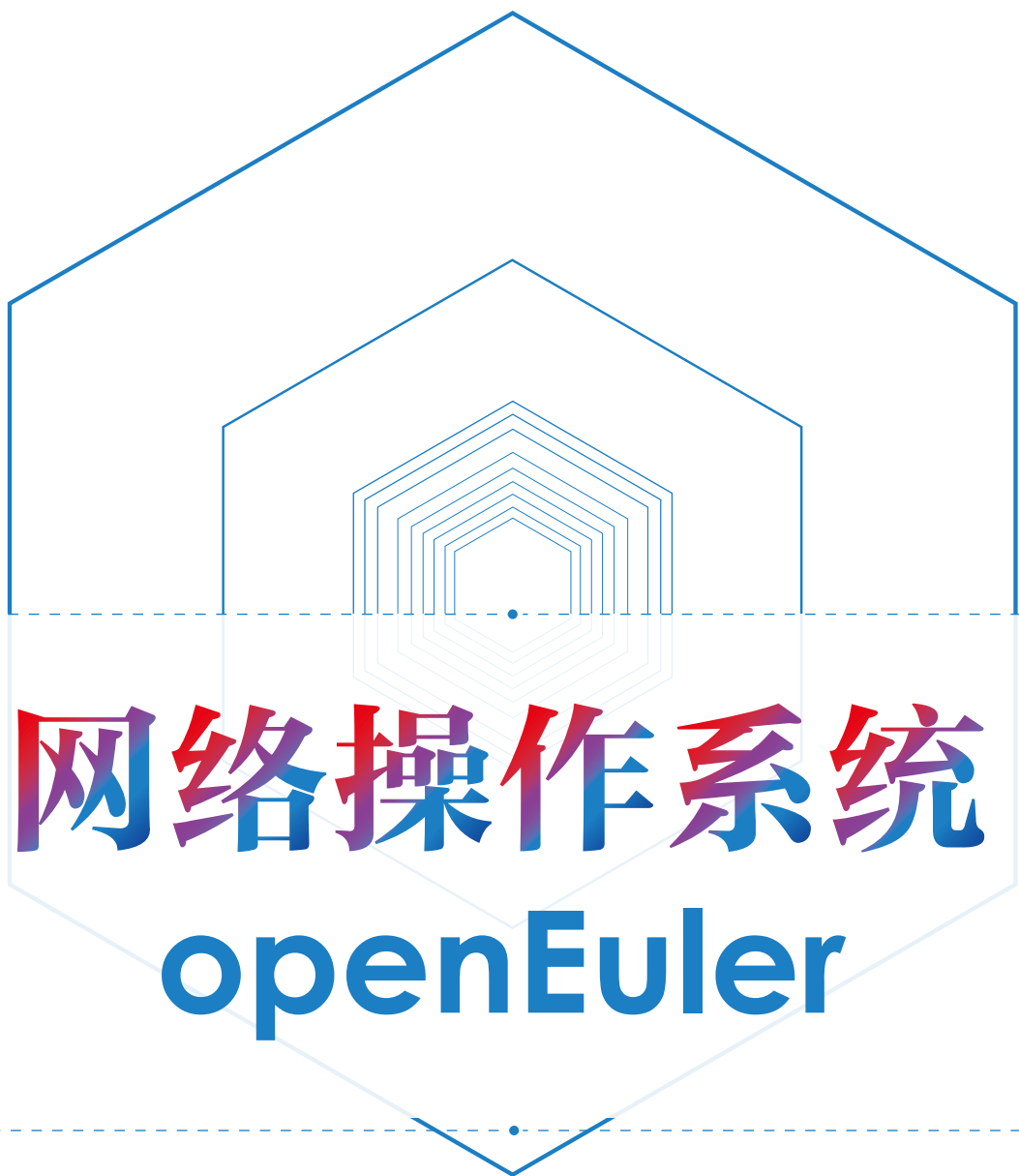
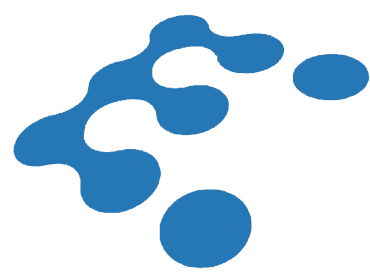




主 编 ◎ 王瑜坤



Contents | 目录

第 1 章 | Linux 基础

1.1	了解 Linux	3
1.1.1	Linux 的发展历史	3
1.1.2	Linux 的版权	4
1.1.3	Linux 的软件组成	4
1.2	openEuler 的安装、登录和关闭	5
1.2.1	配置虚拟机	6
1.2.2	安装 Linux	11
1.2.3	登录和关闭 Linux	13
1.2.4	修改软件安装源	15
1.3	openEuler 的救援	16

第 2 章 | 命令行操作基础

2.1	命令的基础知识	21
2.1.1	Linux 命令的语法格式	21
2.1.2	Linux 的操作技巧	22
2.1.3	Linux 的命令分类	23
2.2	登录和开关机命令	23
2.2.1	login 命令	23
2.2.2	last 命令	24
2.2.3	exit 命令	25
2.2.4	shutdown 命令	25
2.3	文件目录操作命令	26
2.3.1	Linux 文件目录结构	26
2.3.2	Linux 文件路径	27
2.3.3	pwd 命令	27

2.3.4	cd 命令	28
2.3.5	ls 命令	29
2.3.6	mkdir 命令	31
2.3.7	cp 命令	32
2.3.8	touch 命令	33
2.3.9	mv 命令	35
2.3.10	rm 命令	36
2.3.11	cat 命令	38
2.3.12	head 命令	39
2.3.13	tail 命令	40
2.3.14	more 命令	41
2.3.15	less 命令	43
2.3.16	find 命令	43
2.3.17	locate 命令	46
2.3.18	which 命令	48
2.4	其他操作命令	48
2.4.1	gzip 命令	49
2.4.2	tar 命令	51
2.4.3	ln 命令	56

第3章 | 文本编辑器及文本处理

3.1	Linux 常用文本编辑器	63
3.1.1	gedit 编辑器	63
3.1.2	kedit 编辑器	63
3.1.3	vim 编辑器	63
3.2	进入 vim 编辑器	64
3.3	vim 的编辑操作	64
3.3.1	三种常见模式	65
3.3.2	命令模式	65
3.3.3	末行模式	67
3.4	文本处理	69

3.4.1	文本处理之 grep	69
3.4.2	文本处理之 sed	74
3.4.3	文本处理之 awk	78

第4章 | 用户和权限管理

4.1	管理用户和组	85
4.1.1	用户和组的概念	85
4.1.2	用户和组的配置文件	86
4.2	用户的管理	90
4.3	组的管理	94
4.4	文件权限的管理	98
4.4.1	文件类型	98
4.4.2	权限位说明	98
4.4.3	访问权限操作命令	99
4.4.4	预设权限掩码	102

第5章 | 软件包的安装和服务管理

5.1	RPM 管理软件包	107
5.1.1	RPM 软件包文件命名规则	107
5.1.2	RPM 管理软件包的优缺点分析	107
5.1.3	rpm 命令	107
5.2	使用 dnf 管理软件包	109
5.2.1	配置 dnf	110
5.2.2	管理软件包	114
5.2.3	更新软件包	118
5.3	源代码安装软件包	118
5.3.1	源代码安装的优缺点	119
5.3.2	源代码安装的基本步骤	119
5.4	使用 systemd 管理服务	121
5.4.1	Systemd Unit 的基本概念	121

5.4.2 使用 systemd 管理服务	122
-----------------------------	-----

第6章 | 文件系统和磁盘管理

6.1 文件系统的基本概念	127
6.1.1 openEuler 的文件系统	127
6.1.2 swap 交换空间	128
6.1.3 磁盘基本知识	128
6.2 磁盘分区	130
6.2.1 磁盘分区的工具	131
6.2.2 分区表类型选择	131
6.3 磁盘管理	131
6.3.1 MBR 分区	132
6.3.2 分区的命名	132
6.3.3 分区操作	133
6.3.4 格式化和挂载	138
6.3.5 启动自动挂载	140
6.3.6 swap 分区的使用	143
6.4 逻辑卷管理	144

第7章 | 系统管理

7.1 任务管理	155
7.1.1 定时任务 at	155
7.1.2 周期任务 cron	157
7.2 网络管理	159
7.2.1 网络管理基础概念	160
7.2.2 网卡设备配置文件	160
7.2.3 NetworkManager	162
7.3 进程管理	167
7.3.1 进程信息查询	168
7.3.2 进程的信号	170

7.3.3	进程优先级	171
7.3.4	进程管理集成命令	173

第 8 章 | Shell 脚本

8.1	Shell 基础	179
8.1.1	Shell 的发展历史	179
8.1.2	查看 Shell	180
8.1.3	Shell 脚本内容	180
8.1.4	编写 Shell 脚本	181
8.1.5	Shell 脚本的执行	182
8.2	Shell 编程基础	182
8.2.1	重定向和管道	183
8.2.2	Shell 特殊字符	184
8.2.3	Shell 变量	185
8.2.4	Shell 基本运算	186
8.3	Shell 简单编程	187
8.3.1	Shell 调试模式	188
8.3.2	Shell 编程实例	189

第 9 章 | Samba 服务

9.1	Samba 简介	195
9.1.1	Samba 的功能	195
9.1.2	Samba 协议	196
9.2	Samba 工作原理	196
9.2.1	Samba 的工作流程	197
9.2.2	Samba 软件	198
9.3	配置 Samba 服务	199
9.3.1	安装并启动 Samba 服务	199
9.3.2	主要配置文件	200

9.3.3	日志文件和密码文件	202
9.4	配置案例和验证	203
9.4.1	用户主目录共享	203
9.4.2	匿名共享目录	205
9.4.3	普通用户共享目录	207
9.4.4	从 Linux 客户端访问共享目录	209
9.5	Samba 服务器运维	212
9.5.1	日志分析	212
9.5.2	故障排查	213

第 10 章 | DNS 服务

10.1	DNS 简介	219
10.2	DNS 服务的安装和启动	219
10.2.1	DNS 的分类	220
10.2.2	DNS 的查询	221
10.3	DNS 服务的配置	221
10.4	DNS 服务器配置案例和验证	225
10.4.1	纯缓存 DNS 服务器	225
10.4.2	本地 DNS 服务器	229
10.4.3	备份 DNS 服务器	235

第 11 章 | Web 服务

11.1	常见 Web 服务器	243
11.2	HTTP 协议	243
11.3	Apache 服务器的安装和配置	244
11.3.1	安装和启动 Apache 服务器	244
11.3.2	配置文件	246
11.4	Apache 服务器配置案例和验证	247
11.4.1	默认网页实例	247
11.4.2	用户个人主页	249

11.4.3	虚拟目录	251
11.4.4	配置基于 IP 地址的虚拟主机	252
11.4.5	配置基于端口号的虚拟主机	254
11.4.6	配置基于域名的虚拟主机	255

