第 3 章

Linux 系统常用命令

操作系统安装完成后，可以通过一些命令来查看操作系统的相关信息。登录系统后输入 uname -a 命令，可显示计算机及操作系统的相关信息。输入 cat /proc/version 命令可查看正在运行的内核版本。输入 cat /etc/issue 命令可显示发行版本信息。lsb_release -a 命令适用于所有的 Linux，包括 RedHat、SuSE、Debian 等发行版，但是在 Debian 下需要安装 lsb。如果不知道命令的意思，则可以通过“man 命令名称”查看它的使用方式及详细信息。本章所介绍的命令都会在后续的云计算环境部署和开发中经常使用，这些是 Linux 使用者必须掌握的基础知识。

3.1 熟用 Linux 系统最常用的 ls 命令

这个命令可以说是 Linux 系统中用得最频繁的命令之一，它可以用来查看文件目录信息，也可以与其他选项组合使用。ls 命令的含义是 list（显示）当前目录中的文件名字，ls 命令跟 Dos 下的 dir 命令是一样的，都用来列出目录下的文件。注意，当不加参数时它显示除隐藏文件外的所有文件及目录的名字。

1. 基本使用

（1）ls：显示当前目录下的文件和目录。

（2）ls -l：显示当前目录下的文件和目录的详细信息。

（3）ls -l * |grep "^d"：显示当前目录下的子目录信息。

（4）ls -l * |grep "^-"：显示当前目录下的文件信息。

（5）ls -lh：显示当前目录下的文件及其详细信息，并把相关信息转换为方便阅读的单位。

（6）ls -a：显示当前目录中的所有文件，包含隐藏文件。

（7）ls|wc -l：统计当前目录下的文件和目录数量。

（8）ls -lt：显示当前目录下的文件和目录的详细信息，并按最后修改时间排序。

（9）ls -lS：显示当前目录下的文件和目录的详细信息，并按文件大小排序。

（10）其他命令。

-a：列出目录下的所有文件，包括以“.”开头的隐含文件。

-b：把文件名中不可输出的字符用反斜杠加字符编号（就像在C语言里一样）的形式列出。

-c：输出文件的i节点的修改时间，并以此排序。

-d：将目录像文件一样显示，而不是显示其下的文件。

-e：输出时间的全部信息，而不是输出简略信息。

-f -U：对输出的文件不排序。

-g：没有特殊作用。

-i：输出文件的i节点的索引信息。

-k：以k字节的形式表示文件的大小。

-l：列出文件的详细信息。

-m：横向输出文件名，并以“,”作为分隔符。

-n：用数字的UID和GID代替名称。

-o：显示文件除组信息外的详细信息。

-p -F：在每个文件名后附上一个字符以说明该文件的类型，“*”表示可执行的普通文件；“/”表示目录；“@”标示符号链接；“|”表示FIFOs；“=”表示套接字（sockets）。

-q：用“?”代替不可输出的字符。

-r：对目录反向排序。

-s：在每个文件名后输出该文件的大小。

-t：以时间排序。

-u：以文件上次被访问的时间排序。

-x：按列输出，横向排序。

-A：显示除“.”和“..”外的所有文件。

-B：不输出以“～”结尾的备份文件。

-C：按列输出，纵向排序。

-G：输出文件的组的信息。

-L：列出链接文件名而不是链接到的文件。

-N：不限制文件长度。

-Q：把输出的文件名用双引号括起来。

-R：列出所有子目录下的文件。

-S：以文件大小排序。

-X：以文件的扩展名（最后一个“.”后的字符）排序。

-1：一行只输出一个文件。

--color=no：不显示彩色文件名。

--help：在标准输出上显示帮助信息。

--version：在标准输出上输出版本信息并退出。

命令组合是一个非常灵活的运用，根据自己的应用场景及自己需要查看的信息，进行命令项的选择和组合，这里主要对常用的命令进行了举例，目的在于引导读者如何进行命令项的组合使用。在实际运用中，通过不断的练习达到熟能生巧的目的，如果有些选项不记得了，则可以用 help 命令进行查看。

2. 从系统目录认识 Linux 系统

ls 命令的组合选项除了查看当前目录的信息，还可以查看指定的文件目录，例如查看系统的根目录时可使用命令 ls /，结果如图 3-1 所示。

```
[root@localhost ~]# ls /
bin   dev  home  lost+found  mnt  proc  sbin     srv  tmp  var
boot  etc  lib   media       opt  root  selinux  sys  usr
```

图 3-1　Linux 根目录文件夹

有些文件夹目录采用的是连接的方式，与 Windows 系统下的快捷方式有点类似，转换为对应的树状目录结构如图 3-2 所示。

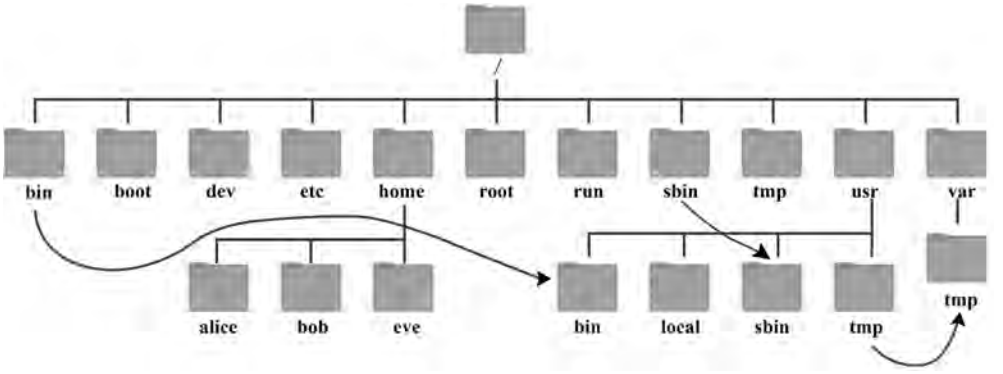

图 3-2　Linux 根目录结构图

以下分别对这些目录进行简单说明。

(1) /bin：bin 是 Binaries(二进制文件)的缩写，该目录下存放着最经常使用的命令。

(2) /boot：这里存放的是启动 Linux 时使用的一些核心文件，包括一些链接文件及镜像文件。

(3) /dev：dev 是 Device(设备)的缩写，该目录下存放的是 Linux 的外部设备，在 Linux 中访问设备的方式和访问文件的方式是相同的。

(4) /etc：etc 是 Etcetera(等)的缩写，该目录用来存放所有的系统管理所需要的配置文件和子目录。

(5) /home：用户的主目录，在 Linux 中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的，如图 3-2 中的 alice、bob 和 eve。

(6) /lib：lib 是 Library(库)的缩写，该目录里存放着系统最基本的动态连接共享库，其作用类似于 Windows 里的 DLL 文件。绝大多数的应用程序需要用到这些共享库。

(7) /lost+found：该目录一般情况下是空的，当系统非法关机后，这里就会存放一些与非法关机相关的文件。

(8) /media：Linux 系统会自动识别一些设备，例如 U 盘、光驱等，识别后，Linux 会把识别的设备挂载到这个目录下。

(9) /mnt：系统提供该目录是为了让用户临时挂载别的文件系统，用户可以将光驱挂载在/mnt/上，然后进入该目录就可以查看光驱里的内容了。

(10) /opt：opt 是 Optional(可选)的缩写，这是给主机额外安装软件所创建的目录。例如安装一个 Oracle 数据库就可以放到这个目录下，该目录默认为空。

(11) /proc：proc 是 Processes(进程)的缩写，/proc 是一种伪文件系统(虚拟文件系统)，存储的是当前内核运行状态的一系列特殊文件。该目录是一个虚拟的目录，它是系统内存的映射，可以通过直接访问这个目录获取系统信息。

这个目录的内容不在硬盘上而是在内存里，也可以直接修改里面的某些文件，例如可以通过下面的命令屏蔽主机的 ping 命令，使别人无法 ping 该机器：

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

(12) /root：该目录为系统管理员的主目录，也称作超级权限者的用户主目录。

(13) /sbin：s 就是 Super User 的意思，是 Super User Binaries(超级用户的二进制文件)的缩写，这里存放的是系统管理员使用的系统管理程序。

(14) /seLinux：是 RedHat/CentOS 所特有的目录，SeLinux 是一个安全机制，类似于 Windows 的防火墙，但是这套机制比较复杂，该目录就是存放 SeLinux 相关的文件的。

(15) /srv：该目录用于存放一些服务启动后需要提取的数据。

(16) /sys：这是 Linux 2.6 内核的一个很大的变化。该目录下安装了 2.6 内核中新出现的一个文件系统 sysfs。sysfs 文件系统集成了 3 种文件系统的信息：针对进程信息的 proc 文件系统、针对设备的 devfs 文件系统及针对伪终端的 devpts 文件系统。该文件系统是内核设备树的一个直观反映。当一个内核对象被创建时，对应的文件和目录也在内核对象子系统中被创建。

(17) /tmp：tmp 是 Temporary(临时)的缩写，该目录用来存放一些临时文件。

(18) /usr：usr 是 UNIX Shared Resources(共享资源) 的缩写，这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似于 Windows 下的 Program Files 目录。

(19) /usr/bin：系统用户使用的应用程序。

(20) /usr/sbin：超级用户使用的比较高级的管理程序和系统守护程序。

(21) /usr/src：内核源代码默认的存放目录。

(22) /var：var 是 Variable(变量)的缩写，该目录中存放着在不断扩充着的目录和文件，很多用户习惯将那些经常被修改的目录放在这个目录下，包括各种日志文件。

(23) /run：临时文件系统，用于存储系统启动以来的信息。当系统重启时，该目录下的文件应该被删掉或清除。如果系统上有/var/run 目录，则应该让它指向 run。

在 Linux 或 UNIX 操作系统中，所有的文件和目录都被组织成以一个根节点开始的倒置的树状结构。文件系统的最顶层是由根目录开始的，系统使用“/”来表示根目录。在根目录下既可以是目录，也可以是文件，而每个目录中又可以包含子目录和文件。如此反复就可以构成一个庞大的文件系统。在 Linux 文件系统中有两个特殊的目录，一个是用户所在的工作目录，也叫当前目录，可以使用一个点“.”来表示；另一个是当前目录的上一级目录，也叫父目录，可以使用两个点 “..”来表示。

.：代表当前的目录，也可以用“./”表示。

..：代表上一级目录，也可以用“../”表示。

如果一个目录或文件名以一个点“.”开始，则表示这个目录或文件是一个隐藏目录或文件（如.bashrc），即以默认方式查找时，不显示该目录或文件。

在 Linux 系统中，有几个目录是比较重要的，平时需要注意不要误删除或者随意更改内部文件。

（1）/etc：系统中的配置目录，如果更改了该目录下的某个文件，则可能会导致系统不能启动。

（2）/bin、/sbin、/usr/bin 和/usr/sbin：系统预设的执行文件的放置目录，例如 ls 就在/bin/ls 目录下。

值得注意的是，/bin 和/usr/bin 用于存放系统用户使用的指令（除 root 外的普通用户），而/sbin 和/usr/sbin 则用于存放供 root 使用的指令。

（3）/var：这是一个非常重要的目录，系统上运行了很多程序，每个程序都会有相应的日志产生，而这些日志会被记录到该目录的/var/log 目录下，另外 mail 的预设也放置在这里。

3.2 巧用 cat 命令进行文件的读与写

cat 命令主要用来查看文件内容、创建文件、文件合并和追加文件内容等。cat 是一个文本文件查看和连接工具。cat --help 可以查看 cat 帮助信息，如各种参数的使用方法，当然也可以用 man cat 来查看，建议遇到不懂的命令用法时，用--help 或 man 来查看帮助信息，养成好习惯。filename 为文件名，即系统中需要查看的文件名字。与这个命令功能相似的命令有 tac、less、head、tail 和 more。

1. 创建文件

1）创建一个新文件

创建一个新文件会用到“>”符号，这个符号是重定向的意思，会覆盖原来文件的内容，没有文件时会自动创建。创建文件时要设置文件结束标志，也就是<< EOF，可以把 EOF 换成别的字符，注意大小写，当文件内容写完后要输入结束标志 EOF，这时命令会正确结束，表示成功创建文件并且写进内容，命令如下：

```
cat > f1.txt << EOF
> Hello, my name is yangyi.
> Nice to meet you.
> EOF
```

2）将内容追加到文件

将内容追加到文件会用到“>>”符号，即表示追加内容，不会覆盖原文件内容，只会在原文件内容下面追加所输入的内容，命令如下：

```
cat >> f1.txt << EOF
> Something content is added.
> Just do it.
> EOF
```

3）合并多个文件

把文件 f2.txt、f3.txt 和 f4.txt 的内容写入 f5.txt 文件中，如果 f5.txt 文件以前有内容，则先会清除它们，然后写入合并后的内容。如果不想清除原文件内容，则可以把单边号“>”变成双边号“>>”，命令如下：

```
cat f2.txt f3.txt f4.txt > f5.txt
```

