第5章 Linux文件和目录管理（下）

阿铭linux
2023年10月20日 10:33

已关注

9
0
0
0
1

5.6 更改文件的权限

上面讲了那么多文件的属性，你虽然不能一下子明白每列信息所表示的具体含义，但随着后续章节的逐步深入，阿铭相信你一定能理解和掌握它们。

5.6.1 命令chgrp

chgrp（change group的简写）命令可以更改文件的所属组，其格式为：chgrp [组名] [文件名]，示例命令如下：

```
# groupadd testgroup
# mkdir /tmp/4_6 //创建实验用的目录
# cd /tmp/4_6
# touch test1
# ls -l test1
-rw-r--r-- 1 root 0 12月 30
07:43 test1
# chgrp testgroup test1
# ls -l test1
-rw-r--r-- 1 root testgroup 0 12月 30 07:43 test1
```

上例中用到了groupadd命令，其含义为增加一个用户组。

chgrp命令还可以更改目录的所属组，示例命令如下：

```
# mkdir dir2
# touch dir2/test2
# ls –ld dir2
drwxr-xr-x 2 root 19 12月 30
07:44 dir2
# chgrp testgroup dir2
# ls -ld dir2
drwxr-xr-x 2 root testgroup 19 12月 30 07:44 dir2
# ls –l dir2
总用量 0
-rw-r--r-- 1 root 0 12月 30
07:44 test2
```

上例中，chgrp命令只更改了目录本身，而目录下的文件并没有更改。如果要想级联更改子目录以及子文件，加-R选项可以实现，示例命令如下：

```
# chgrp -R testgroup dir2
# ls -l dir2
总用量 0
```

回到
旧版

顶部

chgrp命令的语法和用户组类似有一个命令可以替代它，那就是chown。

5.6.2 命令chown

chown（change owner的简写）命令可以更改文件的所有者，其格式为：chown [-R] 账户名 文件名或者chown [-R] 账户名:组名 文件名。这里的-R选项只适用于目录，作用是级联更改，即不仅更改当前目录，连目录里的目录或者文件也全部更改。示例命令如下：

```
# mkdir dir3
# useradd user1 // 创建用户user1，useradd命令会在5.2.3节中介绍
# touch dir3/test3 // 在dir3目录下创建test3文件
# chown user1 dir3
# ls -ld dir3 // dir3目录所有者已经由root改为user1
drwxr-xr-x 2 user1 root 19 12月 30
07:46 dir3
# ls -l dir3 // 但是dir3目录下的test3文件的所有者依旧是root
总用量 0
-rw-r--r-- 1 root 0 12月 30
07:46 test3
# chown -R user1:testgroup dir3
# ls -l dir3
总用量 0
-rw-r--r-- 1 user1 testgroup 0 12月 30 07:46 test3
```

上例中，chown -R user1:testgroup会把test目录以及该目录下的文件都修改成所有者为user1，所属组为testgroup。

5.6.3 命令chmod

为了方便更改文件的权限，Linux使用数字代替rwx，具体规则为：r等于4，w等于2，x等于1，-等于0。例如，rwxrwx---用数字表示就是770，其具体算法为：rwx=4+2+1=7，rwx=4+2+1=7，---=0+0+0=0。

命令chmod（change mode的简写）用于改变用户对文件/目录的读写执行权限，其格式为：chmod [-R] xyz 文件名（这里的xyz表示数字）。其中，-R选项的作用等同于chown命令的-R选项，也表示级联更改。值得注意的是，在Linux系统中，一个目录的默认权限为755，而一个文件的默认权限为644。下面我们举例说明一下：

```
# ls -ld dir3
drwxr-xr-x 2 user1 testgroup 19 12月 30 07:46 dir3
# ls -l dir3
总用量 0
-rw-r--r-- 1 user1 testgroup 0 12月 30 07:46 test3
# chmod 750 dir3
# ls -ld dir3
drwxr-x--- 2 user1 testgroup 19 12月 30 07:46 dir3
# ls -l dir3/test3
-rw-r--r-- 1 user1 testgroup 0 12月 30 07:46 dir3/test3
```