第 12 章 | FTP 服务

12.1	FTP 的简介	261
12.1.1	FTP 的工作原理	261
12.1.2	FTP 的用户	262
12.2	FTP 服务器软件包的安装和启停	263
12.3	FTP 服务的配置文件	264
12.4	匿名用户配置案例	267
12.5	本地用户配置案例	272
12.6	虚拟用户配置案例	275
12.6.1	配置虚拟用户的基本步骤	275
12.6.2	配置虚拟用户的过程	276
12.6.3	验证虚拟用户访问 FTP 服务	278

第 13 章 | 防火墙和安全模块

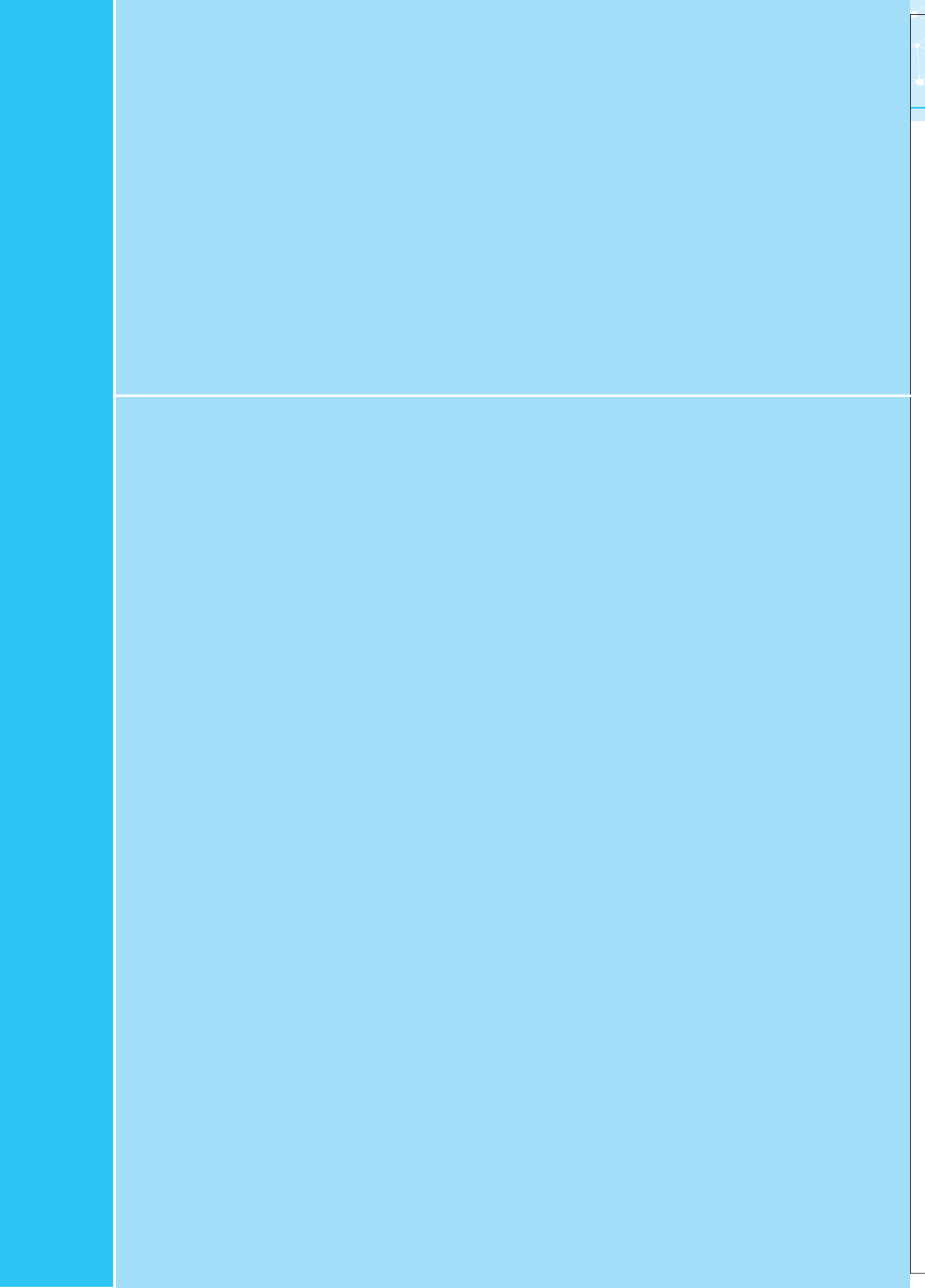
13.1	防火墙概述	285
13.2	安全模块概述	285
13.2.1	自主访问控制	285
13.2.2	强制访问控制	286
13.2.3	SELinux 的工作机制	286
13.3	防火墙配置	287
13.4	安全模块配置	293
13.5	防火墙和安全模块配置案例	299

► 第3章 文本编辑器及文本处理

文本处理是操作系统的实用工具对文本文件的处理操作；文本编辑器是计算机软件中主要用于编写和查看文本文件的软件。在 Linux 操作系统中，存在许多文本文件格式的配置文件，如用户账户文件，因此对配置文件进行文本编辑是 Linux 操作维护的重要操作。不同的文件编辑器功能类似，但也有各自独特的功能。

本章的学习要点如下。

- (1) 理解文本编辑器在系统管理和开发中的基础作用。
- (2) 掌握各编辑器的特点。
- (3) 掌握 vim 核心操作。
- (4) 掌握文本处理方式。
- (5) 了解命令组合和脚本实现自动化处理的基本思想。



3.1 Linux 常用文本编辑器

文本编辑器是操作系统的基础功能软件之一。根据使用环境的不同，Linux 的文本编辑器有很多类型。

常见的 Linux 文本编辑器如下。

- (1) gedit 编辑器。
- (2) kedit 编辑器。
- (3) vim 编辑器。

3.1.1 gedit 编辑器

gedit 是一个在 GNOME 图形桌面环境下兼容 UTF-8 的文本编辑器。它简单易用，有良好的语法高亮，对中文支持很好，支持包括 GB2312、GBK 在内的多种字符编码。gedit 是一款自由软件。gedit 包含语法高亮和多文件编辑并行的功能。利用 GNOME VFS 库，它还可以编辑远程文件。它支持完整的恢复、重做，以及查找和替换功能。

gedit 的优点是图形化界面，易上手，操作习惯与 Windows 类似，包括常用的快捷键（复制、粘贴等）。

gedit 的缺点是需要安装图形化桌面才能使用。

3.1.2 kedit 编辑器

与 gedit 类似，kedit 是 KDE 图形化桌面中常用的一种文本编辑器。Linux 的图形化界面有两种：KDE 和 GNOME，为了适应图形化界面的技术，形成了不同的文本编辑器。

kedit 是一个非常小的编辑器，特别适用于浏览文本和各种配置文件。它的优点、缺点与 gedit 类似。优点是图形化界面，易上手，操作习惯与 Windows 类似，包括常用的快捷键（复制、粘贴等）。缺点是需要安装图形化 KDE 桌面才能使用。

3.1.3 vim 编辑器

vi 是标准的 Unix 文本编辑器，也是最古老的文本编辑器。所有的 Unix 都默认带有 vi 文本编辑器，但是 vi 不是开源软件，人们无法自由使用它。vim 是开源的文本编辑器，它是模仿 vi 功能开发的，在部分 Linux 中，vi 和 vim 指向同一个程序。vim 具备程序编码增强的许多功能，如代码自动补全、语法高亮显示等，被广泛使用。

vim 首个版本由布莱姆·米勒于 1991 年发布。英文全称为 Vi IMitation，随着功能持续补充，其官方全称调整为 Vi Improved。从命名可以看出，vim 是 vi 的增强版本，它一直是在开放源代码方式下发行的自由软件。

vim 具有以下三种工作模式。

- (1) 命令模式，用于移动光标、翻页、复制、删除等操作。
- (2) 编辑模式，用于输入文本。



（3）末行模式，用于存盘、退出、查找、设置等操作。

vim 还具有多种模式，例如，基本模式中包含普通模式、插入模式、可视模式、选择模式、命令行模式、Ex 模式；派生模式中包含操作符等待模式、插入普通模式、插入可视模式、插入选择模式、替换模式等。本章主要围绕三种基本工作模式展开，让大家掌握 vim 的基本操作。

3.2

进入 vim 编辑器

语法：

```
vi [选项] [目标文件]...
```

vim 命令选项如表 3-1 所示。

表 3-1 vim 命令选项

选项	解释
-c	打开文件之前执行指定的命令
-R	以只读方式打开，但是可以强制保存
-M	以只读方式打开，不可以强制保存
-r	回复崩溃的会话
+num:	从第 num 行开始

● vim 基础操作 – 打开文件

```
root@localhost ~]# vi filename
```

如果 filename 文件存在，则会打开文件并显示文件内容。

如果 filename 文件不存在，vim 会在下面提示 [New File]，并且会在第一次保存时创建该文件。

```
root@localhost ~]# vi test.txt
~
~
~
"test.txt" [New File]
```

3.3

vim 的编辑操作

vim 是一个文本编辑器，对于初学者具有一定的难度。首先要明确区分 vim 的工作模式，在不同的模式下，键盘录入字符的含义是不一样的。

3.3.1 三种常见模式

vim 的工作模式如图 3-1 所示。

(1) 命令模式，也称普通（normal）模式，可以实现移动光标、复制、粘贴、删除文本等操作。

(2) 编辑模式，也称插入（insert）模式，用于编辑文本操作。

(3) 末行模式，也称扩展命令（extended command）模式，用于保存、退出、设置等操作。

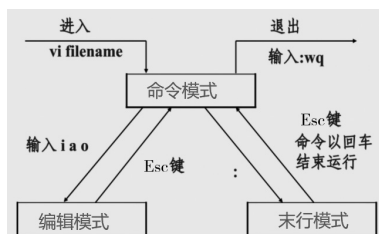


图 3-1 vim/vi 工作模式

工作模式的切换对于初学者来说有些复杂，常常让大家感到疑惑。建议初学者以命令模式为核心，当忘记当前的工作模式的时候，请立即按 Esc 键，即可立即返回命令模式。通过图 3-1 可以看出，命令模式是可以切换为编辑模式或者末行模式的，而编辑模式和末行模式之间没有切换路径，这就是以命令模式为核心的操作思路。

● 编辑模式练习

输入 i/a/o 任何一个字母，都可以进入编辑模式，然后自由输入字符。

输入 I/A/O 任何一个字母，也可以进入编辑模式，观察光标的位置是否有变化。

3.3.2 命令模式

命令模式又称普通模式，功能十分强大，涉及的操作命令很多，本书只介绍常用的操作命令。在此模式下，输入指令并不会在屏幕上显示，界面显示非常不友好，需要记忆一定数量的命令操作。

