linux基础篇（二）

> 作者：强哥
>
> 版本：QF1.0
>
> 版权：千锋java教研院

第一章：linux快捷键与命令格式

1.1、常用快捷键

```
`tab    命令或路径等的补全键，linux用的最多的一个快捷键 `
```

光标移动

```
`ctrl+a    光标回到行首 `
`ctrl+e    光标回到行尾`
`ctrl+f    光标向右移动一个字符 forwards`
`ctrl+b    光标向左移动一个字符 behind`
`esc+b     移动到当前单词的开头`
`esc+f     移动到当前单词的结尾`
`esc+t     当前单词向前移动`
```

剪切

```
`ctrl+u     剪切光标处到行首的所有字符，也就是删除`
`ctrl+k     剪切光标处到行尾的所有字符`
`ctrl+w     剪切光标前的一个单词(word)`
`ctrl+h     删除光标前的一个字符（相当于退格键）`
```

中断

```
`ctrl+c    中断终端正在执行的任务并开启一个新的一行`
```

1.2、linux命令终端格式

1.2.1 命令的提示符

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

- `[]`：这是提示符的分隔符号，没有特殊含义。
- `root`：显示的是当前的登录用户， 目前使用的是root用户登录。
- `@`：分隔符号，没有特殊含义。
- `localhost`：当前系统的简写主机名（完整主机名是localhost.localdomain）。
- `~`：代表用户当前所在的目录，此例中用户当前所在的目录是家目录。
- `#`：命令提示符。超级用户是#，普通用户是$

1.2.2 命令的基本格式

> 格式如下

```
[root@localhost ~]# command命令 [选项options] [参数parameter]
```

说明:

- `command`：命令名, 相应功能的英文单词或单词的缩写
- `[-options]`：选项, 可用来对命令进行控制, 也可以省略
- `parameter`：传给命令的参数, 可以是 **零个、一个 或者 多个**

举例:

```
如ls -l /home/，ls是命令，-l是选项，/home是参数。
```

1.3、help帮助指令

Linux常用命令**help命令** 用于显示shell内部命令的帮助信息。help命令只能显示shell内部的命令帮助信息。而对于外部命令的帮助信息只能使用man或者info命令查看，下面为大家分享一下Linux常用命令help命令具体使用方法。

> 命令格式

```
help [-dms] [PATTERN...]
```

选项说明

```
-d
 输出每个命令的简短描述
-m
 以类似于 man 手册的格式描述命令
-s
 只显示命令使用格式
```

实例

1.查看 help 自身的帮助信息。

```
help help
```

2.以类似于 man 手册格式查看 help 命令的帮助信息。

```
help -m help
```

3.查看 help 命令的简短描述。

```
help -d help
help - Display information about builtin commands.
```

4.查看 help 和 cd 命令使用格式。

```
help -s help cd
help: help [-dms] [pattern ...]
cd: cd [-L|[-P [-e]]] [dir]
```

> --help 选项

绝大多数命令都可以使用`--help`选项来查看帮助，者也是一种获取帮助的方法。

例如 `ls --help`，这种方法非常简单，输出的帮助信息基本上是`man`命令的信息简要版。

14、man指令

man命令是最常见的帮助命令，也是Linux最主要的帮助命令，基本信息如下：

> 命令格式：

`man [选项] [章节] 命令`

`man`命令交互快捷键：

- `上箭头`：向上移动一行
- `下箭头`：向下移动一行
- `PgUP`：向上翻一页
- `PgDn`：向下翻一页
- `g`：移动到第一页
- `G`：移动到最后一页
- `q`：退出
- `/字符串`：从当前向下搜索字符串
- `?字符串`：从当前向上搜索字符串
- `n`：当搜索字符串时，可以用n键找到下一个字符串
- `N`：当搜索字符串时，使用N键反向查询字符串。也就是说，如果使用“/字符

串”方式搜索，则N键表示向上搜索字符串；如果使用“?字符串”方式搜索，则N键表示向下搜索字符串

`man`手册的格式：

- `NAME`：命令名称及功能简要说明
- `SYNOPSIS`：用法说明，包括可用的选项
  - `[]`：可选内容
  - `<>`：必选内容
  - `a|b`：二选一
  - `{}`：分组
  - `...`：同一内容可出现多次
- `DESCRIPTION`：命令功能的详细说明，可能包括每一个选项的意义
- `OPTIONS`：说明每一项的意义
- `EXAMPLES`：使用示例

* `FILES`：此命令相关的配置文件
* `AUTHOR`：作者
* `COPYRIGHT`：版本信息
* `REPORTTING BUGS`：bug信息

- `SEE ALSO`：参考其他帮助

示例：
我们输入 `man ls`，它会在最左上角显示“LS（1）”，在这里，“LS”表示手册名称，而“（1）”表示该手册位于第一节章，同样，我们输 `man ifconfig` 它会在最左上角显示“IFCONFIG（8）”。也可以这样输入命令：“man [章节号] 手册名称”。

```
1 man`是按照手册的章节号的顺序进行搜索的，比如：`man sleep` 只会显示sleep命令的手册，如果想查看库函数sleep，就要输入：`man 3 sleep
```

第二章：目录及文件操作指令

2.1、目录操作命令

2.1.1 pwd 指令

`Linux`中使用`pwd`来查看`当前工作目录`的完整路径。

> 命令格式

```
pwd [参数选项]
```

常用参数

```
-L, --logical 打印环境变量"$PWD"的值，可能为符号链接。
-P, --physical （默认值）打印当前工作目录的物理位置。
--help 显示帮助信息并退出。
--version 显示版本信息并退出。
```

- -P 如果是链接目录,则输出真实的目录
- -L 目录连接链接时，输出连接路径

常用案例

1.查看当前的工作目录

```
> pwd
/home
```

2.目录连接链接时，pwd -P 显示出实际路径，而非使用连接（link）路径；pwd显示的是连接路径

```
> cd /etc/init.d
> pwd
/etc/init.d
> pwd -P
/etc/rc.d/init.d
> pwd -L
/etc/init.d
```

> /etc/init.d`是`/etc/rc.d/init.d`软连接,也就是`/etc/init.d`的真实目录是`/etc/rc.d/init.d

3.当前目录被删除了，而pwd命令仍然显示那个目录

```
> cd /tmp
> mkdir test
> cd test
> pwd
/tmp/test
>rm -rf ../test
>pwd
/tmp/test
```