2. 查看文件

cat f1.txt：查看 f1.txt 文件的内容。

cat-n f1.txt：查看 f1.txt 文件的内容，并且由 1 开始对所有输出行进行编号。

cat-b f1.txt：查看 f1.txt 文件的内容，用法与 -n 相似，只不过对于空白行不编号。

cat-s f1.txt：当遇到连续两行或两行以上的空白行时，替换为一行的空白行。

cat-e f1.txt：在输出内容的每行后面加一个 $ 符号。

cat f1.txt f2.txt：同时显示 f1.txt 和 f2.txt 文件的内容，注意文件名之间以空格分隔，而不是逗号。

cat -n f1.txt > f2.txt：对 f1.txt 文件中的每行加上行号后，写入 f2.txt 文件中，会覆盖原来的内容，如果文件不存在，则创建它。

cat -n f1.txt >> f2.txt：对 f1.txt 文件中的每行加上行号后，追加到 f2.txt 文件中，不会覆盖原来的内容，如果文件不存在，则创建它。

3. 其他选项

-A：--show-all 等价于-vET。

-b：--number-nonblank 对非空输出行编号，即在每行前显示所在的行号。

-e：等价于-vE。

-E：--show-ends 在每行结束处显示 $ 。

-n：--number 对输出的所有行编号，即在每行前显示所在的行号。

-s：--squeeze-blank 不输出多行空行。

-t：与-vT 等价。

-T：--show-tabs 将跳字符显示为^I。

-u：被忽略。

-v：--show-nonprinting 使用^和 M-引用，除了 LFD 和 TAB 外。

--help：显示此帮助信息并离开。

3.3 强大的 grep 文本搜索工具

Linux 系统中 grep 命令是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。grep 的全称是 Global Regular Expression Print，作为 Linux 中最为常用的三大文本(awk、sed、grep)处理工具之一，掌握好其用法很有必要。

UNIX 的 grep 宗族包含 grep、egrep 和 fgrep。egrep 和 fgrep 的指令只跟 grep 有很小的不同。egrep 是 grep 的扩展，支持更多的 re 元字符，fgrep 就是 fixedgrep 或 fastgrep，它们把一切的字母都看作单词，也就是说，将正则表达式中的元字符表示回其自身的字面含义，不再特殊。Linux 运用 GNU 版本的 grep，它功用更强，可以经过-G、-E、-F 指令行选项来运用 egrep 和 fgrep 的功能。

1. 工作原理

grep 的工作方式是在一个或多个文件中查找字符串模板。假如模板包含空格，则有必要被引用，模板后的一切字符串被看作文件名。查找的结果被送到规范输出，不影响原文件内容。grep 可用于 Shell 脚本，因为 grep 通过一个状况值来说明查找的状况，假如模板查找成功，则返回 0；假如查找不成功，则返回 1；假如查找的文件不存在，则返回 2。利用这些返回值就可进行一些自动化的文本处理工作。

2. 正则表达式

1) 基本正则表达式

匹配字符，表达式如下。

.：任意一个字符。

[abc]：表示匹配一个字符，这个字符必须是 abc 中的一个。

[a-zA-Z]：表示匹配一个字符，这个字符必须是 a～z 或 A～Z 这 52 个字母中的一个。

[^123]：匹配一个字符，这个字符可以是除了 1、2、3 以外的所有字符。

对于一些常用的字符集，系统的定义如下：

[A-Za-z]等价于 [[:alpha:]]。

[0-9]等价于 [[:digit:]]。

[A-Za-z0-9]等价于 [[:alnum:]]。

tab，space 等价于空白字符 [[:space:]]。

[A-Z]等价于 [[:upper:]]。

[a-z]等价于 [[:lower:]]。

标点符号等价于 [[:punct:]]。

匹配次数,表达式如下。

\{m,n\}:匹配其前面出现的字符至少 m 次,至多 n 次。

\?:匹配其前面出现的内容 0 次或 1 次,等价于\{0,1\}。

:匹配其前面出现的内容任意次,等价于\{0,\},所以 "."表示任意字符任意次,即无论什么内容全部匹配。

位置锚定,表达式如下。

^:锚定行首。

$:锚定行尾,"^$"用于匹配空白行。

\b 或\<:锚定单词的词首,如"\blike"不会匹配 alike,但会匹配 liker。

\b 或\>:锚定单词的词尾,如"\blike\b"不会匹配 alike 和 liker,只会匹配 like。

\B:与\b 的作用相反。

分组及引用,表达式如下。

\(string\):将 string 作为一个整体方便后面引用。

\1:引用第 1 个左括号及其对应的右括号所匹配的内容。

\2:引用第 2 个左括号及其对应的右括号所匹配的内容。

\n:引用第 n 个左括号及其对应的右括号所匹配的内容。

2)扩展正则表达式

注意,当使用扩展的正则表达式时要加-E 选项,或者直接使用 egrep。

匹配字符:这部分和基本正则表达式一样。

匹配次数,表达式如下。

*:和基本正则表达式一样。

?:基本正则表达式是\?,问号前没有“\”。

{m,n}:相比基本正则表达式也没有了“\”。

+:匹配其前面的字符至少一次,相当于{1,}。

位置锚定:和基本正则表达式一样。

分组及引用,表达式如下。

(string):相比基本正则表达式也没有了“\”。

\1:引用部分和基本正则表达式一样。

\n:引用部分和基本正则表达式一样。

a|b:匹配 a 或 b,注意 a 是指 | 的左边的整体,b 也同理。例如 C|cat 表示的是 C 或 cat,而不是 Cat 或 cat,如果要表示 Cat 或 cat,则应该写为(C|c)at。记住(string)除了用于引用还用于分组。

默认情况下,正则表达式的匹配工作处于贪婪模式下,也就是说它会尽可能长地去匹配,例如某行有字符串 abacb,如果搜索内容为 "a.*b",则会直接匹配 abacb 这个字符串,

而不会只匹配 ab 或 acb。所有的正则字符，如 [、* 、(等，若要搜索 *，而不是想把 * 解释为重复先前字符任意次，则可以使用 * 来转义。

3. 应用实践

1）grep 正则表达式元字符集

^锚定行的开始，如'^grep'，匹配所有以 grep 开头的行。

$ 锚定行的结束，如'grep$'，匹配所有以 grep 结尾的行。

. 匹配一个非换行符的字符，如'gr.p'，匹配 gr 后接一个任意字符，然后是 p。

* 匹配零个或多个先前字符，如'*grep'，匹配所有一个或多个空格后紧跟 grep 的行。. 和 * 一起用代表任意字符。

[]匹配一个指定范围内的字符，如'[Gg]rep'，匹配 Grep 和 grep。

[^]匹配一个不在指定范围内的字符，如'[^A-FH-Z]rep'，匹配不包含 A～F 和 H～Z 的一个字母开头，紧跟 rep 的行。

\(..\)标记匹配字符，如'\(love\)'，love 被标记为 1。

\>锚定单词的结束，如'grep\>'，匹配包含以 grep 结尾的单词的行。

x\{m\}重复字符 x 共 m 次，如'o\{5\}'，匹配包含 5 个 o 的行。

x\{m,\}重复字符 x 至少 m 次，如'o\{5,\}'，匹配至少包含 5 个 o 的行。

x\{m,n\}重复字符 x 至少 m 次，不多于 n 次，如'o\{5,10\}'，匹配 5～10 个 o 的行。

\w 匹配文字和数字字符，也就是[A-Za-z0-9]，如'G\w*p'，匹配以 G 后跟 0 个或多个文字或数字字符，然后是 p。

\b 单词锁定符，如'\bgrep\b'，只匹配 grep。

2）grep 匹配的实例

grep -c "48" test.txt，统计所有以 48 字符开头的行有多少。

grep -i "May" test.txt，不区分大小写查找 May 所有的行。

grep -n "48" test.txt，显示行号；显示匹配字符 48 的行及行号，相同于 nl test.txt | grep 48。

grep -v "48" test.txt，显示输出没有字符 48 的所有的行。

grep "471" test.txt，显示输出字符 471 所在的行。

grep "48;" test.txt，显示输出以字符 48 开头，并在字符 48 后是一个 Tab 键所在的行。

grep "48[34]" test.txt，显示输出以字符 48 开头，第 3 个字符是 3 或 4 的所有的行。

grep "^[^48]" test.txt，显示输出行首不是字符 48 的行。

grep "[Mm]ay" test.txt，设置大小写查找：显示输出第 1 个字符以 M 或 m 开头，以字符 ay 结束的行。

grep "K…D" test.txt，显示输出第 1 个字符是 K，第 2、第 3 和第 4 个字符是任意字符，第 5 个字符是 D 所在的行。

grep "[A-Z][9]D" test.txt，显示输出第 1 个字符的范围是 A～Z，第 2 个字符是 9，第 3 个字符是 D 的所有的行。

grep "[35]..1998" test.txt,显示第1个字符是3或5,第2和第3个字符是任意字符,以1998结尾的所有行。

grep "4\{2,\}" test.txt,模式出现概率查找:显示输出字符4至少重复出现两次的所有行。

grep "9\{3,\}" test.txt,模式出现概率查找:显示输出字符9至少重复出现3次的所有行。

grep "9\{2,3\}" test.txt,模式出现概率查找:显示输出字符9重复出现的次数在一定范围内,重复出现2次或3次的所有行。

grep-n "^$" test.txt,显示输出空行的行号。

ls-l |grep "^d",如果要查询目录列表中的目录,则同:ls-d *。

ls-l |grep "^d[d]",在一个目录中查询不包含目录的所有文件。

ls-l |grpe "^d.....x..x",查询其他用户和用户组成员有可执行权限的目录集合。

在当前目录中,查找后缀有file字样的文件中包含test字符串的文件,并打印该字符串的行。此时,可以使用的命令如下:

```
grep test *file
```

以递归的方式查找符合条件的文件。例如,查找指定目录/etc/acpi及其子目录(如果存在子目录)下所有文件中包含字符串update的文件,并打印该字符串所在行的内容,使用的命令如下:

```
grep -r update /etc/acpi
```

反向查找,前面各个例子是查找并打印符合条件的行,通过-v参数可以打印不符合条件行的内容。

查找文件名中包含test的文件中不包含test的行,命令如下:

```
grep -v test *test*
```

从根目录开始查找所有扩展名为.log的文本文件,并找出包含ERROR的行,命令如下:

```
$ find / -type f -name "*.log" | xargs grep "ERROR"
```

例如从当前目录开始查找所有扩展名为.in的文本文件,并找出包含thermcontact的行,命令如下:

```
find . -name "*.in" | xargs grep "thermcontact"
```

在网络配置文件/etc/sysconfig/network-scripts/ifcfg-ens33 中检索出所有的 IP，命令如下：

```
grep -o -E "inet addr:[0-9.]+" /etc/sysconfig/network-scripts/ifcfg-ens33
```

4. 详细参数

1）匹配模式

-E,--extended-regexp：扩展正则表达式 egrep。

-F,--fixed-strings：一个换行符分隔的字符串的集合 fgrep。

-G,--basic-regexp：基本正则。

-P,--perl-regexp：调用的 perl 正则。

-e,--regexp=PATTERN：后面跟正则模式，默认无。

-f,--file=FILE：从文件中获得匹配模式。

-i,--ignore-case：不区分大小写。

-w,--word-regexp：匹配整个单词。

-x,--line-regexp：匹配整行。

-z,--null-data：以字节 0 而不是换行符结尾的数据行。

2）其他项

-s,--no-messages：不显示错误信息。

-v,--invert-match：显示不匹配的行。

-V,--version：显示版本号。

--help：显示帮助信息。

--mmap：如果可能，则使用系统内存映象作为输入。

3）输入控制

-m,--max-count=NUM：匹配的最大数。

-b,--Byte-offset：显示匹配串在匹配到的行中的位置。

-n,--line-number：输出匹配行内容并显示行号。

--line-buffered：刷新输出每行。

-H,--with-filename：当搜索多个文件时，显示匹配文件名前缀。

-h,--no-filename：当搜索多个文件时，不显示匹配文件名前缀。

-q,--quiet,--silent：不显示任何东西。

-a,--text：匹配二进制的东西。

-I：不匹配二进制的东西。

-d,--directories=ACTION：目录操作，读取，递归，跳过。

-D,--devices=ACTION：设置对设备、FIFO、管道的操作，读取，跳过。

-R,-r,--recursive：递归调用。

-L,--files-without-match：匹配多个文件时，显示不匹配的文件名。

-l,--files-with-matches：匹配多个文件时，显示匹配的文件名。

-c,--count：显示匹配了多少次。

4）文件控制

-B,--before-context=NUM：打印匹配本身及前面的几行由 NUM 控制。

-A,--after-context=NUM：打印匹配本身及随后的几行由 NUM 控制。

-C,--context=NUM：打印匹配本身及随后，前面的几行由 NUM 控制。

-NUM：根-C 的用法一样。

3.4 强大的 awk 文本分析工具

awk 被称为文本处理三剑客之一，其名称得自于它的创始人 Alfred Aho、Peter Weinberger 和 Brian Kernighan 姓氏的首个字母。实际上 awk 的确拥有自己的语言，即 awk 程序设计语言，三位创建人已将它正式定义为“样式扫描和处理语言”。它允许创建简短的程序，这些程序用于读取输入文件、为数据排序、处理数据、对输入执行计算及生成报表，还有很多其他的功能，所以说 awk 是一个强大的文本分析工具，相对于 grep 的查找，sed 的编辑，awk 在对数据分析并生成报告时，显得尤为强大。简单来讲 awk 就是把文件逐行地读取，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理。

