient and easy to figure out the distribution of the sum of independent random variables using characteristic functions.

**推论 戊.5.** *The characteristic function of the sum of two independent random variables  $X_1$  and  $X_2$  is the product of characteristic functions of  $X_1$  and  $X_2$ , i.e.,*

$$\varphi_{X_1+X_2}(t) = \varphi_{X_1}(t)\varphi_{X_2}(t)$$

**练习 戊.1** (Characteristic Function of the Sample Mean). Let  $\bar{X} = \sum_{i=1}^n \frac{1}{n} X_i$  be the sample mean of  $n$  independent and identically distributed random variables, each with characteristic function  $\varphi_X$ . Compute the characteristic function of  $\bar{X}$ .

**解答.** Applying Theorem 戊.3, we have

$$\varphi_{\bar{X}}(t) = \prod_{i=1}^n \varphi_{X_i}\left(\frac{t}{n}\right) = \left[\varphi_X\left(\frac{t}{n}\right)\right]^n.$$

### 戊.2.3 Special headers

There are a few special types of first-level headers that will be processed differently in **bookdown**. The first type is an unnumbered header that starts with the token (PART). This kind of headers are translated to part titles. If you are familiar with LaTeX, this basically means `\part{}`. When your book has a large number of chapters, you may want to organize them into parts, e.g.,

```
1 # (PART) Part I {-}
2
```

```
3 # Chapter One
4
5 # Chapter Two
6
7 # (PART) Part II {-}
8
9 # Chapter Three
```

A part title should be written right before the first chapter title in this part. You can use `(PART\*)` (the backslash before `*` is required) instead of `(PART)` if a part title should not be numbered.

The second type is an unnumbered header that starts with `(APPENDIX)`, indicating that all chapters after this header are appendices, e.g.,

```
1 # Chapter One
2
3 # Chapter Two
4
5 # (APPENDIX) Appendix {-}
6
7 # Appendix A
8
9 # Appendix B
```

The numbering style of appendices will be automatically changed in LaTeX/PDF and HTML output (usually in the form A, A.1, A.2, B, B.1, ...). This feature is not available to e-books or Word output.

#### 戊.2.4 Text references

You can assign some text to a label and reference the text using the label elsewhere in your document. This can be particularly useful for long figure/table captions (Section 戊.4 and 戊.5), in which case you normally will have to write the whole character string in the chunk header (e.g., `fig.cap = "A long long figure caption."`) or your R code (e.g., `kable(caption = "A long long table caption.")`). It is also useful when these captions contain special HTML or LaTeX characters, e.g., if the figure caption contains an underscore, it works in the HTML output but may not work in LaTeX output because the underscore must be escaped in LaTeX.

The syntax for a text reference is `(ref:label) text`, where `label` is a unique label<sup>1</sup> throughout the document for `text`. It must be in a separate paragraph with empty lines above and below it. The paragraph must not be wrapped into multiple lines, and should not end with a white space. For example,

```
1 $(ref:foo)$ Define a text reference **here**.
```

Then you can use `(ref:foo)` in your figure/table captions. The text can contain anything that Markdown supports, as long as it is one single paragraph. Here is a complete example:

```
1 A normal paragraph.
2
3 $(ref:foo)$ A scatterplot of the data `cars` using **base** R
 graphics.
4
5 ````{r foo, fig.cap='(ref:foo)'}
6 plot(cars) # a scatterplot
7 ````
```

Text references can be used anywhere in the document (not limited to figure captions). It can also be useful if you want to reuse a fragment of text in multiple places.

## 戊.3 R code

There are two types of R code in R Markdown/**knitr** documents: R code chunks, and inline R code. The syntax for the latter is ``r R_CODE``, and it can be embedded inline with other document elements. R code chunks look like plain code blocks, but have `{r}` after the three backticks and (optionally) chunk options inside `{}`, e.g.,

```
1 ````{r chunk-label, echo = FALSE, fig.cap = 'A figure caption.'}
2 1 + 1
3 rnorm(10) # 10 random numbers
4 plot(dist ~ speed, cars) # a scatterplot
5 ````
```

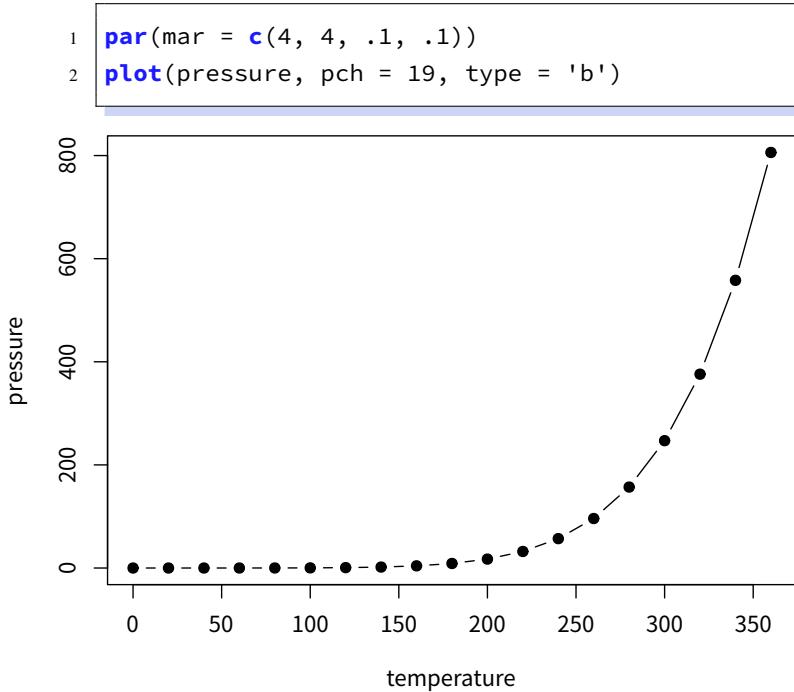
---

<sup>1</sup>You may consider using the code chunk labels.

To learn more about **knitr** chunk options, see Xie [19] or the web page <http://yihui.name/knitr/options>. For books, additional R code can be executed before/after each chapter; see `before_chapter_script` and `after_chapter_script` in Section ??.

## 戊.4 Figures

By default, figures have no captions in the output generated by **knitr**, which means they will be placed wherever they were generated in the R code. Below is such an example.



The disadvantage of typesetting figures in this way is that when there is not enough space on the current page to place a figure, it may either reach the bottom of the page (hence exceeds the page margin), or be pushed to the next page, leaving a large white margin at the bottom of the current page. That is basically why there are “floating environments” in LaTeX: elements that cannot be split over multiple pages (like figures) are put in floating environments, so they can float to a page that has enough space to hold them. There is also a disadvantage of floating things forward or backward, though. That is, readers may have to jump to a different page to find the figure mentioned on the current page. This is simply a natural consequence of having to typeset things on multiple pages of fixed sizes. This issue does not exist in HTML, however, since everything can

be placed continuously on one single page (presumably with infinite height), and there is no need to split anything across multiple pages of the same page size.