2.1.2 cd 指令

cd是切换所在目录的命令

> 命令格式:

`cd [-L|-P] [dir]`

选项:

`-L`:（默认值）如果要切换到的目标目录是一个符号连接，那么切换到符号连接的目录。

`-P`: 如果要切换到的目标目录是一个符号连接，那么切换到它指向的物理位置目录。

参数:

dir（可选）： 指定要切换到的目录，可以是绝对路径(以根目录为参照物)或相对路径(以当前目录为参照物)

简单用法:

`cd ~`: 当前用户的家目录
`cd -`: 上一次所在目录
`cd .`: 当前目录
`cd ..`: 上级目录

练习:

1 查看当前所在目录
2 切换到 /usr/local(绝对路径)
3 切换到 上一级 /usr
4 切换到 /usr/tmp (相对路径)
5 切换回 /usr/local
6 后退到上一次所在目录

答案:

```
pwd
cd /usr/local
cd ..
cd tmp
cd /usr/local
cd -
```

2.1.3 ls 指令

ls 是最常见的目录操作命令，主要作用是显示目录下的内容

> 命令格式：

`ls [选项] [目录名]`

- 选项

```
-a：显示所有文件
-h：人性化显示，按照我们习惯的单位显示文件大小
-l：长格式显示
```

示例：

```
[root@localhost ~]# ls -l
#权限      引用计数 所有者 所属组   大小   文件修改时间          文件名
-rw-------. 1      root root   1446    12月 19 16:15 anaconda-ks.cfg
```

“-l” 选项用于显示文件的详细信息，那么“-l”选项显示的这 7 列分别是什么含义?
第一列：权限。
第二列：引用计数。文件的引用计数代表该文件的硬链接个数，而目录的引用计数代表该目录有多少个一级子目录。
第三列：所有者，也就是这个文件属于哪个用户。默认所有者是文件的建立用户
第四列：所属组。默认所属组是文件建立用户的有效组，一般情况下就是建立用户的所在组。
第五列：大小。默认单位是字节。
第六列：文件修改时间。文件状态修改时间或文件数据修改时间都会更改这个时间，注意这个时间不是文件的创建时间。
第七列：文件名。

练习

1 查看 /usr内容
2 查看所有 /usr内容(既包含隐藏,也包含非隐藏)
3 查看 /usr详细内容
4 简化 查看 /usr详细内容
5 易懂简化版 查看 /usr详细内容

答案

```
[root@localhost /]# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@localhost /]# ls -a
.  ..  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@localhost /]# ls -l
总用量 20
lrwxrwxrwx.   1 root root    7 8月  13 05:05 bin -> usr/bin
dr-xr-xr-x.   5 root root 4096 8月  13 05:19 boot
drwxr-xr-x.  20 root root 3240 8月  14 01:42 dev
drwxr-xr-x.  75 root root 8192 8月  14 05:55 etc
drwxr-xr-x.   2 root root    6 8月  15 02:56 home
lrwxrwxrwx.   1 root root    7 8月  13 05:05 lib -> usr/lib
lrwxrwxrwx.   1 root root    9 8月  13 05:05 lib64 -> usr/lib64
drwxr-xr-x.   2 root root    6 11月  5 2016 media
drwxr-xr-x.   2 root root    6 11月  5 2016 mnt
drwxr-xr-x.   2 root root    6 11月  5 2016 opt
dr-xr-xr-x. 112 root root    0 8月  14 01:42 proc
dr-xr-x---.   2 root root  151 8月  16 22:37 root
drwxr-xr-x.  23 root root  680 8月  14 05:55 run
lrwxrwxrwx.   1 root root    8 8月  13 05:05 sbin -> usr/sbin
drwxr-xr-x.   2 root root    6 11月  5 2016 srv
dr-xr-xr-x.  13 root root    0 8月  14 01:42 sys
drwxrwxrwt.  11 root root 4096 8月  16 03:15 tmp
drwxr-xr-x.  13 root root  155 8月  13 05:05 usr
```

```
drwxr-xr-x.  19 root root  267 8月  13 05:20 var
[root@localhost /]# ll
总用量 20
lrwxrwxrwx.   1 root root    7 8月  13 05:05 bin -> usr/bin
dr-xr-xr-x.   5 root root 4096 8月  13 05:19 boot
drwxr-xr-x.  20 root root 3240 8月  14 01:42 dev
drwxr-xr-x.  75 root root 8192 8月  14 05:55 etc
drwxr-xr-x.   2 root root    6 8月  15 02:56 home
lrwxrwxrwx.   1 root root    7 8月  13 05:05 lib -> usr/lib
lrwxrwxrwx.   1 root root    9 8月  13 05:05 lib64 ->
usr/lib64
drwxr-xr-x.   2 root root    6 11月  5 2016 media
drwxr-xr-x.   2 root root    6 11月  5 2016 mnt
drwxr-xr-x.   2 root root    6 11月  5 2016 opt
dr-xr-xr-x. 112 root root    0 8月  14 01:42 proc
dr-xr-x---.   2 root root  151 8月  16 22:37 root
drwxr-xr-x.  23 root root  680 8月  14 05:55 run
lrwxrwxrwx.   1 root root    8 8月  13 05:05 sbin -> usr/sbin
drwxr-xr-x.   2 root root    6 11月  5 2016 srv
dr-xr-xr-x.  13 root root    0 8月  14 01:42 sys
drwxrwxrwt.  11 root root 4096 8月  16 03:15 tmp
drwxr-xr-x.  13 root root  155 8月  13 05:05 usr
drwxr-xr-x.  19 root root  267 8月  13 05:20 var
[root@localhost /]# ls -lh
总用量 20K
lrwxrwxrwx.   1 root root    7 8月  13 05:05 bin -> usr/bin
dr-xr-xr-x.   5 root root 4.0K 8月  13 05:19 boot
drwxr-xr-x.  20 root root 3.2K 8月  14 01:42 dev
drwxr-xr-x.  75 root root 8.0K 8月  14 05:55 etc
drwxr-xr-x.   2 root root    6 8月  15 02:56 home
lrwxrwxrwx.   1 root root    7 8月  13 05:05 lib -> usr/lib
lrwxrwxrwx.   1 root root    9 8月  13 05:05 lib64 ->
usr/lib64
drwxr-xr-x.   2 root root    6 11月  5 2016 media
drwxr-xr-x.   2 root root    6 11月  5 2016 mnt
drwxr-xr-x.   2 root root    6 11月  5 2016 opt
dr-xr-xr-x. 112 root root    0 8月  14 01:42 proc
```

```
dr-xr-x---.   2 root root  151 8月  16 22:37 root
drwxr-xr-x.  23 root root  680 8月  14 05:55 run
lrwxrwxrwx.   1 root root    8 8月  13 05:05 sbin -> usr/sbin
drwxr-xr-x.   2 root root    6 11月  5 2016 srv
dr-xr-xr-x.  13 root root    0 8月  14 01:42 sys
drwxrwxrwt.  11 root root 4.0K 8月  16 03:15 tmp
drwxr-xr-x.  13 root root  155 8月  13 05:05 usr
drwxr-xr-x.  19 root root  267 8月  13 05:20 var
```

