第5章 常用命令

本章学习目标：

- 了解 Linux 命令的基本语法
- 掌握常用的 Bash 快捷键
- 掌握命令重定向
- 熟悉最基本的命令用法

看到黑白屏幕上闪动的光标是一件多么神奇的事情，你会猜测计算机的操作者一定是一个令人羡慕的计算机高手，每个命令就像音乐家弹出的音符从他的指间蹦出。高效、自由、快捷……这就是 Linux 命令带给我们的愉悦，就连一直傲慢的微软都在其最新的 Windows 服务器版中借鉴 Linux 的命令手法，推出 PowerShell。掌握常用的命令是 Linux 使用者必备的基本素质。

5.1 命令基本语法与类型

5.1.1 命令类型与语法

Linux 默认安装后的命令超过两千个，这些命令分布在/usr/bin 和/usr/sbin 等目录中，平均每个命令可携带 5 个参数，这样“衍生”出了上万种命令用法。但庆幸的是，常用命令不会超过 200 个，最基本的常用命令大约 100 个，每个命令最常用的参数平均为 3 个，因此总的最基本“衍生”用法也就 300 种左右。

Linux 命令分为两类：一类是外部命令；另一类是内部命令。外部命令一定对应一个磁盘上的二进制文件，用户在命令行输入命令后，Bash 首先扩展匹配参数，然后再去执行对应的磁盘二进制文件。采用命令 which 可以找到对应的二进制文件所在的目录，例如 which ls 返回/usr/bin/ls，表示外部命令 ls 对应的二进制程序在/usr/bin 目录下。内部命令直接对应 Bash 程序里的代码段(可以理解为函数)，这样 Bash 在扩展命令行匹配参数后直接调用对应的函数代码，显然内部命令执行的效率更高，不带参数运行命令 help 可以列出全部的内部命令。存在一些既是内部命令又是外部命令的情况，例如 test 命令，如果只执行内部命令，因此把磁盘上的二进制文件/usr/bin/test 删除不会影响此命令的执行。

用户在命令窗口中输入命令，按 Enter 键后登录 Shell 程序就读取用户输入的命令串，然后扩展匹配参数，最后执行命令，执行完后又显示命令行，等待用户输入下一个命令。用

户默认的登录 Shell 程序就是 Bash。

Bash 在用户环境变量 PATH 所定义的目录中查找外部命令对应的二进制文件，例如 root 用户的 PATH 变量的默认值如下。

```
/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

Bash 从左至右遍历这些目录直到找到同名二进制文件为止，如果遍历完所有的目录还没有找到，Bash 就返回“命令没找到”的错误信息。如果编译的程序放在磁盘上的某个其他目录中，这时有两种方法执行它：一是把那个目录加到 PATH 中，然后直接输入程序名即可运行；二是带路径执行，如在命令行上输入/opt/arm/bin/myprogram 即可运行命令 myprogram，或者先进入目录/opt/arm/bin，然后带相对路径执行，即./myrpogram。

每个命令都允许带若干参数，参数用来影响命令的行为，参数有单字符参数和多字符参数之分，单字符参数前用“-”前导，多字符参数前用“--”前导，加前导字符的目的是与命令的“目标”区分开来，例如命令 ls 可带的参数中就有-f,--directory。单字符参数可以合在一起，只加一个前导字符即可，例如命令“ls -la”中的参数-la 实际上是由两个单字符参数合并而成的，这个命令也可以写成“ls -l -a”。

5.1.2 在线帮助文档

默认安装后，所有的命令都有在线帮助文档，命令带“--help”参数获取简要的帮助信息，用 man 可以获取外部命令的帮助信息，help 可以获取内部命令的帮助信息，info 命令提供更详细的信息。man 帮助文档很庞大，因此被划分成不同的“章节”，每个“章节”被赋予一个唯一序号，可以指定 man 命令只显示特定节中的帮助信息。man 的用法如表 5.1 所示。

表 5.1 man 帮助文档被划分为章节

序号	章节	说　明
1	1	外部命令的帮助信息
2	2	系统调用函数的帮助信息(内核提供的接口函数)
3	3	库函数的帮助信息
4	4	设备文件的帮助信息
5	5	文件格式和规范，例如/etc/passwd 文件的格式
6	6	游戏的帮助信息
7	7	其他帮助信息
8	8	系统管理命令的帮助信息(一般是 root 使用的命令)
9	9	内核工具帮助信息(非标准)

man 命令的简单语法如下：

```
man -a|[章节] 命令|文档|函数
```

如果省略章节，就在第 1 章搜索。有的命令帮助信息分布在多个章节中，如果要翻阅全部章节的内容，就带上-a 参数，例如“man -a read”。进入 man 界面后，可以使用快捷键：Space 键表示前翻一页；b 表示后翻一页；d 表示前翻半页；u 表示后翻半页；/pattern 表示查找关键字，n 表示继续前找，N 表示继续后找；h 表示取得帮助；q 表示退出 man 命令。

对于 Bash 的内部命令,直接采用 help 获取帮助信息。

(1) 列出全部内部命令:

```
help
```

(2) 取得内部命令 alias 的帮助信息:

```
help alias
```

(3) 获取 passwd 命令的帮助信息:

```
man passwd
```

(4) 获取/etc/passwd 文件的格式说明信息:

```
man 5 passwd
```

(5) 获取 C 语言函数 read 的调用方法:

```
man 3 read
```

5.2 Bash 快捷键、重定向和管道

5.2.1 历史命令与 Bash 快捷键

Linux 的命令行(见图 2.8 光标所在的行)其实就是行编辑器——在这里编辑将要执行的命令。在这个小小的编辑器里(只有一行),Bash 提供了许多快捷键,掌握最常用的几个快捷键并灵活使用,能带来意想不到的高效和便利。表 5.2 列举常用的一些快捷键。

表 5.2 Bash 常用快捷键

序号	快捷键	说 明
1	Ctrl+A	光标跳到行首(相当于 Home 键)
2	Ctrl+E	光标跳到行尾(相当于 End 键)
3	Ctrl+U	删除从光标位置到行首的所有字符
4	Ctrl+K	删除从光标位置到行尾的所有字符
5	Ctrl+W	删除光标左侧的一个单词
6	Ctrl+L	清屏(相当于执行命令 clear)
7	Ctrl+C	中断当前正在执行的命令
8	Tab	对命令或命令目标进行补齐。例如你只记得命令的前几个字母是 stra,那么输入 stra 后按一次 Tab 键,Bash 会自动补齐这个命令为 strace,但如果有多个命令的前 4 个字母是 stra,那么需要连续按两次 Tab 键,Bash 会显示所有以 stra 开头的命令。再如你想进入一个目录,但你只记目录的开头是/etc/init,这时你输完 cd /etc/init 后按 Tab 键补齐为 cd /etc/init.d。同样当存在多个符合要求的目录时,需要连续按两次 Tab 键列出全部的匹配目录
9	Ctrl+R	进入历史命令查找状态,然后输入关键字找到符合要求的历史命令,找到后可以编辑或者直接按 Enter 键执行

续表

序号	快捷键	说　明
10	↑，↓	用上、下光标键查阅历史命令
11	!!	再次执行最近执行的命令
12	!string	再次执行最近的以 string 开头的历史命令。如!man
13	!?string	执行最近的包含 string 的历史命令
14	!n	执行第 *n* 条历史命令。可用 history 查看全部的历史目录
15	!-n	执行倒数第 *n* 条历史命令。如!-15

与历史命令密切相关的两个 Shell 变量是 HISTFILE 和 HISTFILESIZE，前者定义记录历史命令的文件，后者定义记录历史命令的个数。默认值是 HISTFILE＝～/.bash_history，HISTFILESIZE＝1000，建议把 HISTFILESIZE 的值改成 5000，采用下面命令即可：

```
echo "export HISTFILESIZE = 5000" >>～/.bash_profile
```

然后重新登录或者重启计算机，新的用户环境变量就起作用。

5.2.2 命令重定向

不管是内部命令还是外部命令，最终都是执行一段程序代码，如果把这段代码看成一个黑盒子，那么流入黑盒的信息称为输入，从黑盒流出的信息称为输出。在 Linux 系统中，输入也称为标准输入，默认从键盘输入，输出又分为错误输出和标准输出，默认都是输出到屏幕，如图 5.1 所示。

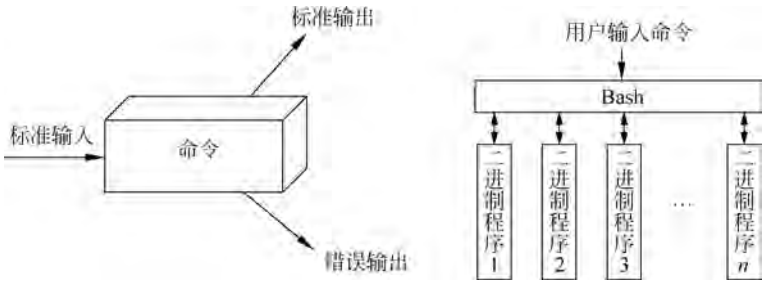


图 5.1　Linux 命令的“一进二出”

图 5.1 右侧部分说明用户输入的命令最终都会由 Bash 去找到同名的二进制程序，并执行那个二进制程序。

Linux 执行命令的过程如下：

(1) Bash 读取命令行上用户输入的字符串。

(2) Bash 扩展字符串中包含通配符的参数。

(3) 复制如下 3 个文件描述符。

0 号文件描述符(标准输入)，默认指向/dev/stdin(表示键盘)，缩写为 stdin；

1 号文件描述符(标准输出)，默认指向/dev/stdout(表示屏幕)，缩写为 stdout；

2 号文件描述符(标准错误输出)，默认指向/dev/stderr(表示屏幕)，缩写为 stderr；

注意：如果命令中存在重定向，那么这3个文件描述符的指向就可能发生变化。

(4) 在用户环境变量PATH定义的目录中搜索命令对应的二进制程序，然后执行它。

(5) 把二进制程序退出的状态保存到特殊变量“?”中。

(6) 显示命令状态行并等待用户再次输入。

0、1和2号文件描述符的指向允许通过命令行上的特殊参数来改变，即输入输出重定向，简称重定向。重定向用到的元字符有<、>、&、-。

(1) 列出/etc目录中的文件，结果放在/tmp/ls.txt文件中，而不显示在屏幕上，如果/tmp/ls.txt文件已经存在，就先清空它：

```
ls /etc 1 > /tmp/ls.txt
```

因为默认输出重定向就是针对标准输出的，所以上面命令中的“1”可以省略。

(2) 删除/opt/abc，报错信息重定向到/tmp/err文件中，正常信息放入/tmp/f123中：

```
rm /opt/abc > /tmp/f123 2 > /tmp/err
```

(3) 报错信息追加到/tmp/err.txt文件的末尾：

```
cp *.conf /tmp/ 2 >> /tmp/err.txt
```

(4) 正常信息和报错信息都重定向到/dev/null文件：

```
gcc abc.c &> /dev/null
```

/dev/null是一个“黑洞”，不管往里面写多少数据，都会消失，不会占用磁盘空间。对于某些命令，不想让它的执行结果显示在屏幕上，同时我们也不关心这些信息，就可以扔到这个“黑洞”里去。

(5) 分区命令输入重定向到cmd.txt文件：