If we assign a figure caption to a code chunk via the chunk option `fig.cap`, R plots will be put into figure environments, which will be automatically labeled and numbered, and can also be cross-referenced. The label of a figure environment is generated from the label of the code chunk, e.g., if the chunk label is `foo`, the figure label will be `fig:foo` (the prefix `fig:` is added before `foo`). To reference a figure, use the syntax `\ref {label}`,<sup>1</sup> where `label` is the figure label, e.g., `fig:foo`.

To take advantage of Markdown formatting *within* the figure caption, you will need to use text references (see Section 戊.2.4). For example, a figure caption that contains `_italic text_` will not work when the output format is LaTeX/PDF, since the underscore is a special character in LaTeX, but if you use text references, `_italic text_` will be translated to LaTeX code when the output is LaTeX.



If you want to cross-reference figures or tables generated from a code chunk, please make sure the chunk label only contains *alphanumeric* characters (a-z, A-Z, 0-9), slashes (/), or dashes (-).

The chunk option `fig.asp` can be used to set the aspect ratio of plots, i.e., the ratio of figure height/width. If the figure width is 6 inches (`fig.width = 6`) and `fig.asp = 0.7`, the figure height will be automatically calculated from `fig.width * fig.asp = 6 * 0.7 = 4.2`. Figure 戊-1 is an example using the chunk options `fig.asp = 0.7`, `fig.width = 6`, and `fig.align = 'center'`, generated from the code below:

```
1 par(mar = c(4, 4, .1, .1))
2 plot(pressure, pch = 19, type = 'b')
```

The actual size of a plot is determined by the chunk options `fig.width` and `fig.height` (the size of the plot generated from a graphical device), and we can specify the output size of plots via the chunk options `out.width` and `out.height`. The possible value of these two options depends on the output format of the document. For example, `out.width = '30%'` is a valid value for HTML output, but not for LaTeX/PDF output.

---

<sup>1</sup>Do not forget the leading backslash! And also note the parentheses () after `ref`; they are not curly braces {}.

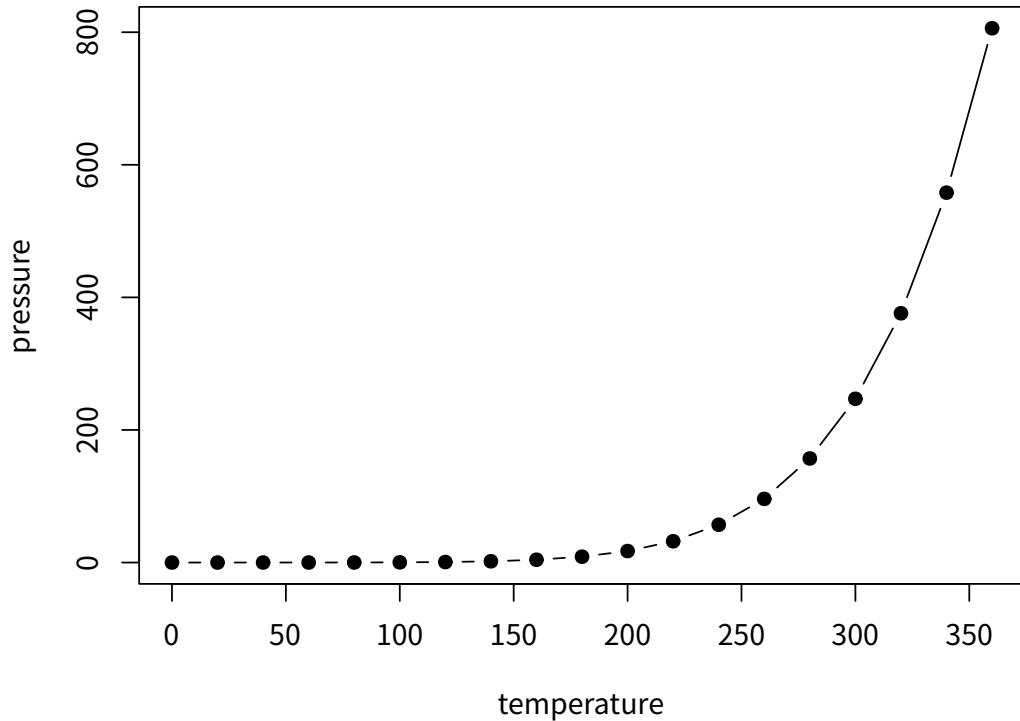


图 戊-1 A figure example with the specified aspect ratio, width, and alignment.

However, **knitr** will automatically convert a percentage value for `out.width` of the form `x%` to `(x / 100) \linewidth`, e.g., `out.width = '70'` will be treated as `.7\linewidth` when the output format is LaTeX. This makes it possible to specify a relative width of a plot in a consistent manner. Figure 戊-2 is an example of `out.width = 70%`.

```
1 par(mar = c(4, 4, .1, .1))
2 plot(cars, pch = 19)
```

If you want to put multiple plots in one figure environment, you must use the chunk option `fig.show = 'hold'` to hold multiple plots from a code chunk and include them in one environment. You can also place plots side by side if the sum of the width of all plots is smaller than or equal to the current line width. For example, if two plots have the same width 50%, they will be placed side by side. Similarly, you can specify `out.width = '33%'` to arrange three plots on one line. Figure 戊-3 is an example of two plots, each with a width of 50%.

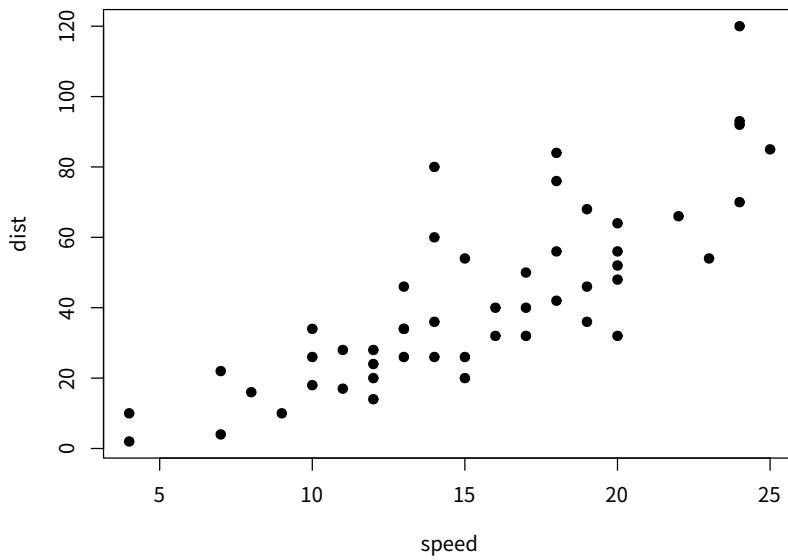


图 戊-2 A figure example with a relative width 70%.

```

1 par(mar = c(4, 4, .1, .1))
2 plot(pressure, pch = 19, type = 'b')
3 plot(cars, pch = 19)

