第六章 Linux 文件与目录管理

一、目录与路径

1.绝对路径：路径的写法一定由根目录/写起；

相对目录：路径写法不是由/写起。相对路径指相对于目前工作目录的路径。

2.目录的相关操作

在 Linux 下，根目录下没有上层目录存在。

符号/命令	含义
.	代表此目录
..	上级目录
-	前一个工作目录
~account	代表“account”用户的主文件夹

命令	含义	使用方式
cd	变换目录	cd [相对路径或者绝对路径]
pwd	显示当前目录	pwd [-P]（-P：显示出当前的路径，而非使用连接路径）
mkdir	新建一个目录	mkdir [-mp] 目录名称（-m：配置文件夹的权限；-p 递归建立层级目录）
rmdir	删除一个空目录	rmdir [-p] 目录名称（-p：连同上层“空”目录一起删除）

```
[root@study ~]# cd ~dmtsai
# 代表去到 dmtsai 这个用户的主目录，亦即 /home/dmtsai
[root@study dmtsai]# cd ~
# 表示回到自己的主目录，亦即是 /root 这个目录
[root@study ~]# cd
# 没有加上任何路径，也还是代表回到自己主目录的意思喔！
[root@study ~]# cd ..
# 表示去到目前的上层目录，亦即是 /root 的上层目录的意思；
[root@study /]# cd -
# 表示回到刚刚的那个目录，也就是 /root ～
```

```
范例：单纯显示出目前的工作目录：
[root@study ~]# pwd
/root     <== 显示出目录啦～
范例：显示出实际的工作目录，而非连接文件本身的目录名而已
[root@study ~]# cd /var/mail     <==注意，/var/mail 是一个连接文件
[root@study mail]# pwd
/var/mail              <==列出目前的工作目录
[root@study mail]# pwd -P
/var/spool/mail     <==怎么回事？有没有加 -P 差很多～
[root@study mail]# ls -ld /var/mail
```

```
lrwxrwxrwx. 1 root root 10 May  4 17:51 /var/mail -> spool/mail
# 看到这里应该知道为啥了吧？因为 /var/mail 是连接文件，连接到 /var/spool/mail。所以，加上 pwd -P 的选项后，会不以连结文件的数据显示，而是显示正确的完整路径！
```

```
[root@study tmp]# mkdir test      <==建立一名为 test 的新目录
[root@study tmp]# mkdir test1/test2/test3/test4
mkdir: cannot create directory ‘test1/test2/test3/test4’: No such file or directory
[root@study tmp]# mkdir -p test1/test2/test3/test4
#加了这个-p 的选项，可以自行帮你建立多层目录！
范例：建立权限为 rwx--x--x 的目录
[root@study tmp]# mkdir -m 711 test2
```

3.关于执行文件路径的变量：$PATH

执行命令 echo $PATH 看哪些目录被定义出来了。注意 echo 有“显示”的意思，PATH 前面的$表示后面接的是变量。

```
xiaoao@xiaoao-virtual-machine:~/桌面/work$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
xiaoao@xiaoao-virtual-machine:~/桌面/work$ su
密码:
root@xiaoao-virtual-machine:/home/xiaoao/桌面/work# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

图 1.1 $PATH 变量内容

PATH 这个变量的内容是由一堆目录所组成的，每个目录中间用冒号（:）来隔开，每个目录是有顺序之分的。

例题 1：:假设你是 root，如果你将 ls 由/bin/ls 移动成为/root/ls（可用「mv/bin/ls/root」指令达成），然后你自己本身也在/ root 目录下，请问（1）你能不能直接输入 ls 来执行？（2）若不能，你该如何执行 ls 这个指令？（3）若要直接输入 ls 即可执行，又该如何进行？

答案：（1）不能。因为**/root 这个目录并没有被添加到 PATH 中去**，所以即使你在/root 目录下也无法搜索到 ls 这个命令。

（2）使用绝对路径或者相对路径来执行这个命令：/root/ls（绝对路径）、./ls（相对路径）。

（3）将当前目录加入到 PATH 中去。执行 **PATH=”${PATH}:/root”**。

例题 2：为什么 ${PATH} 搜寻的目录不加入本目录(.)?

答案：如果在 PATH 中加入本目录(.)后，确实我们就能够在指令所在目录进行指令的执行了。但是由于你的工作目录并非固定（常常会使用 cd 来切换到不同的目录）， 因此能够执行的指令会有变动，这对用户来说并非好事。

另外，如果有用户在/tmp 底下做了一个指令，因为/tmp 是大家都能够写入的环境，所以他当然可以这样做。假设该指令可能会窃取用户的一些资料，如果你使用 root 的身份来执行这个指令，那不是很糟糕？如果这个指令的名称又是经常会被用到的 ls 时，那『中标』的机率就更高了！

所以，**为了安全起见，不建议将『.』加入 PATH 的搜寻目录中**。

由此我们可以得到以下结论：

（1）不同身份用户预设的 PATH 不同，预设能够随意执行的指令也不同；

（2）相同的指令，${PATH}中哪个目录首先被查询，就执行哪个目录下的指令；

（3）PATH 是可以修改的；

（4）使用绝对路径或相对路径直接指定某个指令的文件名来执行，会比搜寻 PATH 来的正确；

（5）指令应该要放置到正确的目录下，执行才会比较方便；

（6）本目录(.)最好不要放到 PATH 当中。

二、文件与目录管理

1.查看文件与目录：ls

默认仅显示：**非隐藏文件的文件名、以文件名进行排序**及**文件名代表的颜色显示**。