● 退出 vim

(1) ZZ: 保存退出。

(2) ZQ: 不保存退出。

● 光标跳转、字符间跳转

(1) h: 左。

(2) l: 右。

(3) j: 下。

(4) k: 上。

● 单词间跳转

(1) w: 跳转至下一个单词的词首。



(2) e: 跳转至当前或下一个单词的词尾。

(3) b: 跳转至当前或前一个单词的词首。

● 当前页跳转

(1) H: 跳转至页首。

(2) M: 跳转至页中间行。

(3) L: 跳转至页底。

(4) zt: 将光标所在当前行移至屏幕顶端。

(5) zz: 将光标所在当前行移至屏幕中间。

(6) zb: 将光标所在当前行移至屏幕底端。

● 行首行尾跳转

(1) ^: 跳转至行首的第一个非空白字符。

(2) O: 跳转至行首。

(3) \$: 跳转至行尾。

(4) h,l: 行间移动。

(5) #G: 跳转至第 # 行, 相当于扩展命令模式下的 :#。

(6) G: 跳转至最后一行。

(7) 1G, gg: 跳转至第一行。

● 字符编辑练习

字符编辑练习请参考表 3-2。

表 3-2 字符编辑表

选项	解释
x	删除光标处的字符
#x	删除光标处起始的 # 个字符
xp	将光标处的字符与其后面的字符交换
~	转换大小写
J	删除当前行后的换行符

● 替换练习

替换练习请参考表 3-3。

表 3-3 替换命令表

选项	解释
r	只替换光标所在处的一个字符
R	切换到 REPLACE 模式（末行出现 -- REPLACE -- 提示），按 Esc 回到命令模式

● 删除练习

可结合光标跳转字符实现范围内的删除操作。

删除练习请参考表 3-4。

表 3-4 删除命令表

选项	解释
d\$	删除到行尾
d^	删除到非空行首
d0	删除到行首
dw	删除到某个单词的结尾位置
db	删除到某个单词的开始位置
dd	剪切光标所在的行

● 复制练习

可结合光标跳转字符实现范围内的复制操作。使用 p 或者 P 粘贴复制的内容。复制练习请参考表 3-5。

表 3-5 复制命令表

选项	解释
y	v 命令选中文本后，用 y 进行复制
yy	复制当前行，然后用 p 进行粘贴
5yy	复制从当前行开始的 5 行
y\$	从当前位置复制到行尾
y0	从当前位置复制到行首

● 撤销修改

在命令模式下按字母 u 撤销修改。

3.3.3 末行模式

按冒号 “:” 可以进入末行模式（也称 Ex，或扩展命令模式），创建一个命令提示符 “:” 处于屏幕底部的左侧。末行模式可以实现存盘、退出、查找、替换等操作，还可以修改 vim 设置，如显示行号等。

● 存盘或退出练习

存盘或退出练习请参考表 3-6。

表 3-6 存盘或退出命令表

选项	解释
w	写（存）磁盘文件
wq	写入并退出
x	有修改存盘退出，无修改直接退出
q	退出
q!	不存盘退出，即时更改都将丢失



● 地址定界练习

地址定界练习请参考表 3-7。

表 3-7 地址定界表

选项	解释
#	# 表示行号，例如 2 表示第二行
#, #	第一个 # 表示起始行，第二个 # 表示结尾行，定义查找或替换的范围
\$	\$ 表示最后一行

● 查找练习

查找命令为末行模式下的操作，可以向前 / 向后查找。

查找练习请参考表 3-8。

表 3-8 查找命令表

选项	解释
/PATTERN	从当前光标所在处向文件尾部查找，查找下一个按字母 n
?PATTERN	从当前光标所在处向文件首部查找，查找下一个按字母 n

● 查找并替换练习

输入冒号“:”进入末行模式，按照格式 s/ 要查找的内容 / 替换后的内容 / 修饰符输入，修饰符参考表 3-9。

表 3-9 修饰符表

修饰符	解释
I	忽略大小写
g	全局替换，默认情况下，每一行只替换第一次出现的内容
gc	全局替换，每次替换前询问

配置 vim 为高效开发环境，启用语法高亮（支持 Python/JS 等多种语言）、自动缩进与行号显示，优化光标样式与搜索高亮，兼顾代码可读性与操作流畅性，适配前后端开发场景，有助于提升代码逻辑与编程效率。

● 显示或者取消行号练习

输入冒号“:”进入末行模式，参考表 3-10 进行练习。

表 3-10 行号设置表

选项	解释
set nu	显示行号，也可以用全拼 set number
set nonu	取消显示，也可以用全拼 set nonumber

● 显示或者取消语法高亮练习

输入冒号“:”进入末行模式，参考表 3-11 进行练习。

表 3-11 语法高亮设置表

选项	解释
syntax on	启用语法高亮
syntax off	禁用语法高亮

● 显示或者取消光标所在行的标示线练习

输入冒号“:”进入末行模式，参考表 3-12 进行练习。

注意: 标示线只支持在图形界面显示, 用字符终端登录操作 Linux 时, 无法显示标示线。

表 3-12 光标所在行的标示线设置表

选项	解释
set cursorline	启用光标所在行的标示线
set nocursorline	禁用光标所在行的标示线

当我们安装了图形界面，打开图形界面的终端程序，输入 vim 命令，设置显示标示线，如图 3-2 所示。

```
--  
-- Table structure for table `users`  
--  
  
DROP TABLE IF EXISTS `users`;  
/*!40101 SET @saved_cs_client      = @@character_set_client */;  
/*!50503 SET character_set_client = utf8mb4 */;  
:set cursorline
```

图 3-2 显示标示线

3.4

文本处理

Linux 凭借强大的工具链实现高效文本处理。grep 精准搜索匹配模式文本，sed 流式编辑替换内容，awk 按字段解析统计数据，三者组合可实现复杂过滤与格式化。此外，还有 cut 提取指定列数据，sort/uniq 排序去重，tr 转换字符编码等命令。配合管道符与重定向，可构建灵活的脚本“流水线”，实现批量处理日志、配置文件等，适用于系统管理、数据清洗、开发调试等场景，以简洁的命令实现文本的分析、转换与自动化处理。本节只针对 grep、sed 和 awk 命令进行学习。

3.4.1

文本处理之 grep

grep 命令源自短语 Global search REgular expression and Print out the line，该命令主要用于文本搜索。grep 命令根据用户指定的模式（pattern）对目标文本逐行进行匹配检查，打印匹配到的行。所谓模式，是由正则表达式字符及文本字符所编写的过滤条件。



语法：

```
grep [选项] [模式] [目标文件]
```

grep 命令选项如表 3-13 所示。

表 3-13 grep 命令选项

选项	解释
-v	显示不被模式匹配到的行
-i	忽略字符大小写
-n	显示匹配的行号
-c	统计匹配的行数
-A #	after, 后 # 行
-B #	before, 前 # 行
-C #	context, 前后各 # 行
-w	匹配整个单词
-f file	根据模式文件中的每一行来匹配目标文件中的行
-o	仅显示匹配到的字符串
-E	使用扩展正则表达式规则 ERE，相当于使用 egrep 命令

● 匹配 root 字符串，目标文件为 /etc/passwd

```
[root@localhost ~]# grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

● 匹配 nologin 字符串，显示不匹配的行，并显示行号，目标文件为 /etc/passwd

```
[root@localhost ~]# grep -n -v nologin /etc/passwd
1:root:x:0:0:root:/root:/bin/bash
6:sync:x:5:0:sync:/sbin:/bin/sync // 第一项为行号
7:shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
8:halt:x:7:0:halt:/sbin:/sbin/halt
28:tom:x:1000:1000:./home/tom:/bin/bash
```

● 显示匹配的行数，条件和上一操作相同

```
[root@localhost ~]# grep -c -v nologin /etc/passwd
5
[root@localhost ~]#
```

● 匹配 sync 字符串，显示行号，显示该行以及后面两行

```
[root@localhost ~]# grep -A 2 -n sync /etc/passwd
6:sync:x:5:0:sync:/sbin:/bin/sync
7-shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
```

```
8-halt:x:7:0:halt:/sbin:/sbin/halt
--
25:systemd-timesync:x:988:988:systemd Time Synchronization:/:usr/sbin/nologin
26-named:x:25:25:Named:/var/named:/sbin/nologin
27-apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
[root@localhost ~]#
```

● 匹配 sync 字符串，匹配完整单词，显示行号，显示该行以及前面两行

```
[root@localhost ~]# grep -B2 -n -w sync /etc/passwd
4-adm:x:3:4:adm:/var/adm:/sbin/nologin
5-lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
6:sync:x:5:0:sync:/sbin:/bin/sync
[root@localhost ~]#
```

● 使用文件进行字符串匹配

```
[root@localhost ~]# vi aaa
[root@localhost ~]# cat aaa // aaa 文件的行是需要匹配的字符串
root
sync
[root@localhost ~]# grep -f aaa /etc/passwd// 实现多个字符串匹配的操作
root:x:0:0:root:/root:/bin/bash
sync:x:5:0:sync:/sbin:/bin/sync
operator:x:11:0:operator:/root:/sbin/nologin
systemd-timesync:x:988:988:systemd Time Synchronization:/:usr/sbin/nologin
[root@localhost ~]#
```

正则表达式是一种用于描述字符串模式的强大工具，常用于文本处理和搜索操作。在 `grep`、`locate` 等命令中，正则表达式用于查找符合特定模式的文本行。正则表达式的规则包括三个方面：模式匹配字符、量词和控制字符。由于正则表达式中的规则使用了特殊字符，为了保证正则表达式的正确性，强烈建议使用单引号将正则表达式分隔出来。

模式匹配字符如表 3-14 所示。

表 3-14 模式匹配字符

模式匹配字符	解释
.	小数点，匹配任意单个字符
[列表]	匹配字符串列表内的任意单个字符
[范围]	匹配指定字符范围内的任意单个字符
[^ 列表]	匹配指定字符范围内，或者字符串列表内的任意单个字符



● 匹配文件中以 r 字符开头，以 t 字符结尾，长度为 4 的字符串

```
[root@localhost ~]# grep 'r..t' /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

● 匹配文件中以 a/b/c 任意一个字符开头，以 t 字符结尾，长度为 3 的字符串

```
[root@localhost ~]# grep '[abc].t' /etc/passwd
halt:x:7:0:halt:/sbin:/sbin/halt
saslauthd:x:998:76:Saslauthd user:/run/saslauthd:/sbin/nologin
[root@localhost ~]#
```

● 匹配文件中以 a ~ f 字符中任意一个字符开头，以 t 字符结尾，长度为 3 的字符串

```
[root@localhost ~]# grep '[a-f].t' /etc/passwd
halt:x:7:0:halt:/sbin:/sbin/halt
saslauthd:x:998:76:Saslauthd user:/run/saslauthd:/sbin/nologin
systemd-network:x:990:990:systemd Network Management:/:usr/sbin/nologin
systemd-timesync:x:988:988:systemd Time Synchronization:/:usr/sbin/nologin
tom:x:1000:1000:/:home/tom:/bin/bash
[root@localhost ~]#
```

● 匹配文件中不是以 a ~ w 字符中任意一个字符开头，以 t 字符结尾，长度为 3 的字符串

```
[root@localhost ~]# grep '[^a-w].t' /etc/passwd
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:sbin/nologin
unbound:x:997:996:Unbound DNS resolver:/etc/unbound:/sbin/nologin
systemd-network:x:990:990:systemd Network Management:/:usr/sbin/nologin
systemd-resolve:x:989:989:systemd Resolver:/:usr/sbin/nologin
systemd-timesync:x:988:988:systemd Time Synchronization:/:usr/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
```