1. awk 原理

awk 的用法如下：

```
awk 'BEGIN{ commands } pattern{ commands } END{ commands }'
```

第 1 步：执行 BEGIN{commands}语句块中的语句。

第 2 步：从文件或标准输入(stdin)读取一行，然后执行 pattern{commands}语句块，它逐行扫描文件，从第 1 行到最后一行重复这个过程，直到文件全部被读取完毕。

第 3 步：当读至输入流末尾时，执行 END{commands}语句块。

BEGIN 语句块在 awk 开始从输入流中读取行之前被执行，这是一个可选的语句块，例如变量初始化、打印输出表格的表头等语句通常可以写在 BEGIN 语句块中。

END 语句块在 awk 从输入流中读取完所有的行之后即被执行，例如打印所有行的分析结果这类信息汇总都是在 END 语句块中完成的，它也是一个可选语句块。

pattern 语句块中的通用命令是最重要的部分，它也是可选的。如果没有提供 pattern 语句块，则默认执行{print}，即打印每个读取的行，awk 读取的每行都会执行该语句块。

2. awk 支持

awk 支持很多内置的变量和运算符，awk 支持的内置变量见表 3-1，awk 支持的运算符见表 3-2。

表 3-1 awk 内置变量

变　量	描　述
\$n	当前记录的第 *n* 个字段，字段间由 FS 分隔
\$0	完整的输入记录
ARGC	命令行参数的数目
ARGIND	命令行中当前文件的位置(从 0 开始计算)
ARGV	包含命令行参数的数组
CONVFMT	数字转换格式(默认值为%.6g)ENVIRON 环境变量关联数组
ERRNO	最后一个系统错误的描述
FIELDWIDTHS	字段宽度列表(用空格键分隔)
FILENAME	当前文件名
FNR	各文件分别计数的行号
FS	字段分隔符(默认为空格)
IGNORECASE	如果为真，则进行忽略大小写的匹配
NF	一条记录的字段的数目
NR	已经读取的记录数，也就是行号，从 1 开始
OFMT	数字的输出格式(默认值为%.6g)
OFS	输出记录分隔符(输出换行符)，输出时用指定的符号代替换行符
ORS	输出记录分隔符(默认值为一个换行符)
RLENGTH	由 match 函数所匹配的字符串的长度
RS	记录分隔符(默认为一个换行符)
RSTART	由 match 函数所匹配的字符串的第 1 个位置
SUBSEP	数组下标分隔符(默认值为/034)

表 3-2 awk 运算符

运　算　符	描　述
=、+=、-=、*=、/=、%=、^=、* 、*=	赋值
?、:	C 条件表达式
||	逻辑或
&&	逻辑与
~和!~	匹配正则表达式和不匹配正则表达式
<、<=、>、>=、!=、==	关系运算符
空格	连接
+、-	加和减
*、/、%	乘、除与求余
+、-、!	一元加、减和逻辑非
^ * 、**	求幂
++、--	增加或减少，作为前缀或后缀
$	字段引用
in	数组成员

3. awk 使用

awk 的调用有 3 种方式。

(1) 命令行方式,命令如下:

```
awk [ -F field-separator] 'commands' input-file(s)
```

其中,commands 是真正的 awk 命令,[-F 域分隔符]是可选的。input-file(s)是待处理的文件。

在 awk 中,文件的每行中,由域分隔符分开的每项称为一个域。通常,在不指明-F 域分隔符的情况下,默认的域分隔符是空格。

(2) Shell 脚本方式,命令如下:

```
awk 'BEGIN{ print "start" } pattern{ commands } END{ print "end" }' file
```

一个 awk 脚本通常由 BEGIN 语句块、能够使用模式匹配的通用语句块、END 语句块 3 部分组成,这 3 部分是可选的。任意一部分都可以不出现在脚本中,脚本通常在单引号或双引号中,命令如下:

```
awk 'BEGIN{ i=0 } { i++} END{ print i }' filename
awk "BEGIN{ i=0 } { i++} END{ print i }" filename
```

(3) 将所有的 awk 命令插入一个单独文件,然后调用,命令如下:

```
awk -f awk-script-file input-file(s)
```

其中,-f 选项表示加载 awk-script-file 中的 awk 脚本,input-file(s)与上面的命令行方式相同。

打印/etc/passwd 下所有的用户名,命令如下:

```
awk -F: '{print $1}' /etc/passwd
```

打印/etc/passwd 下所有的用户名及 UID,命令如下:

```
awk -F: '{print $1,$3}' /etc/passwd
```

以 username: XXX 和 uid: XXX 格式输出,命令如下:

```
awk -F: '{print "username: " $1 "\t\tuid: " $3}' /etc/passwd
```

awk 赋值运算,赋值语句运算符包括=、+=、-=、*=、/=、%=、^=、**、**=。

例如 a+=5 等价于 a=a+5,命令如下:

```
awk 'BEGIN{a = 5;a += 5;print a}'
```

awk 正则运算，输出包含 root 的行，并打印用户名、UID 及原行内容，命令如下：

```
awk -F: '/root/ {print $1, $3, $0}' /etc/passwd
```

awk 三目运算，三目运算其实是一个判断运算，如果为真，则输出“?”后的内容；如果为假，则输出“:”后的内容，命令如下：

```
awk 'BEGIN{a = "b";print a == "b"?"ok":"err"}'
awk 'BEGIN{a = "b";print a == "c"?"ok":"err"}'
```

if 语句运用，每条命令后用“;”结尾，命令如下：

```
awk 'BEGIN{ test = 100;if(test > 90){ print "vear good";} else{print "no pass";}}'
```

awk 的用法有很多种，在不同的场景中用不同的方式，主要的方式有以下几种。

1）awk 的循环运用

awk 支持的循环包括 while、for 和 do 循环，如果需要计算从 1 累加到 100 的值，则可以通过下面几种方式完成。

while 循环的运用，命令如下：

```
awk 'BEGIN{test = 100;num = 0;while(i <= test){num += i; i++;}print num;}'
```

for 循环的运用，命令如下：

```
awk 'BEGIN{test = 0;for(i = 0;i <= 100;i++){test += i;}print test;}'
```

do 循环的运用，命令如下：

```
awk 'BEGIN{test = 0;i = 0;do{test += i;i++}while(i <= 100)print test;}'
```

2）awk 的数组运用

数组是 awk 的灵魂，处理文本中最不能少的就是数组处理。因为数组索引（下标）可以是数字和字符串，在 awk 中数组叫作关联数组（Associative Arrays）。awk 中的数组不必提前声明，也不必声明大小。数组元素用 0 或空字符串来初始化，根据上下文而定。一般而言，awk 中的数组用来从记录中收集信息，可以用于计算总和、统计单词及跟踪模板被匹配的次数等。

显示/etc/passwd 的账户，命令如下：

```
awk -F: 'BEGIN{count=0;} {name[count] = $1;count++;}; END{for(i = 0; i < NR; i++) print i,
name[i]}' /etc/passwd
```

3）awk 字符串函数的运用

sub 是用于匹配记录中最大、最靠左边的子字符串的正则表达式，并用替换字符串替换这些字符串。如果没有指定目标字符串就默认使用整个记录，并且替换只发生在第 1 次匹配时，命令如下：

```
sub(regular expression, substitution string):
sub(regular expression, substitution string, target string)
```

实际举例，命令如下：

```
awk '{ sub(/test/, "mytest"); print }' testfile
awk '{ sub(/test/, "mytest"); $1}; print }' testfile
```

第 1 个例子在整个记录中匹配，替换只发生在第 1 次匹配发生时。如果要在整个文件中进行匹配，则需要用到 gsub。

第 2 个例子在整个记录的第 1 个域中进行匹配，替换只发生在第 1 次匹配发生时。

gsub 在整个文档中进行匹配，命令如下：

```
gsub(regular expression, substitution string)
gsub(regular expression, substitution string, target string)
```

例如，在整个文档中匹配 test，匹配的字符串都被替换成 mytest，命令如下：

```
awk '{ gsub(/test/, "mytest"); print }' testfile
```

在整个文档的第 1 个域中匹配，所有匹配的字符串都被替换成 mytest，命令如下：

```
awk '{ gsub(/test/, "mytest" , $1) }; print }' testfile
```

awk 结合 match 使用，提取匹配字符串，假如 filename 文件的内容如下：

```
<key>hostname</key>
<value>0.0.0.0</key>
```

要求提取 0.0.0.0，命令如下：

```
cat filename | grep -A 1 "hostname" |awk 'match($0, "<value>(.*)</value>", a) {print a[1]}'
```

其中，grep-A1 表示多往下输出 1 行。

3.5 强大的 sed 文本处理工具

Linux 命令 sed 是 Stream Editor 的缩写，也就是流编辑器，它一次处理一行内容，处理时，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，把缓冲区的内容输出到屏幕。接着处理下一行，这样不断地重复，直到文件末尾。sed 是 Linux 下一款功能强大的文本处理工具，可以替换、删除、追加文件内容，支持正则表达式使用。

1. 基本用法

1）命令格式

sed 的命令格式：sed [options] 'command' file(s)；

sed 的脚本格式：sed [options]-f scriptfile file(s)；

2）命令选项

sed 详细的命令选项参数见表 3-3。

表 3-3 sed 选项参数

选　项	含　义
--version	显示 sed 版本
--help	显示帮助文档
-n,--quit,--silent	静默输出，默认情况下，sed 程序在所有的脚本指令执行完毕后，将自动打印模式空间中的内容，该选项可以屏蔽自动打印
-e script	允许多个脚本指令被执行
-f script-file	从文件中读取脚本指令，对编写自动脚本程序很实用
-i,--in-place	慎用，该选项将直接修改源文件
-l,N	该选项指令 l 可以输出的行长度，l 指令为输出而非打印字符
--posix	禁用 GNU sed 扩展功能
-r	在脚本指令中使用扩展正则表达式
-s,--separate	默认情况下，sed 将输入的多个文件名作为一个长的连续的输入流，而 GNU sed 则允许把它们当作单独的文件
-u,--unbuffered	最低限度地缓存输入和输出

2. 主要参数

sed 常用参数见表 3-4。

表 3-4 sed 常用参数

参　数	说　明
a\	在当前行下面插入文本
i\	在当前行上面插入文本
c\	把选定的行改为新的文本

续表

参　数	说　明
d	删除，删除选择的行
D	删除模板块的第 1 行
s	替换指定字符
h	将模板块的内容复制到内存中的缓冲区
H	将模板块的内容追加到内存中的缓冲区
g	获得内存缓冲区的内容，并替代当前模板块中的文本
G	获得内存缓冲区的内容，并追加到当前模板块文本的后面
l	列表不能打印字符的清单
n	读取下一个输入行，用下一个命令处理新的行而不是用第 1 个命令
N	将下一个输入行追加到模板块后面并在二者间嵌入一个新行，改变当前行号码
p	打印模板块的行。P(大写)表示打印模板块的第 1 行
q	退出 sed
b	lable 分支到脚本中带有标记的地方，如果分支不存在，则分支到脚本的末尾
r	file，从 file 中读行
t	label if 分支，从最后一行开始，条件一旦满足，将导致分支到带有标号的命令处，或者到脚本的末尾
T	label 错误分支，从最后一行开始，一旦发生错误，将导致分支到带有标号的命令处，或者到脚本的末尾
w	file，写并将模板块追加到 file 末尾
W	file，写并将模板块的第 1 行追加到 file 末尾
!	表示后面的命令对所有没有被选定的行发生作用
=	打印当前行号
^ #	把注释扩展到下一个换行符以前

sed 替换标记见表 3-5。

表 3-5　sed 替换标记

标　记	说　明
g	表示行内全面替换
p	表示打印行
w	表示把行写入一个文件
x	表示互换模板块中的文本和缓冲区中的文本
y	表示把一个字符翻译为另外的字符(但是不用于正则表达式)
\1	子串匹配标记
&	已匹配字符串标记

sed 元字符集见表 3-6。

表 3-6 sed 元字符集

元字符集	说明
^	匹配行开始，如/^sed/匹配所有以 sed 开头的行
$	匹配行结束，如/sed$/匹配所有以 sed 结尾的行
.	匹配一个非换行符的任意字符，如/s.d/匹配 s 后接一个任意字符，最后是 d
*	匹配 0 个或多个字符，如/*sed/匹配所有模板是一个或多个空格后紧跟 sed 的行
[]	匹配一个指定范围内的字符，如/[ss]ed/匹配 sed 和 Sed
[^]	匹配一个不在指定范围内的字符，如/[^A-RT-Z]ed/匹配不包含 A～R 和 T～Z 的一个字母开头，紧跟 ed 的行
\(..\)	匹配子串，保存匹配的字符，如 s/\(love\)able/\1rs，loveable 被替换成 lovers
&	保存搜索字符用来替换其他字符，如 s/love/**&**/，love 被替换成 **love**
\<	匹配单词的开始，如/\
\>	匹配单词的结束，如/love\>/匹配包含以 love 结尾的单词的行
x\{m\}	重复字符 x 共 m 次，如/0\{5\}/匹配包含 5 个 0 的行
x\{m,\}	重复字符 x 至少 m 次，如/0\{5,\}/匹配至少有 5 个 0 的行
x\{m,n\}	重复字符 x 至少 m 次，不多于 n 次，如/0\{5,10\}/匹配 5～10 个 0 的行