```
[root@study ~]# ls [-aAdfFhilnrRSt] 文件名或目录名称
[root@study ~]# ls [--color={never,auto,always}] 文件名或目录名称
[root@study ~]# ls [--full-time] 文件名或目录名称
选项与参数：
-a ：全部的文件，连同隐藏档( 开头为 . 的文件) 一起列出来(常用)
-A ：全部的文件，连同隐藏档，但不包括 . 与 .. 这两个目录
-d ：仅列出目录本身，而不是列出目录内的文件数据(常用)
-f ：直接列出结果，而不进行排序 (ls 预设会以文件名排序！)
-F ：根据文件、目录等信息，给予附加数据结构，例如：
      *:代表可执行文件； /:代表目录； =:代表 socket 文件； |:代表 FIFO 文件；
-h ：将文件容量以人类较易读的方式(例如 GB, KB 等等)列出来；
-i ：列出 inode 号码，inode 的意义下一章将会介绍；
-l ：长数据串行出，包含文件的属性与权限等等数据；(常用)
-n ：列出 UID 与 GID 而非用户与用户组的名称 (UID 与 GID 会在账号管理提到！)
-r ：将排序结果反向输出，例如：原本文件名由小到大，反向则为由大到小；
-R ：连同子目录内容一起列出来，等于该目录下的所有文件都会显示出来；
-S ：以文件容量大小排序，而不是用文件名排序；
-t ：依时间排序，而不是用文件名。
--color=never ：不要依据文件特性给予颜色显示；
--color=always ：显示颜色
--color=auto ：让系统自行依据设定来判断是否给予颜色
--full-time ：以完整时间模式 (包含年、月、日、时、分) 输出
--time={atime,ctime} ：输出 access 时间或改变权限属性时间 (ctime)
                        而非内容变更时间 (modification time)
```

2.复制、删除与移动：cp、rm、mv0

（1）cp：复制、建立快捷方式、对比文件的新旧而予以更新，复制整个目录。

```
[root@study ~]# cp [-adfilprsu] 来源文件(source) 目标文件(destination)
[root@study ~]# cp [options] source1 source2 source3 .... directory
选项与参数：
-a ：相当于 -dr --preserve=all 的意思，至于 dr 请参考下列说明；(常用)
-d ：若来源文件为连接文件的属性(快捷方式)，则复制连接文件属性而非文件本身；
-f ：为强制(force)的意思，若目标文件已经存在且无法开启，则移除后再尝试一次；
-i ：若目标文件(destination)已经存在时，在覆盖时会先询问动作的进行(常用)
-l ：进行硬连接(hard link)的连结文件建立，而非复制文件本身；
-p ：连同文件的属性(权限、用户、时间)一起复制过去，而非使用默认属性(备份常用)；
-r ：递归持续复制，用于目录的复制行为；(常用)
-s ：复制成为符号连接文件 (symbolic link)，也就是快捷方式；
```

```
-u  ：destination 比 source 旧才更新 destination，或 destination 不存在的情况下才复制。
--preserve=all：除了-p 权限相关参数，还加入 SELinux 的属性, links, xattr 等也复制了。
```

注意：

①如果来源档有两个以上，则最后一个目的文件一定要是**目录**才行！

②不同身份者执行这个指令会有不同的结果，特别是-a/-p 选项。

③在预设条件中，cp 的来源文件和目的文件的权限是不同的，目的文件的拥有者通常会是指令操作者本身。

④如果没有加上任何选项，cp 复制的是源文件，而非连接文件属性。

（2）mv：移动目录与文件，更名。

```
[root@study ~]# mv [-fiu] source destination
[root@study ~]# mv [options] source1 source2 source3 .... directory
选项与参数：
-f  ：force 强制的意思，如果目标文件已经存在，不会询问而直接覆盖；
-i  ：若目标文件 (destination) 已经存在时，就会询问是否覆盖！
-u  ：若目标文件已经存在，且 source 比较新，才会更新 (update)
```