量词如表 3-15 所示。

表 3-15 量词

量词	解释
*	匹配一个字符零次或者多次
\{n\}	匹配一个字符 n 次
\{n,\}	匹配一个字符至少 n 次
\{n,m\}	匹配一个字符 n~m 次

● 匹配字符 s 多次

```
[root@localhost ~]# vi aaa
```

```
[root@localhost ~]# cat aaa                                     // 添加几行，方便练习
root
sync
ss
sss
ssss
[root@localhost ~]# grep 's*' aaa
root
sync
ss
sss
ssss
```

● 匹配字符 s 连续 2 次

```
[root@localhost ~]# grep 's\{2\}' aaa
ss
sss
ssss
[root@localhost ~]#
```

● 匹配字符 s 连续 2 ~ 3 次

```
[root@localhost ~]# grep 's\{2,3\}' aaa
ss
sss
ssss
[root@localhost ~]#
```

● 匹配字符 s 至少 3 次

```
[root@localhost ~]# grep 's\{3,\}' aaa
sss
ssss
[root@localhost ~]#
```

控制字符如表 3-16 所示。

表 3-16 控制字符

控制字符	解释
^	只在每行开头处匹配
\$	只在每行结束处匹配
\	引用特殊字符



● 匹配每行只有 4 个字符的行

```
[root@localhost ~]# grep '^....$' aaa
root
sync
ssss
[root@localhost ~]#
```

● 使用 locate 正则表达式，查找文件名末尾为 2025 其后跟随至少 4 个数字的文件

```
[root@localhost ~]# locate -r '2025[0-9]\{4,\}$'
/var/log/btmp-20250904
/var/log/hawkey.log-20250904
/var/log/hawkey.log-20250913
/var/log/hawkey.log-20250914
/var/log/hawkey.log-20250922
/var/log/messages-20250915
/var/log/rpmpkgs-20250904
/var/log/rpmpkgs-20250913
/var/log/rpmpkgs-20250914
/var/log/rpmpkgs-20250922
/var/named/data/named.run-20250927
[root@localhost ~]#
```

3.4.2 文本处理之 sed

sed 即 Stream Editor，和 vi 不同，sed 是行编辑器。sed 是从文件或管道中读取一行，处理一行，输出一行；再读取一行，再处理一行，再输出一行，直到最后一行。每处理一行时，把当前处理的行存储在临时缓冲区中，称为模式空间（pattern space），接着用 sed 命令处理缓冲区中的内容，处理完成后，把缓冲区中的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。一次处理一行的设计模式使得 sed 性能很高。sed 在读取大文件时不会出现卡顿的现象。如果使用 vi 命令打开较大文件时，会出现明显卡顿，这是因为 vi 命令打开文件是先一次性将文件加载到内存中，然后再打开的。sed 就避免了这种情况，一行一行地处理，打开速度非常快，执行速度也很快。

sed 文本处理流程图如图 3-3 所示。

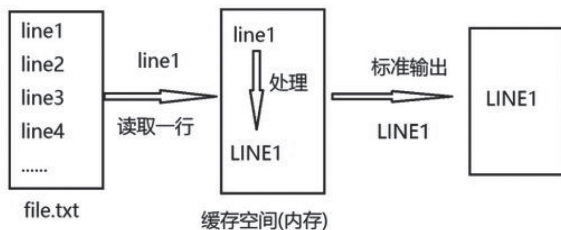


图 3-3 sed 文本处理流程图

语法：

```
sed [选项] '脚本；执行选项' [目标文件]
```

sed 命令选项如表 3-17 所示。

表 3-17 sed 命令选项

选项	解释
-n	不输出缓存空间内的内容到屏幕，推荐使用此选项，减少信息干扰
-e	多点编辑
-f FILE	从指定文件中读取编辑脚本
-r, -E	使用扩展正则表达式
-i .bak	备份文件并直接修改原文件，备份文件的扩展名为 bak； 若只有选项 -i，则直接修改原文件

sed 命令脚本选项如表 3-18 所示。

表 3-18 sed 命令脚本选项

脚本选项	解释
#	指定第 # 行
\$	最后一行
#, #	从 # 行到第 # 行
#, +#	从 # 行到 +# 行
/pattern/	斜杠 (/) 内 pattern 为正则表达式，按照此规则实现搜索匹配

sed 命令 executet 选项如表 3-19 所示。

表 3-19 sed 命令 executet 选项

执行选项	解释
p	打印当前模式空间内容，追加到默认输出之后
d	删除模式空间匹配的行，并立即启用下一轮循环
!	对模式空间中不匹配的行执行命令
a [\\]text	在指定行后添加 text 文本，支持使用 \\n 实现多行追加
i [\\]text	在行前插入文本
c [\\]text	替换行为单行或多行文本
w file	保存当前模式空间中匹配的行至指定文件
r file	读取指定文件的文本至模式空间中匹配到的行后
s/pattern/string/	查找替换，支持使用其他分隔符

executet 为执行选项，当 sed 命令搜索匹配到对应的字符串时，按照执行选项进行处理。

● 测试 -n 选项的作用

```
[root@localhost ~]# sed '/127.0.0.1/p' /etc/hosts
```



```
// 匹配 127.0.0.1 字符串
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
// 实现
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
// 此行符合条件要求，打印到屏幕
:::1    localhost localhost.localdomain localhost6 localhost6.localdomain6
[root@localhost ~]# sed -n '/127.0.0.1/p' /etc/hosts
// 添加 -n 选项，对比结果
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
// 只显示符合条件的行
[root@localhost ~]# cat /etc/hosts // 对比原文件
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
:::1    localhost localhost.localdomain localhost6 localhost6.localdomain6
[root@localhost ~]#
```

● 过滤空行

```
[root@localhost ~]# sed -n '/^$/p' anaconda-ks.cfg // 使用正则表达式
```

● 过滤 # 开头的注释行

```
[root@localhost ~]# sed -n '/^#/p' anaconda-ks.cfg
# Generated by Anaconda 36.16.5
# Generated by pykickstart v3.47
#version=DEVEL
# Use graphical install
# Keyboard layouts
# System language
# Use hard drive installation media
# Run the Setup Agent on first boot
# Generated using Blivet version 3.4.2
# Partition clearing information
# System timezone
# Root password
```

● 同时过滤空行和注释行

```
[root@localhost ~]# sed -n -e '/^#/p' -e '/^$/p' anaconda-ks.cfg
// 使用多点编辑选项 -e

# Generated by Anaconda 36.16.5
```

```
# Generated by pykickstart v3.47
#version=DEVEL
# Use graphical install

# Keyboard layouts
# System language

# Use hard drive installation media

# Run the Setup Agent on first boot

# Generated using Blivet version 3.4.2
# Partition clearing information

# System timezone

# Root password
```

● 过滤空行和注释行，把剩余的信息保留在原文件中，并做好备份

```
[root@localhost ~]# sed -n -r '/^(#|$)/!p' anaconda-ks.cfg
// 使用扩展正则表达式，把过滤后剩余行打印到屏幕上

graphical
keyboard --vckeymap=cn --xlayouts='cn'
lang zh_CN.UTF-8
harddrive --dir= --partition=LABEL=openEuler-22.03-LTS-SP4-x86_64
%packages
@^minimal-environment
%end
firstboot --enable
ignoredisk --only-use=sda
autopart
clearpart --none --initlabel
timezone Asia/Shanghai --utc
rootpw --iscrypted $y$j9T$riWawwVJ8yvJcUwYnHTCRqXn$8DzmBO.d3.5MjPw/
DiQNWJoOcF6q2y6Fehjowm0dSM1

[root@localhost ~]# sed -i.bak -n -r '/^(#|$)/!p' anaconda-ks.cfg
```



```
// 使用 -i.bak 选项

[root@localhost ~]# ll
总用量 12
-rw-----. 1 root root 375 10月 18 11:20 anaconda-ks.cfg
-rw-----. 1 root root 702 10月 18 11:19 anaconda-ks.cfg.bak
// 自动保留原文件内容
```

3.4.3 文本处理之 `awk`

`awk` 是一种处理文本的工具，其内部的命令集合可以实现强大的文本分析功能。它被认为是专门为文本处理设计的编程语言，也是文本文件的行处理软件，通常用于扫描、过滤、统计汇总工作。文本数据可以来自标准输入，也可以来自管道或磁盘文件。

`awk` 的名字取自三位创始人 Alfred Aho、Peter Weinberger 和 Brian Kernighan 的家族姓的首字母。`awk` 命令由 GNU/Linux 发布方自由软件基金会（FSF）进行开发和维护，通常也称为 GNU AWK。

语法：

```
awk [选项] 'pattern{处理命令}' [目标文件]
```

`awk` 命令常用选项如表 3-20 所示。

表 3-20 `awk` 命令常用选项表

选项	解释
<code>-F</code>	定义字段分隔符，默认分隔符是空格或制表符
<code>-v</code>	定义变量并赋值
<code>-f</code>	指定文件中的命令来处理文件

`pattern` 是可选项目，可以是正则表达式，也可以是关系表达式。当 `pattern` 缺省时，文本文件的每行都会被处理。当 `pattern` 存在时，则根据匹配的结果决定是否处理该行的文本。

处理命令部分包括形式规定和内置变量两方面。

1. 形式规定

首先是 `BEGIN` 部分，完成读取文件数据之前的操作，通常用于定义变量；接着是处理部分，通过命令对数据按照行的顺序进行分析和处理；最后是完成处理文本之后的操作。

`BEGIN{}`：读文件之前的动作，通常用于定义变量。

`{}`：处理文件，此种方式最为常用。

`END{}`：文件处理完毕之后的动作。

2. 内置变量

内置变量是已经预先定义的变量，可以直接使用，表 3-21 为部分常见的内置变量。

表 3-21 awk 命令内置变量

变量	含义
NR	行号 NR==1, NR<=5
NF	行的字段总数（注：也称列数）
FS	分隔符，默认为空格
OFS	定义输出字段分隔符，默认空格
RS	定义输出字段分隔符，默认空格
ORS	记录分隔符，默认换行
\$0	当前处理行的所有内容
\$1,\$2...	文件中每行以间隔符号分隔的不同字段

● 打印第一列内容，没有指定字段分隔符，默认是以空格分隔

```
[root@localhost ~]# awk '{print $1}' /etc/hosts
127.0.0.1
::1
```

● 打印第一列内容，没有指定字段分隔符，默认是以空格分隔

```
[root@localhost ~]# awk -F: '{print $1}' /etc/passwd
root
bin
daemon
```

● 以冒号分隔，打印第一列和第二列内容

```
[root@localhost ~]# awk -F: '{print $1,$2}' /etc/passwd
root x
bin x
daemon x
adm x
```

● 使用 BEGIN{}.....END{} 结构

处理分为三段：处理之前的操作→行处理操作→处理完毕之后的操作，不一定三段都出现。若增加处理条件，处理条件要写在第二步行处理之前。

```
[root@localhost ~]# awk 'BEGIN{FS=":"} $3<5 {print $1} END {print NR}' /etc/passwd
root // 第一段定义字段间隔符号
bin // 条件为 uid 小于 5 的行
daemon // 输出用户名
adm
lp
28 // 处理完毕显示文件总行数
```



```
[root@localhost ~]#
```