2.1.4 mkdir创建目录

通过 `mkdir` 命令 创建目录

> 基本语法

```
mkdir [-p] 要创建的目录
```

选项

- -p： 帮助你直接将所需要的目录(包含上一级目录)递归创建起来!

案例

```
[root@linux ~]# mkdir test
[root@linux ~]# mkdir -p test/test1
```

注意

```
通过 mkdir -p 目录名 命令 创建目录
注意: 新建目录的名称 不能与当前目录中 已有的目录或文件同名
```

2.1.5 rmdir 删除目录

`rmdir`命令的作用十分有限，只能删除空目录，一旦目录中有内容就会报错。所以一般不论删除的是文件还是目录，都会使用`rm`命令

> 基本语法：

```
rmdir   目录名称
```

案例

```
[root@linux121 ~]# mkdir test2
[root@linux121 ~]# rmdir test2
```

注意

```
如果该目录中存在文件或其他目录是该命令是不能删除的.
```

2.2、文件操作命令

2.2.1 touch 指令

touch命令用于修改文件或者目录的时间属性，包括存取时间和更改时间。若文件不存在，系统会建立一个新的文件。

> 语法

```
用法: touch [OPTION]… FILE…
```

实例

实例1：创建不存在的文件

`touch 不存在的文件`

```
[root@bogon ~]# touch demo.log
[root@bogon ~]# ll
总用量 4
-rw-------. 1 root root 1233 4月  29 19:03 anaconda-ks.cfg
-rw-r--r--. 1 root root    0 5月  10 09:26 demo.log
```

实例2：修改文件的末次访问时间

`touch 存在的文件`

```
[root@bogon ~]# touch demo.log
[root@bogon ~]# ll
总用量 4
-rw-------. 1 root root 1233 4月  29 19:03 anaconda-ks.cfg
-rw-r--r--. 1 root root    0 5月  10 09:27 demo.log
```

2.2.2 cp 复制指令

cp指令用于拷贝文件和目录。

> 语法

```
cp [选项] 文件1 备份文件名称
cp [选项] 文件1 文件2 文件3 .... 目录
```

选项：

```
-r  recursive (递归)递归复制目标目录的内容
```

案例:

```
 (1) 复制文件
[root@linux121 opt]# cp test.txt test1.txt
 (2) 递归复制整个文件夹
[root@linux121 opt]# cp -r abc /tmp
```

2.2.3 mv 移动剪切

通过`mv`命令可以用来 移动 文件 或 目录, 也可以给 文件或目录重命名

> 语法

```
mv    旧文件名        新文件名称
mv    被移动目录      目标目录
```

案例

```
1）重命名
[root@linux]# mv file1 file11  （把file1文件夹改名为file11）

2）移动文件
[root@linux]# mv file11 test  （把file11文件夹放到test文件夹内
```

2.2.4 rm 删除指令

rm (移除文件或目录)

> 语法:

```
rm [-fr] 文件或目录
```

参数

- -f： 就是 force 的意思，忽略不存在的文件，不会出现警告信息；
- -r： 递归删除啊！最常用在目录的删除了

案例

```
1）删除空目录
[root@linux]# rmdir test1
2）递归删除目录中所有内容
[root@linux]# rm -rf test2
```

2.3、查看文件指令

2.3.1 cat 指令

`cat`看完文件内容，一次显示所有的内容, 适合 查看内容较少 的文本文件

> 语法

```
cat [选项]  fileName
```

参数

- `-n`或`--number`: 由`1`开始对所有输出的行数编号。
- `-b`或`--number-nonblank`: 和`-n`相似，只不过对于空白行不编号

案例

1.查看内容

```
[root@localhost usr]# cat abc.txt
babbaba
abababab
ababbabaZZ
```

2.查看内容添加-n

```
[root@localhost usr]# cat -n abc.txt
     1  babbaba
     2  abababab
     3  ababbabaZZ
```

3.查看内容添加-b

```
[root@localhost usr]# cat -b abc.txt
     1  babbaba
     2  abababab
     3  ababbabaZZ
```

2.3.2 more 指令

more 命令是常用的文本文件阅读工具，类似于 cat，不过以一页一页的形式显示，更方便使用者逐页阅读。一般文件过大时使用 more 浏览,文件较小时使用 cat。

> 命令格式

```
more [OPTIONS] FILE [...]
```

参数

```
-NUM 指定每屏显示的行数为 NUM
+/STRING   从匹配搜索字符串 STRING 所在行的前两行开始显示
+NUM       从文件第 NUM 行开始显示
```

最常用交互式的指令有：

```
(1) 回车键向下滚动一行;
(2) 空格键（Space）显示下一页;
(3) b 键（back）回显上一页;
(4) q 或 Q 键退出。
```

案例

（1）查看文件内容, 按下回车逐行向下浏览。

```
more /etc/passwd
```

（2）指定每屏显示行数，使用选项 -NUM，这里指定显示 5 行。

```
more -5 /etc/passwd

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

（3）从文件第 NUM 行开始显示，使用选项 +NUM，这里从第 3 行开始显示。

```
more -5 +3 /etc/passwd

daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
```

（4）从匹配搜索字符串 STRING 的文件位置开始显示。比如从 /etc/passwd 中搜索 adm 所在行的前两行开始显示。