```
范例一：复制一文件，建立一目录，将文件移动到目录中
[root@study tmp]# mkdir mvtest
[root@study tmp]# mv bashrc mvtest
# 将某个文件移动到某个目录去，就是这样做！
范例二：将刚刚的目录名称更名为 mvtest2
[root@study tmp]# mv mvtest mvtest2 <== 这样就更名了！简单～
# 其实在 Linux 底下还有个有趣的指令，名称为 rename ，该指令专职进行多个文件名的同时更名，并非针对单一文件名变更，与 mv 不同。请 man rename。
范例三：再建立两个文件，再全部移动到 /tmp/mvtest2 当中
[root@study tmp]# cp ~/.bashrc bashrc1
[root@study tmp]# cp ~/.bashrc bashrc2
[root@study tmp]# mv bashrc1 bashrc2 mvtest2
# 注意到这边，如果有多个来源文件或目录，则最后一个目标文件一定是『目录！』
# 意思是说，将所有的数据移动到该目录的意思！
```

（3）rm：移除文件

```
[root@study ~]# rm [-fir]  文件或目录
选项与参数：
-f：就是 force 的意思，忽略不存在的文件，不会出现警告讯息；
-i：互动模式，在删除前会询问用户是否动作
-r：递归删除啊！最常用在目录的删除了！这是非常危险的选项！！！
```

注意：如果我们要删除以“-”开头的文件，由于“-”会被认为是选项的前缀，所以删除的时候可以使用前面添加当前目录的方式进行。例如要删除“-aaa”文件，我们可以输入“rm ./-aaa”。

3.取得路径的文件名与目录名称

```
[root@study ~]# basename /etc/sysconfig/network
network                 <== 很简单！就取得最后的文件名～
[root@study ~]# dirname /etc/sysconfig/network
```

```
/etc/sysconfig   <== 取得的变成目录名了！
```

三、文件内容查阅

常用命令及其作用如下表所示。

命令	作用	命令	作用
cat	由第一行开始显示文件内容	more	一页一页地显示文件内容
tac	从最后一行开始显示	head	只看头几行
nl	显示的时候，顺便输出行号	tail	只看结尾几行
less	与 more 类似，但是它可以往前翻页	od	以二进制方式读取文件内容

1.cat（con**cat**enate，串联）

```
[root@study ~]# cat [-AbEnTv]
选项与参数：
-A  ：相当于 -vET 的整合选项，可列出一些特殊字符而不是空白而已；
-b  ：列出行号，仅针对非空白行做行号显示，空白行不标行号！
-E  ：将结尾的断行字符 $ 显示出来；
-n  ：打印出行号，连同空白行也会有行号，与 -b 的选项不同；
-T  ：将 [tab] 按键以 ^I 显示出来；
-v  ：列出一些看不出来的特殊字符
```

2.tac（反向列显）

由最后一行到第一行反向显示在屏幕上。

3.nl（添加行号打印）

```
[root@study ~]# nl [-bnw] 文件
选项与参数：
-b  ：指定行号指定的方式，主要有两种：
      -b a ：表示不论是否为空行，也同样列出行号(类似 cat -n)；
      -b t ：如果有空行，空的那一行不要列出行号(默认值)；
-n  ：列出行号表示的方法，主要有三种：
      -n ln ：行号在屏幕的最左方显示；
      -n rn ：行号在自己字段的最右方显示，且不加 0 ；
      -n rz ：行号在自己字段的最右方显示，且加 0 ；
-w  ：行号字段的占用的字符数。
```

上面提到的命令都是一次性将数据显示在屏幕上面。

4.more（一页一页翻动）

使用格式：

more [文件名]

more 运行过程中的按键及作用：

命令	作用	命令	作用
空格键	向下翻**一页**	:f	立刻显示出文件名及**目前显示**的行数
Enter	向下翻**一行**	q	代表立刻离开 more，不再显示该文件内容
/字符串	在**显示的内容**中，向下搜寻“字符串”关键词	b 或 ctrl+b	代表往回翻页，只对文件有用，对管道无用。

5.less（一页一页翻动）

使用格式：

less [文件名]

命令	作用	命令	作用
空格键	向下翻一页	n	重复前一个搜寻
[pagedown]	向下翻一页	N	反向的重复前一个搜寻
[pageup]	向上翻一页	g	前进到这个数据的第一行去
/字符串	向下搜寻“字符串”	G	前进道这个数据的最后一行去
?字符串	向上搜寻“字符串”	q	退出 less

注意：**man 指令就是调用 less 来显示说明文件的内容**。

我们可以将输出的数据做一个最简单的选取，那就是取出前面（head）几行或者后面（tail）几行文字的功能。注意：**head 和 tail 都是以行为单位进行数据选取的**。

6.head（取出前面几行）

```
[root@study ~]# head [-n number]  文件名
选项与参数：
-n   ：后面接数字，代表显示几行的意思。预设为 10 行。
```

若后面接的数字为负数，则代表列出前面所有行数，但不包括后面的行数。例如-100，就代表不列出最后 100 行。

7.tail（取出后面几行）