- 查找用户名为 root 的账户使用的 shell 程序

```
[root@localhost ~]# awk -F: '/^root/{ print $7}' /etc/passwd  
/bin/bash
```

- 打印第七列等于字符串 "/bin/bash" 的行

```
[root@localhost ~]# awk -F: '$7=="/bin/bash"' /etc/passwd  
root:x:0:0:root:/root:/bin/bash
```

- 打印每一行和每一行的行号，中间用空格分隔，两个文件 NR 行号是累加的

```
[root@localhost ~]# awk '{print NR,$0}' /etc/passwd /etc/hosts  
1 root:x:0:0:root:/root:/bin/bash  
2 bin:x:1:1:bin:/bin:/sbin/nologin  
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin  
4 adm:x:3:4:adm:/var/adm:/sbin/nologin  
5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin  
6 sync:x:5:0:sync:/sbin:/bin/sync  
7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown  
8 halt:x:7:0:halt:/sbin:/sbin/halt  
9 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin  
10 operator:x:11:0:operator:/root:/sbin/nologin  
11 127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
12 ::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
```

本章练习题

1 以下哪款文本编辑器因其几乎在所有 Unix/Linux 系统上都默认安装而被称为“通用”编辑器？（ ）

- A. gedit B. kedit C. vi D. VS Code

2 在 vim 中，如何从“命令模式”切换到“插入模式”以开始编辑文本？（ ）

- A. 按 : 键 B. 按 i 键 C. 按 Esc 键 D. 按 / 键

3 在 vim 的命令模式下，执行哪条命令可以保存文件并退出？（ ）

- A. :q! B. :w C. :wq D. :z

4 使用 grep 命令在文件 log.txt 中查找包含单词“error”的行，并且忽略大小写，应该使用哪个命令？（ ）

- A. `grep error log.txt`
- C. `grep -v error log.txt`

- B. `grep -i error log.txt`
- D. `grep -n error log.txt`

5 `sed` 命令的核心工作流程是（ ）。

- A. 一次性将整个文件加载到内存中进行处理
- B. 逐行读取、处理、输出，再处理下一行
- C. 主要用于交互式文本编辑
- D. 只能用于替换文本，不能执行其他操作

6 在 `vim` 中，在命令模式下按下 `dd` 会执行什么操作？（ ）

- A. 删除当前光标所在处的字符
- B. 复制当前行
- C. 剪切（删除）当前行
- D. 粘贴剪贴板内容

7 `awk` 在处理 `/etc/passwd` 文件时，正确指定字段分隔符的命令是（ ）。

- A. `awk -F: '{print $1}' /etc/passwd`
- B. `awk -f: '{print $1}' /etc/passwd`
- C. `awk '/:/ '{print $1}' /etc/passwd`
- D. `awk '{print $1}' /etc/passwd`

8 如果要在 `vim` 中显示行号，应该在扩展命令模式下输入（ ）命令。

- A. `set cursorline`
- B. `syntax on`
- C. `set nonu`
- D. `set nu`

9 `grep` 命令的 `-v` 选项的作用是什么？（ ）

- A. 显示匹配行的行号
- B. 忽略匹配时的大小写差异
- C. 显示不被模式匹配到的行（反向选择）
- D. 只输出匹配到的字符串部分

10 在 `vim` 的命令模式下，按下 `$` 键可以实现什么功能？（ ）

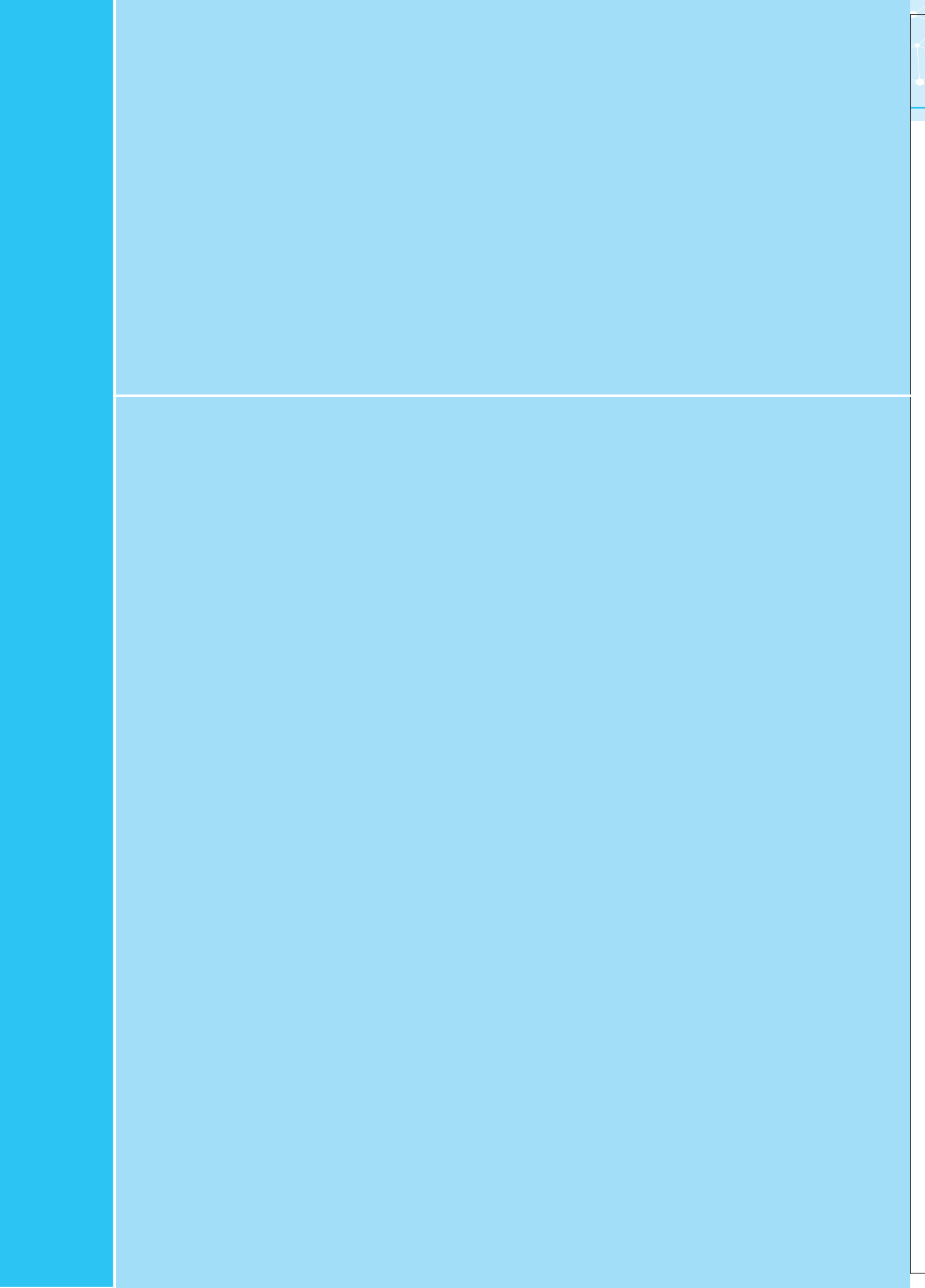
- A. 跳转到当前行的行首
- B. 跳转到当前行的行尾
- C. 跳转到文件的第一行
- D. 跳转到文件的最后一行

► 第 4 章 用户和权限管理

在 Linux 系统中，用户用于区分不同的使用者。每个用户拥有独立的权限，可以访问特定的文件和资源。组则是将用户进行分类管理的机制，方便为一组用户统一分配权限，如一个项目组的成员共享对项目文件的读写权限。

本章的学习要点如下。

- (1) 了解用户和组的基本概念。
- (2) 了解用户和组的管理和配置信息。
- (3) 掌握用户管理的基本命令。
- (4) 掌握组管理的基本命令。
- (5) 掌握权限管理的基本命令。



4.1 管理用户和组

Linux 是一个多用户的操作系统，所有要使用系统资源的用户需要先向系统管理员申请一个账号，之后以此账号进入系统，可以在系统上建立多个用户，而多个用户可以在同一时间登录至同一系统执行不同的任务，并不会相互影响。

4.1.1 用户和组的概念

用户的概念可以理解能够为获取系统资源的权限的集合，用户意味着权限。在 Linux 系统中，每个用户都会被分配一个唯一识别号，英文为 UID（全称为 User ID）。关于用户的概念，我们分以下几点进行讲解。

（1）用户的 UID 标识唯一一个用户，UID 是系统使用的一个数字身份号，类似于中国社会的身份证号码。通过 UID 可以区分不同用户的类别。注意：用户在登录系统时是通过 UID 来区分用户的，而不是通过用户名。

（2）超级用户：也称为 root 用户，它的 UID 为 0，超级用户拥有系统的完全控制权，在 Linux 中这个账号可以进行修改、删除文件等操作，也可以运行各种程序。当使用 root 用户时，操作人员一定要十分谨慎。

（3）普通用户：也称为一般用户，它的 UID 为 1000 ~ 60000，普通用户可以对自己目录下的文件进行访问和修改，也可以对经过授权的文件进行访问。这种用户权限是有限且可控的。

（4）系统用户，它的 UID 为 1 ~ 999，系统用户往往也有较高的系统使用权限或者配置权限，也要谨慎使用。一般来说，系统用户不提供密码登录功能，它们的存在主要是为了方便系统的管理。例如，用户可以用系统用户身份来运行某些程序。所以，有时候我们也把系统用户称为虚拟用户。

通过命令，我们可以查看不同用户的 UID，通过 UID 的数值可以区分用户的类别为超级用户、普通用户或系统用户。

Linux 系统中的用户组也是一个重要的概念。多个用户可以归类到一个组中，每个用户组都有自己的权限设置，便于统一管理文件和资源的访问权限。组的概念可以简化权限管理，提高系统安全性，同时方便团队协作。例如，一个项目团队的所有用户可以被分配到同一个用户组，共享项目相关文件资源。

用户对应有 UID，组对应也有一个 ID，叫作 GID，即组 ID。GID 也是用整数来表达的，取值范围和 UID 类似。用户可以属于多个组，但是默认组有且只有一个，其余的组称为附加组，附加组可以添加多个。例如，一个大学生在入学登记时，必须被分配到一个专业班，专业班相当于默认组的概念；如果该学生又选修了某公共课程，那么他就相当于添加了一个附加组，进入附加组来参与某公共课程的班级学习活动。

总结一下，用户和组的关系是多样的。

（1）一对一：一个用户只属于一个默认组。



- (2) 一对多：一个用户可以加入一个或多个附加组，该用户具有多个组的权限。
- (3) 多对一：多个用户属于一个组，这些用户具有相同的组权限。
- (4) 多对多：多个用户可以同时存在于多个用户组中，实际上就是以上三种关系的扩展。