3. 用法举例

替换文本中的字符串，命令如下：

```
sed 's/book/books/' file
```

-n 选项和 p 命令一起使用表示只打印那些发生替换的行，命令如下：

```
sed -n 's/test/TEST/p' file
```

直接编辑文件选项-i，会匹配 file 文件中每行的第 1 个 book 并替换为 books，命令如下：

```
sed -i 's/book/books/g' file
```

将 2～3 行中的 hello 替换成 hey，命令如下：

```
sed '2,3s/hello/hey/g' fin.txt
```

找出包含字符 Pony 的那些行，将这些行中的 hello 替换成 hey，命令如下：

```
sed '/Pony/s/hello/hey/g' fin.txt
```

使用后缀/g 标记会替换每行中所有匹配的字符串，命令如下：

```
sed 's/book/books/g' file
```

当需要从第 N 处匹配开始替换时，可以使用/Ng，命令如下：

```
sed 's/sk/SK/3g' file
```

在以上命令中字符 / 在 sed 中作为定界符使用，也可以使用任意的定界符，命令如下：

```
sed 's:test:TEXT:g'
sed 's|test|TEXT|g'
```

当定界符出现在样式内部时，需要进行转义，命令如下：

```
sed 's/\/bin/\/usr\/local\/bin/g'
```

删除空白行，命令如下：

```
sed '/^$/d' file
```

注意，有时候有些空白行包含空格，这种情况下写成的命令如下：

```
sed '/^\s*$/d' fin.txt
```

其中，\s 表示空格，星号 * 表示前面的字符重复 0 次或多次，所以这种写法可以匹配那些包含任意个空格的空白行。

删除文件的第 2 行，命令如下：

```
sed '2d' file
```

删除文件中从第 2 行到末尾的所有行，命令如下：

```
sed '2,$d' file
```

删除文件的最后一行，命令如下：

```
sed '$d' file
```

删除文件中所有开头是 test 的行，命令如下：

```
sed '/^test/'d file
```

正则表达式\w\+用于匹配每个单词，使用[&]替换它，& 对应于之前所匹配到的单词，命令如下：

```
echo this is a test line | sed 's/\w\+/[&]/g'
```

所有以 192.168.0.1 开头的行都会被替换成它自己加 localhost，命令如下：

```
sed 's/^192.168.0.1/&localhost/' file 192.168.0.1localhost
```

匹配给定样式的其中一部分，命令如下：

```
echo this is digit 7 in a number | sed 's/digit \([0-9]\)/\1/'
```

命令中 digit 7，被替换成了 7。样式匹配到的子串是 7，\(..\)用于匹配子串，对于匹配到的第 1 个子串标记为\1，以此类推，匹配到的第 2 个子串标记为\2，示例命令如下：

```
echo aaa BBB | sed 's/\([a-z]\+\) \([A-Z]\+\)/\2 \1/'
```

love 被标记为 1，所有 loveable 会被替换成 lovers，并打印出来，命令如下：

```
sed -n 's/\(love\)able/\1rs/p' file
```

sed 表达式可以使用单引号来引用，但是如果表达式内部包含变量字符串，则需要使用双引号，命令如下：

```
sed "s/$test/HELLO" file
```

所有在模板 test 和 check 所确定的范围内的行都被打印，命令如下：

```
sed -n '/test/,/check/p' file
```

打印从第 5 行开始到第 1 个包含以 test 开始的行之间的所有行，命令如下：

```
sed -n '5,/^test/p' file
```

对于模板 test 和 west 之间的行，每行的末尾用字符串 aaa bbb 替换，命令如下：

```
sed '/test/,/west/s/$/aaa bbb/' file
```

-e 选项允许在同一行里执行多条命令，命令如下：

```
sed -e '1,5d' -e 's/test/check/' file
```

上面 sed 表达式的第 1 条命令用于删除 1～5 行，第 2 条命令用 check 替换 test。命令的执行顺序对结果有影响。如果两个命令都是替换命令，则第 1 个替换命令将影响第 2 个替换命令的结果。

和-e 等价的命令是--expression，命令如下：

```
sed --expression = 's/test/check/' --expression = '/love/d' file
```

file 里的内容被读取，显示在与 test 匹配的行后面，如果匹配多行，则 file 的内容将显示在所有匹配行的下面，命令如下：

```
sed '/test/r file' filename
```

在 example 中所有包含 test 的行都被写入 file 里，命令如下：

```
sed -n '/test/w file' example
```

将 this is a test line 追加到以 test 开头的行后面，命令如下：

```
sed '/^test/a\this is a test line' file
```

在指定某一行的前面或者后面添加一行，命令如下：

```
sed -i '1i\welcome' fin.txt
```

这里的 1 表示第 1 行，i 表示在这一行前面添加一行，如果要在第 1 行后面添加一行，则用字母 a，命令如下：

```
sed -i '1a\welcome' fin.txt
```

字母 a 是 append 的首字母，表示在后面添加一行，字母 i 是 insert 的首字母，表示在前面添加一行，在 test.conf 文件第 2 行后插入 this is a test line，命令如下：

```
sed -i '2a\this is a test line' test.conf
```

i\命令将 this is a test line 追加到以 test 开头的行前面，命令如下：

```
sed '/^test/i\this is a test line' file
```

在 test.conf 文件的第 5 行前插入 this is a test line，命令如下：

```
sed -i '5i\this is a test line' test.conf
```

如果 test 被匹配，则移动到匹配行的下一行，替换这一行的 aa 为 bb，并打印该行，然后继续，命令如下：

```
sed '/test/{ n; s/aa/bb/; }' file
```

删除不包含字符 Pony 的行，命令如下：

```
sed '/Pony/\!d' fin.txt
```

这里的感叹号“!”表示反选，也就是选择那些不符合正则表达式 /Pony/ 的行，右斜杠表示转义，因为在有些系统下“!”会被识别成其他的意思。

把 1～10 行内所有的 abcde 转换为大写，命令如下：

```
sed '1,10y/abcde/ABCDE/' file
```

h 命令和 G 命令在 sed 处理文件时，每行都被保存在一个叫作模式空间的临时缓冲区中，除非行被删除或者输出被取消，否则所有被处理的行都将打印在屏幕上。接着模式空间被清空，并存入新的一行等待处理，命令如下：

```
sed -e '/test/h' -e '$G' file
```

在这个例子里，匹配 test 的行被找到后，将存入模式空间，h 命令将其复制并存入一个被称为保持缓存区的特殊缓冲区内。第 2 条语句的意思是，当到达最后一行后，G 命令取出保持缓冲区的行，然后把它放回模式空间中，并且追加到现在已经存在于模式空间中的行的末尾。在这个例子中就是追加到最后一行。简单来讲，任何包含 test 的行都被复制并追加到该文件的末尾。

互换模式空间和保持缓冲区的内容，也就是把包含 test 与 check 的行互换，命令如下：

```
sed -e '/test/h' -e '/check/x' file
```

sed 脚本是一个 sed 命令的清单，启动 sed 时以-f 选项引导脚本文件名。sed 对于脚本中输入的命令非常挑剔，在命令的末尾不能有任何空白或文本，如果在一行中有多个命令，则要用分号分隔。以 # 开头的行为注释行，并且不能跨行，命令如下：

```
sed [options] -f scriptfile file(s)
```

例如打印奇数行或偶数行，可以用下面的方法实现。

方法 1，命令如下：

```
sed -n 'p;n' test.txt          #奇数行
sed -n 'n;p' test.txt          #偶数行
```

方法 2,命令如下:

```
sed -n '1～2p' test.txt              #奇数行
sed -n '2～2p' test.txt              #偶数行
```

打印匹配字符串的下一行,命令如下:

```
grep -A 1 SCC URFILE
sed -n '/SCC/{n;p}' URFILE
awk '/SCC/{getline; print}' URFILE
```

3.6 多功能的 curl 网络传输工具

curl(Command Line Uniform Resource Locator),即在命令行中利用 URL 进行数据或者文件传输,它是 Linux 下的 HTTP 命令行工具,其功能十分强大,参数非常丰富。与其类似的进行网络文件下载的命令还有 wget 等。

curl 是基于 URL 语法在命令行方式下工作的文件传输工具,它支持 FTP、FTPS、HTTP、HTTPS、GOPHER、TELNET、DICT、FILE 及 LDAP 等协议。curl 支持 HTTPS 认证、HTTP 的 POST、PUT 等方法、FTP 上传、kerberos 认证、HTTP 上传、代理服务器、Cookies、用户名/密码认证、下载文件断点续传、上传文件断点续传、HTTP 代理服务器管道(Proxy Tunneling)等。此外,curl 还支持 IPv6、SOCKS5 代理服务器,通过 HTTP 代理服务器上传文件到 FTP 服务器等,功能十分强大。

1. 常用参数

curl 后面可以跟很多种参数,具体见表 3-7。

表 3-7 curl 常用参数

参 数	说 明
-A/--user-agent < string >	设置用户代理,发送给服务器
-e/--referer < URL >	来源网址
--cacert < file >	CA 证书 (SSL)
-k/--insecure	允许忽略证书进行 SSL 连接
--compressed	要求返回压缩的格式
-H/--header < line >	自定义首部信息,传递给服务器
-i	显示页面内容,包括报文首部信息
-I/--head	只显示响应报文首部信息
-D/--dump-header < file >	将 URL 的 header 信息存放在指定文件中
--basic	使用 HTTP 基本认证
-u/--user < user[:password]>	设置服务器的用户名和密码
-L	如果有 3xx 响应码,则重新将请求发送到新位置

续表

参　数	说　明
-O	使用 URL 中默认的文件名，将文件保存到本地
-o < file >	将网络文件保存到指定的文件中
--limit-rate < rate >	设置传输速度
-0/--http1.0	数字 0，使用 HTTP 1.0
-v/--verbose	更详细
-C	选项可对文件使用断点续传功能
-c/--Cookie-jar < file name >	将 URL 中 Cookie 存放在指定文件中
-x/--proxy < proxyhost[:port]>	指定代理服务器地址
-X/--request < command >	向服务器发送指定请求方法
-U/--proxy-user < user:password >	代理服务器的用户名和密码
-T	选项可将指定的本地文件上传到 FTP 服务器上
--data/-d	指定使用 POST 方式传递数据
-b name=data	从服务器响应 set-Cookie 得到值，返回服务器

2. 基本用法

访问网页的命令如下：

```
curl http://www.linux.com
```

执行后，www.linux.com 的网页就会显示在屏幕上了。由于安装 Linux 时很多时候没有安装桌面，也就意味着没有浏览器，因此这种方法经常用于测试一台服务器是否可以访问一个网站。此时如果想要 curl 像浏览器一样跟随链接的跳转，获取最终的网页内容，则可以在命令中添加-L 选项来跟随链接重定向，命令如下：

```
curl -L http://codebelief.com
```

如果直接使用 curl 打开某些被重定向后的链接，在这种情况下则无法获取想要的网页内容。通过浏览器打开该链接时，会自动跳转到重定向之后的网址。

保存访问的网页，使用 Linux 的重定向功能保存，命令如下：

```
curl http://www.linux.com >> linux.html
```

也可以使用 curl 的内置 option:-o(小写)保存网页，命令如下：

```
curl -o linux.html http://www.linux.com
```

执行完成后会显示如下界面，如果显示 100%，则表示保存成功，命令如下：

```
% Total    % Received % Xferd Average Speed Time   Time   Time Current
```

```
                                 Dload Upload Total Spent    Left Speed
100 79684    0 79684    0    0  3437k    0 --:--:-- --:--:-- --:--:-- 7781k
```

可以使用 curl 的内置 option:-O(大写)保存网页中的文件,要注意这里后面的 URL 要具体到某个文件,否则无法下载保存,命令如下:

```
curl -O http://www.linux.com/hello.sh
```

测试网页的返回值是否在脚本中,这是很常见的测试网站是否正常的方法,命令如下:

```
curl -o /dev/null -s -w %{http_code} www.linux.com
```

使用-H 自定义 header,当需要传递特定的 header 时,命令如下:

```
curl -H "Referer: https://rumenz.com" -H "User-Agent: Custom-User-Agent" https://
json.im
```

指定 proxy 服务器及其端口,很多时候上网需要用到代理服务器(例如使用代理服务器上网或者因为使用 curl 访问别人网站而被别人屏蔽 IP 地址时),幸运的是,curl 通过内置 option:-x 来支持设置代理,命令如下:

```
curl -x 192.168.100.100:1080 http://www.linux.com
```

有些网站使用 Cookie 来记录 session 信息。对于 Chrome 浏览器,可以轻易处理 Cookie 信息,但在 curl 中只要增加相关参数也是可以很容易地处理 Cookie,保存 HTTP 的 response 里面的 Cookie 信息。内置 option:-c(小写),命令如下:

```
curl -c Cookiec.txt  http://www.linux.com
```