```
[root@study ~]# tail [-n number]  文件名
选项与参数：
-n   ：后面接数字，代表显示几行的意思。默认显示最后 10 行。
-f   ：表示持续侦测后面所接的文件名，要等到按下[ctrl]-c 才会结束 tail 的检测
```

若只想列出前面**第 m 行以后的数据**，那么可以使用“**tail -n +m [文件名]**”。

由于有的文件随时会有数据写入，如果想要该文件有数据写入时立刻显示到屏幕上，就要使用“-f”这个选项，它可以一直检测这个文件，新加入的数据就会被显示到屏幕上，直到你按下 ctrl+c 才会停止。

例题 1：假如我想要显示 /etc/man_db.conf 的第 11 到第 20 行，该怎么办呢？
答：在第 11 到第 20 行，那么我取前 20 行，再取后 10 行，所以结果就是：『head -n 20 /etc/man_db.conf | tail -n 10』，这样就可以得到第 11 到第 20 行之间的内容了。
注意：这两个指令中间有个管道 (|) 的符号存在，这个管线的意思是：**前面的指令所输出的信息，请通过管道交由后续的指令继续使用**的意思。 所以，head -n 20 /etc/man_db.conf 会将文件内的前 20 行取出来，但不输出到屏幕上，而是转交给后续的 tail 指令继续处理。因此 tail 不需要接文件名。

例题 2：承上一题，那如果我想要列出正确的行号呢？
答：还是利用管道，即写为“cat -n /etc/man_db.conf | head -n 20 | tail -n 10”

6.非纯文本文件：od（二进制）

```
[root@study ~]# od [-t TYPE]  文件名
选项或参数：
-t   ：后面可以接各种『类型 (TYPE)』的输出，例如：
       a          ：利用默认的字符来输出；
       c          ：使用 ASCII 字符来输出
       d[size]  ：利用十进制(decimal)来输出数据，每个整数占用 size bytes ；
       f[size]  ：利用浮点数(floating)来输出数据，每个数占用 size bytes ；
       o[size]  ：利用八进制(octal)来输出数据，每个整数占用 size bytes ；
```

x[size] ：利用十六进制(hexadecimal)来输出数据，每个整数占用 size bytes ；

例题 1：如果我想立即找到 xidian 这几个字的 ASCII 码，该怎样透过 od 来判断呢？
答：使用管道（|）。echo 可以在屏幕上显示任何信息，而这个信息不由屏幕显示，而是交给 od 去继续处理，这样就可以得到对应的 ASCII 码了。如下图所示。

```
root@xiaoao-virtual-machine:/tmp# echo xidian | od -t oCc
0000000 170 151 144 151 141 156 012
          x   i   d   i   a   n  \n
0000007
```

图 3.1

7.修改文件时间或创建新文件：touch

首先来介绍一下文件时间。文件时间主要有三种，它们分别是：

（1）modification time(mtime)：即**修改时间**。当文件的**内容数据**发生改变的时候，就会更新这个时间。

（2）status time(ctime)：即文件的**状态**改变的时候，就会更新这个时间。例如修改文件的权限或者属性。

（3）access time(atime)：即**读取时间**。当文件的内容被读取的时候，就会更新这个读取时间。

在默认的情况下，**ls 显示出来的是该文件的 mtime，也就是这个文件的内容上次被更改的时间**。我们可以看下图：

```
root@xiaoao-virtual-machine:/tmp# date;ls -l /etc/issue;ls -l --time=atime /etc
/issue
2018年 12月 03日 星期一 17:37:50 CST 当前时间
-rw-r--r-- 1 root root 24 4月  24  2018 /etc/issue 文件建立时间（mtime）
-rw-r--r-- 1 root root 24 12月  3 14:54 /etc/issue 读取内容时间（atime）
root@xiaoao-virtual-machine:/tmp# ls -l --time=ctime /etc/issue
-rw-r--r-- 1 root root 24 9月   5 22:02 /etc/issue 更新状态时间(ctime)
root@xiaoao-virtual-machine:/tmp#
```

图 3.2 时间测试

文件的时间是很重要的，因为，如果文件的时间误判的话，可能会造成某些程序无法顺利的运行。但是，如果发现了一个文件来自未来，该如何让该文件的时间变成现在的时间呢？答案是**使用 touch 命令**。