```

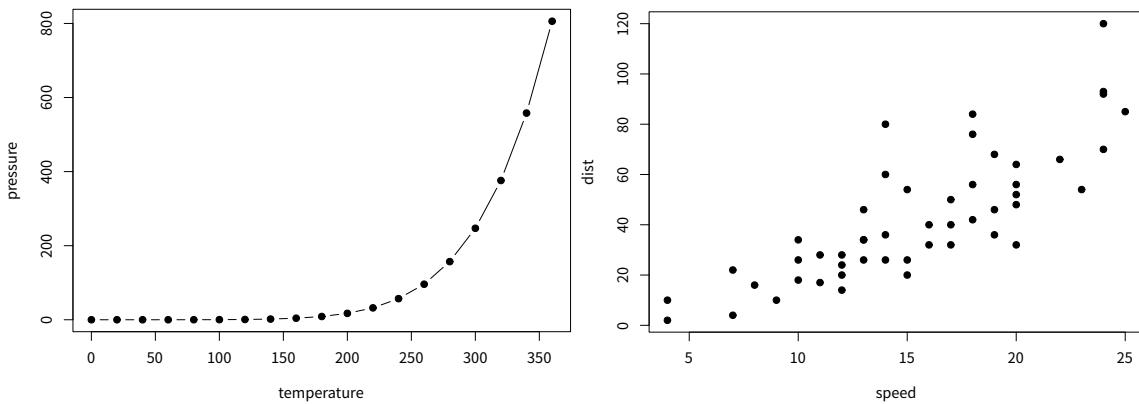


图 戊-3 Two plots placed side by side.

Sometimes you may have certain images that are not generated from R code, and you can include them in R Markdown via the function `knitr:::include_graphics()`. Figure 戊-4 is an example of three **knitr** logos included in a figure environment. You may pass one or multiple image paths to the `include_graphics()` function, and all chunk options that apply to normal R plots also apply to these images, e.g., you can use `out.width = '33%` to set the widths of these images in the output document.

```
1 knitr:::include_graphics(rep('images/bookdown/knit-logo.png', 3))
```

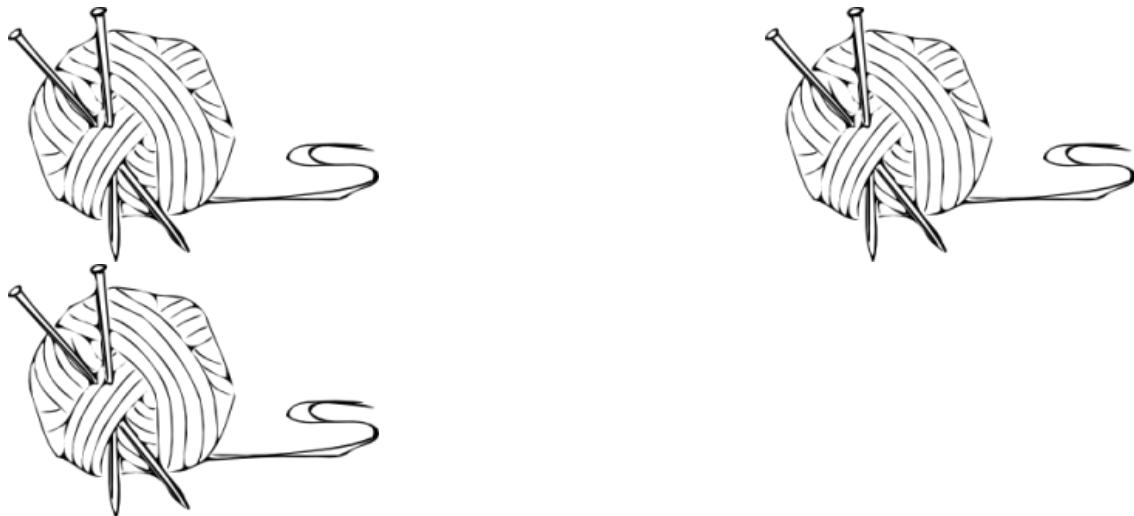


图 戊-4 Three knitr logos included in the document from an external PNG image file.

There are a few advantages of using `include_graphics()`:

1. You do not need to worry about the document output format, e.g., when the output format is LaTeX, you may have to use the LaTeX command `\includegraphics{}` to include an image, and when the output format is Markdown, you have to use ``. The function `include_graphics()` in **knitr** takes care of these details automatically.
2. The syntax for controlling the image attributes is the same as when images are generated from R code, e.g., chunk options `fig.cap`, `out.width`, and `fig.show` still have the same meanings.
3. `include_graphics()` can be smart enough to use PDF graphics automatically when the output format is LaTeX and the PDF graphics files exist, e.g., an image path `foo/bar.png` can be automatically replaced with `foo/bar.pdf` if the latter exists. PDF images often have better qualities than raster images in LaTeX/PDF output. To make use of this feature, set the argument `auto_pdf = TRUE`, or set the global option `options(knitr.graphics.auto_pdf = TRUE)` to enable this feature globally in an R session.
4. You can easily scale these images proportionally using the same ratio. This can be done via the `dpi` argument (dots per inch), which takes the value from

the chunk option `dpi` by default. If it is a numeric value and the chunk option `out.width` is not set, the output width of an image will be its actual width (in pixels) divided by `dpi`, and the unit will be inches. For example, for an image with the size 672 x 480, its output width will be 7 inches (`7in`) when `dpi = 96`. This feature requires the package **png** and/or **jpeg** to be installed. You can always override the automatic calculation of width in inches by providing a non-NULL value to the chunk option `out.width`, or use `include_graphics(dpi = NA)`.

## 戊.5 Tables

For now, the most convenient way to generate a table is the function `knitr::kable()`, because there are some internal tricks in **knitr** to make it work with **bookdown** and users do not have to know anything about these implementation details. We will explain how to use other packages and functions later in this section.

Like figures, tables with captions will also be numbered and can be referenced. The `kable()` function will automatically generate a label for a table environment, which is the prefix `tab:` plus the chunk label. For example, the table label for a code chunk with the label `foo` will be `tab:foo`, and we can still use the syntax `\ref {label}` to reference the table. Table 戊-2 is a simple example.

```

1 knitr::kable(
2 head(mtcars[, 1:8], 10), booktabs = TRUE,
3 caption = 'A table of the first 10 rows of the mtcars data.'
4)

```

If you want to put multiple tables in a single table environment, wrap the data objects (usually data frames in R) into a list. See Table 戊-3 for an example. Please note that this feature is only available in HTML and PDF output.

```

1 knitr::kable(
2 list(
3 head(iris[, 1:2], 3),
4 head(mtcars[, 1:3], 5)
5),
6 caption = 'A Tale of Two Tables.', booktabs = TRUE
7)

```

表 戊-2 A table of the first 10 rows of the mtcars data.

	mpg	cyl	disp	hp	drat	wt	qsec	vs
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1

表 戊-3 A Tale of Two Tables.