```
parted /dev/nvme0n1 < cmd.txt
```

例如我的cmd.txt文件的内容如下，即创建一个分区，显示分区表，然后退出：

```
mkpart primay 50% 80%
print
quit
```

也可以这样执行：

```
parted /dev/nvme0n1 << EOF
mkpart primay 50% 80%
print
quit
EOF
```

也就是EOF之间的内容作为标准输入，EOF也可以换成任何其他字符串，只要首尾相同即可。大家看看下面的例子完成了什么任务：

```
cat > /tmp/file.txt << HA
```

```
Hi, 我是光脚老师.
有空吗?
喝一杯.
HA
```

(6) sudo 命令直接从命令行上读取密码 A2b2C3,然后执行命令 ls -l /tmp:

```
sudo -S ls -l /tmp <<< A1b2C3
```

sudo 命令要用参数-S 才允许从重定向的标准输入读取密码。

5.2.3 其他元字符

1. 管道|、|&

用"|"或"|&"隔开的两个命令之间形成了一个管道,左边命令的标准输出(用"|"连接)或者标准和错误输出(用"|&"连接)信息流入右边命令的标准输入,即左边命令的输出作为右边命令的标准输入,好像在两个命令之间铺设了一根水管。

(1) 翻页显示列出来的文件:

```
ls -l /proc | more
```

(2) 翻页显示编译时产生的信息:

```
gcc abc.c |& more
```

关于编译 C 语言的内容请参考 8.2 节。管道两边允许出现任何命令,只要满足工作需要,另外可以连续使用很多管道连接众多命令,像流水线似的。

(3) 显示全部的非系统用户名:

```
cat /etc/passwd | grep -v nologin | awk -F: '{print $1}'
```

grep 和 awk 命令后面会介绍,现在可以先不理它。

2. 命令序列

用";""&""&&"和"||"连接在一起的命令称为命令序列。用";"连接的命令从左至右依次被执行,最后执行的命令的返回状态就是整个命令序列返回的状态。在一个命令后加"&",表示该命令将在后台执行,即在子 Bash 中执行,对于一些执行时间较长又无须交互的程序适合在后台执行,例如下面的打包压缩命令执行时间比较长,把它放到后台执行,在前台可继续输入其他命令。

```
tar -cJf /tmp/etc.tar.xz /etc &
```

也可以用另外一种方法把一个任务切换到后台执行:先在前台执行命令,在命令还没有执行完前按 Ctrl+Z 快捷键把任务切换到后台,此时被切换到后台的命令暂停执行。这两种开启后台任务的方法由一个缺点,就是当用户注销后,后台任务都会退出,在命令前增加 nohup 就不存在这个缺点了,例如:

```
nohuo tar -cJf /opt/data.tar.xz /opt/erp &
```

假设下班前输入此命令,然后就可以注销走人了,这个命令会继续运行。关于 tar 命令可参考 5.3.4 节。

用“&&”连接的命令序列是这样执行的:当且仅当左边的命令执行成功(退出状态为0)时才执行右边的命令。在 Linux 系统中,命令退出状态为 0 表示执行成功,非 0 表示执行失败。如命令序列:

```
mkdir /opt/abc && cd /opt/abc
```

表示当创建目录成功才进入那个目录,否则就不执行“&&”右边的命令,创建目录不一定会成功,例如权限不够,从而没法创建目录。允许使用多个“&&”连接很多命令。

用“||”连接的命令序列是这样执行的:当且仅当左边的命令执行失败时才执行右边的命令,例如命令序列:

```
cd /tmp/abc || mkdir /tmp/abc
```

即如果不能进入目录(进入目录/tmp/abc 失败),那就创建那个目录。

混合使用“&&”和“||”可以达到意想不到的效果,如命令序列:

```
tar -cjf /tmp/etc.tar.bz2 /etc 2&>1 >/dev/null && echo "成功" || echo "失败"
```

这样当最左边的 tar 命令执行成功时会在屏幕上显示“成功”,否则就显示“失败”。

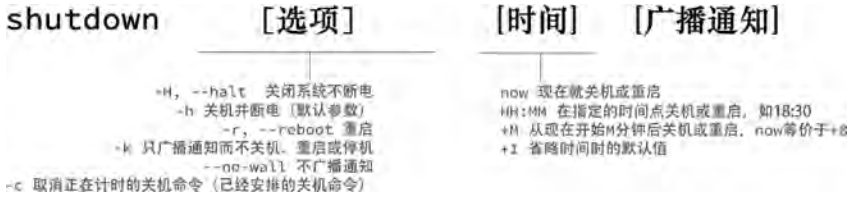
注意:最左侧的命令做了输出重定向处理,即标准输出和错误输出都定向到设备文件/dev/null。至于为什么 tar 命令执行失败会执行“echo"失败""?可以这样理解:tar 命令执行失败就不执行“&&”后面的命令“echo "成功""(相当于此命令不存在),那么对“||”来说,相当于最左侧的命令执行失败,所以就执行右侧的命令。

5.3 命令举例

前面章节已经介绍的命令这里不再赘述,本节补充一些常用的命令。

5.3.1 关机/重启/睡眠

关机和重启命令 shutdown 的语法如下:



即在指定的时间点关机或重启,关机或重启前会向已经登录操作系统的用户广播通知。现在的计算机支持操作系统关闭的同时主板不断电,这样就可以远程唤醒(例如通过网络远程启动计算机)。关机或重启前 5min 内不再允许用户登录,如果指定了广播通知,那么就要一定指定时间点。

(1) 关机并且断电：

```
shutdown
```

本命令省略选项、时间和广播通知，全部采用默认值，这等价于下面的命令：

```
shutdown -h +1 "The system is going down for poweroff at xxxxxx 时间"
```

(2) 现在就重启操作系统：

```
shutdown -r now
```

(3) 对于下面的例子，大家看得懂吗？

```
shutdown -h 23:59 "各位注意了，今晚 0 时关闭系统!"
```

(4) 取消之前已经安排的关机或重启命令：

```
shutdown -c
```

另一个停机命令是 halt，另一个关机命令是 poweroff，另一个重启命令是 reboot，不过这 3 个命令都是为了兼容早期版本，建议少用。进一步发现 shutdown、halt、poweroff、reboot 都是指向 systemctl 命令的符号连接，如图 5.2 所示。

```
root@Debian-11:/usr/sbin# ls -l shutdown reboot halt poweroff
lrwxrwxrwx 1 root root 14 Jul 14  2021 halt -> /bin/systemctl
lrwxrwxrwx 1 root root 14 Jul 14  2021 poweroff -> /bin/systemctl
lrwxrwxrwx 1 root root 14 Jul 14  2021 reboot -> /bin/systemctl
lrwxrwxrwx 1 root root 14 Jul 14  2021 shutdown -> /bin/systemctl
root@Debian-11:/usr/sbin#
```

图 5.2 关机和重启命令

所以可以采用 systemctl 命令关机或重启，此命令会在第 9 章介绍。

(5) 使系统进入睡眠：

```
systemctl hibernate
```

5.3.2 Bash 内部命令

1. history

history：管理历史命令。

(1) 显示最近执行的 $ HISTSIZE 条历史命令：

```
history
```

(2) 清除全部历史命令：

```
history -c
```

(3) 把保存在缓存中的历史命令写入磁盘文件中：

```
history -w
```

2. alias 和 unalias

alias 和 unalias：定义命令的别名和删除别名，通常把比较长且又经常使用的命令定义一个别名，以后输入这个别名即可执行那个长命令。

(1) 显示全部已经定义的别名：

```
alias
```

(2) 定义命令 man 的别名为 h：

```
alias h=man
```

此后执行 h 也可以获取外部命令的帮助信息，例如“h ls”命令获取 ls 的在线帮助信息。

(3) 把“ls -l --color=auto”命令定义成别名 ll：

```
alias ll='ls -l --color=auto'
```

(4) 删除别名 h：

```
unalias h
```

3. echo

echo：显示信息。

(1) 显示“王先生，早上好!”：

```
echo "王先生,早上好!"
```

(2) 分两行显示，第一行显示“苹果”，第二行显示“5 元一斤”：

```
echo -e "苹果\n5 元一斤"
```

带参数-e 的情况下，“\n”表示换行符，另外“\a”表示响铃，“\b”表示退格键，“\r”表示 Enter 键，“\t”表示 Tab 键，“\\”表示反斜杠。

(3) 大家看看这是怎么显示的：

```
echo -e "Alice\\t18 years old\\nJohn\\t20 years old"
```

(4) 显示后不换行(主要是-n 参数)：

```
echo -n 显示 $HISTSIZE 条历史命令
```

4. source

source：从文件中读命令并执行。通常会把定义环境变量的命令统一放在一个文件中，然后采用 source 命令来执行它，从而输出这些环境变量。

执行 allenvs，这样里面定义的环境变量和别名就起作用了：

```
source allenvs
```

allenvs 文件内容类似：

```
alias ll = 'ls -l --color = auto'
export HISTFILESIZE = 5000
export HISTSIZE = 1000
export HISTTIMEFORMAT = '%F %T '
export LANGUAGE = zh_CN:zh
export FCEDIT = vim
```

source 命令和点命令(.)是等价的,所以本例也可以写成:

```
. allenvs
```

5. fc

fc:重新编辑并执行历史命令,编辑器由 FCEDIT 环境变量定义。该命令对于特别长的命令非常有用,不需要每次都重新输入。

(1) 编辑刚刚执行过的历史命令,存盘后立即执行:

```
fc
```

(2) 编辑并再次执行最近执行过的以 tar 开头的历史命令:

```
fc tar
```

(3) 编辑并再次执行第 456 条历史命令:

```
fc 456
```

可用 fc -l 列出历史命令。

6. pwd

pwd:显示当前目录。

显示当前目录:

```
pwd
```

5.3.3 系统信息相关命令

1. date

date:显示或设置系统时间。

(1) 按默认格式显示计算机的系统时间:

```
date
```

显示的时间类似这样:2022 年 01 月 22 日 星期六 07:14:45 CST。

(2) 按格式显示日期和时间,如“2022-01-22 07:16”:

```
date +"%Y-%m-%d%H:%M"
```

常用的时间格式有:%c(当地的日期时间格式)、%d(月中的第几日)、%F(年-月-

日)、%H(小时,二十四小时制)、%I(小时,十二小时制)、%j(年中第几日,取值范围是 001 至 366)、%m(月份)、%M(分钟)、%n(换行)、%N(纳秒,000000000..999999999)、%9(季度)、%s(从 1970 年 1 月 1 日零时至今的秒数)、%S(分钟内的秒数)、%t(Tab 键)、%T(等价于%H:%M:%S)、%u(星期几)、%U(一年的第几周)、%Y(年份)。

(3) 大家看看下面的命令显示什么:

```
date +"现在是 %H 时 %M 分 %S 秒"
```

(4) 把计算机时间改为 2022 年 6 月 5 日上午 9 点 30 分:

```
date 060509302022
```

新的日期时间格式为: MMDDhhmm[[CC]YY][.ss],即两位月份两位日期两位小时两位分钟四位年份一个点两位秒数,其中年份和秒数允许省略。只有 root 用户有权修改系统时间。

2. cal

cal: 显示日历。

(1) 显示本月的日历:

```
cal
```

(2) 显示本年的日历:

```
cal -y
```

(3) 显示指定年份的日历:

```
cal 2025
```

cat 命令还可以显示指定日期的日历,日期的格式为[[DD] MM] YYYY,如

```
cal 05 2018
cal 25 06 2019
```

3. clear

clear: 清除屏幕上的内容,相当于快捷键 Ctrl+L。

4. bc

bc: 任意精度计算器。更准确地说,它是一门类似 C 的计算器语言: 能定义变量和函数,存在判断语句和循环语句,还提供了数学函数库(例如三角函数)。

(1) 交互式使用计算器:

```
bc
```

进入 bc 后,就可以不断输入算数表达式,按 Enter 键就会看到结果,输入 quit 退出计算器,如图 5.3 所示。

图 5.3 交互式使用 bc

(2) 通过重定向标准输入直接把表达式输入命令行：

```
bc <<< "99 * (77-55)/5"
```

(3) 从 bc.exp 文件中读取计算机语言：

```
bc -q bc.exp
```

例如 bc.exp 的内容如下，这是计算 10 的阶乘：

```
define f(x) {
    if (x <= 1) return (1);
    return (f(x-1) * x);
}
f(10)
quit
```

关于 bc 更详细的用法可参考在线文档。

5. w

w：查看系统连续运行的时间、当前用户和负载，如图 5.4 所示。

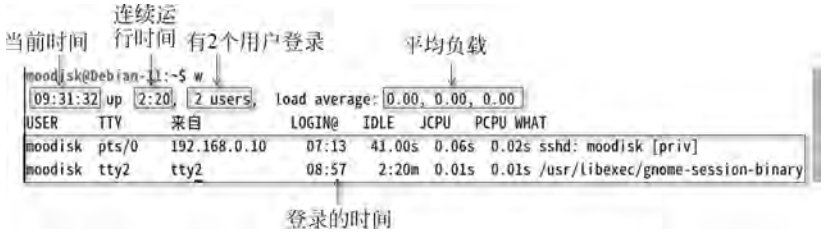


图 5.4 w 命令的输出

图 5.4 中平均负载有三个数字，分别表示最近 5min、10min 和 15min 内的平均负载。这里的负载就是处于运行状态和不可中断状态的进程数目。其他完成功能的命令还有 uptime、who。

6. uname

uname：查看系统信息。带-a 参数显示全部系统信息，其他参数只显示相应信息。系

统信息如图 5.5 所示。

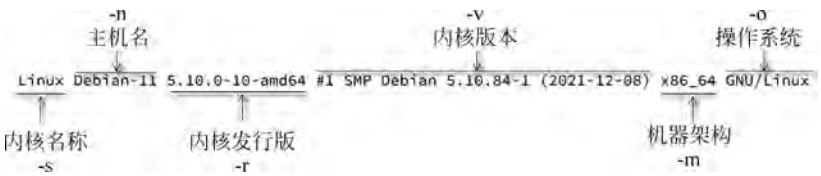


图 5.5 系统信息

只查看主机名称：

```
uname -n
```

7. df

df：查看已经挂载分区的使用情况。最常用的参数就是下面的例子。

```
df -Th
```

显示的结果如图 5.6 所示。

图 5.6 df 命令的显示结果

8. dmesg

dmesg：操纵开机信息。

(1) 显示全部的开机信息：

```
dmesg
```

(2) 只显示错误和警告信息：

```
dmesg -l err,warn
```

其他的信息类型还有 emerg(导致系统不稳定)、alert(需尽快处理的报警信息)、crit(致命错误信息)、notice(值得注意的信息)、info(正常信息)、debug(调试信息)。

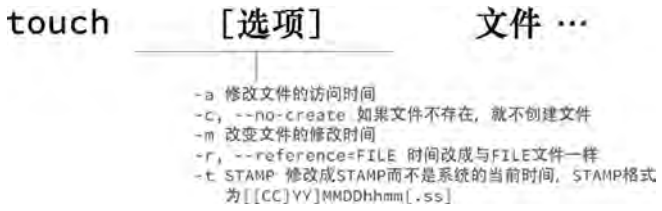
(3) 清除开机信息：

```
dmesg -C
```

5.3.4 文件操作命令

1. touch

touch：修改文件的时间戳(访问时间、修改时间等),其使用语法如下。



(1) 修改 abc.txt 的时间戳为当前的系统时间,如果 abc.txt 不存在,就创建它：

```
touch abc.txt
```

(2) 把 file.123 的时间戳改成与/etc/passwd 一样：

```
touch -r /etc/passwd file.123
```

(3) 大家看看下面的命令做了什么：

```
touch -t 201806040630 /opt/erp
```

2. tree

tree：以树形结构显示目录中的内容,其语法如下。

(1) 以属性结构列出当前目录中的内容：

```
tree
```

(2) 以属性结构列出/etc 目录中的目录：

```
tree -d /etc
```

(3) 列出根目录的两层树形结构：

```
tree -d -L 2 -x /
```

-x 参数不列出挂载点(挂载了分区)中的内容,例如在/mnt 目录上挂载了某个分区,那么/mnt 目录的内容不会出现在树形结构中。

3. du

du：统计文件占用磁盘的空间大小，其使用语法如下。

```
du      [选项]          [文件 …]
        -a, --all 统计全部内容（包括文件和目录）
        -b, --bytes 以B为单位
        -k 以KB为单位（默认单位）
        -m 以MB为单位
        -c, --total 计算合计
        -h, --human-readable 灵活采用合适的单位，如1K|,234M|,1.1G|
        -s 只统计命令上的文件或目录，不单独统计目录中的每个文件
        -x 忽略挂载点中的内容
```

如果省略“文件…”，就统计当前目录。

(1) 以 KB 为单位统计/boot 中每个目录占用磁盘容量：

```
du /boot
```

(2) 统计根目录下各个一级子目录占用磁盘容量：

```
du -sh -x /*
```

4. grep

grep：在文件中搜索包含特定内容的行，其语法如下。

```
grep    [选项]          匹配模式        [文件 …]
        -e PATTERNS, --regexp=PATTERN 搜索的匹配模式，可多次使用此参数
        -i, --ignore-case 比较时忽略字母大小写
        -v, --invert-match 显示非匹配的行
        -w, --word-regexp 要求单词匹配
        -x, --line-regexp 整行匹配
        -l, --files-with-matches 只显示匹配成功的文件名
        -A NUM 匹配行下面的NUM行也显示出来
        -B NUM 匹配行上面的NUM行也显示出来
        -r, --recursive 递归在各个子目录的文件中搜索
```

如果省略“文件 …”，那么就在当前目录中搜索。-e 参数可以出现多次，例如“grep -e root -e 123456 /etc/ssh/*”就是搜索包含 root 或者 123456 的行。“匹配模式”就是要搜索的正则表达式，出现在正则表达式中的大部分字符，如字母、数字代表自身，中括号内部的字符表示匹配任何一个即可，例如[a8kjd]只要匹配 a、8、k、j、d 中的任何一个就行，中括号内可以使用范围字符-，如[2-8a-h]代表 2～8 的数字和 a～h 的字母。^匹配一行的开头，$ 匹配一行的结尾，例如^Root 表示行首是 Root 才匹配成功，amen $ 表示 amen 出现在行尾才算匹配成功。可用转义符“\”把一个具有特殊意义字符转变为普通字符，如[234a\[]。关于正则表达式更详细的介绍可参阅在线文档 man 7 regex。

(1) 查找/etc/passwd 文件中包含 nologin 的行：

```
grep nologin /etc/passwd
```

(2) 查找/etc/profile 中首字母为 # 的行：

```
grep  ^# /etc/profile
```

(3) 在/etc/ssh 目录的全部文件中搜索包含 root 的行(忽略大小写)：

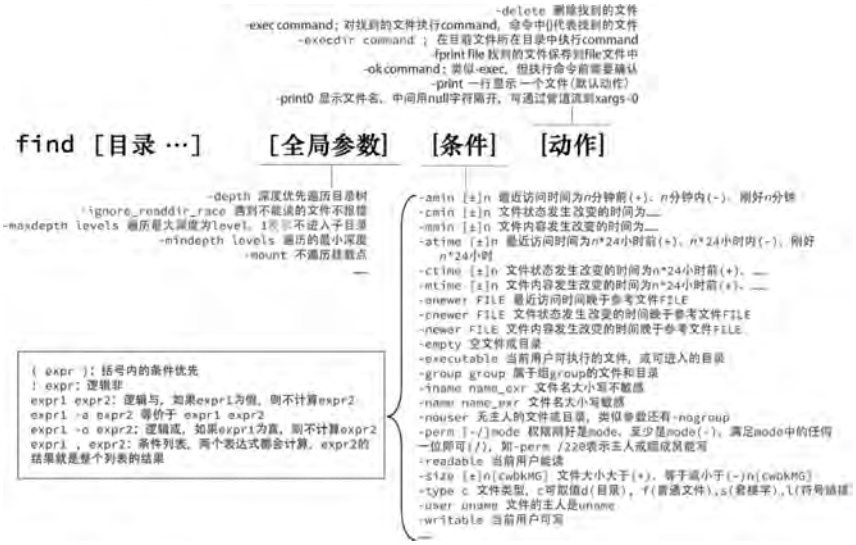
```
grep -i root /etc/ssh/*
```

(4) 递归在/etc/目录及其子目录中搜索包含255.255.255或gateway的行：

```
grep -r -i -e 255.255.255 -e gateway /etc/*
```

5. find

find：在目录树中查找目标文件，并对找到的文件施行某种操作。此命令语法复杂，功能强大，其语法如下。



即find在若干“目录”中查找符合“条件”的文件(包含目录)，对于找到的文件执行某个“动作”。语法中只列出大部分常用的“条件”，其他更复杂且不常用的条件可参考在线文档(man find)，例如条件“-mmin -30”表示半个小时之内修改过的文件。多个条件可以通过()、-a、-o、和，整合使用，例如“-name *.conf -a -size +1M”表示以.conf结尾且大于1MB的文件。

(1) 在当前目录下找出任何文件和目录：

```
find
```

此命令就是递归列出目录中的任何文件，类似的例子有“find /boot”。

(2) 递归找出/etc及其子目录中所有以.conf结尾的文件：

```
find /etc -name \*.conf
```

星号前加转义符(即*)是告诉Bash这里的星号不是通配符，而是普通字符，让find命令去处理它。

(3) 在/var、/usr、/home三个目录中查找符合这些条件的文件：当前用户可读且大于50MB的普通文件：

```
find /var /usr /home -readable -size +50M -type f
```

(4) 在根目录及其一、二级子目录中(不包括挂载点)查找空文件或者没有主人的文件

和目录：

```
find  / -maxdepth 2 -mount -empty -o -nouser
```

删除一个用户后，那么以此用户为主人的文件就变成没有主人的文件了。

(5) 删除/var/log 及其子目录中以“.n”(*n* 为 1～9 的数字)结尾的文件：

```
find  /var/log -name \*.[1-9] -delete
```

(6) 删除根分区上大于 200MB 的文件(要特别小心，不要误删了文件)：

```
find / -mount -size +200M -exec rm {} \;
```

最好先执行命令“find / -mount -size ＋200M”看看找到的文件可否删除。命令中的“{}”代表被找到的文件名。当涉及改动文件(包括删除)时采用-ok 动作更安全，执行操作前需要进行确认，如：

```
find / -mount -size +200M -ok rm {} \;
```

(7) 将/etc 目录下(不含子目录)的以.conf 结尾的文件复制成.conf.old：