```
[root@study ~]# touch [-acdmt]  文件名
选项与参数：
-a  ：仅修订  access time；
-c  ：仅修改文件的时间，若该文件不存在则不建立新文件；
-d  ：后面可以接欲修改的日期而不用目前的日期，也可以使用  --date="日期或时间"
-m  ：仅修改文件修改时间（mtime）
-t  ：后面可以接欲修改的时间而不用目前的时间，格式为[YYYYMMDDhhmm]
```

注意：默认情况下，如果 touch 后面有接文件，则该文件的三个时间（atime/ctime/mtime）都会被修改为**目前时间**；若该文件不存在，则会主动**新建一个空的文件**。

如图 3.3 所示。前 4 行为延时的创建一个空文件的过程。第 5 行~第 10 行演示了修改 bashrc 文件，将日期修改为两天前的方法和结果；第 11~16 行延时了将 bashrc 的日期修改为 2018 年 12 月 2 日 11 点 19 分。

我们发现，通过 touch 这个指令，我们可以轻易修改文件的日期和时间，并且可以新建一个空的文件。不过要注意的是，即使我们复制一个文件，并且复制所有的属性的时候，我们也无法复制 ctime 这个属性。

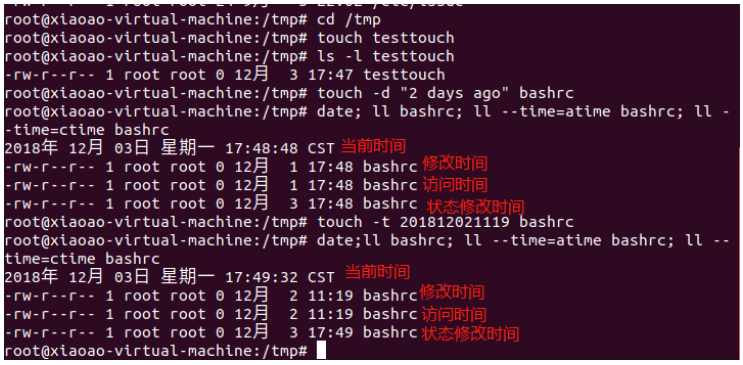

图 3.3 修改时间延时

所以一定要记住，touch 这个指令最常用的情况是：

（1）建立一个空文件；

（2）将某个文件的日期修改为目前（mtime 和 atime）。

四、文件与目录的默认权限和隐藏权限

1.文件默认权限：umask

当建立一个新的文件或目录的时候，它的默认权限跟 umask 有关，umask 就是指定“目前用户在监理文件或目录时候的权限默认值”，那么如何得知或者设定 umask 呢？

```
[root@study ~]# umask
0022              <==与一般权限有关的是后面三个数字
[root@study ~]# umask -S
u=rwx,g=rx,o=rx
```

查询的方式有两种，如上所示。一是直接输入 umask，然后显示出数字形态的权限设定。一种是加入-S，就会以符号类型的方式显示出权限。

预设情况如下：

（1）若用户建立为“**文件**”，则预设**没有可执行权限**，亦只有 rw 两个权限，所以最大为 666，即-rw-rw-rw-。

（2）若用户建立为“**目录**”，由于 x 与是否可以进入该目录有关，所以**所有权限均开放**，即为 777，预设权限如下：drwxrwxrwx。

要注意的是，umask 指的是**该默认值需要减掉的权限**。我们举例 umask 值为 022，那么用户建立文件和目录时权限如下：

	预设权限	实际权限
建立文件	-rw-rw-rw-(666)	-rw-r--r--
建立目录	drwxrwxrwx(777)	drwxr-xr-x

如果想要修改 umask 的值，只需要执行下面的语句即可：

umask xxx

例如我想要给新建的文件同用户组用户都可以编辑的时候，那么我们就不能拿掉 group 里面的 w 权限，所以我们需要设计 umask 为 002（包括但不限于这个，只需要拿掉 group 中的 w 即可）。

umask 002

2.文件隐藏属性 chattr, lsattr

（1）chattr

注意：**chattr 指令只能在 Ext2/Ext3/Ext4 的 Linux 传统文件系统上面完整生效**，其他的文件系统可能就无法完整的支持这个指令。

[root@study ~]# chattr [+-=][ASacdistu] 文件或目录名称

选项与参数：
+：增加某一个特殊参数，其他原本存在参数则不动。
-：移除某一个特殊参数，其他原本存在参数则不动。
=：设定一定，且仅有后面接的参数。
A：当设定了 A 这个属性时，若你有存取此文件(或目录)时，他的访问时间 atime 将不会被修改，可避免 I/O 较慢的机器过度的存取磁盘。(目前建议使用文件系统挂载参数处理这个项目)
S：一般文件是异步写入磁盘的(原理请参考前一章 sync 的说明)，如果加上 S 这个属性时，当你进行任何文件的修改，该更动会『同步』写入磁盘中。
a：当设定 a 之后，这个文件将只能增加数据，而不能删除也不能修改数据，只**有 root 才能设定这属性**。
c：这个属性设定之后，将会自动的将此文件『压缩』，在读取的时候将会自动解压缩，但是在储存的时候，将会先进行压缩后再储存(看来对于大文件似乎蛮有用的！)
d：当 dump 程序被执行的时候，设定 d 属性将可使该文件(或目录)不会被 dump 备份。
i：这个 i 可就很厉害了！他可以让一个文件『不能被删除、改名、设置连接也无法写入或新增数据！』对于系统安全性有相当大的帮助！**只有 root 能设定此属性。**
s：当文件设定了 s 属性时，如果这个文件被删除，他将会被完全的移除出这个硬盘空间，所以如果误删了，完全无法救回来了喔！
u：与 s 相反的，当使用 u 来配置文件案时，如果该文件被删除了，则数据内容其实还存在磁盘中，可以使用来救援该文件喔！

注意：

（1）属性设定常见的是 a 与 i 的设定值，而且很多设定值必须要身为 root 才能设定。

（2）xfs 文件系统仅支援 AadiS 而已。

这个指令非常重要，特别是在系统的数据安全上面。由于这些属性都是隐藏的属性，所以需要使用下面的 lsattr 才能看到这个属性。

另外，如果是 log file 这种登录文件，就更需要+a 这个属性，它可以增加，但不能修改或者删除旧的数据，非常适合 log file。

（2）lsattr

[root@study ~]# lsattr [-adR] 文件或目录
选项与参数：
-a ：将隐藏文件的属性也展示出来；
-d ：如果接的是目录，仅列出目录本身的属性而非目录内的文件名；
-R ：连同子目录的数据也一并列出来！

如图 4.1 所示，第一行新建了一个名为 attrtest 的文件，第二行展示其隐藏属性（使用 lsattr），看到其并没有设置 chattr 能够设置的属性，然后给其设置一个 i 属性（无法新增、修改、删除），然后执行一下删除操作，发现就连 root 用户都无法删除，这个时候使用 lsattr 查看下其隐藏属性，发现确实为 i。最后删除其 i 属性，然后再次利用 lsattr 查看其属性，发现已经没有了 i 属性，并且可以被成功删除。

使用这两个指令要特别小心，否则可能会出现很大的困扰。