语法：

```
id [option] [user_name]
```

id 命令是显示用户身份（UID）和组身份（GID）的命令，用于检查用户的身份类别，掌握用户身份权限，涉及安全上下文信息。

id 命令选项如表 4-1 所示。

表 4-1 id 命令选项

选项	解释
-u	只显示 UID
-g	只显示 GID
-Z	安全上下文信息

● 查看用户身份

```
[root@localhost ~]# id
用户 id=0(root) 组 id=0(root) 组 =0(root) 上下文 =unconfined_u:unconfined_
r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]# id -u
0
[root@localhost ~]# id -g
0
[root@localhost ~]# id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

4.1.2 用户和组的配置文件

在 Linux 系统中，用户和组的关联文件有 `/etc/passwd` 和 `/etc/group`。`/etc/passwd` 文件存储用户信息，包括用户名、用户 ID、用户组 ID、用户主目录（也称为家目录）等。`/etc/group` 文件则存储组信息，包括组名、组 ID 和所属附加组用户列表。这两个文件共同管理用户和组的关系，确保系统权限分配和资源管理的正确性。

`/etc/passwd`：作为用户账号文件，每一行代表着一个记录，每一行用户记录中定义了用户各个方面的相关属性。相关属性由七个字段的数据组成，且字段之间用“:”隔开，字段的内容如图 4-1 所示。

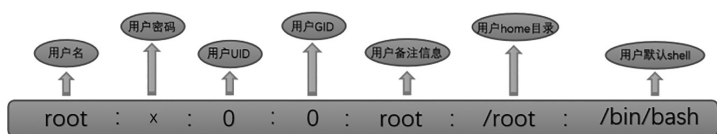


图 4-1 用户账号文件的字段

我们可以看到,第一个字段为用户名,它是用户登录操作系统时使用的名字,非常重要。然而,操作系统管理用户和组时,实际上使用的是第三和第四个字段,即使用 UID 和 GID 来管理用户和组。用户账号文件的每个字段的详细说明如表 4-2 所示。

表 4-2 用户账号文件字段说明表

字段序号	字段含义
1	用户名,用户名字符可以是大小写字母、数字、减号、点以及下划线,其他符号不合法
2	用户密码,在此字段用“x”表示,具体密码需要到对应的 <code>/etc/shadow</code> 文件中查看
3	用户 UID,用于对用户进行识别,从而判断用户类型
4	用户 GID,对应 <code>/etc/group</code> 中的组信息,将用户分组管理
5	用户备注信息,例如会注释用户的真实姓名、地址等信息,该信息为一般信息,不重要
6	用户 home 目录,即用户登录时所处目录
7	用户默认 shell,用户登录时运行的程序,默认的 shell 程序将输入命令提交给操作系统内核运行。如果不允许用户登录,则可以选择 <code>nologin</code> 程序,拒绝用户登录

● 查看用户账号文件

```
[root@localhost ~]# head /etc/passwd           // 对比表格字段说明,学习文件格式
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
```

`/etc/shadow` 文件是用户账号密码文件(又称“影子文件”),用于存储系统中用户的密码信息。`/etc/passwd` 文件允许所有用户读取,容易导致密码泄露,因此将密码信息从该文件中分离出来,单独放置在 `/etc/shadow` 文件中。

`/etc/shadow` 文件只有超级用户(`root` 用户)具有读取权限,从而保证了用户密码的安全性。密码经 `/etc/shadow` 保护后,在 `/etc/passwd` 文件的用户记录中只会以“x”的形式呈现。



与 `/etc/passwd` 文件类似，每一行记录代表一个用户，且以 “:” 隔开，不同之处在于，`/etc/shadow` 文件中每行记录被分为九个字段。

观察图 4-2 可以看到，从第三个字段开始，一直到第八个字段都和时间的有关。这些时间参数对于密码的安全有重大意义。但也是这些时间的设置，常常导致人们使用密码感到不便。例如，若密码的有效期限设置得较短，会导致系统频繁要求用户更换密码。合理地设置密码参数有利于系统的维护，表 4-3 为 `/etc/shadow` 文件中各个字段的详细说明。

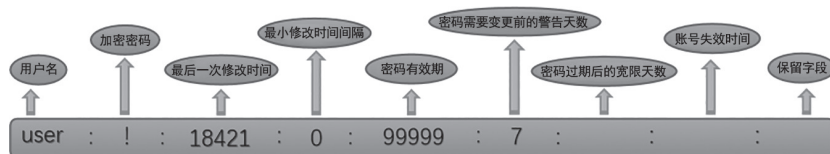


图 4-2 用户账号加密文件的字段

表 4-3 `/etc/shadow` 文件字段说明

字段序号	字段含义
1	用户名，与文件 <code>/etc/passwd</code> 中的用户名具有相同的意义
2	加密密码，若该字段第一个字符显示的为 “*” 或 “!”，说明该账户为禁止登录账户
3	此字段表示最后一次修改密码的时间
4	最小修改时间间隔，如果设置为 0，则随时可进行密码修改
5	密码有效期，强制用户定期修改密码，提高系统的安全性
6	密码需要更改前的警告天数，当用户密码快过期时，发出警告信息提醒用户进行密码修改
7	密码过期后宽限天数，若是在密码过期后设置的宽限天数内仍未修改密码，则禁用该用户
8	用户失效时间，失效后该用户将不再是一个合法用户，无法登录系统
9	保留字段，目前为空，以备将来扩展

● 查看用户密码文件

```
[root@localhost ~]# head /etc/shadow // 对比表格字段说明，学习文件格式
root:$y$j9T$riWawwVJ8yvJcUwYnHTCRqXn$8DzmBO.d3.5MjPw/DiQNWJoOcF6q2y6Fehj
owm0dSM1::0:99999:7:::
bin:*:19894:0:99999:7:::
daemon:*:19894:0:99999:7:::
adm:*:19894:0:99999:7:::
lp:*:19894:0:99999:7:::
sync:*:19894:0:99999:7:::
shutdown:*:19894:0:99999:7:::
halt:*:19894:0:99999:7:::
```

```
mail:*:19894:0:99999:7:::
operator:*:19894:0:99999:7:::
```

Linux 系统中的 `/etc/group` 文件是一个用于定义用户组信息的核心配置文件，它详细记录了系统中所有用户组的名称、密码（注：通常仅为占位符“x”，实际密码被搬到 `gshadow` 文件）、组标识号以及属于该组的用户列表。该文件中的每一行都代表一个独立的用户组，各个字段之间使用冒号分隔，其基本格式为

组名: 密码 :GID: 用户列表

具体来说，“组名”是用户组的唯一标识；“密码”字段在现代 Linux 系统中通常用“x”或留空表示，真正的组密码（现已较少使用）会存储在专门的 `/etc/gshadow` 文件中；GID 是系统用于识别不同组的唯一数字 ID；而“用户列表”则是一个用逗号分隔的用户名集合，列出了哪些用户是该组的附加成员。

● 查看组文件

```
[root@localhost ~]# tail /etc/group
printadmin:x:993:
chrony:x:992:
sgx:x:991:
systemd-network:x:990:
systemd-resolve:x:989:
systemd-timesync:x:988:
named:x:25:
apache:x:48:
soft:x:1000:wyk
wyk:x:1001:
```

// wyk 为 soft 的附加组成员

`/etc/gshadow` 文件为 Linux 组的密码文件，它与 `/etc/group` 文件相互补充，专门负责管理用户组的密码、组管理员以及组成员关系等加密信息。该文件默认只有 root 用户有读取权限，每一行对应一个用户组的记录，各记录均以冒号分隔为四个字段。第一个字段是用户组的名称，与 `/etc/group` 文件中的组名完全对应；第二个字段存储的是用户组的加密密码，该字段可能为空、包含感叹号 (!) 或有效的加密字符串，若为空则表示该组未设密码且仅组成员可使用 `newgrp` 命令切换至该组，若为感叹号 (! 或 !!) 则表示组密码已禁用，不允许非组成员通过密码加入；第三个字段列出了组管理员的用户名，组管理员有权通过 `gpasswd` 命令添加或移除该组的普通成员，从而分担 root 用户的管理负担，实际上该字段很少使用；第四个字段则显示该组的附加组成员列表，多个成员间用逗号分隔，其内容通常与 `/etc/group` 文件中相应组的成员列表一致。

● 查看组的密码文件

```
[root@localhost ~]# tail /etc/gshadow
printadmin!:::
chrony!:::
```



```
sgx:!*::
systemd-network:!*::
systemd-resolve:!*::
systemd-timesync:!*::
named:!::
apache:!::
soft:$6$5.5ykuF28CYRfjZP$rPql.8BVnK5mums4lo6ixRTylH65/jO1IFQgGzn.4cgT0uW
qHv3VV1jcgHP6csBx0uPpWCGC/92fRMHaJ43/C/::wyk
// soft 组添加了组密码，wyk 为该组附加组成员，与 group 文件一致
wyk:!::
```

4.2

用户的管理

在 Linux 系统中，创建账户时，会给每个普通用户一个账号，配置内容主要包括用户名、密码和主目录等信息。Linux 系统中的一些特殊用户，具有特殊的存在意义，其中最重要的用户就是管理员账户，默认的用户名为 root（即超级用户）。我们必须以 root 身份登录，才有权限通过操作命令实现创建用户、配置用户参数和删除用户等操作。这些操作的本质，就是对用户账号文件和用户账号加密文件对应记录进行增加、修改、删除等操作。为什么我们不直接编辑用户账号文件和用户账号加密文件来管理用户呢？大家可以带着这个问题，一边学习一边思考。

下面我们学习用户管理相关命令。

1. 创建用户 - useradd

useradd 命令可以用来创建用户账号，并保存在 /etc/passwd 和 /etc/shadow 文件中。

语法：

```
useradd [options] user_name
```

useradd 命令选项如表 4-4 所示。

表 4-4 useradd 命令选项

选项	解释
-u	指定用户 UID
-o	配合 -u 选项，允许 UID 重复
-g	指明用户所属基本组，既可为用户组名，也可为 GID（该组必须已存在）
-d	指定用户的 home 目录，并自动创建用户 home 目录
-s	指明用户的默认 shell 程序
-D	显示或更改默认配置

- 创建一个测试用户 user，通过 useradd 命令创建一个新用户

```
[root@localhost ~]# useradd user
```

- 创建一个用户 user1 并指定一个组名为 user

```
[root@localhost ~]# useradd user1 -g user          // -g 选项指定组名
useradd: user 'user1' already exists
```

2. 修改用户 - usermod

usermod 可以用来修改用户账号的各类信息，通过选项来指定修改的字段。

语法：

```
usermod [options] user_name
```

usermod 命令选项如表 4-5 所示。

表 4-5 usermod 命令选项

选项	解释
-u	修改用户 UID
-g	修改用户所属用户组
-l	修改用户账号名称
-d	修改用户 home 目录
-s	修改用户默认 shell 程序

- 修改用户 user 的 UID

使用 usermod 命令及 -u 选项修改 user 的 UID，并查看 user 的 ID 变化。

```
[root@localhost ~]# id user
uid=1001(user) gid=1001(user) groups=1001(user)
[root@localhost ~]# usermod -u 1002 user          // 使用 usermod 命令及 -u 选项修改 UID
[root@localhost ~]# id user
uid=1002(user) gid=1001(user) groups=1001(user)
```

- 修改 user 用户的账号名称为 user2