```
find /etc -maxdepth 1 -name \*.conf -exec cp {} {}.old \;
```

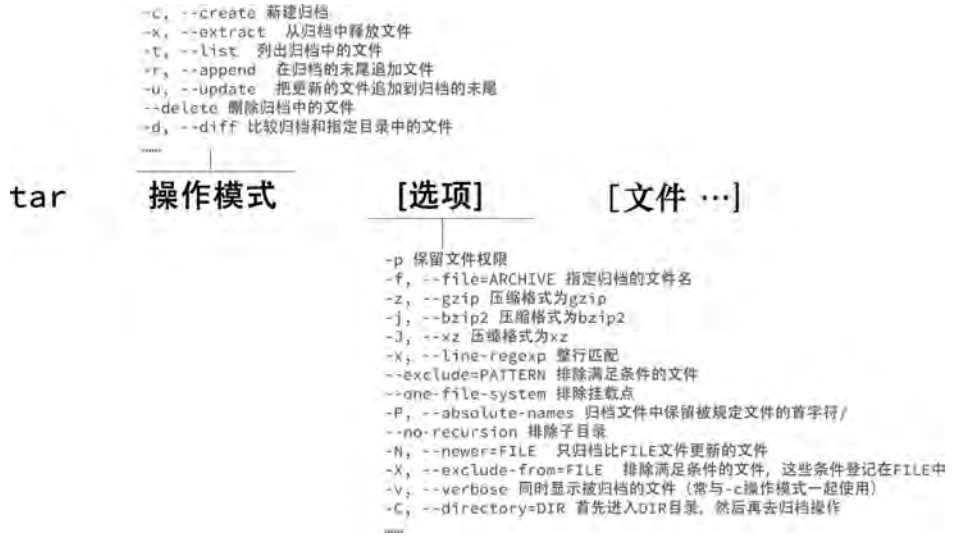
(8) 大家看看下面的例子完成什么任务：

```
find / \( -perm -4000 -fprint /tmp/suid.txt \) , \( -size +100M -fprint /tmp/big.txt \) 2 >
/dev/null
```

注意：命令中的逗号逻辑运算符前后要有一个空格。出现在 find 命令中的小括号、星号和分号前都要加一个转义符\。

6. tar

tar：归档命令，它把多个文件打包压缩到一个文件中，或者反方向操作，常用于备份数据。其语法如下。



“操作模式”每次只能选择一种，要么选择-c 新建归档，要么选择-x 从归档中释放文件，等等。至于其他不太常用的操作模式和选项，可参考在线帮助文档(man tar)。

(1) 把/etc 打包成/tmp/etc_backup.tar：

```
tar -c -f /tmp/etc_backup.tar /etc
```

归档文件名可以任意取，例如 f123、abc、etc.tar 等都可以，不过强烈建议以.tar 结尾。“-c -f”可以合在一起，即“-cf”。

(2) 打包并压缩/etc 目录下的以.conf 结尾的文件：

```
tar -cJvf conf.tar.xz /etc/*.conf
```

xz 的压缩率高于 bzip2，而 bzip2 又高于 gzip，尤其是归档文件很大时，xz 更能体现优势。既打包又压缩的归档文件名最好以.tar.xz(采用 xz 压缩)、.tar.bz2(采用 bzip2 压缩)或.tar.gz(采用 gzip 压缩)结尾，这是业界约定俗成的。

(3) 列出归档中的文件：

```
tar -tf abc.tar.bz2
```

不管采用哪种压缩格式，都可以采用上述命令列出里面的文件。

(4) 从 Linux 内核归档中释放文件到当前目录中：

```
tar -xf linux-5.16.2.tar.xz
```

从 www.kernel.org 下载的内核归档都采用 xz 压缩格式，不管采用何种压缩格式，都可以采用上面的命令释放里面的内容。

(5) 从归档中释放指定的文件(即 etc/sudo.conf etc/nfs.conf)：

```
tar -xvf conf.tar.xz etc/sudo.conf etc/nfs.conf
```

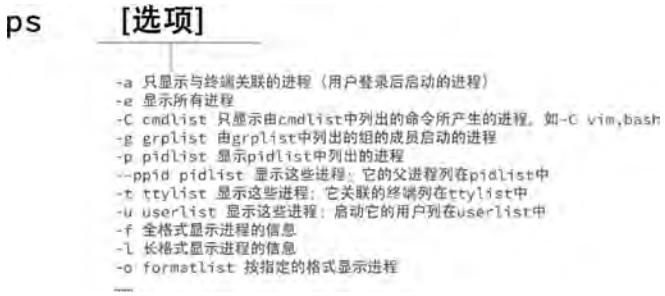
(6) 把归档中的文件释放到/tmp 目录中：

```
tar -xf abc.tar.bz2 -C /tmp
```

5.3.5 进程及任务管理相关命令

1. ps

ps：显示进程。其语法如下。



允许同时使用多个参数,但有的参数是互斥的。参数-o 定义显示的列,常见的列名有args(完整的命令)、exe(命令所在目录)、pgid(启动进程的用户的主要组群)、pid(进程号)、ppid(父进程号)、state(状态)等,如-o exe,args,pid,state。

(1) 显示与当前终端关联的且由当前用户启动的进程:

```
ps
```

示例如图 5.7 所示。

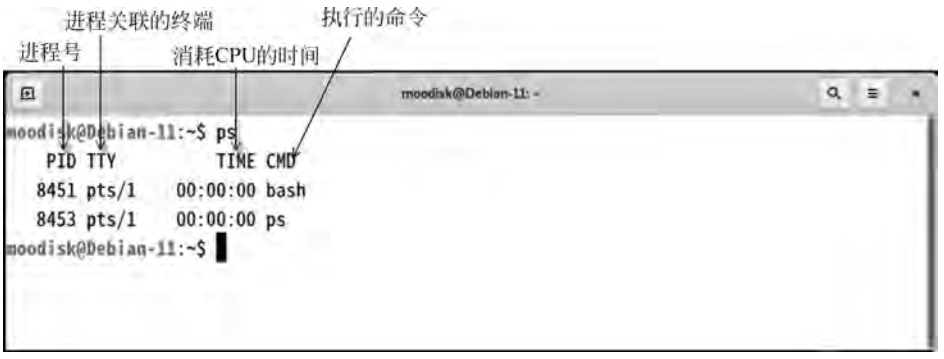


图 5.7 ps 命令示例

(2) 查看用户登录后启动的进程:

```
ps -a
```

(3) 以全格式显示全部的进程:

```
ps -ef
```

结果中的 UID 表示启动进程的用户 ID,PPID 表示父进程,STIME 表示启动时间。

(4) 以全格式显示由 moodisk 和 teacher 用户启动的进程:

```
ps -f -u moodisk,teacher
```

(5) 按指定的格式显示进程:

```
ps -a -o pid,user,exe,args
```

由于进程之间存在父子关系,因此用 pstree 命令显示完整的进程家族树。

2. kill

kill:向进程发送信号,信号不同,目标进程收到信号后的行为也会不同,例如信号 9 会导致目标进程被立即终止。其语法如下。

```
kill [选项] [进程号…]
-l, --list 列出全部的信号,此参数必须省略进程号
-<signal> 指定发送的信号,如-9
```

常用的信号有:1(挂起)、2(中断,相当于快捷键 Ctrl+C)、3(退出)、9(立即终止)、15(终止进程,这是默认信号。-1 进程号表示所有进程,正数表示具体的进程,除-1 之外的

其他负数表示组号(即由此组成员启动的全部进程)。

(1) 列出全部信号:

```
kill -l
```

(2) 给 2453 进程发信号 15:

```
kill 2453
```

(3) 终止进程 8865 和 7765:

```
kill -9 8865 7765
```

(4) 给所有进程发送挂起信号:

```
kill -1 -1
```

其他的一些发送信号的命令还有 pkill、killall 等。

(5) 终止用户 osadmin 的全部进程,用户被动退出系统:

```
pkill -9 -u osadmin
```

(6) 终止由程序 sendmail 产生的全部进程:

```
killall -9 sendmail
```

3. jobs、fg、bg

jobs、fg、bg: 管理任务。

这是三个内部命令。之前讲过,在一个命令末尾添加 &,此命令就在后台运行(称为后台任务),或者按 Ctrl+Z 快捷键把正在前台运行的命令切换到后台(执行命令 ls -R /,此命令还没结束前按 Ctrl+Z 快捷键试试),此时任务暂停运行。

(1) 列出全部的后台任务:

```
jobs
```

(2) 使 2 号任务继续在后台运行:

```
bg 2
```

如果省略任务号,bg 就使刚刚切换到后台的任务继续运行。

(3) 把刚刚切换到后台的任务切回到前台:

```
fg
```

也可以带具体的任务号执行 fg 命令,例如 fg 3 就把 3 号任务切换到前台。

5.3.6 网络相关命令

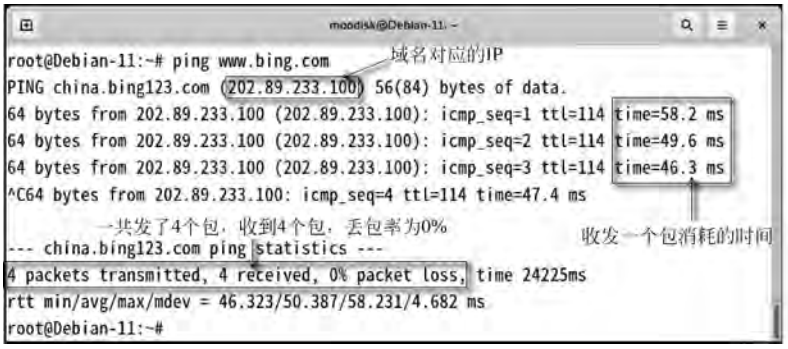
1. ping

ping: 测试网络通断情况。

(1) 测试是否与 www.bing.com 连通：

```
ping www.bing.com
```

示例如图 5.8 所示。




```
root@Debian-11:~# ping www.bing.com
PING china.bing123.com (202.89.233.100) 56(84) bytes of data.
64 bytes from 202.89.233.100 (202.89.233.100): icmp_seq=1 ttl=114 time=58.2 ms
64 bytes from 202.89.233.100 (202.89.233.100): icmp_seq=2 ttl=114 time=49.6 ms
64 bytes from 202.89.233.100 (202.89.233.100): icmp_seq=3 ttl=114 time=46.3 ms
^C64 bytes from 202.89.233.100: icmp_seq=4 ttl=114 time=47.4 ms

--- china.bing123.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 24225ms
rtt min/avg/max/mdev = 46.323/50.387/58.231/4.682 ms
root@Debian-11:~#
```

图 5.8　ping 示例

使用此命令会不断给对方发数据包，直到按下 Ctrl+C 快捷键终止命令为止。

(2) 给对方发 3 个数据包：

```
ping -c 3 192.168.0.100
```

2. ss

ss：显示网络套接字信息。ss 是 netstat 的替代命令，具有更好的性能。其语法如下。

```
ss [选项] [过滤器 …]
```

```
-t, --tcp 显示TCP套接字
-u, --udp 显示UDP套接字
-4, --ipv4 显示IPv4套接字
-6, --ipv6 显示IPv6套接字
-w 显示RAW套接字
-d 显示DCCP套接字
-D FILE 不显示而写入FILE文件中

-a, --all 显示全部套接字
-n, --numeric 不解释服务名称，以数字方式显示：ip:port
-r, --resolve 解释服务名称：域名:服务名
-l, --listening 只显示正在监听的套接字
-p, --processes 显示使用网络的进程
-m, --memory 显示套接字使用内存的情况
```

ss 命令更多不常用的参数可参考在线帮助文档。

(1) 显示 TCP 的全部套接字：