```
xiaoao@xiaoao-virtual-machine:/tmp$ touch attrtest
xiaoao@xiaoao-virtual-machine:/tmp$ lsattr attrtest
-------------e--- attrtest
xiaoao@xiaoao-virtual-machine:/tmp$ chattr +i attrtest
chattr: 不允许的操作 设置 attrtest 的标志时
xiaoao@xiaoao-virtual-machine:/tmp$ su
密码:
root@xiaoao-virtual-machine:/tmp# chattr +i attrtest
root@xiaoao-virtual-machine:/tmp# rm attrtest
rm: 无法删除'attrtest': 不允许的操作
root@xiaoao-virtual-machine:/tmp# lsattr attrtest
----i--------e--- attrtest
root@xiaoao-virtual-machine:/tmp# chattr -i attrtest
root@xiaoao-virtual-machine:/tmp# lsattr attrtest
-------------e--- attrtest
root@xiaoao-virtual-machine:/tmp# rm attrtest
root@xiaoao-virtual-machine:/tmp#
```

图 4.4 chattr 和 lsattr 演示

（3）文件特殊权限：SUID, SGID, SBIT

我们在 terminal 中执行 ls -l /会发现有两个文件夹会有下面的结果：

drwxrwxrw**t**. 14 root root 4096 Jun 16 01:27 /tmp

-rw**s**r-xr-x. 1 root root 27832 Jun 10　2014 /usr/bin/passwd

这个 s 和 t 就是 SUID/SGID 的东西。

Set UID 简称 SUID，它有这样的功能：

①SUID 权限仅对**二进制程序(binary program)**有效；

②用户对于该程序需要**具有 x 的可执行权限**；

③本权限仅在**执行该程序的过程**中有效 (run-time)；

④用户将具有该程序**拥有者**(owner)的权限。

Set GID 简称 SGID。与 SUID 不同，SGID 是针对**文件或目录**来设置的。SGID 有如下功能：

①SGID 对**二进制程序**有用；

②程序用户对于该程序来说，需**具备 x 的权限**；

③用户在执行的过程中将会获得该程序**用户组**的支持。

SGID 用在目录上的时候，它具有以下功能：

①用户若对于此目录具有 r 与 x 的权限时，该用户能够进入此目录；

②用户在此目录下的有效群组(effective group)将会变成该目录的群组；

③用途：若用户在此目录下具有 w 的权限(可以新建文件)，则用户所创建的新文件的用户组与此目录的用户组相同。

Sticky Bit，简称 SBIT，目前只针对**目录**有效。它对于目录的作用是：**当用户在该目录下建立文件或目录时，仅有自己与 root 才有权力删除该文件**。

（4）SUID/SGID/SBIT 权限设定

SUID 为 4，SGID 为 2，SBIT 为 1。设定的时候，**数字在 RWX 数字之前**。

特别要说一个例子，如下所示：

```
[root@study tmp]# chmod 7666 test; ls -l test <==空的 SUID/SGID 权限
-rwSrwSrwT 1 root root 0 Jun 16 02:53 test
```

原因：

因为 s 与 t 都是取代 x 这个权限的，但是我们下达的是 7666。也就是说，user/group/others 都没有 x 这个可执行的标志。所以，这个 S 和 T 代表的就是“空”。

除了数字表示法，也可以用符号法来处理。SUID 是 u+s，SGID 是 g+s，SBIT 是 o+t。

（5）查看文件类型：file

如果想知道某个文件的基本数据，例如它是 ASCII、data，还是 binary 文件，且其中有没有使用到动态函数库等信息，就可以利用 file 命令来查看。例如图 4.5 所示。