```
more -5 +/adm /etc/passwd

...skipping
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
```

2.3.3 less 指令

命令简介

less 命令是常用的文本文件阅读工具，类似于 more，是加强版的 more 命令。less 主要用于浏览大文件，加载文件时不会读取整个文件，相比于 vim 或 nano 等文本编辑器，启动会更快。

> 格式

```
less [OPTIONS] [FILE]...
```

参数

```
-m 显示类似more命令的百分比
-N 显示每行的行号
```

`less 的命令行选项很少使用，浏览文件时常用的是交互式命令。`

交互式命令

```
b 向后翻一页
Q 退出less 命令
空格键 滚动一页
回车键 滚动一行

1) 向前搜索
    / ： 使用一个模式进行搜索，并定位到下一个匹配的文本
    n ： 向前查找下一个匹配的文本
    N ： 向后查找前一个匹配的文本


2) 全屏导航
    ctrl + F ： 向前移动一屏
    ctrl + B ： 向后移动一屏
```

```
    ctrl + D : 向前移动半屏
    ctrl + U : 向后移动半屏

```

案例

（1）查看文件，同时显示行号。

```
less -N /etc/passwd

1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
...
```

（2）执行命令，然后分页显示。

```
ps -ef | less -N
```

（3）浏览多个文件。

```
less a.txt b.txt
```

在浏览 a.txt 时，输入 :n 后，切换到 b.txt，输入 :p 后，切换回 a.txt。也可以使用 :e 命令打开另一个文件。

2.3.4 head 指令

命令简介

head 命令用于显示文件开头的内容。在默认情况下，显示文件的头 10 行内容。

> 命令格式

```
head [OPTION]... [FILE]...
```

参数说明

```
-c, --bytes=[-]K
  显示每个文件的前 K 字节内容；如果附加 - 参数，则显示每个文件最后 K 字节外的所有内容
-n, --lines=[-]K
  显示每个文件的前 K 行内容；如果附加 - 参数，则显示每个文件最后 K 行外的所有内容
```

案例

（1）显示文件 /etc/passwd 的前 10 行。

```
head /etc/passwd

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
```

（2）显示文件 /etc/passwd 的前 3 行。

```
head -n3 /etc/passwd

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

（3）同时显示多个文件的前 3 行，以 /etc/passwd 和 /etc/group 为例。

```
head -n3 /etc/passwd /etc/group

==> /etc/passwd <==
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin

==> /etc/group <==
root:x:0:
bin:x:1:
daemon:x:2:
```

2.3.5 tail 指令

命令简介

tail 命令用于显示文件尾部内容，与命令 head 作用相反。默认情况下，显示文件的末尾 10 行内容。

> 命令格式

```
tail [OPTION]... [FILE]...
```

可以指定多个文件 FILE，种情况下，输出的内容前会列出所属文件名。如果未给定 FILE 或者 FILE 是 -，则从标准输入读取。

参数说明

```
-f  按照指定时间间隔输出文件追加的内容;
-n  输出最后 N 行，而非默认的最后 10 行
```

案例

（1）显示文件 /etc/passwd 的末尾 10 行。

```
tail /etc/passwd

rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
saslauth:x:996:76:Saslauthd user:/run/saslauthd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
uucp:x:10:14:Uucp user:/var/spool/uucp:/sbin/nologin
nslcd:x:65:55:LDAP Client User:/:/sbin/nologin
arpwatch:x:77:77::/var/lib/arpwatch:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
mysql:x:27:27:MariaDB Server:/var/lib/mysql:/sbin/nologin
```

(2) 显示文件 /etc/passwd 的末尾 3 行。

```
tail -n3 /etc/passwd

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
mysql:x:27:27:MariaDB Server:/var/lib/mysql:/sbin/nologin
```

(3) 动态追踪 最后10行内容 且 退出

```
tail -10f 文件名 (ctrl + c 是退出)
```

(4) 显示多个文件的最后 3 行，并且默认会显示文件名。

```
tail -n3 /etc/passwd /etc/group

==> /etc/passwd <==
mqq:x:500:501::/usr/local/app:/bin/bash
dev_mqq:x:501:501::/usr/local/dev:/bin/bash
dev:x:502:501::/home/dev:/bin/bash

==> /etc/group <==
screen:x:84:
admin:x:500:
mqq:x:501:
```

第三章：其他常用指令

3.1 日期和时间命令

`date 显示当前时间`

> 基本语法：注意命令与参数之间有空格

```
(1) date （功能描述：显示当前时间）
(2) date +%Y （功能描述：显示当前年年份）
(3) date +%m （功能描述：显示当前月份）
(4) date +%d （功能描述：显示当前是哪一天）
(5) date +%Y%m%d … （功能描述：显示当前年年月日各种格式 ）
(6) date "+%Y-%m-%d %H:%M:%S" 或者单引号也可以 （功能描述：显示年年月月日日时分秒）
```

案例

```
[root@linux /]# date
[root@linux /]# date +"%Y-%m-%d"(注意date后面有个空格再加 )
[root@linux /]# date "+%Y-%m-%d %H:%M:%S"(注意date后面有个空格再加；注意%d空格%H)
```

显示的是字符串串描述的时间，不是当前时间。

`设置系统时间`

> 基本语法：

```
date -s 字符串时间
```

案例

```
[root@linux /]# date -s "2022-05-20 20:52:18"
```

3.2 cal查看日历

1）基本语法：

```
cal [选项] （功能描述：不加选项，显示本月日历）
```

选项：

```
-3 ，显示系统前一个月，当前月，下一个月的日历
具体某一年年，显示这一年年的日历。
```

2）案例：

```
[root@linux /]# cal
[root@linux /]# cal -3
[root@linux /]# cal 2022
```

3.3 搜索命令

3.3.1 find

find命令用来在指定目录下查找文件

> 命令格式:

`find [搜索路径] [选项]`

- 搜索路径：省略则默认为当目录，相当于 "find ."