```
[root@localhost ~]# id user
uid=1002(user) gid=1001(user) groups=1001(user)
[root@localhost ~]# usermod user -l user2        // 使用 usermod 命令及 -l 选项修改用户名
[root@localhost ~]# id user2
uid=1002(user2) gid=1001(user) groups=1001(user)
```

- 修改用户 user2 的所属组为 root

使用 usermod 命令加 -g 选项修改 user2 的所属组为 root，并查看 user 的变化。

```
[root@localhost ~]# id user2
uid=1002(user2) gid=1001(user) groups=1001(user)
[root@localhost ~]# usermod -g root user2
// 使用 usermod 命令及 -g 选项修改用户默认组
```



```
[root@localhost ~]# id user2
uid=1002(user2) gid=0(root) groups=0(root)
```

3. 删除用户 - userdel

userdel 用于删除指定的用户以及与该用户相关的配置文件的记录。

语法：

```
userdel [options] user_name
```

userdel 命令选项如表 4-6 所示。

表 4-6 userdel 命令选项

选项	解释
-f	强制删除用户账号，即使用户当前处于登录状态
-r	删除用户，同时删除与用户相关的所有文件
-h	显示命令的帮助信息

● 删除 user2 用户

```
[root@localhost ~]# userdel user2
[root@localhost ~]# id user2
id: user2: no such user
```

4. 修改用户密码 - passwd

passwd 用于修改用户的密码，root 用户可以修改任何用户的密码，普通用户只能用该命令修改自身的密码。

语法：

```
passwd [OPTION...] user_name
```

passwd 命令选项如表 4-7 所示。

表 4-7 passwd 命令选项

选项	解释
-n	设置修改密码最短天数
-x	设置修改密码最长天数
-w	设置用户在密码失效前多少天收到警告信息
-i	设置密码过期多少天后禁用账户
-d	删除用户密码
-s	显示用户密码信息

● 查看一个测试用户的密码：通过查看 /etc/shadow 文件的密码字段，发现未设置密码，显示为“！”

```
[root@localhost ~]# cat /etc/passwd
```

```
user!!:18421:0:99999:7:::
```

```
[root@localhost ~]# passwd user // 如果没有 user 用户,可重新创建一个
```

```
Changing password for user user.
```

```
New password:
```

```
Retype new password:
```

```
passwd: all authentication tokens updated successfully.
```

```
[root@localhost ~]# cat /etc/shadow // 观察密码字段没有感叹号了
```

```
user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyasYi8TG8XFNUdYUdeMd60lR2hvRC6zx3dAd
bEqQcnQuDoWT7ocu3Ss.zzWSrEb6cZ6Ae6b2EN/:18421:0:99999:7:::
```

● 设置一个测试用户的密码有效期为 7 天

```
[root@localhost ~]# cat /etc/shadow
```

```
user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyasYi8TG8XFNUdYUdeMd60lR2hvRC6zx3dAd
bEqQcnQuDoWT7ocu3Ss.zzWSrEb6cZ6Ae6b2EN/:18421:0:99999:7:::
```

```
[root@localhost ~]# passwd -x 7 user // 使用选项 -x 修改
```

```
[root@localhost ~]# cat /etc/shadow
```

```
user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyasYi8TG8XFNUdYUdeMd60lR2hvRC6zx3dAd
bEqQcnQuDoWT7ocu3Ss.zzWSrEb6cZ6Ae6b2EN/:18421:0:7:7:::
```

● 设置一个测试用户的密码失效前 10 天收到警告

```
[root@localhost ~]# cat /etc/shadow
```

```
user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyasYi8TG8XFNUdYUdeMd60lR2hvRC6zx3dAd
bEqQcnQuDoWT7ocu3Ss.zzWSrEb6cZ6Ae6b2EN/:18421:0:7:7:::
```

```
[root@localhost ~]# passwd -w 10 user // 使用选项 -w 修改
```

```
[root@localhost ~]# cat /etc/shadow
```

```
user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyasYi8TG8XFNUdYUdeMd60lR2hvRC6zx3dAd
bEqQcnQuDoWT7ocu3Ss.zzWSrEb6cZ6Ae6b2EN/:18421:0:7:10:::
```

● 设置一个测试用户的密码失效 3 天后禁用账户

```
[root@localhost ~]# cat /etc/shadow
```

```
user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyasYi8TG8XFNUdYUdeMd60lR2hvRC6zx3dAd
bEqQcnQuDoWT7ocu3Ss.zzWSrEb6cZ6Ae6b2EN/:18421:0:7:10:::
```

```
[root@localhost ~]# passwd -i 3 user // 使用选项 -i 修改
```

```
[root@localhost ~]# cat /etc/shadow
```

```
user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyasYi8TG8XFNUdYUdeMd60lR2hvRC6zx3dAd
bEqQcnQuDoWT7ocu3Ss.zzWSrEb6cZ6Ae6b2EN/:18421:0:7:10:3:::
```



● 删除一个测试用户的密码

```
[root@localhost ~]# cat /etc/shadow
user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyasYi8TG8XFNUdYUdeMd601R2hvRC6zx3dAd
bEqQcnQuDoWT7ocu3Ss.zzWSrEb6cZ6Ae6b2EN/:18421:0:7:10:3::

[root@localhost ~]# passwd -d user           // 使用选项 -d 删除用户密码
[root@localhost ~]# cat /etc/shadow
user::18421:0:7:10:3::
```

4.3

组的管理

组管理是系统安全管理的重要组成部分，通过组管理命令能够管理组的各种设置，以及切换用户组的身份。用户切换了组身份，对文件的访问权限也会发生变化。

1. 创建组 - groupadd

groupadd 可以用来创建一个新的用户组，并将新用户组信息添加到系统文件中。

语法：

```
groupadd [options] group_name
```

groupadd 命令选项如表 4-8 所示。

表 4-8 groupadd 命令选项

选项	解释
-f	如果组已存在，则成功退出
-g	为新用户组所使用的 GID
-h	显示此帮助信息并退出
-r	创建一个系统账户
-o	允许创建有重复 GID 的组

● 创建一个测试用户组

通过 groupadd 创建一个新的测试用户组：groupadd usergroup。

```
[root@localhost ~]# groupadd usergroup
[root@localhost ~]# cat /etc/group
test04:x:1003:
user:x:1004:
usergroup:x:1005:                                // 增加的记录
```

2. 修改组 - groupmod

groupmod 可以用来更改组名或者 GID。

语法：

```
groupmod [options] group_name
```

groupmod 命令选项如表 4-9 所示。

表 4-9 groupmod 命令选项

选项	解释
-g	修改为要使用的 GID
-h	显示此帮助信息并退出
-n	修改为要使用的组名称
-o	允许创建有重复 GID 的组

● 修改一个测试用户组的 GID

```
[root@localhost ~]# id user
uid=1002(user) gid=1004(user) groups=1004(user)
[root@localhost ~]# groupmod -g 1006 user           // 使用 -g 选项修改
[root@localhost ~]# id user
uid=1002(user) gid=1006(user) groups=1006(user)
```

● 修改一个测试用户组的名称

```
[root@localhost ~]# id user
uid=1002(user) gid=1004(user) groups=1004(user)
[root@localhost ~]# groupmod -n group user          // 组名改为 group
uid=1002(user) gid=1004(group) groups=1004(group)
```

3. 删除组 -groupdel

groupdel 可以用来删除组。

提示：若是组中已有一些用户，需先删除用户账号后再删除组。

语法：

```
groupdel [options] group_name
```

groupdel 命令选项如表 4-10 所示。

表 4-10 groupdel 命令选项

选项	解释
-f	即便是用户的主组也执行删除
-h	显示此帮助信息并退出

● 删除一个测试用户组

```
[root@localhost ~]# cat /etc/group
```



```
test04:x:1003:
user:x:1006:
usergroup:x:1005:                                // 检查 usergroup 组存在
[root@localhost ~]# groupdel usergroup           // 删除
[root@localhost ~]# cat /etc/group               // 检查后，确认已经删除
test04:x:1003:
user:x:1006:
```

4. 切换用户身份 -su

语法：

```
su user_name
```

提示：实际操作中 su 命令常常使用选项“-”，它的作用使得切换身份更加彻底，包括用户主目录也会切换过来。强烈推荐操作时带上选项“-”。

● 测试操作员切换用户身份

```
[root@localhost ~]# useradd jerry                // 添加账户 jerry
[root@localhost ~]# passwd jerry                 // 设置密码
更改用户 jerry 的密码。
新的密码：
无效的密码： 密码少于 8 个字符
重新输入新的密码：
passwd: 所有的身份验证令牌已经成功更新。
[root@localhost ~]# su jerry                     // 切换到用户 jerry 帐号
... ..                                          // 省略
System load:    0.04
Processes:      144
Memory used:    12.5%
Swap used:      0%
Usage On:       11%
IP address:     192.168.61.139
IP address:     192.168.85.199
Users online:   1
To run a command as administrator(user "root"),use "sudo <command>".
[jerry@localhost root]$ id                      // 检查用户 UID
用户 id=1003(jerry) 组 id=1003(jerry) 组=1003(jerry) 上下文=unconfined_
u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

5. 切换组 newgrp

提示：首先要给一个用户添加一个附加组，然后再做切换组的操作。

语法：

```
newgrp group_name
```

● 测试用户添加附加组

```
[root@localhost ~]# usermod -G root jerry
[root@localhost ~]# su - jerry                                // 切换到用户 jerry 帐号
... ..                                                      // 省略
Processes:          138
Memory used:        12.7%
Swap used:          0%
Usage On:           11%
IP address:         192.168.61.139
IP address:         192.168.85.199
Users online:       1
To run a command as administrator(user "root"),use "sudo <command>".
[jerry@localhost ~]$ id    // 组的信息包含两个组 jerry 和 root, 当前组为 jerry
用户id=1003(jerry) 组id=1003(jerry) 组=1003(jerry),0(root) 上下文=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[jerry@localhost ~]$
```

● 测试用户切换到附加组

```
[jerry@localhost ~]$ newgrp root                                // 切换到 root 组
... ..                                                      // 省略
Processes:          140
Memory used:        12.6%
Swap used:          0%
Usage On:           11%
IP address:         192.168.61.139
IP address:         192.168.85.199
Users online:       1
To run a command as administrator(user "root"),use "sudo <command>".
[jerry@localhost ~]$ id    // 组的信息包含两个组 jerry 和 root, 当前组为 root
用户id=1003(jerry) 组id=0(root) 组=0(root),1003(jerry) 上下文=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[jerry@localhost ~]$
```



4.4

文件权限的管理

权限是操作系统用来限制资源访问的一种机制，权限一般分为读、写、执行。在 Linux 系统中，不同的用户所处的地位不尽相同，不同地位的用户拥有不同的权限等级，为了保证系统的安全性，Linux 系统针对不同用户的权限制定了不同的规则。

在 Linux 系统中，每个文件或目录都具有特定的访问权限、所属用户及所属组，通过这些规则可以限制什么用户、什么组可以对特定的文件执行什么样的操作。

● 查看文件权限操作

```
[root@localhost ~]# touch aaa
[root@localhost ~]# ls -l aaa          // 长格式列表，文件第一列信息中包含权限信息
-rw-r--r--. 1 root root 0   9月  2 18:06 aaa
[root@localhost ~]#
```