执行后 Cookie 信息就被存到了 Cookiec.txt 文件里了。

保存 HTTP 的 response 里面的 header 信息。内置 option:-D,命令如下:

```
curl -D Cookied.txt http://www.linux.com
```

执行后 Cookie 信息就被存到 Cookied.txt 文件里了。注意:-c(小写)产生的 Cookie 和-D 里面的 Cookie 是不一样的。

很多网站通过监视访问的 Cookie 信息来判断是否按正常授权访问网站,因此需要使用保存的 Cookie 信息。内置 option:-b,命令如下:

```
curl -b Cookiec.txt http://www.linux.com
```

在 header 中传递 Cookie，命令如下：

```
curl -H "Cookie: JSESSIONID=xxx" https://json.im
```

测试 GET 请求，命令如下：

```
curl http://www.linuxidc.com/login.cgi?user=test001&password=123456
```

测试 POST 请求，命令如下：

```
curl -d "user=nickwolfe&password=12345" http://www.linuxidc.com/login
```

测试 XML 格式的 POST 请求，可以通过下面的方式实现。

方式一：发送磁盘上的 XML 文件，其中 myxmlfile.txt 文件为磁盘上的 XML 文件，后面为请求路径，命令如下：

```
curl -X POST -H 'content-type: application/xml' -d @/apps/myxmlfile.txt http://172.19.
219.xx:8081/csp/faq/actDiaUserInfo.action
```

方式二：在命令行直接发送 XML 结构数据，其中<?xml version...>就是要 POST 的 XML 文件，后面是请求路径，Linux 上双引号或单引号之间嵌套需要使用反斜杠“\”进行转义，命令如下：

```
curl -H 'content-type: application/xml' -X POST -d '<?xml version="1.0" encoding="UTF-
8"?><userinfoReq><subsNumber>13814528620</subsNumber><type>3</type></userinfoReq>'
http://172.19.219.xx:8081/csp/faq/actDiaUserInfo.action
```

或者，命令如下：

```
echo '<?xml version="1.0" encoding="UTF-8"?><userinfoReq><subsNumber>13814528620
</subsNumber><type>3</type></userinfoReq>'|curl -X POST -H'Content-type:text/xm' -d
@- http://172.19.xx.xx:8081/csp/faq/actDiaUserInfo.action
```

测试 JSON 格式的 POST 请求，可以通过下面的方式实现。

方式一：发送磁盘上的 JSON 文件，其中 myjsonfile.txt 文件为磁盘上的 JSON 文件，后面为请求路径，命令如下：

```
curl -X POST -H 'content-type: application/json' -d @/apps/myjsonfile.txt http://192.
168.129.xx/AntiRushServer/api/ActivityAntiRush
```

方式二：在命令行直接发送 JSON 结构数据，命令如下：

```
curl -H 'content-type: application/json' -X POST -d '{"accountType":"4","channel":"1",
"channelId":"YW_MMY","uid":"13154897541","phoneNumber":"13154897541","loginSource":"3",
"loginType":"1","userIp":"192.168.2.3","postTime":"14633ffffffffff81286","userAgent":
"Windows NT","imei":"352600051025733","macAddress":"40:92:d4:cb:46:43","serialNumber":
"123"}' http://192.168.129.xx/AntiRushServer/api/ActivityAntiRush
```

测试 webservice 请求,可以通过下面的方式实现。

方式一:发送磁盘上的请求报文文件,其中 myjsonfile.txt 文件为磁盘上的请求报文文件,后面为请求路径,命令如下:

```
curl -H 'Content-Type: text/xml;charset=UTF-8;SOAPAction:""' -d @/apps/mysoapfile.xml
http://172.18.173.xx:8085/csp-magent-client/madapterservices/madapter/lmCountAccessor
```

方式二:在命令行直接发送 XML 结构数据,命令如下:

```
curl -H 'content-type: application/xml' -d '<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ser=
"http://service.accessor.madapter.csp.huawei.com"><soapenv:Header /><soapenv:Body><ser:
leaveMessageCount><ser:in0><![CDATA[20161011160516XdznbN]]></ser:in0><ser:in1>
<![CDATA[1600106496388382726]]></ser:in1><ser:in2><![CDATA[14]]></ser:in2><ser:in3>
<![CDATA[<extendParams><channelid>1600</channelid><servicetype></servicetype>
<appid></appid><usertype>10</usertype><userid>6496388382726</userid><msisdn>
13814528620</msisdn><email></email><account></account><nickname></nickname>
<questionType></questionType></extendParams>]]></ser:in3></ser:leaveMessageCount>
</soapenv:Body></soapenv:Envelope>' http://172.18.173.xx:8085/csp-magent-client/
madapterservices/madapter/lmCountAccessor
```

模仿浏览器访问,有些网站需要使用特定的浏览器去访问,有些还需要使用某些特定的版本。curl 内置 option:-A 可以指定浏览器去访问网站,命令如下:

```
curl -A "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.0)" \
http://www.linux.com
```

这样服务器端就会认为是使用 IE 8.0 去访问的。

伪造 referer(盗链):很多服务器会检查 HTTP 访问的 referer 从而来控制访问。例如先访问首页,然后访问首页中的邮箱页面,这里访问邮箱的 referer 地址就是访问首页成功后的页面地址,如果服务器发现对邮箱页面访问的 referer 地址不是首页的地址,则断定那是个盗链了。curl 中内置 option:-e 可以设定 referer,命令如下:

```
curl -e "www.linux.com" http://mail.linux.com
```

这样就会让服务器以为是从 www.linux.com 单击某个链接访问的。

下载文件：利用 curl 下载文件。使用内置 option:-o(小写),命令如下：

```
curl -o dodo1.jpg http:www.linux.com/dodo1.JPG
```

使用内置 option:-O(大写),命令如下：

```
curl -O http://www.linux.com/dodo1.JPG
```

这样就会以服务器上的名称将文件保存到本地。

循环下载：有时候需下载的图片可能前面的部分名称是一样的,只有最后的后缀名不一样,命令如下：

```
curl -O http://www.linux.com/dodo[1-5].JPG
```

这样就会把 dodo1、dodo2、dodo3、dodo4 和 dodo5 全部保存下来。

同时下载多个文件,可以使用-o 或-O 选项来同时指定多个链接,命令如下：

```
curl -O http://www.codebelief.com/page/2/ -O\
http://www.codebelief.com/page/3/
```

或者,命令如下：

```
curl -o page1.file http://www.codebelief.com/page/1/ -o page2.file\
http://www.codebelief.com/page/2/
```

下载重命名,命令如下：

```
curl -O http://www.linux.com/{hello,bb}/dodo[1-5].JPG
```

由于下载的 hello 与 bb 中的文件名都是 dodo1、dodo2、dodo3、dodo4 和 dodo5,因此第 2 次下载的文件会把第 1 次下载的文件覆盖,这样就需要对文件进行重命名,命令如下：

```
curl -o #1_#2.JPG http://www.linux.com/{hello,bb}/dodo[1-5].JPG
```

这样 hello/dodo1.JPG 文件下载下来后就会变成 hello_dodo1.JPG,其他文件以此类推,从而有效地避免了文件被覆盖。

分块下载：有时候下载的文件会比较大,这时可以分段下载。使用内置 option:-r,具体的命令如下：

```
curl -r 0-100 -o dodo1_part1.JPG http://www.linux.com/dodo1.JPG
curl -r 100-200 -o dodo1_part2.JPG http://www.linux.com/dodo1.JPG
curl -r 200- -o dodo1_part3.JPG http://www.linux.com/dodo1.JPG
cat dodo1_part* > dodo1.JPG
```

这样就可以查看 dodo1.JPG 文件的内容了。

通过 FTP 下载文件：curl 可以通过 FTP 下载文件，curl 提供了两种通过 FTP 下载文件的语法，具体的命令如下：

```
curl -O -u 用户名:密码 ftp://www.linux.com/dodo1.JPG
curl -O ftp://用户名:密码@www.linux.com/dodo1.JPG
```

显示下载进度条，命令如下：

```
curl - # -O http://www.linux.com/dodo1.JPG
```

不显示下载进度信息，命令如下：

```
curl -s -O http://www.linux.com/dodo1.JPG
```

断点续传：在 Windows 系统中，可以使用迅雷软件进行断点续传。curl 通过内置 option:-C 同样可以达到相同的效果。

如果在下载 dodo1.JPG 的过程中突然掉线了，则可以使用以下的方式续传，命令如下：

```
curl -C -O http://www.linux.com/dodo1.JPG
```

上传文件：curl 不仅可以下载文件，还可以上传文件。通过内置 option:-T 实现，命令如下：

```
curl -T dodo1.JPG -u 用户名:密码 ftp://www.linux.com/img/
```

这样就向 FTP 服务器上传了文件 dodo1.JPG。

显示抓取错误，命令如下：

```
curl -f http://www.linux.com/error
```

只显示响应报文首部信息，通过指定-I/--head 实现，命令如下：

```
curl -I https://blog.csdn.net/u014374009
curl --head https://blog.csdn.net/u014374009
```

显示页面内容，包括报文首部信息，通过指定-i 实现，命令如下：

```
curl -i https://blog.csdn.net/u014374009
```

使用证书检查访问网页，通过指定--cacert < file >实现，命令如下：

```
curl https://blog.csdn.net/u014374009 --cacert\
/etc/httpd/conf.d/ssl/cacert.pem
```

忽略证书检查访问网页，通过指定-k/--insecure 实现，命令如下：

```
curl -k https://blog.csdn.net/u014374009 --cacert\
/etc/httpd/conf.d/ssl/cacert.pem
```

修改名称解析，命令如下：

```
curl --resolve json.im:443:127.0.0.1 https://json.im:443/
```

对 https://json.im 的查询将告诉 curl 从 127.0.0.1 请求该站点，而不是使用 DNS 或/etc/hosts 文件。

限制下载率，命令如下：

```
curl --limit-rate 100K https://json.im/jdk.tar.gz -O
```

HTTP 认证：有些网域需要 HTTP 认证，这时 curl 需要用到--user 参数，命令如下：

```
curl --user name :passwd https://json.im
```

3. 其他选项

-a/--append：上传文件时，附加到目标文件。

--anyauth：可以使用“任何”身份验证方法。

--basic：使用 HTTP 基本验证。

-B/--use-ascii：使用 ASCII 文本传输。

-d/--data <data>：使用 HTTP POST 方式传送数据。

--data-ascii <data>：以 ASCII 的方式 POST 数据。

--data-binary <data>：以二进制的方式 POST 数据。

--negotiate：使用 HTTP 身份验证。

--digest：使用数字身份验证。

--disable-eprt：禁止使用 EPRT 或 LPRT。

--disable-epsv：禁止使用 EPSV。

--egd-file <file>：为随机数据(SSL)设置 EGD Socket 路径。

--tcp-nodelay：使用 TCP_NODELAY 选项。

-E/--cert <cert[:passwd]>：客户端证书文件和密码 (SSL)。

--cert-type <type>：证书文件类型 (DER/PEM/ENG) (SSL)。

--key <key>：私钥文件名 (SSL)。

--key-type <type>：私钥文件类型（DER/PEM/ENG）（SSL）。
--pass <pass>：私钥密码（SSL）。
--engine <eng>：加密引擎使用（SSL），使用--engine list 指定列表。
--cacert <file>：CA 证书（SSL）。
--capath <directory>：CA 证书目录(SSL)。
--ciphers <list>：SSL 密码。
--compressed：要求返回压缩的形式。
--connect-timeout <seconds>：设置最大请求时间。
--create-dirs：建立本地目录的目录层次结构。
--crlf：上传时把 LF 转变成 CRLF。
--ftp-create-dirs：如果远程目录不存在，则创建远程目录。
--ftp-method [multicwd/nocwd/singlecwd]：控制 CWD 的使用。
--ftp-pasv：使用 PASV/EPSV 代替端口。
--ftp-skip-pasv-ip：使用 PASV 时忽略该 IP 地址。
--ftp-ssl：尝试用 SSL/TLS 进行 FTP 数据传输。
--ftp-ssl-reqd：要求用 SSL/TLS 进行 FTP 数据传输。
-F/--form <name=content>：模拟 HTTP 表单提交数据。
-form-string <name=string>：模拟 HTTP 表单提交数据。
-g/--globoff：禁用网址序列和范围使用{}和[]。
-G/--get：以 GET 的方式发送数据。
-h/--help：帮助。
-H/--header <line>：将自定义头信息传递给服务器。
--ignore-content-length：忽略的 HTTP 头信息的长度。
-i/--include：输出时包括 protocol 头信息。
-I/--head：只显示文档信息。
-j/--junk-session-Cookies：读取文件时忽略 session Cookies。
--interface <interface>：使用指定网络接口/地址。
--krb4 <level>：使用指定安全级别的 krb4。
-k/--insecure：允许不使用证书到 SSL 站点。
-K/--config：读取指定的配置文件。
-l/--list-only：列出 ftp 目录下的文件名称。
--limit-rate <rate>：设置传输速度。
--local-port <NUM>：强制使用本地端口号。
-m/--max-time <seconds>：设置最大传输时间。
--max-redirs <num>：设置最大读取的目录数。
--max-filesize <Bytes>：设置最大下载的文件总量。