选项

```
-name 按照文件名称搜索，支持通配符模糊查询
-iname 此参数的效果和指定“-name”参数类似，但忽略字符大小写的差别
-path 查找路径包含范本样式的文件或目录
-regex 正则表达式搜索
-iregex 同"-regex"，忽略大小写
-size [+|-]文件大小[cwbkMG] ：查找符合指定的文件大小的文件
  "+" 的意思是搜索比指定大小还要大的文件，
  "-" 的意思是搜索比指定大小还要小的文件
  "cwbkMG"是单位，c——字节，w——字(2字节)，b——块(512字节)，
  k——千字节，M——兆字节，G——吉字节。如果不写单位默认是b
-atime [+|-]天数：按照文件最后一次访问时间搜索，单位每天
  "+"、"-"的含义，例如"5"表示恰好5天前的那一天，
  "+5"超过5天前的时间，"-5"5天内的时间。(以下按时间搜索选项中"+"、"-"含义相同)
-mtime [+|-]天数：按照文件数据最后一次修改时间搜索，单位每天
-ctime [+|-]天数：按照文件元数据(如权限等)最后一次修改时间搜索，单位每天
-amin [+|-]分钟数：按照文件最后一次访问时间搜索，单位每分钟
-mmin [+|-]分钟数：按照文件数据最后一次修改时间搜索，单位每分钟
-cmin [+|-]分钟数：按照文件元数据(如权限等)最后一次修改时间搜索，单位每分钟
-perm [+|-]权限数值：查找符合指定的权限数值的文件或目录。例如，权限数值为"766"表示权限恰好等于766，"-766"表示文件权限必须全部包含766，"+766"表示文件权限包含766任意一个权限
-uid 用户ID：查找所有者是指定用户ID的文件
-user 用户名：查找所有者是指定用户名的文件
-gid 组ID：查找所有组是指定组ID的文件
-group 组名：查找所有组是指定组名的文件
```

-nouser：查找没有所有者的文件

按照所有者和所有组搜索时，"-nouser"选项比较常用，主要用于查找垃圾文件。没有所有者的文件比较少见，那就是外来文件，比如光盘和U盘的文件是由Windows复制的，在Linux中查看就是没有所有者的文件，再比如手工源码包安装的文件也可能没有所有者

-type 文件类型：只寻找符合指定的文件类型的文件

f——普通文件，l——符号连接，d——目录，c——字符设备，b——块设备，s——套接字，p——Fifo

-empty：查找文件大小为0的文件

案例

```
#==================根据文件名或者正则表达式进行匹配=====================
#列出当前目录及子目录下所有文件和文件夹
[root@localhost ~]# find .
#在`/home`目录下查找以.txt结尾的文件名
[root@localhost ~]# find /home -name "*.txt"
#同上，但忽略大小写
[root@localhost ~]# find /home -iname "*.txt"
#当前目录及子目录下查找所有以.txt和.pdf结尾的文件
[root@localhost ~]# find . \( -name "*.txt" -o -name "*.pdf" \)
或
[root@localhost ~]# find . -name "*.txt" -o -name "*.pdf"
#查找路径包含local的文件或者目录
[root@localhost ~]# find /usr/ -path "*local*"
#基于正则表达式匹配文件路径
[root@localhost ~]# find . -regex ".*\(\.txt\|\.pdf\)$"


#========================逻辑运算符=========================
#查找文件大小超过2k并且是普通文件类型的文件
[root@localhost ~]# find . -size +2k -a -type f
#找出/home下不是以.txt结尾的文件
```

```
[root@localhost tmp]# find . -not -name "*.txt"
或
[root@localhost ~]# find /home ! -name "*.txt"
```

3.3.2 grep

grep命令的作用是在文件中提取和匹配符合条件的字符串行，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.

> 命令格式:

`grep [选项] 搜索内容 文件名`

选项

```
-c:只输出匹配行的计数
-i: 忽略大小写
-n: 输出行号
-w:显示整个单词
-R或-r: 递归查找目录下的所有文件内容
```

案例:

```
1)在applicationContext_dao.xml文件中查找name
2)查找name统计行号
3)统计jdbc的个数
4)查找dataSource 并忽略大小写
5)查找mapper单词
6)递归查询/usr目录下 含有name的字段
```

3.4 重定向输出>和>>

使用重定向运算符“>”和“>>”将终端输出保存到文件中。

>：它将输出重定向到文件并覆盖文件的现有内容。

> 语法：

```
$ command > [filename]
```

例如，要将hostnamectl命令输出保存到文件“myfile.txt”，请运行：

```
$ hostnamectl > myfile.txt
```

>>：它将输出附加到文件的末尾。

> 语法：

```
$ command >> [filename]
```

例如，要将aaa.txt输出附加到文件“myfile.txt”，请运行：

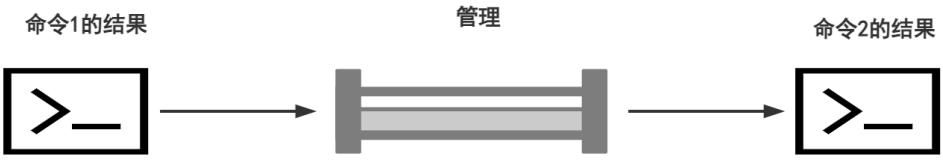
```
$ cat aaa.txt >> myfile.txt
```

现在，您可以使用cat命令从myfile.txt文件中查看这两个命令的输出：

```
$ cat myfile.txt
```

3.5 管道 |

通过 命令1 | 命令2 可以将命令1的结果 通过 命令2 作进一步的处理



目的：将命令1的结果通过管道交给命令2，再做进一步处理

```
#用输出重定向，把"ll"命令的输出保存到/root/testfile
[root@localhost ~]# ll -a /etc/ > /root/testfile
#然后用more分屏显示
[root@localhost ~]# more /root/testfile
```

这样操作实在是不方便，这时可以利用管道符，命令如下：

```
#把"ll"命令的输出作为"more"命令的操作对象
[root@localhost ~]# ll -a /etc/ | more
```

示例2：

```
统计具体的网络连接数量（"grep"命令筛选，"wc"命令统计）
ls /etc/ | grep ^pro
```

示例3：

```
第二步 ： 搜索存在关键字的行数
ls /etc | grep ^pro | wc -l
```

3.6 逻辑控制&&

用&&连接两个命令，前一个命令执行成功，才会执行后一命令。

很多时候我们有这样的需求，比如打开对应的日志信息查看等。

> 格式：

`command1 && command2`

案例:

1 打印1.txt内容 且 打印2.txt内容

```
cat 1.txt && cat 2.txt
```

2.将文件test.txt文件复制备份为test.bak.txt , 然后显示副本test.bak.txt文件内容

```
cp test.txt  test.bak.txt && cat test.bak.txt
```

3.启动tomcat并使用tail命令查看日志

```
./startup.sh && tail -100f catalina.out
```

3.7 history查看历史指令

列出曾经输入的命令

我们直接输入 history 命令:

```
[roc@roclinux ~]$ history
    1  cat .bash_history
    2  echo "" > .bash_history
    3  ls
    4  man ls
    5  date
    6  logout
    7  history
```

这里有一个问题，虽然命令都列了出来，但没有时间戳呀，也不知道命令是什么时候执行的，这怎么能称得上是“history”呢？不用担心，我们试试下面的命令。

```
#我们设置了一个环境变量
```

```
[roc@linux ~]$ export HISTTIMEFORMAT='%F %T '

#再来看history
[roc@linux ~]$ history
    1  2022-06-06 12:15:59 cat .bash_history
    2  2022-06-06 12:15:59 echo "" > .bash_history
    3  2022-06-06 12:15:59 ls
    4  2022-06-06 12:15:59 man ls
    5  2022-06-06 12:15:59 date
    6  2022-06-06 12:15:59 logout
    7  2022-06-06 12:16:16 history
    8  2022-06-06 12:18:15 export HISTTIMEFORMAT='%F %T '
    9  2022-06-06 12:18:20 history
```

清除所有的命令

安全是计算机世界里的头等大事，出于安全的考虑，我们常常需要清除曾经输入的命令，避免被其他别有用心的人看到。这种情况下，我们就要用 -c 选项来帮忙了。

```
[roc@linux ~]$ history -c
```

第四章：Linux打包（归档）和压缩

4.1 打包和压缩的概念和区别

`归档`

也称为打包，指的是一个文件或目录的集合，而这个集合被存储在一个文件中。归档文件没有经过压缩，因此，它占用的空间是其中所有文件和目录的总和。

`压缩`

压缩则是将一个大的文件通过一些压缩算法变成一个小文件

4.2 打包和解包

4.2.1 打tar包

`Linux` 下最常用的打包程序就是 `tar` 了，使用 `tar` 程序打出来的包我们常称为 `tar包`，tar包 文件的命令通常都是以 `.tar` 结尾的。生成 `tar包` 后，就可以用其它的程序来进行压缩了。

`tar` **打包**命令格式

```
# 将 一系列文件 打包成 一个大文件
tar -cvf 打包名.tar 被打包的目录
tar -cvf 打包名.tar 被打包的文件1 被打包的文件2 被打包的文件3
```

`tar` 选项说明

命令	英文	含义
c	create	生成档案文件, 创建打包文件
v	verbosely	显示打包或解打包过程
f	file	指定打包文件名(.tar)或压缩包文件名

练习：

```
#打包不会压缩
[root@localhost ~]# tar -cvf test.cfg.tar  test.cfg
```

4.2.2 解tar包

- **类似将 冬天的衣服 从 袋子里取出来**

`tar` **解包**命令格式

```
# 将一个打包后的 分解成 一系列小文件, 分解位置为 当前目录
tar -xvf 打包名.tar

# 将一个打包后的 分解成 一系列小文件, 分解位置为 指定目录
tar -xvf 打包名.tar -C 解包路径位置
```

命令	英文	含义
x	extract (提取)	解包
C (大写C)	directory (目录)	默认保存到当前目录, 通过`-c`更改提取目录, 注意: 提取目录必须存在

```
#解打包到当前目录
[root@localhost ~]# tar -xvf anaconda-ks.cfg.tar
#解打包到指定目录
[root@localhost ~]# tar -xvf anaconda-ks.cfg.tar -C /testdir/
```

4.3 gzip格式压缩和解压缩

gzip命令是".gz"格式的压缩和解压缩命令,gzip命令对文本文件有60%～70%的压缩率。

> 命令格式

```
# 压缩文件
tar -zcvf 打包压缩文件名.tar.gz 被压缩的文件/目录

# 解压缩文件
tar -zxvf 打包文件.tar.gz

# 解压缩到指定路径
tar -zxvf 打包文件.tar.gz -C 目录路径

```

`tar` 的选项说明

命令	英文	含义
z	gzip	使用gzip压缩和解压缩

练习:

```
#把/tmp/目录直接打包并压缩为".tar.gz"格式
[root@localhost ~]# tar -zcvf tmp.tar.gz /tmp/
#解压缩并解打包".tar.gz"格式文件
[root@localhost ~]# tar -zxvf tmp.tar.gz
```

```
练习1: 将1.txt、2.txt、3.txt 打包压缩成 123.tar.gz文件(gzip压缩格式)
练习2: 将有内容的aaa目录 打包成 aaa.tar.gz 文件(gzip压缩格式)
练习3: 将 123.tar.gz 解压到 当前目录中(gzip压缩格式)
练习4: 将 aaa.tar.gz 解包到 /aaa/bbb 目录中(gzip压缩格式)
```

4.4 bzip2 格式 压缩 和 解压缩

".bz2" 格式是Linux的另一种压缩格式，从理论上来讲，".bz2" 格式的算法更新进、压缩比更好；而 ".gz" 格式相对来讲压缩的时间更快。

- 在 `Linux` 中, bzip2 压缩文件格式是 `xxx.tar.bz2`
- 在 `tar` 命令中有一个选项 -j 可以调用 `bzip2` , 可以实现压缩和解压缩的功能

> 命令格式

```
# 压缩文件
tar -jcvf 打包压缩文件名.tar.bz2 被压缩的文件/目录

# 解压缩文件
tar -jxvf 打包文件.tar.bz2

# 解压缩到指定路径
tar -jxvf 打包文件.tar.bz2 -C 目录路径
```

```
注意事项：如果报错tar (child): bzip2：无法 exec：没有那个文件或目录
要安装bzip2的包
yum install -y bzip2
```