```
root@xiaoao-virtual-machine:/tmp# file ~/.bashrc
/root/.bashrc: ASCII text
root@xiaoao-virtual-machine:/tmp# file /usr/bin/passwd
/usr/bin/passwd: setuid ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2
.0, BuildID[sha1]=d44c96296f224071ed008e442b9eb3f2462840e4, stripped
root@xiaoao-virtual-machine:/tmp# file /var/lib/mlocate/mlocate.db
/var/lib/mlocate/mlocate.db: mlocate database, version 0, require visibility, r
oot /
root@xiaoao-virtual-machine:/tmp#
```

图 4.5 file 查看举例

五、命令与文件的查询

我们常常需要知道那个文件放在哪里，才能够对该文件进行一些修改或维护等动作。有些时候某些软件配置文件的文件名是不变的，但是各分发版本放置的目录则不同。此时我们就需要利用一些搜索命令将该配置文件的完整文件名找到，这样才能进行修改。

1.脚本文件的文件名查询

我们知道在终端模式当中，连续输入两次[tab]按键就能够知道用户有多少命令可以下达。但是我们不知道这些命令的完整文件名放在哪里。这个时候，我们就可以通过 which 或 type 来寻找。

（1）which（寻找“执行文件”）

[root@study ~]# which [-a] command
选项或参数：
-a ：将所有由 PATH 目录中可以找到的命令均列出，而不止第一个被找到的命令名称。

如图 5.1 所示的举例。

```
root@xiaoao-virtual-machine:/tmp# which ifconfig
/sbin/ifconfig
root@xiaoao-virtual-machine:/tmp# which which
/usr/bin/which
root@xiaoao-virtual-machine:/tmp# which -a which
/usr/bin/which
/bin/which
```

图 5.1 which 举例

注意，**这个命令是根据“PATH”这个环境变领所规范的路径去搜索“执行文件”的文件名**。

（2）type

type 将在后续来讲，在此就不说了。

2.文件名的查找

通常来说，find 命令不常用，因为速度很慢。一般常用的是 whereis 和 locate，前者是只**找系统中某些目录底下的文件**，而后者是**利用数据库来搜索文件名**。

（1）whereis

[root@study ~]# whereis [-bmsu] 文件或目录名
选项与参数：

参数	功能	参数	功能
-l	列出 whereis 会去查询的几个主要目录	-b	只找二进制格式的文件
-m	只找在说明文件 manual 路径下的文件	-s	只找 source 来源文件
-u	搜索不在上述项目当中的其他特殊文件		

如图 5.2 所示，第一个例子是找出 ifconfig 这个文件名，第二个例子是找出跟 passwd 的文件名，第三个例子是在 man page 中找到 passwd 的文件名。

```
root@xiaoao-virtual-machine:/tmp# whereis ifconfig
ifconfig: /sbin/ifconfig /usr/share/man/man8/ifconfig.8.gz
root@xiaoao-virtual-machine:/tmp# whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz /usr/share/
man/man1/passwd.1ssl.gz /usr/share/man/man5/passwd.5.gz
root@xiaoao-virtual-machine:/tmp# whereis -m passwd
passwd: /usr/share/man/man1/passwd.1.gz /usr/share/man/man1/passwd.1ssl.gz /usr
/share/man/man5/passwd.5.gz
root@xiaoao-virtual-machine:/tmp#
```

图 5.2 whereis 演示

whereis 只是查询/bin/sbin、/usr/share/man 下的 man page 文件等几个比较特定的目录。具体想知道 whereis 查了多少目录，可以使用 whereis -l 来确认。

（2）locate/updatedb

[root@study ~]# locate [-ir] keyword
选项与参数：
-i：忽略大小写的差异；
-c：不输出文件名，仅计算找到的文件数量
-l：仅输出几行的意思，例如输出五行则是 -l 5
-S：输出 locate 所使用的数据库文件的相关信息，包括该数据库记录的文件/目录数量等
-r：后面可接正规表示法的显示方式

我们要记住无论是 whereis 还是 locate 都是有局限性的。**whereis 只能在所能够检索的目录下检索文件，所以目录之外的文件搜索不到；而 locate 只能在建立的数据库中查找（数据库在/var/lib/mlocate），而这个数据库并不是实时更新，是定期更新，所以有可能刚建立的文件并不能搜索到。**

解决 locate 的方法就是使用 updatedb 命令来手动更新数据库。使用方法非常简单，直接输入“updatedb”，然后回车即可。然后 updatedb 就会去读取/etc/updated.conf 这个配置文件的设置，去搜索文件名，然后更新数据库。

如图 5.3 所示，是 locate 的演示。