-M/--manual：显示全手动。
-n/--netrc：从 netrc 文件中读取用户名和密码。
--netrc-optional：使用.netrc 或者 URL 来覆盖-n。
--ntlm：使用 HTTP NTLM 身份验证。
-N/--no-buffer：禁用缓冲输出。
-p/--proxytunnel：使用 HTTP 代理。
--proxy-anyauth：选择任一代理身份验证方法。
--proxy-basic：在代理上使用基本身份验证。
--proxy-digest：在代理上使用数字身份验证。
--proxy-ntlm：在代理上使用 NTLM 身份验证。
-P/--ftp-port <address>：使用端口地址，而不是使用 PASV。
-Q/--quote <cmd>：文件传输前将命令发送到服务器。
--range-file：读取(SSL)的随机文件。
-R/--remote-time：在本地生成文件时，保留远程文件的时间。
--retry <num>：传输出现问题时，重试的次数。
--retry-delay <seconds>：传输出现问题时，设置重试间隔时间。
--retry-max-time <seconds>：传输出现问题时，设置最大重试时间。
-S/--show-error：显示错误。
--socks4 <host[:port]>：用 SOCKS4 代理给定主机和端口。
--socks5 <host[:port]>：用 SOCKS5 代理给定主机和端口。
-t/--telnet-option <OPT=val>：Telnet 选项设置。
--trace <file>：对指定文件进行 Debug。
--trace-ascii <file>：跟踪但没有 hex 输出。
--trace-time：跟踪/详细输出时，添加时间戳。
--url <URL>：指定所使用的 URL。
-U/--proxy-user <user[:password]>：设置代理用户名和密码。
-V/--version：显示版本信息。
-X/--request <command>：指定什么命令。
-y/--speed-time：放弃限速所要的时间，默认为 30s。
-Y/--speed-limit：停止传输速度的限制。
-z/--time-cond：传送时间设置。
-0/--http1.0：使用 HTTP 1.0。
-1/--tlsv1：使用 TLS v1(SSL)。
-2/--sslv2：使用 SSL v2(SSL)。
-3/--sslv3：使用的 SSL v3(SSL)。
--3p-quote：类似于 -Q 一样发送命令到服务器，进行第三方传输。

--3p-url：使用 URL 进行第三方传送。

--3p-user：使用用户名和密码进行第三方传送。

-4/--ipv4：使用 IPv4。

-6/--ipv6：使用 IPv6。

3.7 提高工作效率的一些实用命令

本节主要基于项目开发和部署对经常使用的一些命令进行整理和说明。先介绍几个查询帮助信息的命令。

一般情况下，type 命令用于判断另外一个命令是否是内置命令。判断一个名字当前是否是 alias、keyword、function、builtin、file 或者什么都不是，命令如下：

```
type if
type -t ls
```

学习一个新的命令之前，不知道其用法，可以查看帮助信息了解，主要有以下两种方式进行调用，命令如下：

```
find - help
help find
```

另一个查询帮助信息的命令是 man，这个命令主要为所有用户提供在线帮助，命令如下：

```
man find
```

whereis 命令只能用于程序名的搜索，而且只能搜索二进制文件（参数-b）、man 说明文件（参数-m）和源代码文件（参数-s）。如果省略参数，则返回所有信息。与 find 相比，whereis 查找的速度更快，这是因为 Linux 系统会将系统内的所有文件都记录在一个数据库文件中，当使用 whereis 和 locate 时，会从数据库中查找数据，而不是像 find 命令那样，通过遍历硬盘来查找，效率自然会很高。但是该数据库文件并不是实时更新的，默认情况下一星期更新一次，因此，在用 whereis 和 locate 查找文件时，有时会找到已经被删除的数据，或者对于刚刚建立的文件却无法查找到，原因是数据库文件没有被更新，示例命令如下：

```
whereis kill
```

locate 命令可以在搜寻数据库时快速找到文件，数据库由 updatedb 程序来更新，updatedb 是由 cron daemon 周期性建立的，locate 命令在搜寻数据库时比通过整个硬盘资料搜寻快，但对于最近才建立或刚更名的文件可能会找不到，在设定值中，updatedb 每天会

更新一次，可以修改 crontab 更新设定值(etc/crontab)。命令如下：

```
locate crontab
```

which 会在 PATH 变量指定的路径中搜索某个系统命令的位置，并且返回第 1 个搜索结果，示例命令如下：

```
which grep
```

find 会根据下列规则判断 path 和 expression，在命令列上第 1 个－、(、)、,、!、之前的部分为 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。

将目前目录及其子目录下所有扩展名是 c 的文件列出来，示例命令如下：

```
find . -name "*.c"
```

将目前目录及其子目录中的所有一般文件列出，示例命令如下：

```
find . -type f
```

将目前目录及其子目录下所有的最近 20 天内更新过的文件列出，示例命令如下：

```
find . -ctime -20
```

查找/var/log 目录中更改时间在 7 日以前的普通文件，并在删除之前询问它们，示例命令如下：

```
find /var/log -type f -mtime +7 -ok rm {} \;
```

查找当前目录中具有读、写权限的文件所有者，以及文件所属组的用户和其他用户具有读权限的文件，示例命令如下：

```
find . -type f -perm 644 -exec ls -l {} \;
```

查找系统中所有文件长度为 0 的普通文件，并列出它们的完整路径，示例命令如下：

```
find / -type f -size 0 -exec ls -l {} \;
```

ln 命令为某个文件在另外一个位置建立一个同步的链接。在 Linux 文件系统中有所谓的链接(Link)，可以将其视为文件的别名，而链接又可分为两种：硬链接(Hard Link)与软链接(Symbolic Link)。硬链接的意思是一个文件可以有多个名称，软链接的方式则是生

成一个特殊的文件，该文件的内容指向另一个文件的位置。硬链接存在于同一个文件系统中，而软链接却可以跨越不同的文件系统。

给文件创建软链接，并显示操作信息，示例命令如下：

```
ln -sv source.log link.log
```

给文件创建硬链接，并显示操作信息，示例命令如下：

```
ln -v source.log link1.log
```

给目录创建软链接，示例命令如下：

```
ln -sv /opt/soft/test/test3 /opt/soft/test/test5
```

关于文件目录的一些使用，示例命令如下：

```
cd test                 #切换到 test 目录下
cd ..                   #切换到上一级目录
cd /                    #切换到系统根目录下
cd ~                    #切换到当前用户的根目录下
cd -                    #切换到上一级所在的目录

mkdir test              #在当前目录下创建一个 test 目录
mkdir -p test/a/b       #在 test 目录下的 a 目录下创建一个 b 目录，如果上一级目录不存在，则连
#它的父目录一起创建(注意：若不加 -p，并且原本 test 或者 a 目录不存在，则会产生错误)
rmdir test              #删除当前目录下的 test 目录(注意：该命令只能删除空目录)

touch test.txt          #在当前目录下创建一个 test.txt 文件
rm test.txt             #删除 test.txt 文件(带询问的删除，需输入 y 才能删除)
rm -f test.txt          #直接删除 text.txt 文件
rm -r test              #递归删除，即删除 test 目录及其目录下的子目录(带询问的删除，需输入
#y 才能删除)
rm -rf test             #直接删除 test 目录及其目录下的子目录
```

在 Linux 系统中修改文件/目录权限的命令是 chmod、chown 等相关命令。

将/test 下的 aaa.txt 文件的权限修改为所有者拥有全部权限，所有者所在的组有读、写权限，其他用户只有读的权限，示例命令如下：

```
chmod u=rwx,g=rw,o=r aaa.txt
```

还可以使用数字表示：

```
chmod 764 aaa.txt
```

chown 可将指定文件的所有者改为指定的用户或组，用户可以是用户名或者用户 ID；组可以是组名或者组 ID；如果文件是以空格分开的，则要改变权限的文件列表，支持通配符。

改变所有者和群组并显示改变信息，示例命令如下：

```
chown -c mail:mail log2012.log
```

将文件夹及子文件目录所有者，以及其属组更改为 mail，示例命令如下：

```
chown -cR mail: test/
```

关于文件的权限和属性的一些使用（使用“+”设置，使用“-”取消），示例命令如下：

```
ls -lh                          #显示权限
ls /tmp | pr -T5 -W $COLUMNS    #将终端划分成 5 栏显示
chmod ugo+rwx directory1        #将目录的所有者(u)、群组(g)及其他人(o)设置为拥有读(r)、
                                #写(w)和执行(x)的权限
chmod go-rwx directory1         #删除群组(g)与其他人(o)对目录的读、写执行权限
chown user1 file1               #改变一个文件的所有者属性
chown -R user1 directory1       #改变一个目录的所有者属性并同时改变该目录下所有文件的属性
chgrp group1 file1              #改变文件的群组
chown user1:group1 file1        #改变一个文件的所有者和群组属性
find / -perm -u+s               #列出一个系统中所有使用了 SUID 控制的文件
chmod u+s /bin/file1            #设置一个二进制文件的 SUID 位，运行该文件的用户也被赋予
                                #和所有者同样的权限
chmod u-s /bin/file1            #禁用一个二进制文件的 SUID 位
chmod g+s /home/public          #设置一个目录的 SGID 位，类似 SUID，不过这是针对目录的
chmod g-s /home/public          #禁用一个目录的 SGID 位
chmod o+t /home/public          #设置一个文件的 STIKY 位，只允许合法所有人删除文件
chmod o-t /home/public          #禁用一个目录的 STIKY 位

chattr +a file1                 #只允许以追加方式读、写文件
chattr +c file1                 #允许这个文件被内核自动压缩/解压
chattr +d file1                 #在进行文件系统备份时，dump 程序将忽略这个文件
chattr +i file1                 #设置成不可变的文件，不能被删除、修改、重命名或者创建链接
chattr +s file1                 #允许一个文件被安全地删除
chattr +S file1                 #一旦应用程序对这个文件执行了写操作，系统就立刻把修改的
                                #结果写到磁盘
chattr +u file1                 #若文件被删除，系统会允许以后恢复这个被删除的文件
lsattr                          #显示特殊的属性
```

在 Linux 系统中有很多种查看文件内容的方式，下面对常用几种命令及用法做一个简单的介绍，在实际搭建项目环境时会经常用来查看配置文件和日志信息等。

cat 从第 1 行开始显示文件内容，示例命令如下：

```
cat /etc/issue
```

tac 从最后一行开始显示文件内容，可以看出 tac 是 cat 的倒写，示例命令如下：

```
tac /etc/issue
```

nl 在显示文件内容的同时输出行号，示例命令如下：

```
nl /etc/issue
```

more 一页一页地显示文件内容，示例命令如下：

```
more /etc/man.config
```

more 显示文件中从第 3 行起的内容，示例命令如下：

```
more +3 text.txt
```

more 显示所列出文件目录的详细信息，借助管道每次显示 5 行，示例命令如下：

```
ls -l | more -5
```

less 与 more 类似，但是比 more 更好的是可以往前翻页，示例命令如下：

```
less /etc/man.config
```

head 只看头几行，示例命令如下：

```
head /etc/man.config
```

head 默认情况下只显示前面 10 行。若要显示前 20 行，则示例命令如下：

```
head -n 20 /etc/man.config
```

tail 只看最后几行，示例命令如下：

```
tail /etc/man.config
```

tail 默认情况下只显示最后 10 行。若要显示最后 20 行，则示例命令如下：

```
tail -n 20 /etc/man.config
```

关于查看系统信息的一些示例，命令如下：

```
arch                              # 显示机器的处理器架构
uname -m                          # 显示机器的处理器架构
uname -r                          # 显示正在使用的内核版本
dmidecode -q                      # 显示硬件系统部件，SMBIOS/DMI
hdparm -i /dev/hda                # 列出一个磁盘的架构特性
hdparm -tT /dev/sda               # 在磁盘上执行测试性读取操作
cat /proc/cpuinfo                 # 显示 CPU info 的信息
cat /proc/interrupts              # 显示中断
cat /proc/meminfo                 # 校验内存使用
cat /proc/swaps                   # 显示哪些 swap 被使用
cat /proc/version                 # 显示内核的版本
cat /proc/net/dev                 # 显示网络适配器及统计
cat /proc/mounts                  # 显示已加载的文件系统
lspci -tv                         # 列出 PCI 设备
lsusb -tv                         # 显示 USB 设备
cal 2022                          # 显示 2022 年的日历表
clock -w                          # 将时间修改后保存到 BIOS
```