`tar` 的选项说明

命令	英文	含义
j	bzip2	使用bzip2压缩和解压缩

练习：

```
#把/tmp/目录直接打包并压缩为".tar.bz2"格式
[root@localhost ~]# tar -jcvf tmp.tar.gz /tmp/
#解压缩并解打包".tar.bz2"格式文件
[root@localhost ~]# tar -jxvf tmp.tar.gz
```

```
练习1：将1.txt、2.txt、3.txt 打包压缩成 123.tar.bz2文件(bzip2压缩格式)
练习2：将有内容的aaa目录 打包成 aaa.tar.bz2 文件(bzip2压缩格式)
练习3：将 123.tar.bz2 解压到 当前目录中(bzip2压缩格式)
练习4：将 aaa.tar.bz2 解包到 /ccc/bbb 目录中(bzip2压缩格式)
```

小结

```
打包压缩：tar -jcvf 打包之后的文件名.tar.bz2    被打包压缩的目录或文件名
解包解压缩：tar -jxvf 打包之后的文件名.tar.bz2  [ -C  指定解包位置 ]
```

第五章：vi编辑器

5.1、vi与vim

5.1.1 vi介绍

vi编辑器通常被简称为vi，而vi又是visual editor的简称，是**效率很高的文本编辑器**。它可以执行输出、删除、查找、替换、块操作等众多文本操作，而且用户可以根据自己的需要对其进行定制，这是其他编辑程序所没有的。

5.1.2 vim介绍

Vim是从 Vi 发展出来的一个文本编辑器，可以看做是 Vi 的增强版本，可以主动的以字体颜色辨别语法的正确性，方便程序设计代码补全、编译及错误跳转等方便编程的功能特别丰富，vim适用于coding。

注意：`如果在命令行模式下输入“vim”，输出结果为“Command not found”，则表示此系统中未安装 Vim`

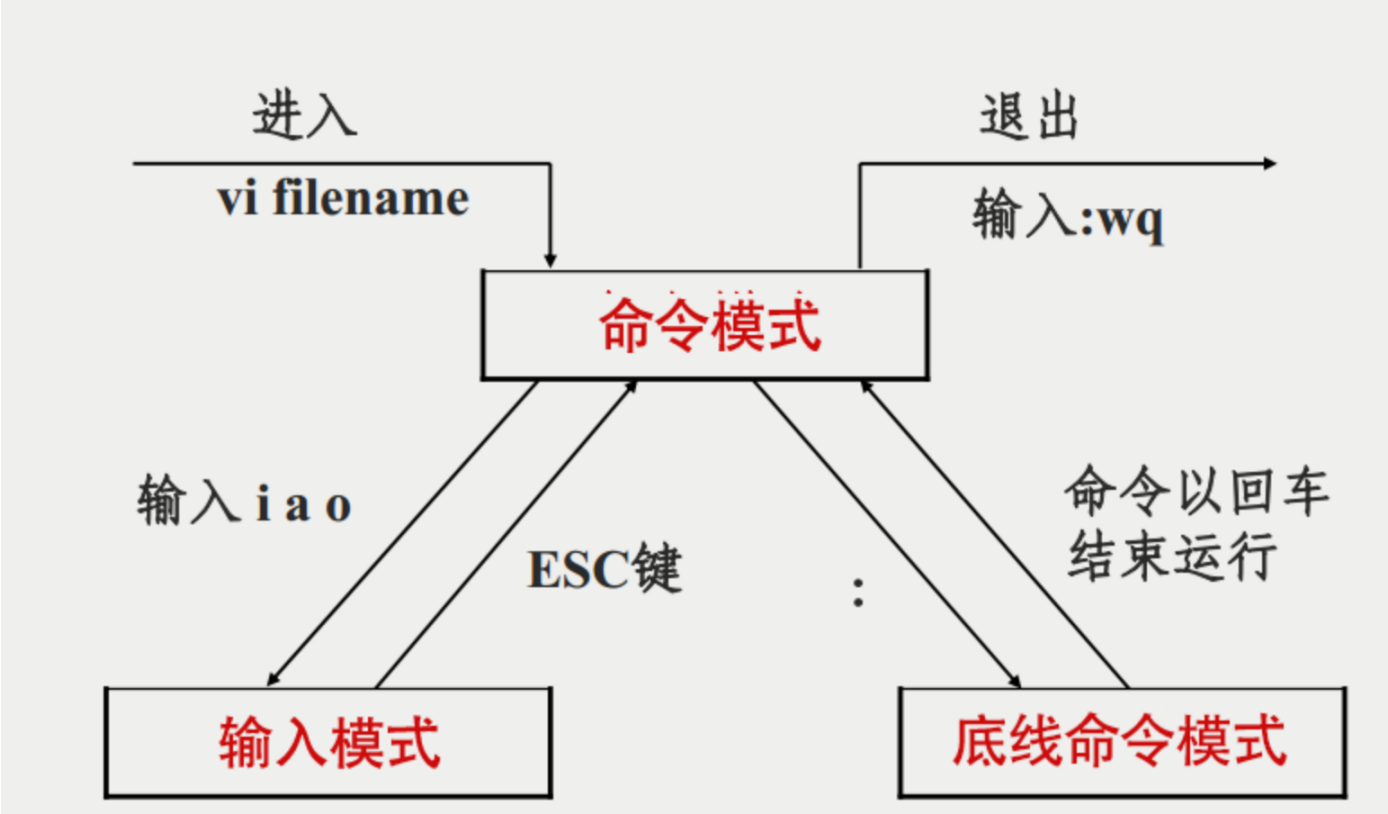
CentOS 系统中，使用如下命令即可安装 Vim：

```
yum -y install vim
```

5.2、vi编辑器三种模式

1. 命令模式
   - **打开文件首先进入命令模式**, 是使用vi的**入口**
   - 通过 **命令** 对文件进行常规的编辑操作, 例如 **定位 翻页 复制 粘贴 删除**
   - 在其他图形编辑器下, 通过 快捷键 或者 鼠标 实现的操作, 都在 **命令模式** 下实现
2. 末行模式 – 执行

   保存 退出等操作
   - 要退出 vi 返回到控制台, 需要在莫行模式下输入命令
   - **末行模式** 是 vi 的**出口**
3. **编辑模式** – 正常的编辑文字



5.2.1 命令/一般模式:

用户刚刚启动 vi/vim，便进入了命令模式。

此状态下敲击键盘动作会被Vim识别为命令，而非输入字符。

以下是常用的几个命令:

```
i/a/o 切换到插入模式，以输入字符。`(i：光标不动 o：另起一行 a：光标到下一个字符)`
: 切换到底线命令模式，以在最底一行输入命令。
```