Sepal.Length	Sepal.Width	mpg	cyl	disp
5.1	3.5	Mazda RX4	21.0	6 160
4.9	3.0	Mazda RX4 Wag	21.0	6 160
4.7	3.2	Datsun 710	22.8	4 108
		Hornet 4 Drive	21.4	6 258
		Hornet Sportabout	18.7	8 360

When you do not want a table to float in PDF, you may use the LaTeX package **longtable**, which can break a table across multiple pages. To use **longtable**, pass `longtable = TRUE` to `kable()`, and make sure to include `\usepackage{longtable}` in the LaTeX preamble (see Section ?? for how to customize the LaTeX preamble). Of course, this is irrelevant to HTML output, since tables in HTML do not need to float.

```

1 knitr::kable(
2 iris[1:55,], longtable = TRUE, booktabs = TRUE,
3 caption = 'A table generated by the longtable package.'

```

4 )

表 戊-4 A table generated by the longtable package.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa
5.4	3.4	1.7	0.2	setosa
5.1	3.7	1.5	0.4	setosa
4.6	3.6	1.0	0.2	setosa
5.1	3.3	1.7	0.5	setosa
4.8	3.4	1.9	0.2	setosa
5.0	3.0	1.6	0.2	setosa
5.0	3.4	1.6	0.4	setosa

5.2	3.5	1.5	0.2	setosa
5.2	3.4	1.4	0.2	setosa
4.7	3.2	1.6	0.2	setosa
4.8	3.1	1.6	0.2	setosa
5.4	3.4	1.5	0.4	setosa
5.2	4.1	1.5	0.1	setosa
5.5	4.2	1.4	0.2	setosa
4.9	3.1	1.5	0.2	setosa
5.0	3.2	1.2	0.2	setosa
5.5	3.5	1.3	0.2	setosa
4.9	3.6	1.4	0.1	setosa
4.4	3.0	1.3	0.2	setosa
5.1	3.4	1.5	0.2	setosa
5.0	3.5	1.3	0.3	setosa
4.5	2.3	1.3	0.3	setosa
4.4	3.2	1.3	0.2	setosa
5.0	3.5	1.6	0.6	setosa
5.1	3.8	1.9	0.4	setosa
4.8	3.0	1.4	0.3	setosa
5.1	3.8	1.6	0.2	setosa
4.6	3.2	1.4	0.2	setosa
5.3	3.7	1.5	0.2	setosa
5.0	3.3	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4.0	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor

Pandoc supports several types of [Markdown tables](#), such as simple tables, multiline tables, grid tables, and pipe tables. What `knitr::kable()` generates is a simple table like this:

```

1 Table: A simple table in Markdown.
2
3 Sepal.Length Sepal.Width Petal.Length Petal.Width
4 -----
5 5.1 3.5 1.4 0.2
6 4.9 3.0 1.4 0.2
7 4.7 3.2 1.3 0.2
8 4.6 3.1 1.5 0.2
9 5.0 3.6 1.4 0.2
10 5.4 3.9 1.7 0.4

```

You can use any types of Markdown tables in your document. To be able to cross-reference a Markdown table, it must have a labeled caption of the form `Table: (\#\#label) Caption here`, where `label` must have the prefix `tab:`, e.g., `tab:simple-table`.

If you decide to use other R packages to generate tables, you have to make sure the label for the table environment appears in the beginning of the table caption in the form `(\#\#label)` (again, `label` must have the prefix `tab:`). You have to be very careful about the *portability* of the table generating function: it should work for both HTML and LaTeX output automatically, so it must consider the output format internally (check `knitr::opts_knit$get('rmarkdown.pandoc.to')`). When writing out an HTML table, the caption must be written in the `<caption></caption>` tag. For simple tables, `kable()` should suffice. If you have to create complicated tables (e.g., with certain cells spanning across multiple columns/rows), you will have to take the aforementioned issues into consideration.

## 戊.6 Cross-references

We have explained how cross-references work for equations (Section 戊.2.1), theorems (Section 戊.2.2), figures (Section 戊.4), and tables (Section 戊.5). In fact, you can also reference sections using the same syntax `\ref {label}`, where `label` is the section ID. By default, Pandoc will generate an ID for all section headers, e.g., a section `# Hello World` will have an ID `hello-world`. We recommend you to manually assign an ID to a section header to make sure you do not forget to update the reference label after you change the section header. To assign an ID to a section header, simply add `{##id}` to the end of the section header. Further attributes of section headers can be set using standard

### Pandoc syntax.

When a referenced label cannot be found, you will see two question marks like ??, as well as a warning message in the R console when rendering the book.

You can also create text-based links using explicit or automatic section IDs or even the actual section header text.

- If you are happy with the section header as the link text, use it inside a single set of square brackets:
  - [Section header text]: example “[A single document]” via [A single document]
- There are two ways to specify custom link text:
  - [link text][Section header text], e.g., “[non-English books][Internationalization]” via [non-English books][Internationalization]
  - [link text](##ID), e.g., “**Table stuff**” via [Table stuff](##tables)

The Pandoc documentation provides more details on [automatic section IDs](#) and [implicit header references](#).

Cross-references still work even when we refer to an item that is not on the current page of the PDF or HTML output. For example, see Equation (戊-1) and Figure 戊-4.

## 戊.7 Custom blocks

You can generate custom blocks using the `block` engine in `knitr`, i.e., the chunk option `engine = 'block'`, or the more compact syntax ````{block}`. This engine should be used in conjunction with the chunk option `type`, which takes a character string. When the `block` engine is used, it generates a `<div>` to wrap the chunk content if the output format is HTML, and a LaTeX environment if the output is LaTeX. The `type` option specifies the class of the `<div>` and the name of the LaTeX environment. For example, the HTML output of this chunk

```

1 ```{block, type='FOO'}
2 Some text for this block.
3 ```


```

will be this:

```

1 <div class="FOO">
2 Some text for this block.
3 </div>
```

and the LaTeX output will be this:

```

1 \begin{FOO}
2 Some text for this block.
3 \end{FOO}
```

It is up to the book author how to define the style of the block. You can define the style of the `<div>` in CSS and include it in the output via the `includes` option in the YAML metadata. Similarly, you may define the LaTeX environment via `\newenvironment` and include the definition in the LaTeX output via the `includes` option. For example, we may save the following style in a CSS file, say, `style.css`:

```

1 div.FOO {
2 font-weight: bold;
3 color: red;
4 }
```

And the YAML metadata of the R Markdown document can be:

```

1 ---
2 output:
3 bookdown::html_book:
4 includes:
5 in_header: style.css
6 ---
```

We have defined a few types of blocks for this book to show notes, tips, and warnings, etc. Below are some examples:

The `knitr` block engine was designed to display simple content (typically a paragraph of plain text). You can use simple formatting syntax such as making certain words bold or italic, but more advanced syntax such as citations and cross-references will not work. However, there is an alternative engine named `block2` that supports arbitrary Markdown



R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see <http://www.gnu.org/licenses/>.



R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see <http://www.gnu.org/licenses/>.



R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see <http://www.gnu.org/licenses/>.



R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see <http://www.gnu.org/licenses/>.