文件以长格式列表显示后，可以看到第一列的第一个字符是“-”，代表普通文件，从第二个字符开始，连续九个字符是权限信息的内容。长格式列表字段内容如图 4-3 所示。

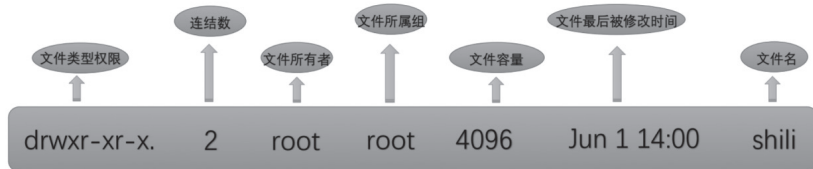


图 4-3 文件长格式列表字段内容

4.4.1

文件类型

Linux 有七种文件类型，如表 4-11 所示。

表 4-11 Linux 文件类型

文件类型	解释
-	普通文件，除去其他六种类型文件
d	目录
b	块设备文件，可随机存取装置间
c	字符设备文件，键盘、鼠标等一次性读取装置
l	符号链接文件，指向另一文件（link file）
p	命名管道文件（pipe）
s	套接字文件（socket）

4.4.2

权限位说明

Linux 文件或目录的权限位是由九个权限字段来控制的，每三位为一组，都是 r、w、

x 三个参数的组合，共三组，第一组是属主权限，第二组是属组权限，第三组是其他用户权限。其中 r 代表的是读权限（read），允许读取文件或读取目录下全部文件列表；w 代表的是写权限（write），允许写文件或在目录下创建、删除文件；x 代表的是执行权限（execute），允许执行文件或进入目录。三个权限位的位置在文件或目录中是不变的。若是没有对应的权限，用“-”表示。

图 4-4 所示为文件或目录权限字段。

文件类型	属主权限	属组权限	其他用户权限
0	1 2 3	4 5 6	7 8 9
d	rwx	r-x	r-x
目录文件	读 写 执行	读 写 执行	读 写 执行

图 4-4 文件或目录权限字段

访问权限也可以用数值来表示，假设一个文件的权限位的内容如下：

```
rwxr-xr-x
```

每一组的权限可以对应为二进制数值，1 代表拥有权限，0 代表没有权限。可在命令中直接实现二进制数值到八进制数值的转换。因此，我们要掌握好权限的数值表达。表 4-12 为上述文件或目录权限转换为数值表达的详细记录表，供大家参考。

表 4-12 文件或目录权限转换为数值表达

位置	权限代号	对应二进制	对应八进制	权限详情
第 123 位	rwX	111	4+2+1=7	文件属主可读可写可执行
第 456 位	r-X	101	4+1=5	同组用户可读不可写可执行
第 789 位	r-X	101	4+1=5	其他组用户可读不可写可执行

根据表 4-12 所表达的权限转换结果，文件权限用数值表达的结果应该是：755。

4.4.3 访问权限操作命令

访问权限相关的操作命令有 chmod、chown 和 chgrp，chmod 任何用户都可以使用，chown 和 chgrp 只有 root 用户才能使用。

1. chmod

文件调用权限针对文件所有者、所属组以及其他用户，使用 chmod 可以对文件调用权限进行修改。

语法：

```
chmod MODE[,MODE]... FILE...
```

MODE 操作模式的内容包括操作对象、操作符和赋予权限三个部分，其内容如表 4-13 所示。



表 4-13 chmod 命令操作模式内容

操作对象
u: 用户 user, 表示文件或目录所有者
g: 用户组 group, 表示文件或目录所属组
o: 其他用户 other, 表示其他组
a: 所有用户 all
操作符
+: 添加权限
-: 减少权限
=: 给特定权限
赋予权限
r: 读权限
w: 写权限
x: 执行权限

● 同时修改测试文件的权限，查看修改权限的结果

```
[root@localhost ~]# touch bbb
[root@localhost ~]# ls -l bbb
-rw-r--r--. 1 root root 0  9月  2 18:59 bbb
[root@localhost ~]# chmod g+w bbb
[root@localhost ~]# ls -l bbb
-rw-rw-r--. 1 root root 0  9月  2 18:59 bbb
[root@localhost ~]# chmod a-r bbb
[root@localhost ~]# ls -l bbb
--w--w----. 1 root root 0  9月  2 18:59 bbb
[root@localhost ~]# chmod ug=rw bbb
[root@localhost ~]# ls -l bbb
-rw-rw----. 1 root root 0  9月  2 18:59 bbb
[root@localhost ~]# chmod 446 bbb
[root@localhost ~]# ls -l bbb
-r--r--rw-. 1 root root 0  9月  2 18:59 bbb
```

2. chown

利用 chown 可以将指定文件的所有者改为指定的用户或组，只有 root 用户才能进行此操作，要特别注意的是，普通用户不能修改自己文件的属主和属组。

语法：

```
chown [OPTION]... [OWNER][:[GROUP]] FILE...
```

chown 命令选项如表 4-14 所示。

表 4-14 chown 命令选项

选项	解释
-c	显示更改部分的信息
-f	忽略错误信息
-h	修复符号链接
-v	显示详细的处理信息
-R	处理指定目录以及其子目录下的所有文件

● 修改文件的属主

```
[root@localhost ~]# ls -l bbb
-r--r--rw-. 1 root root 0  9月  2 18:59 bbb
[root@localhost ~]# chown jerry bbb
[root@localhost ~]# ls -l bbb
-r--r--rw-. 1 jerry root 0  9月  2 18:59 bbb
[root@localhost ~]# ^C
[root@localhost ~]#
```

● 同时修改属主和属组

```
[root@localhost ~]# ls -l
drw-r--r--. 2 root root 4096 Jun 8 11:10 usertxt
[root@localhost ~]# chown user:usergroup usertxt
[root@localhost ~]# ls -l
drw-r--r--. 2 user usergroup 4096 Jun 8 11:10 usertxt
```

3. chgrp

chgrp 可以用来修改文件或目录的所属组。

语法：

```
chgrp [OPTION] GROUP FILE...
```

chgrp 命令选项如表 4-15 所示。

表 4-15 chgrp 命令选项

选项	解释
-c	效果类似“-v”选项，但是只显示更改的部分
-f	忽略错误信息
-h	只修改符号链接的文件，而不对其他任何相关文件进行改动
-v	显示指令执行过程
-R	处理指定目录及其子目录下的所有文件



● 更改文件所属组

```
[root@localhost ~]# ls -l                                // 通过 ls -l 来查看文件的所属组
drw-r--r--. 2 user usergroup 4096 Jun 8 11:10 usertxt
[root@localhost ~]# chgrp user usertxt                  // 使用 chgrp 命令进行权限的修改
[root@localhost ~]# ls -l
drw-r--r--. 2 user user 4096 Jun 8 11:10 usertxt
```

4.4.4 预设权限掩码

umask 是 Linux 文件权限管理的一个重要命令，用于设定文件和目录在创建时的默认权限。Linux 系统默认的文件权限为 666，目录为 777，所有用户都一样。umask 是系统默认权限的掩码，意思是 umask 对应为 1 的权限位，会被取消权限。不同的用户设置不同的 umask 码，可以让不同用户默认的文件和目录的权限具有个性化特征，满足各自不同的需求。

举例来说，若 umask 设为 022，创建文件时，实际权限就是“ $666 - 022 = 644$ ”，这里的减法是逐位计算的；创建目录时，实际权限是“ $777 - 022 = 755$ ”。

常见 umask 值及对应的文件或目录权限如表 4-16 所示。

表 4-16 常见 umask 值及对应的文件或目录权限

umask 值	文件权限	目录权限
022	644	755
027	640	750
002	664	775
006	660	771
007	660	770

● 修改 umask 值，将影响新创建文件的权限

```
[root@localhost ~]# umask
0022
[root@localhost ~]# touch ccc
[root@localhost ~]# ls -l ccc
-rw-r--r--. 1 root root 0  9月  2 19:21 ccc
[root@localhost ~]# umask 027
[root@localhost ~]# touch cba
[root@localhost ~]# ls -l cba
-rw-r-----. 1 root root 0  9月  2 19:22 cba
```

本章练习题

1 在 Linux 系统中, 用于唯一标识一个用户, 并且系统在登录时真正用来区分用户的是 ()。

- A. 用户名 B. 用户主目录 C. 用户 ID (UID) D. 登录 Shell

2 以下关于 UID 的叙述中, 哪一项是正确的? ()

- A. root 用户的 UID 是 1
B. 所有普通用户的 UID 都必须大于 1000
C. UID 为 1 ~ 999 的用户通常是系统用户, 不可登录
D. 一个 UID 可以同时分配给多个不同的用户名

3 管理员想要创建一个名为 testuser 的新用户, 并指定其主目录为 /opt/testuser, 应该使用以下哪个命令? ()

- A. useradd -m testuser B. useradd -d /opt/testuser testuser
C. usermod -d /opt/testuser testuser D. useradd -s /opt/testuser testuser

4 在 Linux 中, 使用 ls -l 命令查看一个文件的详细信息, 显示为 -rwxr--r--, 这表示 ()。

- A. 文件所有者有读、写、执行权限; 所属组有读、写权限; 其他用户有读权限
B. 文件所有者有读、写、执行权限; 所属组有读权限; 其他用户有读权限
C. 文件所有者有读、写权限; 所属组有读、执行权限; 其他用户有执行权限
D. 这是一个目录, 所有者有全部权限

5 命令 chmod 755 filename 的效果等同于下列哪一项? ()

- A. chmod u=rwx, g=rx, o=rx filename B. chmod u=rw, g=r, o=r filename
C. chmod a+rwx filename D. chmod o+w filename

6 以下哪个文件存储了所有用户的加密密码信息, 并且通常只有 root 用户才有权限读取 ()。

- A. /etc/passwd B. /etc/group C. /etc/shadow D. /etc/gshadow

7 如果想要将文件 file.txt 的所有者更改为用户 john, 应该使用哪个命令? ()

- A. chgrp john file.txt
B. chmod john file.txt



- C. `chown john file.txt`
- D. `usermod -o john file.txt`

8 `umask` 设置用于解决什么问题？（ ）

- A. 提供个性化的默认文件或者目录权限
- B. 提供比传统 UGO（用户 - 组 - 其他）模型更精细的权限控制
- C. 替代 `/etc/passwd` 和 `/etc/shadow` 文件
- D. 用于加密文件内容

9 普通用户想要临时以 `root` 权限执行一条系统管理命令（如 `fdisk -l`），最安全的做法是（ ）。

- A. 直接使用 `su` 命令切换到 `root` 用户再执行
- B. 使用 `su -` 命令切换到 `root` 用户再执行
- C. 使用 `sudo fdisk -l` 命令
- D. 让管理员告知 `root` 密码，用 `root` 登录后执行

10 使用 `userdel` 命令删除一个用户时，如果希望同时删除该用户的主目录和邮件池，应该添加以下哪个选项？（ ）

- A. `-f`
- B. `-r`
- C. `-h`
- D. `-Z`