关于文件系统的一些示例，命令如下：

```
# 文件系统分析
badblocks -v /dev/hda1            # 检查磁盘 hda1 上的坏磁块
fsck /dev/hda1                    # 修复/检查 hda1 磁盘上 Linux 文件系统的完整性
fsck.ext2 /dev/hda1               # 修复/检查 hda1 磁盘上 ext2 文件系统的完整性
e2fsck /dev/hda1                  # 修复/检查 hda1 磁盘上 ext2 文件系统的完整性
e2fsck -j /dev/hda1               # 修复/设置文件系统在日志文件的路径
fsck.ext3 /dev/hda1               # 修复/检查 hda1 磁盘上 ext3 文件系统的完整性
fsck.vfat /dev/hda1               # 修复/检查 hda1 磁盘上 fat 文件系统的完整性
fsck.msdos /dev/hda1              # 修复/检查 hda1 磁盘上 dos 文件系统的完整性
dosfsck /dev/hda1                 # 修复/检查 hda1 磁盘上 dos 文件系统的完整性

# 初始化一个文件系统
mkfs /dev/hda1                    # 在 hda1 分区创建一个文件系统
mke2fs /dev/hda1                  # 在 hda1 分区创建一个 Linux ext2 的文件系统
mke2fs -j /dev/hda1               # 在 hda1 分区创建一个 Linux ext3(日志型)的文件系统
mkfs -t vfat 32 -F /dev/hda1      # 创建一个 FAT32 文件系统
fdformat -n /dev/fd0              # 格式化一个软盘
mkswap /dev/hda3                  # 创建一个 swap 文件系统

# swap 文件系统
mkswap /dev/hda3                  # 创建一个 swap 文件系统
swapon /dev/hda3                  # 启用一个新的 swap 文件系统
swapon /dev/hda2 /dev/hdb3        # 启用两个 swap 分区
```

在项目环境搭建中，经常要查看网络信息，包括网卡信息、IP 信息、路由信息、防火墙规则等，查看网络信息主要命令如下：

```
ipconfig
ifconfig
dig
route -n
```

lsof(List Open Files)是一个列出当前系统打开文件的工具。只需输入 lsof 命令便可以生成大量的信息，因为 lsof 命令需要访问核心内存和各种文件，所以必须以 root 用户的身份运行它才能充分地发挥其作用。

显示目录下被进程开启的文件，示例命令如下：

```
lsof +d /usr/local/
```

查看当前进程打开了哪些文件，示例命令如下：

```
lsof -c 进程名
```

查看某个端口被占用情况，示例命令如下：

```
lsof -i:6379
```

查看所有 TCP/UDP 链接，示例命令如下：

```
lsof -i tcp
```

查看某个用户打开了哪些文件，示例命令如下：

```
lsof -u rumenz
```

通过某个进程号显示该进程打开的文件，示例命令如下：

```
lsof -p 12345
```

netstat 是一个显示系统中所有 TCP/UDP/UNIX Socket 连接状态的命令行工具。它会列出所有已经连接或者等待连接状态的连接。该工具在识别某个应用监听哪个端口时特别有用，也能用它来判断某个应用是否正常地在监听某个端口。

显示系统所有的 TCP、UDP 及 UNIX 连接，示例命令如下：

```
netstat -a
```

使用 p 选项可以在列出连接的同时显示 PID 或者进程名称，而且它还能与其他选项连用，示例命令如下：

```
netstat -ap
```

列出端口号而不是服务名，示例命令如下：

```
netstat -an
```

top 命令用于显示当前系统正在执行的进程的相关信息，包括进程 ID、内存占用率、CPU 占用率等，这能帮助用户对系统正在发生的事情有个第一认识。

参数-c 用于显示完整的进程命令，示例命令如下：

```
top -c
```

参数-s 用于指定为保密模式，示例命令如下：

```
top -s
```

参数-p <进程号>用于将进程指定为显示，示例命令如下：

```
top -p 2
```

参数-n <次数>用于指定循环显示的次数，示例命令如下：

```
top -n 5
```

kill 命令用于将指定的信号发送到相应进程。如果不指定信号，则将发送 SIGTERM(15)以便终止指定进程。如果仍无法终止该程序，则可使用"-KILL" 参数，其发送的信号为 SIGKILL(9)，将强制结束进程，使用 ps 命令或者 jobs 命令可以查看进程号。root 用户将影响用户的进程，非 root 用户只能影响自己的进程。

先使用 ps 查找进程 pro1，然后用 kill 杀掉，示例命令如下：

```
kill -9 $(ps -ef | grep pro1)
```

在 Linux 系统中，如果想了解内存的状态，则可使用两个很重要的命令，即 free 和 vmstat。

free 命令用于显示系统内存的使用情况，包括物理内存、交互区内存(swap)和内核缓冲区内存。

显示内存使用情况，示例命令如下：

```
free
free -k
free -m
```

以总和的形式显示内存的使用信息，示例命令如下：

```
free -t
```

周期性地查询内存的使用情况，示例命令如下：

```
free -s 10
```

vmstat 命令是最常见的 Linux/UNIX 监控工具之一，可以展现给定时间间隔的服务器的状态值，包括服务器的 CPU 使用率、内存使用情况、虚拟内存交换情况，以及 I/O 读写情况。这个命令是查看 Linux/UNIX 比较好的命令，一是对 Linux/UNIX 都支持，二是相比 top，可以看到整个机器的 CPU、内存、I/O 的使用情况，而不是单单看到各个进程的 CPU 使用率和内存使用率(使用场景不一样)。

vmstat 工具的使用是通过两个数字参数来完成的，第 1 个参数是采样的时间间隔数，单位是秒；第 2 个参数是采样的次数，示例命令如下：

```
vmstat 2 1
```

其中，2 表示每隔 2 秒采集一次服务器状态，1 表示只采集一次，如果不将次数指定为 1，就会一直采集。

df 命令用于显示磁盘空间使用情况，获取硬盘被占用了多少空间，目前还剩下多少空间等信息，如果没有指定文件名，则所有当前被挂载的文件系统的可用空间都将被显示。默认情况下，磁盘空间将以 1KB 为单位显示，除非环境变量 POSIXLY_CORRECT 被指定，那样将以 512 字节为单位显示。

显示磁盘使用情况，示例命令如下：

```
df -l
```

以易读方式列出所有文件系统及其类型，示例命令如下：

```
df -haT
```

du 命令用于统计指定目录和文件所占用磁盘空间的大小。①du -a：为每个指定文件显示使用磁盘的情况；②du -s：为所有指定文件显示使用磁盘的情况。与 df 命令不同的是

Linux du 命令查看的是文件和目录使用的磁盘空间。

以易读方式显示文件夹及子文件夹大小，示例命令如下：

```
du -h scf/
```

显示几个文件或目录各自占用磁盘空间的大小，并统计它们的总和，示例命令如下：

```
du -hc test/ scf/
```

在 Linux 系统中编写脚本时，有时需要使用 date 命令获取日期时间进行判断，或者需要显示或设定系统的日期与时间。这个命令可以跟以下选项以便获取不同格式的日期时间。

-d <字符串>：显示字符串所指的日期与时间，字符串前后必须加上双引号。

-s <字符串>：根据字符串设置日期与时间，字符串前后必须加上双引号。

-u：显示 GMT。

%H：转换为小时(以 00～23 表示)。

%I：转换为小时(以 00～12 表示)。

%M：转换为分钟(以 00～59 表示)。

%s：总秒数，起算时间为 1970 年 1 月 1 日 00:00:00 UTC。

%S：秒(以本地的惯用法来表示)。

%a：星期的缩写。

%A：星期的完整名称。

%d：日期(以 01～31 表示)。

%D：日期(含年、月、日)。

%m：月份(以 01～12 表示)。

%y：年(以 00～99 表示)。

%Y：年(以四位数表示)。

显示明天的日期，示例命令如下：

```
date +%Y%m%d --date="+1 day"
```

显示工作目录，示例命令如下：

```
pwd
```

查看主机名称，示例命令如下：

```
hostname
```

查看操作系统名称等信息，示例命令如下：

```
uname -a
```

查看文件详细状态信息，示例命令如下：

```
stat file.txt
```

wc 命令用于统计文件中的行数、单词数、字节数等信息，后面跟不同的选项参数使用。默认不跟选项参数，wc 将计算指定文件的行数、字数及字节数，使用的命令如下：

```
wc file.txt
```

-l 用于统计行数，示例命令如下：

```
wc -l file.txt
```

-w 用于统计单词个数，示例命令如下：

```
wc -w file.txt
```

-c 用于统计字节数，示例命令如下：

```
wc -c file.txt
```

在 Linux 系统中，wget 命令使用得比较多，主要用来从网站下载文件，选项参数的使用方法有下面几种。

-q 用于无提示下载，示例命令如下：

```
wget -q http://linux.com/file.text
```

-b 用于后台下载，示例命令如下：

```
wget -b http://linux.com/file.text
```

-O 用于指定不同的文件名，示例命令如下：

```
wget -O test.text http://linux.com/file.text
```

-m 用于下载整个网站，示例命令如下：

```
wget -m http://linux.com
```

--no-check-certificate 用于绕过 SSL/TLS 证书的验证，示例命令如下：

```
wget --no-check-certificate http://linux.com/file.text
```

--user=< user_id >--password=< user_password >用于从受密码保护的网站下载文件，示例命令如下：

```
wget --user=admin --password=admin http://linux.com
```

在 Linux 系统中，已经安装的软件可以通过命令查询，以 rpm 命令为例，查询已安装的 RPM 软件信息。

格式：rpm-q 子选项软件名，示例命令如下：

```
rpm -q bashbash-4.1.2-15.el6_4.x86_64
```

-qa 用于查看已安装的所有 RPM 软件列表，示例命令如下：

```
rpm -qa | grep bashbash-4.1.2-15.el6_4.x86_64
```

-qi 用于查看指定软件的详细信息，示例命令如下：

```
rpm -qi bashbash-4.1.2-15.el6_4.x86_64
```

-ql 用于查询软件包的目录，示例命令如下：

```
rpm -ql bashbash-4.1.2-15.el6_4.x86_64
```

查询未安装的 RPM 包文件的命令格式为 rpm-qb 子选项 RPM 包文件。

-qpi 用于查看该软件的详细信息，示例命令如下：

```
rpm -qpi bashbash-4.1.2-15.el6_4.x86_64
```

-qpl 用于查看包内所含的目录和文件列表，示例命令如下：

```
rpm -qpl bashbash-4.1.2-15.el6_4.x86_64
```

安装升级 RPM 包文件的命令格式为 rpm 选项 RPM 包文件。

-i：安装一个新的 RPM 软件包（install）。

-U：升级，若未安装，则进行安装。

-h：以“#”号显示安装的进度。

-v：显示安装过程中的详细信息。

-F：更新某个 RPM 软件，若未安装，则放弃安装。

挂载光盘，示例命令如下：

```
mount /dev/sr0 /media/
```

卸载光盘，示例命令如下：

```
umount /dev/rs0
```

关于挂载一个文件系统的常用命令，示例命令如下：

```
mount /dev/hda2 /mnt/hda2              #挂载一个叫作 hda2 的磁盘，确定目录'/mnt/hda2'已经
                                       #存在
umount /dev/hda2                       #卸载一个叫作 hda2 的磁盘，先从挂载点 '/mnt/hda2'退出
fuser -km /mnt/hda2                    #当设备繁忙时强制卸载
umount -n /mnt/hda2                    #运行卸载操作而不写入 /etc/mtab 文件，当文件为只读
                                       #或当磁盘写满时非常有用
mount /dev/fd0 /mnt/floppy             #挂载一个软盘
mount /dev/cdrom /mnt/cdrom            #挂载一个 CD ROM 或 DVD ROM
mount /dev/hdc /mnt/cdrecorder         #挂载一个 CD RW 或 DVD ROM
mount -o loop file.iso /mnt/cdrom      #挂载一个文件或 ISO 镜像文件
mount -t vfat /dev/hda5 /mnt/hda5      #挂载一个 Windows FAT32 文件系统
mount /dev/sda1 /mnt/usbdisk           #挂载一个 USB 键盘或闪存设备
mount -t smbfs -o username=user,password=pass //WinClient/share /mnt/share
                                       #挂载一个 Windows 网络共享
```

在 Linux 系统中，经常会遇到通过源代码安装软件的情况，主要通过编译和安装，根据机器的型号和系统配置的不同，生成的文件稍有不同，主要通过以下几种方式进行编译。

进入设置模式，示例命令如下：

```
./configure
```

编译，示例命令如下：

```
make
```

编译安装，示例命令如下：

```
make install
```

在 Linux 系统中下载或者传输一个文件后，需要查看和校验文件的 md5 值，示例命令如下：

```
md5sum file.txt
```

重启命令，示例命令如下：

```
reboot
shutdown -r now
init 6
```

关机命令，示例命令如下：

```
halt -p
shutdown -h now
init 0
```

passwd 用来设置/更改用户的密码，命令格式为 passwd 选项用户名。
直接修改当前用户的密码，示例命令如下：

```
passwd
```

修改特定用户的密码，示例命令如下：

```
passwd frank
```

-d：清空用户密码。
-l：锁定用户账号。
-S：查看用户账号的状态(是否被锁定)。
-u：解锁用户账号。
-x,--maximum=DAYS：密码的最长有效时限。
-n,--mixiumum=DAYS：密码的最短有效时限。
-w,--warning=DAYS：在密码过期前多少天开始提醒用户。
-i,--inactive=DAYS：当密码过期后经过多少天该账号会被禁用。
关于用户和群组的一些使用，示例命令如下：

```
groupadd group_name                                    #创建一个新用户组
groupdel group_name                                    #删除一个用户组
groupmod -n new_group_name old_group_name              #重命名一个用户组
useradd -c "Name Surname " -g admin -d /home/user1 -s /bin/bash user1
                                                       #创建一个属于 "admin" 用户组的用户
useradd user1                                          #创建一个新用户
userdel -r user1                                       #删除一个用户 ( '-r' 排除主目录)
usermod -c "User FTP" -g system -d /ftp/user1 -s /bin/nologin user1     #修改用户属性
passwd                                                 #修改当前用户的密码
```

```
passwd user1                          #修改一个其他用户的密码 (只允许 root 执行)
chage -E 2005-12-31 user1             #设置用户密码的失效期限
pwck                                  #检查'/etc/passwd'的文件格式和语法修正及存在的用户
grpck                                 #检查'/etc/passwd'的文件格式和语法修正及存在的群组
newgrp group_name                     #登录一个新的群组以改变新创建文件的预设群组
```