R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see <http://www.gnu.org/licenses/>.

syntax, e.g.,

```

1 ````{block2, type='FOO'}
2 Some text for this block [@citation-key].
3
4 - a list item
5 - another item
6
7 More text.
8 ````
```

The `block2` engine should also be faster than the `block` engine if you have a lot of custom blocks in the document, but its implementation was based on a [hack](#), so we are not 100% sure if it is always going to work in the future. We have not seen problems with Pandoc v1.17.2 yet.

One more caveat for the `block2` engine: if the last element in the block is not an ordinary paragraph, you must leave a blank line at the end, e.g.,

```

1 ````{block2, type='FOO'}
2 Some text for this block [@citation-key].
3
4 - a list item
5 - another item
6 - end the list with a blank line
7
8 ````
```

The theorem and proof environments in Section [戊.2.2](#) are actually implemented through the `block2` engine.

For all custom blocks based on the `block` or `block2` engine, there is one chunk option `echo` that you can use to show (`echo = TRUE`) or hide (`echo = FALSE`) the blocks.

## 戊.8 Citations

Although Pandoc supports multiple ways of writing citations, we recommend you to use BibTeX databases because they work best with LaTeX/PDF output. Pandoc can process other types of bibliography databases with the utility `pandoc-citeproc` (<https://pandoc.org/citeproc.html>:

//[github.com/jgm/pandoc-citeproc](https://github.com/jgm/pandoc-citeproc)), but it may not render certain bibliography items correctly (especially in case of multiple authors per item), and BibTeX can do a better job when the output format is LaTeX. With BibTeX databases, you will be able to define the bibliography style if it is required by a certain publisher or journal.

A BibTeX database is a plain-text file (with the conventional filename extension `.bib`) that consists of bibliography entries like this:

```

1 @Manual{R-base,
2 title = {R: A Language and Environment for Statistical
3 Computing},
4 author = {{R Core Team}},
5 organization = {R Foundation for Statistical Computing},
6 address = {Vienna, Austria},
7 year = {2016},
8 url = {https://www.R-project.org/},
9 }
```

A bibliography entry starts with `@type{`, where `type` may be `article`, `book`, `manual`, and so on.<sup>1</sup> Then there is a citation key, like `R-base` in the above example. To cite an entry, use `@key` or `[@key]` (the latter puts the citation in braces), e.g., `@R-base` is rendered as R Core Team [20], and `[@R-base]` generates “20”. If you are familiar with the **natbib** package in LaTeX, `@key` is basically `\citet{key}`, and `[@key]` is equivalent to `\citep{key}`.

There are a number of fields in a bibliography entry, such as `title`, `author`, and `year`, etc. You may see <https://en.wikipedia.org/wiki/BibTeX> for possible types of entries and fields in BibTeX.

There is a helper function `write_bib()` in **knitr** to generate BibTeX entries automatically for R packages. Note that it only generates one BibTeX entry for the package itself at the moment, whereas a package may contain multiple entries in the `CITATION` file, and some entries are about the publications related to the package. These entries are ignored by `write_bib()`.

```

1 # the second argument can be a .bib file
2 knitr:::write_bib(c('knitr', 'stringr'), '', width = 60)
```

---

<sup>1</sup>The type name is case-insensitive, so it does not matter if it is `manual`, `Manual`, or `MANUAL`.

```

1 @Manual{R-knitr,
2 title = {knitr: A General-Purpose Package for Dynamic Report
3 Generation in R},
4 author = {Yihui Xie},
5 year = {2018},
6 note = {R package version 1.20},
7 url = {https://CRAN.R-project.org/package=knitr},
8 }
9 @Manual{R-stringr,
10 title = {stringr: Simple, Consistent Wrappers for Common
11 String Operations},
12 author = {Hadley Wickham},
13 year = {2018},
14 note = {R package version 1.3.1},
15 url = {https://CRAN.R-project.org/package=stringr},
16 }
```

Once you have one or multiple `.bib` files, you may use the field `bibliography` in the YAML metadata of your first R Markdown document (which is typically `index.Rmd`), and you can also specify the bibliography style via `biblio-style` (this only applies to PDF output), e.g.,

```

1 ---
2 bibliography: ["one.bib", "another.bib", "yet-another.bib"]
3 biblio-style: "apalike"
4 link-citations: true
5 ---
```

The field `link-citations` can be used to add internal links from the citation text of the author-year style to the bibliography entry in the HTML output.

When the output format is LaTeX, citations will be automatically put in a chapter or section. For non-LaTeX output, you can add an empty chapter as the last chapter of your book. For example, if your last chapter is the Rmd file `06-references.Rmd`, its content can be an inline R expression:

```

1 `r if (knitr:::is_html_output()) '# References {-}'`
```

## 戊.9 Index

Currently the index is only supported for LaTeX/PDF output. To print an index after the book, you can use the LaTeX package **makeidx** in the preamble (see Section ??):

```
1 \usepackage{makeidx}
2 \makeindex
```

Then insert `\printindex` at the end of your book through the YAML option `includes -> after_body`. An index entry can be created via the `\index{}` command in the book body, e.g., `\index{GIT}`.

## 戊.10 HTML widgets

Although one of R's greatest strengths is data visualization, there are a large number of JavaScript libraries for much richer data visualization. These libraries can be used to build interactive applications that can easily render in web browsers, so users do not need to install any additional software packages to view the visualizations. One way to bring these JavaScript libraries into R is through the **htmlwidgets** package<sup>21</sup>.

HTML widgets can be rendered as a standalone web page (like an R plot), or embedded in R Markdown documents and Shiny applications. They were originally designed for HTML output only, and they require the availability of JavaScript, so they will not work in non-HTML output formats, such as LaTeX/PDF. Before **knitr** v1.13, you will get an error when you render HTML widgets to an output format that is not HTML. Since **knitr** v1.13, HTML widgets will be rendered automatically as screenshots taken via the **webshot** package.<sup>22</sup> Of course, you need to install the **webshot** package. Additionally, you have to install PhantomJS (<http://phantomjs.org>), since it is what **webshot** uses to capture screenshots. Both **webshot** and PhantomJS can be installed automatically from R:

```
1 install.packages('webshot')
2 webshot::install_phantomjs()
```

The function `install_phantomjs()` works for Windows, OS X, and Linux. You may also choose to download and install PhantomJS by yourself, if you are familiar with

modifying the system environment variable PATH.

When **knitr** detects an HTML widget object in a code chunk, it either renders the widget normally when the current output format is HTML, or saves the widget as an HTML page and calls **webshot** to capture the screen of the HTML page when the output format is not HTML. Here is an example of a table created from the **DT** package:<sup>23</sup>

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

Showing 1 to 10 of 150 entries      Previous  2 3 4 5 ... 15 Next

图 戊-5 A table widget rendered via the DT package.

If you are reading this book as web pages now, you should see an interactive table generated from the above code chunk, e.g., you may sort the columns and search in the table. If you are reading a non-HTML version of this book, you should see a screenshot of the table. The screenshot may look a little different with the actual widget rendered in the web browser, due to the difference between a real web browser and PhantomJS's virtual browser.