5.2.2 编辑/插入模式

在命令模式下按下i就进入了输入模式，在输入模式中，可以使用以下按键:

```
字符按键以及Shift组合，输入字符
ENTER，回车键，换行
BACK SPACE，退格键，删除光标前一个字符
DEL，删除键，删除光标后一个字符
方向键，在文本中移动光标
HOME/END，移动光标到行首/行尾
Page Up/Page Down，上/下翻页
Insert，切换光标为输入/替换模式，光标将变成竖线/下划线
ESC，退出输入模式，切换到命令模式
```

5.2.3 底线命令模式

在命令模式下按下:（英文冒号）就进入了底线命令模式。

底线命令模式可以输入单个或多个字符的命令，可用的命令非常多。

```
:q 退出程序
:w 保存文件
```

按ESC键可随时退出底线命令模式。

5.3 三种模式下的指令

5.3.1 一般模式下指令

> 光标移动

移动光标的方法	
h 或 向左箭头键(←)	光标向左移动一个字符
j 或 向下箭头键(↓)	光标向下移动一个字符
k 或 向上箭头键(↑)	光标向上移动一个字符
l 或 向右箭头键(→)	光标向右移动一个字符
[Ctrl] + [f]	屏幕『向下』移动一页，相当于 [Page Down]按键 (常用)
[Ctrl] + [b]	屏幕『向上』移动一页，相当于 [Page Up] 按键 (常用)

> **撤销和恢复撤销**

命令	英文	功能
u	undo	撤销上次的命令(ctrl + z)
Ctrl + r	uredo	恢复撤销的命令

> **复制和剪切**

- vi 中提供有一个 被复制文本的缓冲区
  - 复制 命令会将选中的文字保存在缓冲区
  - 删除 命令删除的文字会被保存在缓冲区
  - 在需要的位置, 使用 粘贴 命令可以将缓冲对的文字插入到光标所在的位置
- 提示:
  - 命令 d 、x 类似于图形界面的 **剪切操作** – ctrl + x

- 命令 y 类似于 图形界面的 **复制操作** – Ctrl + C
- 命令 p 类似于图形界面的 **粘贴操作** – Ctrl + v
- vi中的文本缓冲区只有一个,如果后续做过 复制、剪切操作, 之前缓冲区中的内容会被替换.

命令	英文	功能
y(复制命令)	copy	复制
yy	copy	复制一行,可以nyy复制多行
d(剪切命令)	delete	剪切
dd(剪切)	delete	剪切一行, 可以 ndd 剪切n行
p	paste	粘贴

5.3.2 一般模式切换到编辑模式

> 进入到编辑模式

命令	英文	功能	常用
i	insert	在当前字符前插入文本	常用
I	insert	在行首插入文本	较常用
a	append	在当前字符后添加文本	常用
A	append	在行末添加文本	较常用
o		在当前行后面插入一空行	常用
O		在当前行前面插入一空行	常用

5.3.3 底行模式下命令

> 保存退出

命令	英文	功能
w	write	保存
q	quit	退出,如果没有保存,不允许退出
q!	quit	强行退出,不保存退出
wq	write & quit	保存且退出(末行模式)
x		保存并退出(末行模式)

> **常规查找**

命令	功能
/str	查找str

- 查找到指定内容之后, 使用

```
Next
```

查找下一个出现的位置

- `n` : 查找下一个
- `N` : 查找上一个

> 替换

命令	功能
:s/old/new	用new替换行中首次出现的old
:s/old/new/g	用new替换行中所有的old
:n,m s/old/new/g	用new替换从n到m行里所有的old
:%s/old/new/g	用new替换当前文件里所有的old

5.4、vi 编辑器练习

```
1 使用vim创建一个文件
2 进入编辑模式
3 添加内容  修改内容    删除内容
4 保存后退出
5 一般模式下完成删除复制和粘贴的操作
```

5.5、处理常见问题

如果 vi 异常退出, 在磁盘上可能会保存有 交换文件 下次再使用 vi 编辑文件时, 会看到以下屏幕信息, 按下字母 d 删除交换文件即可,swp文件时在使用vim/vi时产生的一个临时文件，且它是一个隐藏文件，用于保存缓存中还没有保存的数据.

```
E325: ATTENTION
Found a swap file by the name ".1.txt.swp"
          owned by: root   dated: Tue Dec 13 21:20:50 2022
         file name: /usr/test/1.txt
          modified: YES
         user name: root   host name: bogon
        process ID: 88792
While opening file "1.txt"
             dated: Tue Dec 13 18:19:29 2022

(1) Another program may be editing the same file.  If this is the case,
    be careful not to end up with two different instances of the same
    file when making changes.  Quit, or continue with caution.
(2) An edit session for this file crashed.
    If this is the case, use ":recover" or "vim -r 1.txt"
    to recover the changes (see ":help recovery").
    If you did this already, delete the swap file ".1.txt.swp"
    to avoid this message.
"1.txt" 786L, 20259C
Press ENTER or type command to continue
```

查看目录会多出来.swp ...等文件

```
总用量 364
drwxr-xr-x.  4 root root    144 12月 13 21:28 .
drwxr-xr-x. 15 root root    181 5月  10 2022 ..
-rw-r--r--.  1 root root    259 12月 13 19:48 123.tar.bz2
-rw-r--r--.  1 root root  20260 12月 13 21:27 1.txt
-rw-r--r--.  1 root root  16384 12月 13 21:21 .1.txt.swo
-rw-r--r--.  1 root root  16384 12月 13 21:20 .1.txt.swp
-rw-r--r--.  1 root root 306642 12月 13 18:17 2.txt
-rw-r--r--.  1 root root     74 12月 13 18:17 3.txt
drwxr-xr-x.  2 root root      6 12月 13 19:47 aaa
-rw-r--r--.  1 root root    316 12月 13 19:51 aaabbb.tar.bz2
drwxr-xr-x.  3 root root     17 12月 13 19:47 bbb
[root@bogon test]#
```

如果要恢复文件输入指令

```
vim -r 文件名称
```

如果不恢复文件则直接删除改.swp 等文件即可。

千锋教育Java教研院 关注公众号【Java架构栈】下载所有课程代码课件及工具 让技术回归本该有的纯静!