```
ss -t -a
```

示例如图 5.9 所示。

当已经建立链接的套接字(ESTAB 状态)的 Recv-Q 或 Send-Q 都很大时，要引起注意，可能有网络异常。

(2) 以数字形式显示 TCP、UDP 的全部套接字信息(包括相应的进程)：

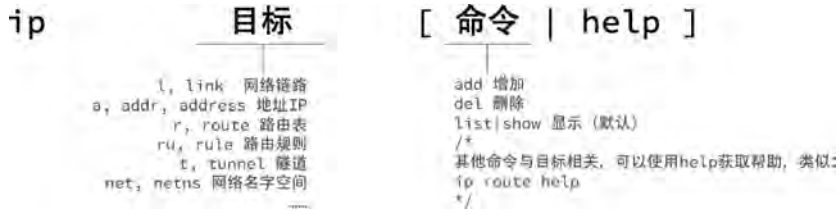
```
ss -t -u -a -n -p
```

图 5.9 ss 命令示例

由于 Linux 命令的单字符参数可以合在一起，因此上面的命令等价于 ss -tuanp。

3. ip

ip：此命令用于显示或操纵路由、网络接口和隧道，它是 ifconfig 的替代命令。其语法如下。



常用的目标有 address、route 和 link，其中 a 和 addr 是 address 的缩写，三者都可以使用。如果省略“命令”，就是显示相应目标的状态信息。

1）操纵网卡的 IP 地址：ip address

（1）显示全部网络接口上的 IP 配置信息：

```
ip address
```

示例如图 5.10 所示。

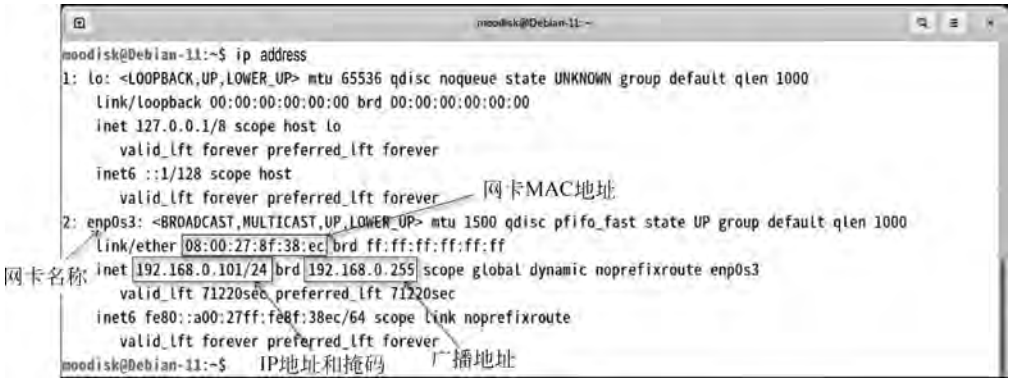


图 5.10 ip address 示例

（2）只显示网卡 enp0s3 的 IP 配置信息：

```
ip a show enp0s3
```

(3) 给网卡添加一个 IP:

```
ip address add 192.168.0.110/24 dev enp0s3
```

一块网卡允许配备多个同一个网段内的 IP,并且掩码一样。代码中的 192.168.0.110/24 也可以写成 192.168.0.110/255.255.255.0。增加的 IP 在机器重启后就失效了。

(4) 删除网卡的 IP 地址:

```
ip address del 192.168.0.110/24 dev enp0s3
```

关于 ip address 更多的信息可参考在线帮助文档(man 8 ip-address)。

2) 操纵网卡的链路: ip link

(1) 显示链路层及收发数据帧情况:

```
ip -s link
```

也可以针对特定的网卡,例如 ip link show enp0s3。

(2) 禁用网卡:

```
ip link set enp0s3 down
```

采用命令 ip link set enp0s3 up 启用网卡。

(3) 修改网卡的 MAC 地址:

```
ip link set enp0s3 down
ip link set enp0s3 address 08:00:27:0c:b8:d2
ip link set enp0s3 up
```

关于 ip link 更详细的介绍,可参考在线帮助文档(man 8 ip-link)。

3) 操纵路由表: ip route

(1) 显示路由表:

```
ip route show
```

(2) 显示默认网关:

```
ip route show default
```

(3) 添加一条默认路由:

```
ip route add default via 192.168.1.1
```

如果存在多块网卡,需要指定从哪块网卡到达网关,类似下面的命令:

```
ip route add default via 192.168.1.1 devenp0s3
```

(4) 通过网卡 enp0s5 到达 10.10.10.0/24 网段:

```
ip route add 10.10.10.0/24 dev enp0s5
```

(5) 删除默认路由:

```
ip route del default
```

(6) 删除一条路由记录：

```
ip route del 10.10.10.0/24 dev enp0s5
```

5.4 安装、卸载和升级软件包

在 Linux 操作系统中，把一些相关的可执行文件、配置文件和说明文档等打包在一起，构成为一个软件包，软件包是安装和卸载的基本单位，把众多的软件包集中存放在某台计算机上，并且建立软件包之间的依赖关系，这台计算机就称为安装源。Debian 和 RHEL 操作系统打包的方式不同，所以它们有各自的安装源，且不能互用，所以本节的举例都会注明适用的操作系统。

为了便于用户安装，Linux 发行版厂家又把一些相关的软件包组合成模块，这样用户只要安装某个模块就把所有的相关软件包都安装了。例如阿帕奇网站服务模块 httpd 由 httpd、httpd-tools、httpd-manual、mod_ldap、mod_ssl 等软件包组成。有的 Linux 发行厂商会根据不同的应用进一步把软件包组合成组，例如用户需要采用 Linux 搭建一套虚拟化应用环境，他只需安装 Virtualization Host 组就可以了，这个组包含了构建虚拟化环境的各种软件包，例如 libvirt、qemu-kvm、libvirt-client、libguestfs 等。

5.4.1 配置安装源

安装源就是被安装软件的存放地，即以后安装软件时自动从安装源下载(位于网络上)或者复制(位于本地存储)相应的软件包并安装。可以配置多个安装源，并设置优先级。如果一个软件包存在于多个安装源中，那么从最高优先级源中安装。在 Windows 上安装软件，首先要从网上搜索并下载相应的软件，然后再安装，或者从光盘安装。但是如果配置好了安装源，在 Linux 上安装软件就容易多了。

对于 Debian 11，登录图形桌面后，单击左上角的“活动”，然后单击左侧的“显示应用程序”，最后再单击“软件和更新”，接下来参照图 5.11 所示的步骤操作即可。

图 5.11 Debian 11 安装源配置

也可以手工修改安装源配置文件/etc/apt/sources.list,默认安装源是 http://cn.archive.ubuntu.com,国内建议采用阿里云的安装源,即 http://mirrors.aliyun.com,速度比较快。把/etc/apt/sources.list 文件中的 cn.archive.ubuntu 全部替换为 mirrors.aliyun 即可,然后运行命令 apt update 更新本地软件包元数据。用上面的图形方式修改安装源实际上也是修改了/etc/apt/sources.list 文件。

对于 RHEL 9.0,可以把系统安装光盘设置成一个本地安装源,首先把光盘放入光驱,然后执行下面几条命令:

```
mkdir /opt/cdrom
echo "/dev/sr0 /opt/cdrom iso9660 ro 0 0">>/etc/fstab
systemctl daemon-reload
mount -a
```

上面第 3 条命令是从修改后的/etc/fstab 文件产生自启动服务,这样下次开机时会自动把光盘挂载到/opt/cdrom 目录上。第 4 条命令是立即把/etc/fstab 文件中已经定义的但是还没有挂载的设备挂载到相应的目录上,目前就是挂载光盘到/opt/cdrom 目录。

采用命令 df -T 可以确认光盘是否挂载到/opt/cdrom 目录,采用如下命令产生文件/etc/yum.repos.d/cdrom.repo:

```
cat > /etc/yum.repos.d/cdrom.repo << EOF
[LocalRepo_BaseOS]
name = LocalRepository_BaseOS
baseurl = file:///opt/cdrom/BaseOS
enabled = 1
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[LocalRepo_AppStream]
name = LocalRepository_AppStream
baseurl = file:///opt/cdrom/AppStream
enabled = 1
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
EOF
```

RHEL 9.0 把软件分成两大类,分别是 BaseOS 和 AppStream,BaseOS 类主要包括 Linux 操作系统软件,AppStream 类主要包括各种应用程序。如果你已经是红帽的订阅用户,那么就可以直接激活订阅,假如订阅用户账号为 moodisk,则可以采用下面的命令激活订阅:

```
subscription-manager register --username=moodisk --auto-attach
```

然后输入订阅时设置的密码即可。不管是采用光盘安装源还是订阅,最好执行 dnf makecache 命令重建软件包元数据缓存,这样就可以直接用命令 dnf 来安装软件了。

(1) 显示全部可用的安装源:

```
dnf repolist                                    #RHEL 9.0
```

如果还要显示不可用的安装源，那么就用命令 dnf repolist all。

注意：第一列为 repo id，即源 ID 号，这个是唯一的。

(2) 显示源 ID 为 LocalRepo_BaseOS 的说明信息：

```
dnf repoinfo LocalRepo_BaseOS                    # RHEL 9.0
```

(3) 把新的安装源的元素据同步下来，每次修改了安装源，记得都要执行本命令：

```
apt update                                       # Debian 11
```

对于 Debian，可以采用命令 apt-mirror 来搭建本地安装源，具体方法可去网上搜索，这里不再赘述。

5.4.2 安装、卸载软件

RHEL 阵营(包括 Suse、CentOS、Fedora 等)和 Debian 阵营(包括 Ubuntu 等)的软件包管理方式完全不同，前者是 rpm 软件包系统，后者是 dpkg 软件包系统。软件包维护既可以用图形方式，也可以用命令方式，下面重点讲解命令方式。

1. RHEL 9.0

主要采用 rpm 和 dnf 命令(注意，yum 已被 dnf 淘汰)。

1) rpm

rpm：一个强大的软件包管理器，用于软件的安装、查询、更新、卸载等。

rpm 使用-q，--query 参数来查询。

(1) 显示全部已经安装的软件包：

```
rpm -q -a
```

此命令的两个参数可以合在一起，即 rpm -qa。由于安装的软件包很多，为了翻页显示，可以通过管道连到 more 命令，或者流到 grep 查找某个软件包是否已经安装，如：

```
rpm -qa | more
rpm -qa | grep vim
```

(2) 列出已经安装的软件包 coreutils 中到底包含什么文件：

```
rpm -ql coreutils
```

(3) 查找/usr/bin/ls 文件被包含在哪个软件包中：

```
rpm --query -f /usr/bin/ls
```

(4) 列出软件包 tar-1.34-3.el9.x86_64.rpm 中包含的文件：

```
rpm -qpl tar-1.34-3.el9.x86_64.rpm
```

红帽操作系统的软件包文件都以.rpm 结尾，可从安装光盘复制或者网站下载。

(5) 查看已安装软件包 openssh-server 的说明和版本信息：

```
rpm -q openssh-server -i
```

(6) 查看软件包 gzip 依赖哪些软件包：

```
rpm -q -R gzip
```

软件包之间存在依赖关系，能成功安装某个软件包的前提是它依赖的那些软件包都已经安装。

rpm 使用-i，--install 参数来安装软件包。

(7) 安装软件包，nmap.rpm 就在当前目录下：