There are a number of **knitr** chunk options related to screen-capturing. First, if you are not satisfied with the quality of the automatic screenshots, or want a screenshot of the widget of a particular state (e.g., after you click and sort a certain column of a table), you

may capture the screen manually, and provide your own screenshot via the chunk option `screenshot.alt` (alternative screenshots). This option takes the paths of images. If you have multiple widgets in a chunk, you can provide a vector of image paths. When this option is present, **knitr** will no longer call **webshot** to take automatic screenshots.

Second, sometimes you may want to force **knitr** to use static screenshots instead of rendering the actual widgets even on HTML pages. In this case, you can set the chunk option `screenshot.force = TRUE`, and widgets will always be rendered as static images. Note that you can still choose to use automatic or custom screenshots.

Third, **webshot** has some options to control the automatic screenshots, and you may specify these options via the chunk option `screenshot.opts`, which takes a list like `list(delay = 2, cliprect = 'viewport')`. See the help page `?webshot::webshot` for the full list of possible options, and the [package vignette](#) `vignette('intro', package = 'webshot')` illustrates the effect of these options. Here the `delay` option can be important for widgets that take long time to render: `delay` specifies the number of seconds to wait before PhantomJS takes the screenshot. If you see an incomplete screenshot, you may want to specify a longer delay (the default is 0.2 seconds).

Fourth, if you feel it is slow to capture the screenshots, or do not want to do it every time the code chunk is executed, you may use the chunk option `cache = TRUE` to cache the chunk. Caching works for both HTML and non-HTML output formats.

Screenshots behave like normal R plots in the sense that many chunk options related to figures also apply to screenshots, including `fig.width`, `fig.height`, `out.width`, `fig.cap`, and so on. So you can specify the size of screenshots in the output document, and assign figure captions to them as well. The image format of the automatic screenshots can be specified via the chunk option `dev`, and possible values are `pdf`, `png`, and `jpeg`. The default for PDF output is `pdf`, and it is `png` for other types of output. Note that `pdf` may not work as faithfully as `png`: sometimes there are certain elements on an HTML page that fail to render to the PDF screenshot, so you may want to use `dev = 'png'` even for PDF output. It depends on specific cases of HTML widgets, and you can try both `pdf` and `png` (or `jpeg`) before deciding which format is more desirable.



## 附录己 常见问题

### 己.1 如何修改rm命令，让删除的文件去回收站？

答. 文件被删且难以恢复，想必很多人都曾经遇到过。参考[这里](#)。具体做法如下。

```
$ sudo apt install trash-cli
$ gedit ~/.bashrc
```

打开文件，在末尾加上，`alias rm='trash-put'`，保存退出。运行如下命令。

```
$ source .bashrc
```

重启终端即可。想用原生的`rm`，则为`\rm`。当然这样也有弊端，详细说明见[这里](#)。

### 己.2 文章标题索引时不可以放在标题末尾

答. 索引有个坑，见[这里](#)。另外，如果一些`\ref` `\cite` 的编号出错了，也容易导致在目录里无索引无参考文献。

### 己.3 为什么中文双引号要用`\cqh`和`\cqt`表示？

答. 因为没能很好的解决中文双引号问题，使用`\cqh` (*Chinese Quote Head*) ,`\cqt` (*Chinese Quote Tail*)，折衷解决。见<https://stackoverflow.com/questions/52052231/how-to-write-chinese-quotes-in-bookdown>

### 己.4 代码框里下划线##怎么处置？

答. 直接使用`##`，单个##会变成两个##。待完成本书，去 *pandoc* 提意见，再改。对于代码中出现，作为注释符号时，没有问题，但是出现在字符串中时，可能需要把整段代码改为`\lstset{mathescape=true}`，改完后，再改回`\lstset{mathescape=false}`。作为特殊字符处理，不然报错。例子见[第九章](#)相关代码。

己.5 deepin/win10 双系统 deepin 下其他盘带锁问题。

答. Windows10 进入后，按Super键（也就是Windows图案的键，或者直接点击左下角的微软标志），弹出如图己-1所示操作栏，点击设置。



图 己-1 打开 Windows10 设置

点击系统大按钮，如图己-2所示。



图 己-2 选择系统设置

选择电源和睡眠选项-> 其他电源设置。如图 [己-3](#)所示。



图 己-3 选择其他电源设置

选择电源和按钮的功能选项。如图 己-4所示。

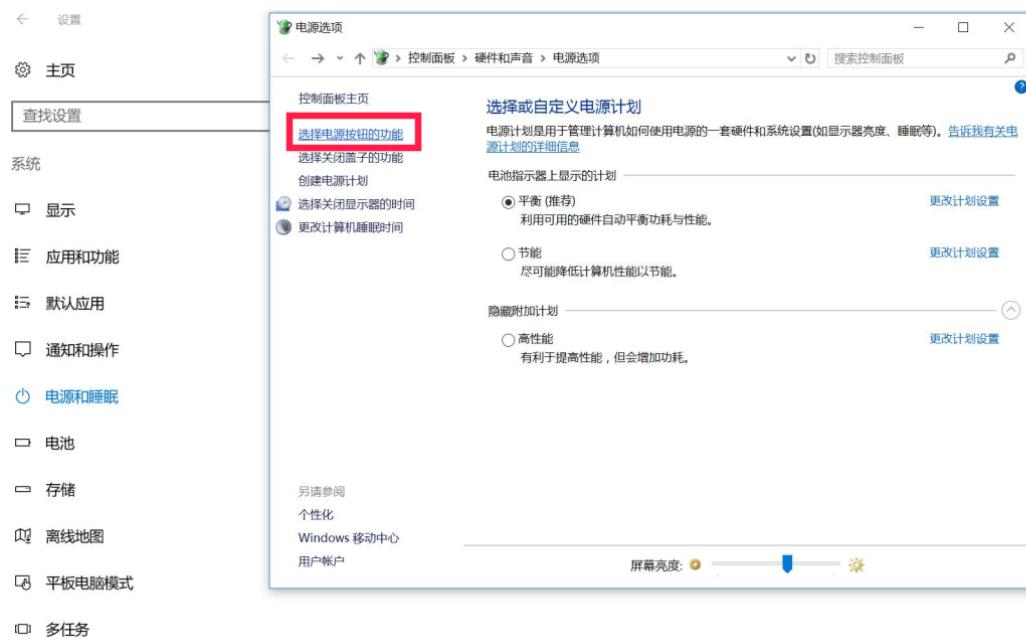


图 己-4 选择电源和按钮的功能

取消快速启动，并确定。如图己-5所示。

## 附录己 常见问题



图 己-5 取消快速启动

## 附录庚 操作系统安装延伸阅读

### 庚.1 计算机引导过程

#### 庚.1.1 传统 BIOS 引导

所谓 BIOS 或 Basic Input-Output System, 就是开机时第一个被执行的程序, 又名固件。一般来说它储存在主板上的一块闪存中, 与硬盘彼此独立。