```
xiaoao@xiaoao-virtual-machine:~$ locate -l 5 passwd
/etc/passwd
/etc/passwd-
/etc/cron.daily/passwd
/etc/pam.d/chpasswd
/etc/pam.d/passwd
xiaoao@xiaoao-virtual-machine:~$ locate -c passwd
208
xiaoao@xiaoao-virtual-machine:~$ locate -S
数据库 /var/lib/mlocate/mlocate.db:
        38,793 文件夹
        422,124 文件
        29,128,172 文件名中的字节数
        10,747,595 字节用于存储数据库
xiaoao@xiaoao-virtual-machine:~$
```

图 5.3 locate 演示

（3）find

[root@study ~]# find [PATH] [option] [action]
选项与参数：
1.与时间有关的选项：共有-atime, -ctime 与-mtime。我们以-mtime 说明：
-mtime n ： n 为数字，意义为在 n 天之前的“**一天之内**”被更动过内容的文件；
-mtime +n ： 列出在 n 天之前(**不含 n 天本身**)被更动过内容的文件文件名；
-mtime -n ： 列出在 n 天之内(**含 n 天本身**)被更动过内容的文件文件名。

-newer file ：file 为一个存在的文件，列出比 file 还要新的文件文件名

2. 与用户或组名有关的参数：

-uid n：n 为数字，这个数字是用户的账号 ID，亦即 UID，这个 UID 是记录在/etc/passwd 里面与账号名称对应的数字。

-gid n：n 为数字，这个数字是组名的 ID，亦即 GID，这个 GID 记录在/etc/group。

-user name：name 为用户账号名称。

-group name：name 为组名。

-nouser：寻找文件的拥有者不存在于/etc/passwd 的人。

-nogroup ：寻找文件的用户组不存在于/etc/group 的文件。

3. 与文件权限及名称有关的参数：

-name filename：搜寻文件名为 filename 的文件。

-size [+-]SIZE：搜寻比 SIZE 还要大(+)或小(-)的文件。（这个 SIZE 的规格有：c 代表 byte，k 代表 1024bytes。所以，要找比 50KB 还要大的文件，就是“-size +50k”。）

-type TYPE：搜寻文件的类型为 TYPE 的。（类型主要有：一般正规文件(f)，设备文件(b, c)，目录(d)，连接文件(l)，socket (s)，及 FIFO(p)等属性。）

-perm mode：搜寻文件权限**等于** mode 的文件，这个 mode 为类似 chmod 的属性值。

-perm -mode：搜寻文件权限**必须要全部包括 mode 的权限**的文件。例如，我们要搜索 0744 的文件，使用“-perm -0744”。但是当一个文件的权限为 4755 时，也会被列出来，因为 -rwsr-xr-x 的属性已经囊括了 -rwxr--r-- 的属性了。

-perm /mode：搜寻文件权限**包含任一 mode 的权限**的文件。例如，我们搜索-rwxr-xr-x，即-perm /755 时，但一个文件属性为-rw-------也会被列出来，因为他有 -rw....的属性。

4. 额外可进行的动作：

-exec com：com 为其他指令，-exec 后面可再接额外的指令来处理搜索到的结果。

-print：将结果打印到屏幕上，这个操作是预设操作。

find 的特殊功能就是能够进行额外的动作。我们将下面的例子以图解来说明。

```
find /usr/bin /usr/sbin -perm /7000 -exec ls -l {} \;
```

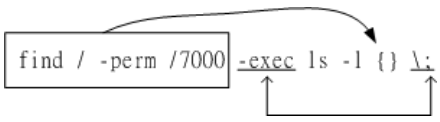

图 5.4 解释说明

该例子中特殊的地方有{}以及\;还有-exec 这个关键词，这些东西的意义为：

①{}代表的是**由 find 找到的内容**，如上图所示，find 的结果会被放置到{}位置中；

②-exec 一直到\;是关键词，代表 find 额外动作的开始(-exec)到结束(\;)，在这中间的就是 find 指令内的额外动作。在本例中就是 **ls-l{}**。

③因为;在 bash 环境下是有特殊意义的，因此利用反斜杠来转义。

六、权限与命令间的关系（非常重要）

1.让用户能进入某目录成为“**可工作目录**”的基本权限为：

（1）可使用的命令：例如 **cd** 等变换工作目录的命令；

（2）目录所需权限：**用户对这个目录至少需要具有 x 的权限**；

（3）额外需求：如果用户想要在这个目录内利用 ls 查阅文件名，则用户对此目录还需要 **r** 的权限。

2.用户在某个目录内读取一个文件的基本权限是：

（1）可使用的命令：例如本章谈到的 cat, more, less 等等

（2）目录所需权限：用户对这个目录至少需要具有 **x** 权限；

（3）文件所需权限：**用户对文件至少需要具有 r 的权限才行**。

3.让用户可以修改一个文件的基本权限为何？

（1）可使用的命令：例如 nano 或未来要介绍的 vi 编辑器等；

（2）目录所需权限：用户在该文件所在的目录至少要有 x 权限；

（3）文件所需权限：**用户对该文件至少要有 r,w 权限**。

4.让一个用户可以建立一个文件的基本权限为何？

目录所需权限：**用户在该目录要具有 w，x 的权限，重点在 w。**

5.让用户进入某目录并执行该目录下的某个命令之基本权限为何？

（1）目录所需权限：用户在该目录至少要有 x 的权限；

（2）文件所需权限：用户在该文件至少需要有 x 的权限。