```
rpm -i ./nmap.rpm
```

(8) 直接从网上安装软件包 expect：

```
rpm -ivh http://rpmfind.net/linux/centos-stream/9-stream/AppStream/x86_64/os/Packages/expect-5.45.4-15.el9.x86_64.rpm
```

(9) 重新安装软件包 expect：

```
rpm --reinstall /opt/cdrom/AppStream/Packages/expect-5.45.4-15.el9.x86_64.rpm
```

rpm 使用-e，--erase 参数来卸载软件包。

(10) 卸载 expect：

```
rpm -e expect
```

2) dnf

dnf 是 yum 的增强版，是一个强大的软件包管理工具，是 yum 的替代命令。不同于 rpm 命令，dnf 能自动安装依赖的软件包。它的使用语法如下。

```
dnf [选项] 操作 [操作参数]

--advisory=<安装源ID>  指明从哪个安装源安装软件
    --disablerepo=<repoid> 禁用某个安装源
  --downloaddir=<path> 下载软件包存放的目录
          --downloadonly 只下载不安装
-y, --assumeyes 安装过程中的问题默认回答"是"
          --assumeno 默认回答"否"

install 安装一个或多个软件包
reinstall 重新安装软件包
search 搜索包名中包含关键字的软件包
provides 按软件包内的文件名搜索对应的软件包
list 列出安装软件的软件包
info 显示关于软件包或软件包组的详细信息
group 软件包组管理
remove 卸载软件包
autoremove 删除原先因为依赖关系安装的而现在不再需要的软件包
upgrade 升级软件包
upgrade-minimal 打上必须的补丁
check-update 检查是否有软件包升级
download
```

语法中的"选项""操作"和"操作参数"三者紧密相关，例如对于 install 操作，有的选项是不能用的，同时又有自己特殊的操作参数。关于 dnf 命令更详细的用法可参考在线帮助文档 man dnf、dnf help 或者 dnf help <操作>，如 dnf help install。

(1) 搜索软件包名称中包含 vim 的软件包：

```
dnf search vim
```

示例如图 5.12 所示。

(2) 搜索哪个软件包中包含文件 gzip：

```
dnf provides gzip
```

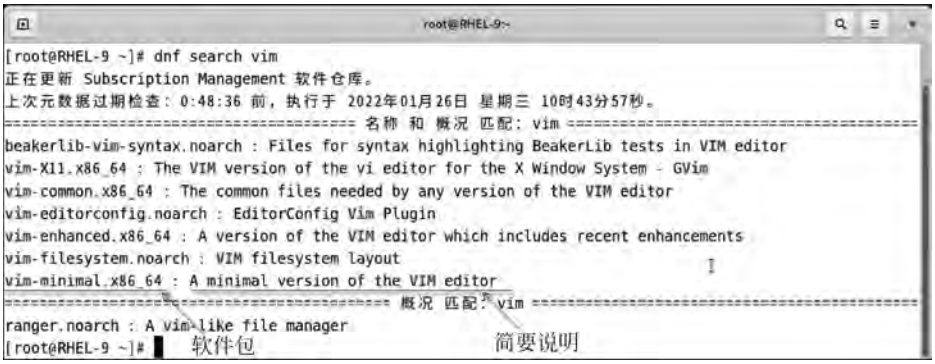


图 5.12 dnf search 示例

或者带全路径搜索，例如 dnf providers /usr/bin/ls。

(3) 查看 wget 软件包的说明信息：

```
dnf info wget
```

(4) 列出全部的软件包(包括安装的和未安装的)：

```
dnf list
```

(5) 列出已经安装的软件包：

```
dnf list --installed
```

(6) 安装 expect 软件包：

```
dnf install expect
```

(7) 安装名字以 vim 开头的所有软件包，且默认回答“是”：

```
dnf -y install vim*
```

(8) 安装已经下载到/tmp/目录中的 xz.rpm 软件包：

```
dnf install /tmp/xz.rpm
```

可以采用类似命令 dnf download wget 把 wget 软件包下载到当前目录。

(9) 安装一个存在于网上的软件包：

```
dnf install http://mirror.stream.centos.org/9-stream/AppStream/x86_64/os/Packages/wget-1.21.1-7.el9.x86_64.rpm
```

(10) 重新安装 vim 软件包。例如一个软件执行异常，可以重新安装它来修复：

```
dnf reinstall vim
```

(11) 安装包含文件/usr/bin/rpmsign 的软件包：

```
dnf install /usr/bin/rpmsign
```

(12) 卸载软件包 expect：

```
dnf remove expect
```

(13) 删除所有原先因为依赖关系被安装的而现在不再需要的软件包。例如用户安装的软件包被卸载了,但是由于某种原因导致卸载软件时没有同时卸载依赖包,可用本命令:

```
dnf autoremove
```

(14) 只下载 samba 及其依赖的软件包而不安装:

```
dnf --downloadonly --downloaddir=/tmp install samba
```

下载的软件包放在/tmp 目录中,也可以直接采用 dnf download samba 下载到当前目录。

(15) 列出所有模块:

```
dnf module list
```

模块由若干相关联的软件包组成,以完成某个特定功能,例如 virt 模块实现虚拟机功能,而软件组实现具体的应用,它包含更多的相关软件包,例如软件组 Container Management 就是实现容器应用。

(16) 安装 virt 模块:

```
dnf install @virt
```

模块名和软件组前要加@以便与普通软件包作区分。

(17) 列出所有的软件组:

```
dnf group list
```

(18) 安装用于开发的软件组:

```
dnf install '@Development Tools'
```

(19) 卸载容器软件组:

```
dnf remove '@Container Management'
```

2. Debian 11

在 Debian 下安装和卸载软件主要采用 dpkg 和 apt 工具,而 apt-get 命令更底层且技术性更强。

1) dpkg

dpkg: 软件包管理工具——安装、卸载、查询等,角色类似于红帽的 rpm 命令。

(1) 列出全部已经安装的软件包:

```
dpkg -l
```

翻页显示,按 q 退出。

(2) 列出名称中包含 vim 的软件包(包括安装的和未安装的):

```
dpkg -l *vim*
```

(3) 列出已安装软件包 bzip2 中的文件：

```
dpkg -L bzip2
```

(4) 显示软件包文件 xxd_8.2.2434-3+deb11u1_amd64.deb 中的文件：

```
dpkg -c/tmp/xxd_8.2.2434-3+deb11u1_amd64.deb
```

(5) 查看哪个软件包中包含文件/bin/ls：

```
dpkg -S /bin/ls
```

(6) 安装已经下载到当前目录中的软件包 xz.deb：

```
dpkg -i xz.deb
```

也可以一次性安装多个软件包，如 dpkg-i /tmp/abc.deb zlib.deb。

(7) 递归找出并安装目录./debs 及其子目录中的软件包文件(以.deb 结尾)：

```
dpkg -i -R ./debs
```

(8) 卸载 expect，但保留配置文件和数据：

```
dpkg -r expect
```

(9) 彻底卸载 vim，不保留配置文件和数据：

```
dpkg -P vim
```

2) apt

apt：软件包管理工具——安装、卸载、查询等，角色类似于红帽的 dnf 命令。

(1) 列出全部软件包(包括安装的和未安装的)：

```
apt list
```

(2) 列出已安装的软件包：

```
apt list --installed
```

(3) 列出所有包名以 vim 开头的软件包：

```
apt list vim*
```

(4) 按关键字 vim 搜索软件包：

```
apt search vim
```

(5) 查看软件包 openssh-server 的说明信息，包括依赖包：

```
apt show openssh-server
```

(6) 安装/tmp 目录下的 name.deb：

```
apt install /tmp/name.db
```

(7) 安装 tasksel 软件包：

```
apt install tasksel
```

(8) 安装名字以 vim 开头的所有软件包,且默认回答“是”:

```
apt -y install vim*
```

(9) 重新安装 gzip:

```
apt reinstall gzip
```

(10) 下载 xxd 软件包到当前目录下:

```
apt download xxd
```

(11) 尝试修复软件包之间的依赖关系,安装缺少的依赖软件包:

```
apt install -f
```

(12) 卸载软件 openssh-server,保留用户修改过的配置文件:

```
apt remove openssh-server
```

(13) 彻底干净地卸载软件 expect:

```
apt purge expect
```

(14) 卸载名称中包含 Office 的软件包,且默认回答“是”:

```
apt -y purge *office*
```

(15) 卸载所有自动安装且不再使用的软件包。由于依赖关系,用户在安装软件时会自动安装大量依赖包,以后当用户卸载软件后,会留下一些不再使用的依赖包,本命令就是把这些不再使用的依赖包卸载掉:

```
apt autoremove
```

(16) 显示全部的软件组:

```
tasksel --list-tasks
```

(17) 显示软件组 web-server 的发行信息:

```
tasksel --task-desc web-server
```

(18) 安装软件组 web-server:

```
tasksel install web-server
```

(19) 卸载软件组 web-server:

```
tasksel remove web-server
```

5.4.3 升级系统

1. RHEL 9.0

主要采用 dnf 工具。

(1) 检查是否有软件包需要升级,或者直接检查某个软件有无新版本:

```
dnf check-update
```

也可以单独检查某个软件包是否有新版本,如 dnf check-update wget。

(2) 升级所有软件(如果存在新版本):

```
dnf upgrade
```

也可以只升级单个软件包,如 dnf upgrade openssh-server。

(3) 修补存在安全漏洞的软件。如果某个软件存在新版本,但是已安装的旧版本并不存在的缺陷,那么此软件不会升级到新版本,这点与 dnf upgrade 命令不同:

```
dnf upgrade-minimal
```

2. Debian 11

主要采用 apt 工具。

(1) 同步安装源里的包索引文件到本地。每次更改安装源后都必须执行这个命令,否者没法安装软件:

```
apt update
```

(2) 检查并升级全部需要升级的软件包。如果某个软件包的新版本要依赖一个新的软件包,那么就安装这个新的依赖包。如果一个软件包 a 原来依赖 b,新版本却依赖 c,那么 a 就不会升级,c 也不会被安装:

```
apt upgrade
```

(3) 与 apt upgrade 命令的功能类似,不同的是:针对上面列举的情况,软件包 a 会被升级,c 会被安装,b 会被卸载。对系统执行 full-upgrade 比执行 upgrade 的风险更大,但是优点很明显,保持整个系统版本一致性:

```
apt full-upgrade
```

5.5 Vim

Vim 由荷兰的布莱姆·米勒开发,遵循 GPL 开源协议。Vim 的功能非常强大,几乎能够编辑当前所有的文件。Vim 的三种工作模式及其转换如图 5.13 所示。

从命令行运行 Vim 命令,首先进入的是命令模式,图 5.14 是不带文件名启动 Vim 后的命令模式,显示了 Vim 版本号、作者姓名。":help"可以获取 Vim 的在线帮助,":q"表示退出 Vim。

在 Vim 中输入:help 并按 Enter 键可获取简明扼要的帮助,里面告诉你如何获取更详细的在线帮助信息。

在命令模式下输入 a、A、i、I、o、O 中的任意一个字符就进入插入模式。注意,左下角的"--插入--"字样,表示当前是插入模式。在这个模式下输入的任何文字都将作为编辑内容被

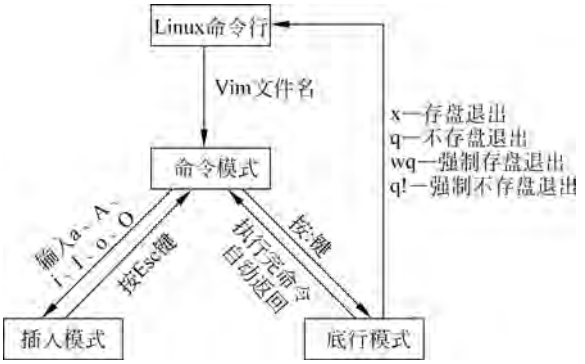


图 5.13 Vim 的三种工作模式及其转换

图 5.14 不带文件名启动 Vim 后的命令模式

显示在屏幕上,例如输入 Hello Vim !。

在插入模式下按 Esc 键(键盘左上角的那个键)返回命令模式,在命令模式下按:(冒号)键进入底行模式,如图 5.15 所示。在底行模式下,可以输入底行命令,按 Enter 键后开始执行底行命令,执行完后又返回命令模式,例如底行命令 w 表示存盘、x 表示存盘并退出 Vim 等。

图 5.15 Vim 的底行模式

在图 5.15 的“:”处输入 q!并按 Enter 键,这样即强制不存盘退出了 Vim 命令。

现在让我们编辑一个只包含一句话“Hello World!”的文件来开始我们的 Vim 学习之旅。先执行命令 rm /tmp/file.txt 把可能已经存在的文件删除掉。

(1) 在命令行上输入如下命令:

```
vim /tmp/file.txt
```

(2) Vim 首先进入命令模式,按 i 键进入插入模式(左下角会提示“-- 插入 --”)。

(3) 在插入模式下连续输入 Hello World!,如图 5.16 所示。

图 5.16 输入 Hello World!

(4) 按 Esc 键返回命令模式(连续按多次 Esc 键还是在命令模式),然后按:进入底行模式,输入底行命令 x(表示存盘并退出),如图 5.17 所示。

图 5.17 存盘并退出

按 Enter 键后,Vim 执行底行命令 x,此命令完成存盘并退出 Vim,从而返回到 Bash 的命令行。到此,一个只包含 Hello World! 的文件产生了,这算是采用 Vim 编辑的处女作了。可采用命令 more /tmp/file.txt 查看文件的内容。

进入 Vim 后,各种编辑命令少说也有上百个,熟练掌握如此众多的命令绝非一朝一夕的事儿,需要智力、体力和耐力,要像鲁班学艺般勤奋。下面开始由易到难用实例来讲述 Vim 的用法。表 5.3~表 5.10 列举了 Vim 的常用编辑命令。

表 5.3 中只列举了常用的几个配置项,在 Vim 的底行模式输入 set! all 可以查看全部的配置项。设置 Vim 的工作环境有两种方法:一是直接作为底行命令;二是把环境设置命令加到一个~/.vimrc 文件中,这样 Vim 启动时会自动读取此文件并设置好环境。图 5.18 就是在底行执行环境设置命令 set nu 来显示行号。

表 5.3 设置 Vim 的工作环境

序号	Vim 命令	说　明
1	set autoindent	自动缩进
2	set nu	显示行号
3	set ts=4	设置一个 Tab 键等于 4 个空格
4	syntax enable	语法高亮显示
5	set wrap	折行显示(超过屏幕宽度的行分成多行显示)
6	set hlsearch	高亮显示匹配的单词
7	set no *	取消设置。例如 set nonu 不显示行号、set nowrap 不折行等
8	help	获取帮助信息

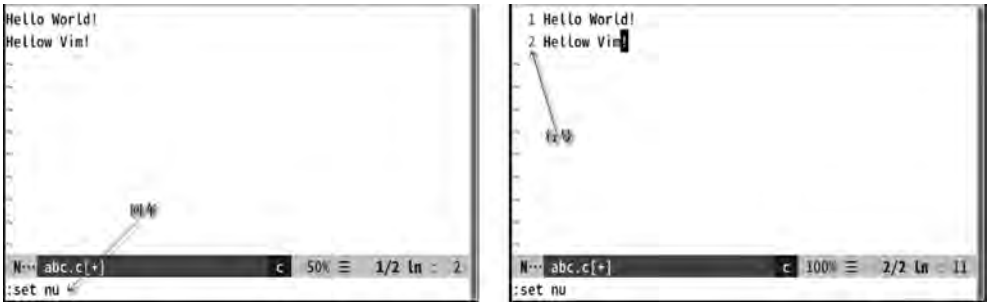


图 5.18 显示行号

强烈建议采用第二种方法,即把需要工作环境设置命令放到~/.vimrc 文件中,Vim 在启动的时候都会先执行这个文件中的命令。例如作者的~/.vimrc 内容如下:

```
set autoindent
syntax enable
syntax on
set nu
set nowrap
set sw = 4 ts = 4
set softtabstop = 4
set shiftwidth = 4
set hlsearch
set ruler
set showcmd
set cindent
set smartindent
set tabstop = 4
set cinoptions = {0,1s,t0,n-2,p2s,(03s, = .5s,> 1s, = 1s, :1s
filetype on
colorscheme desert
```

注意:.vimrc 文件中不能出现错误命令,否则启动 Vim 就会报错,这时建议删除家目录中的.vimrc 后再重新产生此文件。

表 5.4 进入插入模式(在命令模式下)

序号	Vim 命令	说 明
1	i	在当前位置(光标所在的位置)前插入
2	I	在光标所在的行首插入
3	a	在当前位置(光标所在的位置)后插入
4	A	在光标所在的行尾插入
5	o	在当前行下方插入一空行,同时进入插入模式
6	O	在当前行上方插入一空行,同时进入插入模式
7	C	先删除到行尾的全部字符,然后进入插入模式

注意:这些命令只在命令模式下有效,在插入模式就相当于普通字符。

表 5.5 移动光标(在命令模式下)

序号	Vim 命令	说 明
1	h,j,k,l	上、下、左、右移动光标。当然采用光标键也可以,但建议不采用光标键
2	Ctrl+F	上翻页
3	Ctrl+B	下翻页
4	%	跳到匹配的括号处,如光标目前在"{"处,按%后跳到对应的"}"处
5	^	跳到行首
6	$	跳到行尾
7	gg	跳到文件第一行
8	[n]G	跳到第 *n* 行或者文件末尾(省略[n]时),例如 10G 就是跳到第 10 行
9	''	(连续两个单引号) 跳到上一次的位置
10	m <标签>	在光标所在行定义一个标签,标签只能是一个字母,如 ma 定义一个标签 a
11	'<标签>	光标跳到<标签>处。如'a 跳到之前已经定义的标签 a 处

表 5.6 查找和替换(在命令模式下)

序号	Vim 命令	说 明
1	/wlm	往屏幕下方查找 wlm,此后按 N 继续查找下一个(如果文件中包含多个 wlm),如果找到文件末尾还继续按 N,则又从头开始找
2	?key12	往屏幕上方查找 key12,此后按 N 继续查找下一个(如果文件中包含多个 key12)
3	:g/old1/s//new3/g	文件中所有的 old1 替换成 new3
4	:g/old/s//news/gc	类似上面的命令,但替换前要确认
5	:10,40g/abc/s//1234/g	将 10～40 行的 abc 替换成 1234

如果替换的关键字中包含"/"字符,要在前面加一个转义符"\",例如把所有 6|5 替换为 6/5,采用底行命令就是":g/6|5/s//6\/5/g"。

在长时间编辑文件时,经常采用":w"存盘是一个好习惯。在用 Vim 写程序代码时,如果要编译代码,建议采用 Ctrl+Z 快捷键把 Vim 临时切换到后台,这样可在前台编译程序,以后可执行命令 fg 再把 Vim 切回到前台。

表 5.7 存盘与退出(在命令模式下)

序号	Vim 命令	说 明
1	:x	存盘并退出
2	:q!	不存盘强制退出,如果没有改动,可以直接输入:q 退出
3	:w	存盘不退出
4	:wq	存盘并退出(效果与:x 相同)
5	:w newfile.txt	另存为 newfile.txt
6	Ctrl+Z	把 Vim 切换到后台(此后在 Bash 命令行执行 fg 切换回来),前后台更详细的内容参见 5.3.5 节

表 5.8 复制、粘贴与删除(在命令模式下)

序号	Vim 命令	说 明
1	dd	删除光标所在的一行,并同时做了复制
2	9dd	删除光标及光标之下的 9 行
3	d'<标签>	删除从光标到<标签>间的所有行,如 d'g,其中 g 是已定义的标签(如何定义标签,可参见表 5.5 中的第 10 行)
4	x	删除光标所在的一个字符
5	dw	删除一个单词
6	D	删除从光标到行尾的全部字符(Shift+D 键就是快速输入大写 D)
7	yy	复制 1 行(光标所在的行)
8	5yy	复制光标所在的行及以下的 4 行(一共 5 行)
9	"a8yy	复制光标及以下的 8 行到命名寄存器 a 中,命名寄存器可以用 a～z 和 0～9 中任何一个字符表示,一般用字母命名比较好
10	$ y	复制从光标处到行尾的字符
11	p	把刚刚复制的内容或删除的内容粘贴到光标所在行的下面
12	P	把刚刚复制的内容或删除的内容粘贴到光标所在行的上面
13	"bp	把命名寄存器 b 中的内容粘贴到当前行的下面。先要把内容复制到命名寄存器 b(参考上面第 9 条命令)
14	"dP	把命名寄存器 d 中的内容粘贴到当前行的上面

表 5.9 编辑命令(在命令模式下)

序号	Vim 命令	说 明
1	r	替换一个字符。如 rD,则把当前位置的字符替换为 D
2	J	合并两行(把下面一行拼接在当前行末尾)
3	cc 或者 S	替换一行
4	cw	替换一个单词
5	u	撤销之前的操作。可以连续按 u 一直撤销前面的操作
6	Ctrl+R	重做(效果与 u 命令相反)
7	.	重复上一次操作
8	～	字母大小写转换
9	<<	右移一个 Tab(缩进)
10	>>	左移一个 Tab
11	==	(两个等号)自动更正缩进
12	gg=G	按缩进格式进行全文整理。尤其适合整理程序源代码

表中的“.”命令表示重复上一次操作，例如使用 dw 删除了一个单词，然后移动光标到另外某个地方，然后按“.”再次把当前的单词删除了——重复了上次操作 dw。

表 5.10 多文件编辑

序号	Vim 命令	说 明
1	vim file1 file2 file3	同时编辑三个文件(此后可用底行命令“:n”切换到下一个，“:rew”跳回到第一个)
2	vim -o file1 file2	把整个屏幕分屏成两个编辑窗口，每一个编辑窗口里编辑一个文件。切换窗口参见本表第 11 条命令
3	:e abc. txt	在 Vim 中临时编辑 abc. txt(此后可按 Ctrl+Shift+^在两个文件中切换)
4	:10split file2	水平分屏，新屏幕 10 行，在其中编辑 file2 文件。如果直接用“:split file2”则是平均分屏
5	:vsplit file. txt	纵向分屏，然后在新屏中编辑 file. txt
6	:tabedit file3	新建一个分页，然后在新页中编辑文件 file3
7	gt	跳到下一个页
8	gT	跳到上一个页
9	:close	关闭当前页或者当前窗口
10	:only	关闭其他所有窗口，只留下当前窗口
11	Ctrl+W W	(同时按住 Ctrl 键和 W 键，然后松手再单独按一次 W 键)在窗口之间切换
12	Ctrl+W+	(同时按住 Ctrl 键和 W 键，然后松手再单独按一次+键)增大水平窗口行数
13	Ctrl+W-	减少水平窗口行数
14	10Ctrl+W _	当前窗口调整到 10 行
15	Ctrl+W h|j|k|l	在各个窗口之间移动光标。h-移到右边窗口，l-移到左边窗口
16	Ctrl+W H|J|K|L	移动窗口
17	:qall!	强制退出所有窗口和页
18	:xall	全部存盘并退出
19	:wall	全部存盘，但不退出

一个物理屏可以分成多个编辑页，一个编辑页又可以分成多个编辑窗口，每一个窗口中可以编辑独立的文件，如图 5.19 所示，分了三页，第二页又分成两个编辑窗口，在第一个页中编辑文件 abc. c，在第二个页中的第一个窗口中编辑文件. vimrc，第二个窗口中编辑 file.

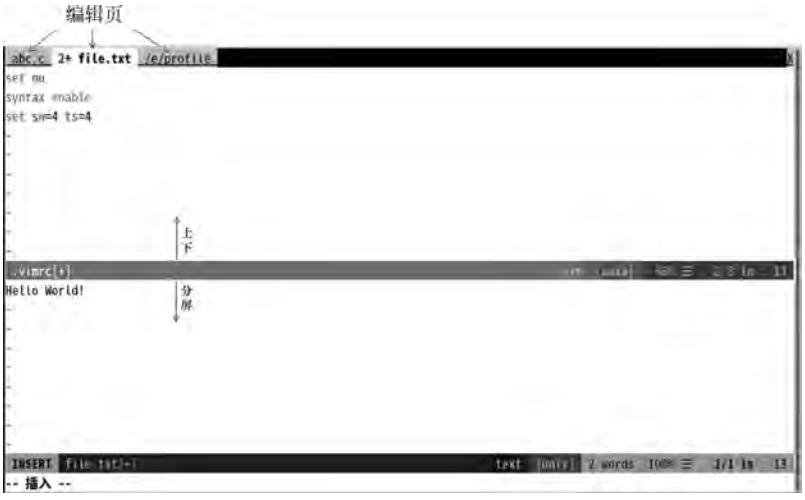


图 5.19 分页分屏编辑多个文件

txt,第三个页中编辑文件/e/profile,这样同时有 4 个文件处于编辑状态,可按 Ctrl+W W 在同一个页的不同窗口切换,按 gt 或者 gT 切换不同的页。

5.6 远程控制:OpenSSH

本节介绍一款使用非常广泛的 Linux 网络工具:安全远程登录和文件传输工具 OpenSSH。

对于机房的 Linux 系统,管理员常常采用远程控制的办法来维护它,这时就要用到一个大名鼎鼎的专门用于安全远程控制的开源软件 OpenSSH。早期经常使用 Telnet 登录 UNIX 机器,这个工具传输的信息是明文,别人很容易从网络上盗窃密码。现在几乎没有人采用它了,都转向了 OpenSSH,它是 SSH(Secure Shell 的缩写)协议的一个具体实现,SSH 协议规定传输的信息必须加密,OpenSSH 安全且功能强大。默认已经安装,也可以采用下面的命令手工安装:

```
dnf install openssh-server              #RHEL
apt install openssh-server              #Debian
```

OpenSSH 的配置文件放在/etc/ssh 目录下,其中 sshd_config 就是 OpenSSH 服务器的配置文件,使用默认配置即可,但为了加快登录速度,可以增加配置项"UseDNS no",即关闭登录时反向解析 IP。另外要注意的是配置参数 Port,它定义 SSH 服务器监听的端口,默认是 22,也可以改为其他端口,例如 7209,因此别人就不知道端口号,根本没法登录,这样在一定程度上增加了安全性。出于安全考虑,默认禁止 root 远程登录,如果实在要开启,就把配置文件中的 PermitRootLogin 定义为 yes。配置文件一旦被修改,就要运行下述命令使新配置参数生效:

```
systemctl reload sshd.service
```

或者干脆重启 SSH 服务:

```
systemctl restart sshd.service
```

SSH 服务的默认端口是 22/TCP,使用命令 ss -tnl 可查看全部被监听的 TCP 端口,如图 5.20 所示。

```
moodisk@Debian-11:~$ ss -t -l -n
State          Recv-Q          Send-Q                    Local Address:Port
LISTEN         0               128                           0.0.0.0:22
LISTEN         0               4096                                *:4369
LISTEN         0               128                              [::]:22
moodisk@Debian-11:~$
```

图 5.20 查看被监听的 TCP 端口

由此可见 OpenSSH 同时支持 IPv4 和 IPv6。

如果服务器上的端口 22 被监听,且没有防火墙阻隔,就可以从世界上任何一台计算机远程登录过去,只要网络是通的。客户端软件包括两类:一类是远程登录工具;另一类是文件传输工具。

Linux 系统和 Windows 10 都可以直接用 SSH 命令登录到另一台已经启动了 SSH 服务的 Linux 系统，另外 SSH 还可以建立安全隧道以及转发 X11 协议等，功能非常强大。

(1) 用当前的用户登录到 192.168.0.21：

```
ssh 192.168.0.21
```

按 Enter 键后会提示你输入对方计算机上用户的密码。例如李木生这个人采用 lisi 账号进入计算机 A，然后执行此命令登录计算机 B(IP 地址是 192.168.0.21)，能成功登录的条件是 B 计算机上存在 lisi 账号，且 alice 知道 B 上 lisi 账号的密码。

(2) 采用 john 用户登录到另一台计算机：

```
ssh john@10.10.1.20
```

(3) SSH 服务的端口改为 7209，SSH 命令需要采用参数-p 7209：

```
ssh -p 7209 root@www.veryopen.org
```

对于 Windows 10 之前的版本，可以采用 putty.exe 或 mobaxterm 登录到 Linux，如图 5.21 所示。

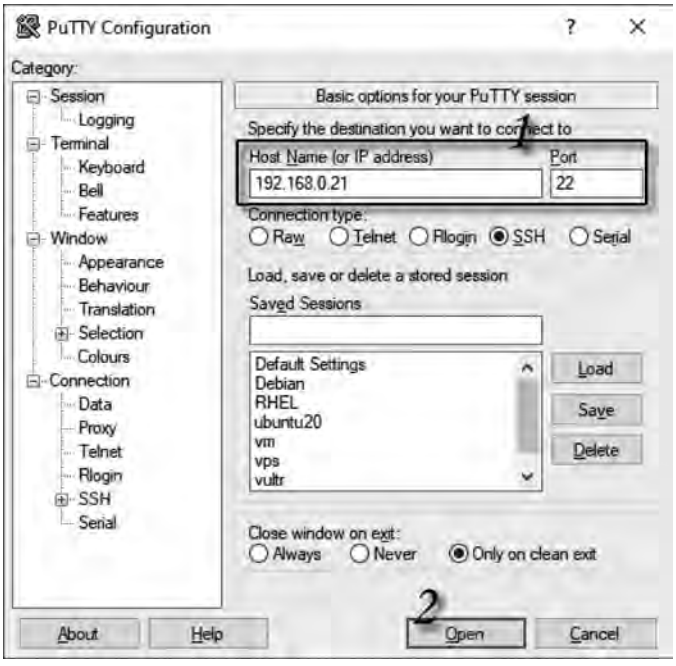


图 5.21 Windows 上的登录工具 putty.exe

基于 SSH 协议的文件传输工具有很多，这里介绍 scp 和 sftp，scp 是非交互命令，而 sftp 是交互命令。

(4) 把当前目录下的文件 file.txt 上传到 192.168.10.156 的/tmp 目录下：

```
scp file.txt 192.168.10.156:/tmp
```

由于省略了用户名，所以采用当前的用户，进一步说明请参考(1)。

(5) 把机器 10.20.10.5 上的文件/etc/profile 下载到当前目录下：

```
scp 10.20.10.5:/etc/profile ./
```

(6) 使用 alice 用户来下载其家目录中的 data.tar.xz 到本机的/tmp 目录：

```
scp alice@192.168.0.50:data.tar.xz /tmp
```

(7) 把目录/opt/erp 上传到对方计算机上 john 用户的家目录中：

```
scp -r /opt/erp john@192.168.0.45:.
```

凡是上传或下载整个目录，需要带参数-r。

(8) 把对方计算机上的/tmp 目录中文件名以 data 开头的所有文件下载到本机：

```
scp -P 7209 root@www.veryopen.org:/tmp/data* ./
```

当端口不是 22 时，scp 命令需要带参数-P 指定具体的端口号。

(9) 限制下载速度为 100kb/s：

```
scp -l 100 ssh_server:1.dat ./
```

(10) 与 192.168.10.100 建立文件传输会话：

```
sftp alice@192.168.10.100
sftp> help
```

成功建立文件传输会话后会出现提示符“sftp>”，此后就可以交互式地上传和下载文件了。输入 help 或者？列出全部的交互命令及简要的说明(上面最后一个例子中的第二行)，最常用的交互命令有 get(下载)、put(上传)、ls(列出对方计算机上的文件)、lls(列举本地计算机上的文件)、cd(改变对方计算机上的当前目录)、lcd(改变本地计算机上的当前目录)、pwd(显示对方计算机的当前目录)、lpwd(显示本机的当前目录)、quit(退出 sftp 命令)等。请看下面交互的例子。

```
sftp> ls                          #列出对方计算机上的文件
sftp> get file.txt                #下载文件 file.txt
sftp> get file.txt /tmp           #下载文件 file.txt 到本地/tmp 目录中
sftp> put /etc/profile            #上传文件/etc/profile
sftp> put file.txt /tmp           #把 file.txt 上传到对方的/tmp 目录下
sftp> cd /tmp                     #改变对方的当前目录到/tmp
sftp> lcd /etc                    #改变本地的工作目录为/etc
sftp> mkdir abc                   #在对方计算机上创建目录/tmp/abc
sftp> rm file2.txt                #删除对方的文件 file2.txt
sftp> quit                        #退出 sftp 程序
```

Windows 10 之前版本可采用 WinSCP 工具，从 https://winscp.ne 网站下载并安装，然后启动它，界面如图 5.22 所示。

图 5.22 的右下角部分，左边是本地硬盘窗口，右边是对方计算机的窗口，右击右边窗口中的文件或者目录，然后选择“下载”命令下载；同样可以右击左边窗口的文件或者目录，然后选择“上传”命令进行上传操作。可以通过按住 Ctrl 键并单击来选择多个文件或目录，以便下载或上传多个文件或目录。

图 5.22 Windows 上的 WinSCP 工具

5.7 知识拓展与作业

5.7.1 知识拓展

(1) 学习 Bash 的文件描述符复制和移动：[n]< &digit-,[n]> &digit-。

(2) 学习几个高级命令：sed、mawk、nc、tcpdump、wireshark。

(3) 了解搭建 Debian 11 本地安装源。

(4) 了解 OpenSSH 的高级功能。端口转发，建立机器间信任关系，修改默认的服务端口 22，局域网穿透技术(参见作者博客)，采用密钥安全登录，利用动态端口转发实现 VPN。

(5) 掌握 Vim 的高级用法。

5.7.2 作业

(1) 解释命令串的作用：grep /bin/bash /etc/passwd 2 >/dev/null | sort -k 1 1 >/tmp/users.txt 2 >/dev/null && cat /tmp/users.txt | wc -l

(2) 把本地目录/etc/init.d 打包压缩为 init.tar.bz2，然后把此文件上传到 192.168.10.101 的/tmp/目录中，请写出命令。