BIOS 被启动后, 会按启动顺序加载磁盘的前 512 字节, 即主引导记录, 前 440 字节包含某个启动引导器, 像 GRUB、Syslinux 和 LILO 之类的第一启动阶段代码。因为空间太小了, 后续的启动代码保存在磁盘上, 最后启动引导器又通过「链式引导」, 或是直接加载内核, 以加载一个操作系统。

整个过程如下:

开机时加电自检。加电自检后, BIOS 初始化一些必要的硬件以准备引导, 比如硬盘和键盘等。BIOS 执行在「BIOS 硬盘顺序」中的第一块硬盘上的前 440 字节代码, 即主引导记录。MBR 接管后, 执行它之后的第二阶段代码, 如果后者存在的话, 它一般就是启动引导器。第二阶段代码会读取它的支持文件和配置文件。

#### 庚.1.2 UEFI 引导

UEFI 不仅能读取分区表, 还能自动支持文件系统。所以它不像 BIOS, 已经没有仅仅 440 字节可执行代码即 MBR 的限制了, 它完全用不到 MBR。

UEFI 主流都支持 MBR 和 GPT 分区表。Apple-Intel Macs 上的 EFI 还支持 Apple 专用分区表。绝大部分 UEFI 固件支持软盘上的 FAT12, 硬盘上的 FAT16、FAT32 文件系统, 以及 CD/DVDs 的 ISO9660 和 UDF。Intel Macs 的 EFI 还额外支持 HFS/HFS+ 文件系统。

不管第一块上有没有 MBR, UEFI 都不会执行它。相反, 它依赖分区表上的一个特殊分区, 叫 EFI 系统分区, 里面有 UEFI 所要用到的一些文件。计算机供应商可以在 /EFI/ 文件夹里放官方指定的文件, 还能用固件或它的 shell, 即 UEFI shell, 来启动引导程序。EFI 系统分区一般被格式化成 FAT32, 或比较非主流的 FAT16。

#### 庚.1.3 UEFI 的多重引导

因为每个操作系统或者提供者都可以维护自己的 EFI 系统分区中的文件, 同时不影响其他系统, 所以 UEFI 的多重启动只是简单的运行不同的 UEFI 程序, 对

应于特定操作系统的引导程序。这避免了依赖 chainloading 机制（通过一个启动引导程序加载另一个引导程序，来切换操作系统）。

UEFI 引导的过程如下：

- 系统开机 - 上电自检（Power On Self Test 或 POST）。
- UEFI 固件被加载，并由它初始化启动要用的硬件。
- 固件读取其引导管理器以确定从何处（比如，从哪个硬盘及分区）加载哪个 UEFI 应用。
- 固件按照引导管理器中的启动项目，加载 UEFI 应用。
- 已启动的 UEFI 应用还可以启动其他应用（对应于 UEFI shell 或 rEFInd 之类的引导管理器的情况）或者启动内核及 initramfs（对应于 GRUB 之类引导器的情况），这取决于 UEFI 应用的配置。

## 庚.2 常见 BIOS 设置

### 庚.2.1 常见启动引导器

BIOS 或 UEFI 加载并初始化硬件完成后，会启动的一个启动引导器来接管硬件设备，引导操作系统启动的工作将有启动引导器来完成。

引导程序引导方式及程序视应用机型种类而不同。例如在普通的个人电脑上，引导程序通常分为两部分：第一阶段引导程序位于主引导记录（MBR），用以引导位于某个分区上的第二阶段引导程序，如 NTLDR、BOOTMGR 和 GNU GRUB 等。

### 庚.2.2 NTLDR/BOOTMGR

NTLDR（NT loader 的缩写）是微软的 Windows NT 系列操作系统（包括 Windows XP 和 Windows Server 2003）的引导程序。

NTLDR 可以从硬盘以及 CD-ROM、优盘等移动存储器运行并引| Windows NT 系统的启动。如果要用 NTLDR 启动其他操作系统，则需要将该操作系统所使用的启动扇区代码保存为一个文件，NTLDR 可以从这个文件加载其它引导程序。

NTLDR 主要由两个文件组成，这两个文件必须放在系统分区（大多数情况下都是 C 盘）：

NTLDR，这是引导程序本身 boot.ini，这是引导程序的配置文件当 boot.ini 丢失时，NTLDR 会启动第一块硬盘第一个分区上的 /Windows 目录中的系统。bootmgr（Windows Boot Manager）是从 Windows Vista 开始引进的新一代开机管理程序，用以替换 NTLDR 。

当电脑运行完 POST (Power On Self Test) 后，传统型 BIOS 会根据引导扇区查找开机硬盘中标记“引导”分区下的 bootmgr 文件；若是 UEFI 则是 bootmgr.efi 文件，接着管理程序会读取开机配置数据库 BCD (Boot Configuration Database) 内的引导数据，接着根据其中的数据加载与默认或用户所选择的操作系统。

### 庚.2.3 GNU GRUB 及其使用

GNU GRUB (简称“GRUB”) 是一个来自 GNU 项目的启动引导程序。GRUB 是多启动规范的实现，它允许用户可以在计算机内同时拥有多个操作系统，并在计算机启动时选择希望运行的操作系统。GRUB 可用于选择操作系统分区上的不同内核，也可用于向这些内核传递启动参数。

#### GRUB Legacy / GURB2

新的 GRUB2 (GRUB 第二版) [F] GRUB 的重写版本，它是 GRUB 的大革新。GRUB2 对 Linux 系统做了更多的优化，支持更多的功能，如动态的载入模块（而在之前的 GRUB 中，新增或[F]除模块要重新编译 GRUB）等。GRUB2 的版本号为 0.98 或更高；旧的 GRUB 的版本号则为 0.97 或更低，也被称为“GRUB Legacy”或“GRUB1”等。GRUB2 的配置、命令等较 GRUB Legacy 有一定的不同。

#### GRUB2 的配置文件

GRUB2 配置文件的文件名和位置随系统的不同而不同；常见为 /boot/grub/grub.conf。

修改 GRUB 的配置文件后，不必把 GRUB 重新安装到 MBR 或者某个分区中。在 Linux 中，“grub-install”命令是用来把 GRUB 的步骤 1 安装到 MBR 或者分区中的。GRUB 的配置文件、步骤 2 以及其它文件必须安装到某个可用的分区中。如果这些文件或者分区不可用，步骤 1 将把用户留在命令行界面。

除了硬盘外，GRUB 也可安装到光盘、软盘和优盘等移动介质中，这样就可以带起一台无法从硬盘启动的系统。

#### 使用 GRUB Shell

## 庚.3 LINUX 启动过程

### 庚.3.1 VMLINUZ

- vmlinuz：一个非压缩的，静态链接的，可执行的，不能 bootable 的 Linux kernel 文件。是用来生成 vmlinuz 的中间步骤。
- vmlinuz：一个压缩的，能 bootable 的 Linux kernel 文件。vmlinuz 是 Linux kernel 文件的历史名字，它实际上就是 zImage 或 bzImage。
- zImage：仅适用于 640k 内存的 Linux kernel 文件。

- bzImage: Big zImage, 适用于更大内存的 Linux kernel 文件。  
对于目前的 Linux 桌面系统, vmlinuz 实际上即 bzImage kernel 文件。