查询用户身份标识的命令格式为 id 用户名,示例命令如下:

```
id root
```

w 用于查询已登录到主机的用户信息,示例命令如下:

```
w
```

与 w 命令类似,查询已登录到主机的用户,示例命令如下:

```
who
```

查询账号的详细信息的命令格式为 finger 用户名,示例命令如下:

```
finger
```

查询当前登录的账号名,示例命令如下:

```
whoami
```

将所有文件以树的形式列出来,示例命令如下:

```
tree
```

临时关闭防火墙,示例命令如下:

```
systemctl stop firewalld
```

永久关闭防火墙,示例命令如下:

```
systemctl disable firewalld
```

临时关闭 SELinux 安全机制,示例命令如下:

```
setenforce 0
```

永久关闭 SELinux 安全机制,示例命令如下:

```
sed -i '7 s/enforcing/disabled/' /etc/seLinux/config
```

ps(Process Status)命令用来查看当前运行的进程状态,一次性查看,如果需要动态连续的结果,则可使用 top。

显示当前所有进程环境变量及进程间关系,示例命令如下:

```
ps -ef
```

显示当前所有进程,示例命令如下:

```
ps -A
```

与 grep 联用查找某进程,示例命令如下:

```
ps -aux | grep apache
```

scp(Secure Copy)是一个在 Linux 下用来进行远程复制文件的命令。有时需要获得远程服务器上的某个文件,该服务器既没有配置 FTP 服务器,也没有进行共享,当无法通过常规途径获得文件时,只需通过简单的 scp 命令便可以达到目的。scp 命令的使用格式如下:

```
scp [可选参数] <file_source> <file_target>
```

-r:递归复制整个目录。

-P <port>:注意是大写的 P,port 是指定数据传输用到的端口号。

从本地将文件复制到远程,示例命令如下:

```
scp local_file remote_username@remote_ip:remote_folder
```

或者,示例命令如下:

```
scp local_file remote_username@remote_ip:remote_file
```

或者,示例命令如下:

```
scp local_file remote_ip:remote_folder
```

或者,示例命令如下:

```
scp local_file remote_ip:remote_file
```

第 1 条和第 2 条命令指定了用户名，命令执行后需要输入密码，第 1 条命令仅指定了远程的目录，文件名不变；第 2 条命令指定了文件名。

第 3 条和第 4 条命令没有指定用户名，命令执行后需要输入用户名和密码，第 3 条命令仅指定了远程的目录，文件名不变；第 4 条命令指定了文件名。

上面 4 条命令对应的应用实例命令如下：

```
scp /home/space/music/1.mp3 root@www.runoob.com:/home/root/others/music

scp /home/space/music/1.mp3\
root@www.runoob.com:/home/root/others/music/001.mp3

scp /home/space/music/1.mp3 www.runoob.com:/home/root/others/music

scp /home/space/music/1.mp3\
www.runoob.com:/home/root/others/music/001.mp3
```

从本地将目录复制到远程，示例命令如下：

```
scp -r local_folder remote_username@remote_ip:remote_folder
```

或者，示例命令如下：

```
scp -r local_folder remote_ip:remote_folder
```

第 1 条命令指定了用户名，命令执行后需要输入密码；第 2 条命令没有指定用户名，命令执行后需要输入用户名和密码。

上面两条命令对应的应用实例命令如下：

```
scp -r /home/space/music/ root@www.runoob.com:/home/root/others/

scp -r /home/space/music/ www.runoob.com:/home/root/others/
```

上面命令将本地 music 目录复制到远程 others 目录下。

从远程复制到本地，只要将从本地复制到远程的命令的后两个参数调换顺序即可，应用实例命令如下：

```
scp root@www.runoob.com:/home/root/others/music /home/space/music/1.mp3

scp -r www.runoob.com:/home/root/others/ /home/space/music/
```

如果远程服务器防火墙为 scp 命令设置了指定的端口，则需要使用-P 参数设置命令的端口号，示例命令如下：

```
scp -P 4588 root@www.runoob.com:/usr/local/sin.sh /home/
```

上述 scp 命令使用端口号 4588，使用 scp 命令要确保使用的用户具有可读取远程服务器相应文件的权限，否则 scp 命令无法起作用。

Linux 中 rcp 命令用于复制远程文件或目录，如同时指定两个以上的文件或目录，并且最后的目的地是一个已经存在的目录，则它会把前面指定的所有文件或目录复制到该目录中。rcp 命令使用格式如下：

```
rcp [-pr][源文件或目录][目标文件或目录]
rcp [-pr][源文件或目录...][目标文件]
```

-p：保留源文件或目录的属性，包括所有者、所属群组、权限与时间。

-r：递归处理，将指定目录下的文件与子目录一并处理。

假设本地主机当前账户为 rootlocal，远程主机账户为 root，要将远程主机(218.6.132.5)主目录下的文件 testfile 复制到本地目录 test 中，则有 3 种命令形式可实现，具体命令如下：

```
rcp root@218.6.132.5:./testfile testfile

rcp root@218.6.132.5:home/rootlocal/testfile testfile

rcp 218.6.132.5:./testfile testfile
```

同样，如果需要把本地文件或者文件夹上传到远程服务器上，只需交换一下上面命令的前后位置，把本地的 work 文件夹上传到远程服务器上的/home/rootlocal 目录下，也有 3 种命令形式可实现，具体命令如下：

```
rcp -pr work root@218.6.132.5:./

rcp -pr root@218.6.132.5:home/rootlocal/

rcp -pr 218.6.132.5:./
```

Linux 中，如果要备份一个文件，则可以使用 cp 命令，把文件复制一份，例如对/etc/resove.conf 文件进行备份，示例命令如下：

```
cp /etc/resove.conf /etc/resove.conf.bak
```

关于 Linux 系统中备份的一些示例，命令如下：

```
dump -0aj -f /tmp/home0.bak /home          #制作一个'/home'目录的完整备份

dump -1aj -f /tmp/home0.bak /home          #制作一个'/home'目录的交互式备份
```

```
restore -if /tmp/home0.bak                    #还原一个交互式备份

rsync -rogpav --delete /home /tmp             #同步两边的目录

rsync -rogpav -e ssh --delete /home ip_address:/tmp        #通过 SSH 通道同步目录

rsync -az -e ssh --delete ip_addr:/home/public /home/local   #通过 SSH 和压缩将一个远程
#目录同步到本地目录

rsync -az -e ssh --delete /home/local ip_addr:/home/public   #通过 SSH 和压缩将本地目录同步
#到远程目录

dd bs=1M if=/dev/hda | gzip | ssh user@ip_addr 'dd of=hda.gz'  #通过 SSH 在远程主机上执行一次
#备份本地磁盘的操作

dd if=/dev/sda of=/tmp/file1                   #将磁盘内容备份到一个文件

tar -Puf backup.tar /home/user                 #执行一次对'/home/user'目录的交互式备份操作

( cd /tmp/local/ && tar c . ) | ssh -C user@ip_addr 'cd /home/share/ && tar x -p'     #通过
SSH 在远程目录中复制一个目录内容

( tar c /home ) | ssh -C user@ip_addr 'cd /home/backup-home && tar x -p'     #通过 SSH 在远
#程目录中复制一个本地目录

tar cf - . | (cd /tmp/backup ; tar xf - )      #本地将一个目录复制到另一个地方,保留原有权
#限及链接

find /home/user1 -name '*.txt' | xargs cp -av --target-directory=/home/backup/ --parents
#从一个目录查找并将所有以.txt 结尾的文件复制到另一个目录

find /var/log -name '*.log' | tar cv --files-from=- | bzip2 > log.tar.bz2     #查找所有
#以.log 结尾的文件并制作成一个 bzip 包

dd if=/dev/hda of=/dev/fd0 bs=512 count=1      #执行一个将 MBR(Master Boot Record)内容复
#制到软盘的动作

dd if=/dev/fd0 of=/dev/hda bs=512 count=1      #从已经保存到软盘的备份中恢复 MBR 内容
```

Linux 中的打包文件一般以.tar 结尾,压缩的命令一般以.gz 结尾,而一般情况下打包和压缩是一起进行的,打包并压缩后的文件的后缀名一般为.tar.gz。tar 命令是 Linux 下比较常用的一种压缩和解压命令。

打包并压缩/test 目录下的所有文件,压缩后的压缩包的指定名称为 xxx.tar.gz,示例命令如下:

```
tar -zcvf xxx.tar.gz aaa.txt bbb.txt ccc.txt
```

或者，示例命令如下：

```
tar -zcvf xxx.tar.gz /test/*
```

将/test 目录下的 xxx.tar.gz 文件解压到当前目录下，示例命令如下：

```
tar -xvf xxx.tar.gz
```

将/test 目录下的 xxx.tar.gz 文件解压到根目录/usr 下，示例命令如下：

```
tar -xvf xxx.tar.gz -C /usr
```

其中，-C 代表指定解压的位置。

在 Windows 系统中对某些文件进行了压缩，将压缩文件传到 Linux 系统中使用，大多数 Windows 系统的压缩文件是 zip 格式的。在 Linux 系统中用到 zip 或者 unzip 时，需要自己进行安装，安装命令如下：

```
yum/dnf install -y unzip zip
#或者
apt-get/apt install -y unzip zip
```

将/home/html/目录下的所有文件和文件夹打包为当前目录下的 html.zip 文件，示例命令如下：

```
zip -q -r html.zip /home/html
```

如果当前在/home/html 目录下进行打包，则示例命令如下：

```
zip -q -r html.zip *
```

从压缩文件 data.zip 中删除文件 a，示例命令如下：

```
zip -dv data.zip a
```

查看压缩文件中包含的文件，示例命令如下：

```
unzip -l abc.zip
```

-v 参数用于查看压缩文件的目录信息，但是不解压该文件，示例命令如下：

```
unzip -v abc.zip
```

关于打包和压缩文件的一些示例，命令如下：

```
bunzip2 file1.bz2                          #解压一个叫作 file1.bz2 的文件
bzip2 file1                                #压缩一个叫作 file1 的文件
gunzip file1.gz                            #解压一个叫作 file1.gz 的文件
gzip file1                                 #压缩一个叫作 file1 的文件
gzip -9 file1                              #最大程度地压缩
rar a file1.rar test_file                  #创建一个叫作 file1.rar 的包
rar a file1.rar file1 file2 dir1           #同时压缩 file1、file2 及目录 dir1
rar x file1.rar                            #压缩 rar 包
unrar x file1.rar                          #解压 rar 包
tar -cvf archive.tar file1                 #创建一个非压缩的 tarball
tar -cvf archive.tar file1 file2 dir1      #创建一个包含了 file1、file2 及 dir1 的文件
tar -tf archive.tar                        #显示一个包中的内容
tar -xvf archive.tar                       #释放一个包
tar -xvf archive.tar -C /tmp               #将压缩包释放到 /tmp 目录下
tar -cvfj archive.tar.bz2 dir1             #创建一个 bzip2 格式的压缩包
tar -jxvf archive.tar.bz2                  #解压一个 bzip2 格式的压缩包
tar -cvfz archive.tar.gz dir1              #创建一个 gzip 格式的压缩包
tar -zxvf archive.tar.gz                   #解压一个 gzip 格式的压缩包
zip file1.zip file1                        #创建一个 zip 格式的压缩包
zip -r file1.zip file1 file2 dir1          #将几个文件和目录同时压缩成一个 zip 格式的压缩包
unzip file1.zip                            #解压一个 zip 格式压缩包
```

在日常开发中有一些有用的快捷键使用方法，Tab 有命令补全与文件补齐的功能，如果 Tab 接在一串指令的第 1 个字的后面，则为命令补全；如果 Tab 接在一串指令的第 2 个字以后，则为文件补齐。

若安装了 bash-completion 软件，则在某些指令后面使用 Tab 按键时，可以进行“选项/参数”的补齐功能。这个软件非常有用，如果经常使用 Linux 命令行，则可以提高命令的输入效率，安装命令如下：

```
yum/dnf install -y bash-completion
#或者
apt-get/apt install -y bash-completion
```

如果在 Linux 系统下输入了错误的指令或参数，想让当前的程序停掉，则可以按快捷键 Ctrl+C 终止。快捷键 Ctrl+D 表示键盘输入结束（End Of File，EOF 或 End Of Input）的意思。另外，它也可以用来取代 Exit 的输入。例如想要直接离开文字输入接口，可以直接按快捷键 Ctrl+D 退出。快捷键 Shift+Page Up 可以实现往前翻页，快捷键 Shift+Page Down 可以实现往后翻页。