### 庚.3.2 INITRD

initrd 的英文含义是 initialized RAM disk, 就是由 bootloader 初始化的内存盘。在 linux 内核启动前, bootloader 会将存储介质中的 initrd 文件加载到内存, 内核启动时会在访问真正的根文件系统前先访问该内存中的 initrd 文件系统。

initrd 分 image-initrd 及 cpio-initrd 两种。

2.4 及以前的内核只支持 image-initrd, 其核心文件是/linuxrc。2.6 及以后的内核两种格式的 initrd 都支持, 并且目前的 Linux 发行版使用的几乎都是 cpio-initrd, 其核心文件是/init。在 bootloader 配置了 initrd 的情况下, 内核启动被分成了两个阶段, 第一阶段先执行 initrd 文件系统中的/init(或早期的/linuxrc), 完成加载驱动模块等任务, 第二阶段才会挂载真正的根文件系统中, 并且 chroot 到真正的根文件系统(例如硬盘上的某个分区)来完成系统的启动。

转载网页:

- [http://wiki.deepin.org/wiki/DEEPIN\\_%E6%A1%8C%E9%9D%A2%E7%B3%BB%E7%BB%9F%E5%AE%89%E8%A3%85%E6%A6%82%E8%BF%B0](http://wiki.deepin.org/wiki/DEEPIN_%E6%A1%8C%E9%9D%A2%E7%B3%BB%E7%BB%9F%E5%AE%89%E8%A3%85%E6%A6%82%E8%BF%B0)

## 附录辛 大事记

- 2017-08-26 开工找模板。
- 2017-09-28 买到《Linux Bible》实体书中文版，可以带徒弟一起写本文了。
- 2017-10-26 第一次开课，给俩兄弟讲 Linux 下 C 语言基础。
- 2017-10-29 R 语言更新，貌似旧的安装方法已经无法在 deepin 下安装，需要调整，暂时没有调整。
- 2017-10-29 注册了优酷土豆账号，用于上传本书的相关视频。
- 2018-05-29 模板转移 SJTU 模板中
- 2018-07-02 第一版草稿成功出炉
- 2018-07-10 成功转移到新模板
- 2018-08-28 定个一百天计划，一周一章，直至元旦，从这周开始。
- 2018-11-22 完成第二部分，现实太残酷，一月一章吧。
- 2018-12-04 家庭事务繁多，今年到此为止，明年开春后再更新。



# 索引

## A

appendix, 231

## B

BibTeX, 246

## C

citation, 246

code chunk, 232

cross-reference, 223, 226, 234, 238, 242

custom block, 243

## E

equation, 223

## F

figure, 233

floating environment, 233

服务器, 177

## H

HTML widget, 249

## I

index, 249

inline R code, 232

## K

knitr::include\_graphics(), 236

## L

LaTeX math expression, 222

Linux, 3

longtable, 239

## M

Markdown, 219

## P

Pandoc, 219

part, 230

## S

samba, 181

shell, 55, 56

深度操作系统, 27

深度科技, 27

## T

table, 238

theorem, 225

## W

Windows, 9, 181

## Y

愚公移山, 193



## 参考文献

- [1] 何闻. 标准动态力发生装置国内外研究现状[J]. 机电工程, 1999(2): 47-49.
- [2] 崔万照, 马伟, 邱乐德, 等. 电磁超介质及其应用[M]. 北京: 国防工业出版社, 2008.
- [3] CHEN H, CHAN C T. Acoustic cloaking in three dimensions using acoustic metamaterials[J]. Applied Physics Letters, 2007, 91: 183518.
- [4] KIM S, WOO N, YEOM H Y, et al. Design and Implementation of Dynamic Process Management for Grid-enabled MPICH[C]// The 10th European PVM/MPI Users' Group Conference. Venice, Italy: [s.n.], 2003.
- [5] JOANNOPOULOS J D, JOHNSON S G, WINN J N. Photonic Crystals: Molding the Flow of Light[M]. [S.l.]: Princeton University Press, 2008.
- [6] 猪八戒. 论流体食物的持久保存[D]. 北京: 广寒宫大学, 2005.
- [7] 1363-2000 I S. IEEE Standard Specifications for Public-Key Cryptography[M]. New York: IEEE, 2000.
- [8] CHEN H, WU B I, ZHANG B, et al. Electromagnetic Wave Interactions with a Metamaterial Cloak[J]. Physical Review Letters, 2007, 99(6): 63903.
- [9] KOCHER C, JAFFE J, JUN B. Differential Power Analysis[C]// WIENER M. Advances in Cryptology (CRYPTO '99). Vol. 1666. [S.l.]: Springer-Verlag, 1999: 388-397.
- [10] 王重阳, 黄药师, 欧阳峰, 等. 武林高手论文集[C]// 第 N 次华山论剑. 西安, 中国: 中国古籍出版社, 2006.
- [11] JEYAKUMAR A R. Metamori: A library for Incremental File Checkpointing[D]. Blacksburg: Virginia Tech, 2004.
- [12] 沙和尚. 论流沙河的综合治理[D]. 北京: 清华大学, 2005.
- [13] ZADOK E. FiST: A System for Stackable File System Code Generation[D]. USA: Computer Science Department, Columbia University, 2001.
- [14] 白云芬. 信用风险传染模型和信用衍生品的定价[D]. 上海: 上海交通大学, 2008.

- [15] WOO A, BAILEY D, YARROW M, et al. The NAS Parallel Benchmarks 2.0[R/OL]. The Pennsylvania State University CiteSeer Archives, 1995. <http://www.nasa.org/>.
- [16] 萧钰. 出版业信息化迈入快车道[J/OL], 2001. <http://www.creader.com/news/20011219/200112190019.html>.
- [17] CHRISTINE M. Plant physiology: plant biology in the Genome Era[J/OL]. Science, 1998, 281: 331-332. <http://www.sciencemag.org/cgi/collection/anatmorp>.
- [18] R Core Team. R: A Language and Environment for Statistical Computing[M/OL]. Vienna, Austria: [s.n.], 2012. <http://www.R-project.org/>.
- [19] XIE Y. Dynamic Documents with R and knitr[M/OL]. 2nd. Boca Raton, Florida: Chapman, Hall/CRC, 2015. <http://yihui.name/knitr/>.
- [20] R Core Team. R: A Language and Environment for Statistical Computing[A/OL]. Vienna, Austria: R Foundation for Statistical Computing, 2018. <https://www.R-project.org/>.
- [21] VAIDYANATHAN R, XIE Y, ALLAIRE J, et al. Htmlwidgets: HTML Widgets for R[A/OL]. R package version 1.2. 2018. <https://CRAN.R-project.org/package=htmlwidgets>.
- [22] CHANG W. Webshot: Take Screenshots of Web Pages[A/OL]. R package version 0.5.1. 2018. <https://CRAN.R-project.org/package=webshot>.
- [23] XIE Y. DT: A Wrapper of the JavaScript Library 'DataTables'[A/OL]. R package version 0.4. 2018. <https://CRAN.R-project.org/package=DT>.