9 0 0 0 1
回到旧版
顶部

-rwx------ 1 user1 testgroup 0 12月 30 07:46 dir3/test3

chmod -R 700 dir3

ls -ld dir3

drwx------ 2 user1 testgroup 19 12月 30 07:46 dir3

ls -l dir3

总用量 0

-rwx------ 1 user1 testgroup 0 12月 30 07:46 test3

如果你创建了一个目录，但又不想让其他人看到该目录的内容，则只需设置成rwxr-----（740）即可。

chmod还支持使用rwx的方式来设置权限。从之前的介绍中可以发现，基本上就9个属性。我们可以使用u、g和o来分别表示user、group和others的属性，用a代表all（即全部）。下面阿铭举例来介绍它们的用法，示例命令如下：

chmod u=rwx,og=rx dir3/test3

ls -l dir3/test3

-rwxr-xr-x 1 user1 testgroup 0 12月 30 07:46 dir3/test3

这样可以把dir3/test3的文件权限修改为rwxr-xr-x。此外，我们还可以针对u、g、o和a，增加或者减少它们的某个权限（读、写或执行），示例命令如下：

 # chmod u-x dir3/test3

 # ls -l dir3

总用量 0

-rw-r-xr-x 1 user1 testgroup 0 12月 30 07:46 test3

 # chmod a-x dir3/test3

 # ls -l dir3/test3

-rw-r--r-- 1 user1 testgroup 0 12月 30 07:46 dir3/test3

 # chmod u+x dir3/test3

 # ls -l dir3/test3

-rwxr--r-- 1 user1 testgroup 0 12月 30 07:46 dir3/test3

5.6.4 命令umask

默认情况下，目录的权限值为755，普通文件的权限值为644，那么这个值是由谁规定的呢？究其原因，便涉及umask了。

命令umask用于改变文件的默认权限，其格式为：umask xxx（这里的xxx代表3个数字）。如果要查看umask的值，只要在命令行输入umask，然后回车即可，如下所示：

umask

0022

这里umask的预设值是0022，这表示什么含义呢？咱们先来看以下两条规则。

- 若用户建立普通文件，则预设没有可执行权限，只有r、w两个权限，最大值为666（-rw-rw-rw-）。
- 若用户建立目录，则预设所有权限均开放，即777（drwxrwxrwx）。

回到旧版

顶部

目录的权限为rwxrwxrwx - ----w--- = rwxr-xr-x

普通文件的权限为rw-rw-rw- -----w--- = rw-r--r--

umask的值是可以自定义的，比如设定umask为002，你再创建目录或者文件时，默认权限分别为：

rwxrwxrwx - -------w- = rwxrwxr-x（目录的权限）

rw-rw-rw- - -------w- = rw-rw-r--（文件的权限）

示例命令如下：

```
# umask 002
# mkdir dir4
# ls -ld dir4
drwxrwxr-x 2 root 6 12月 30
07:53 dir4
# touch test4
# ls -l test4
-rw-rw-r-- 1 root 0 12月 30
07:54 test4
```

这里我们可以看到创建的目录的默认权限变为775，而文件的默认权限变为664。如果要把umask改回来，具体操作方法如下：

```
# umask 022
# touch test5
# ls -l test5
-rw-r--r-- 1 root 0 12月 30
07:54 test5
```

关于umask的计算方法，有的朋友喜欢换算成数字去做减法，比如rwxrwxrwx –----w--w-= 777–022 = 755。乍一看这好像没有任何问题，但有时会出错，比如当umask值为033时，如果使用单纯的减法，文件的默认权限则为666–033=633，但实际权限应该为rw-rw-rw-–----wx-wx = rw-r--r--=644。

umask可以在/etc/bashrc里面更改，默认情况下，root的umask为022，而一般使用者则为002。可写的权限非常重要，因此预设会去掉写权限。可能大家一直有一个疑问，阿铭介绍的umask值一直都是3位数，但为什么系统里面是4位呢？最前面还有一个0呢，这个0加与不加没有影响，它表示umask数值是八进制的。

5.6.5 修改文件的特殊属性

1.命令chattr

命令chattr（change attribute）的格式为：chattr [+-=][Asaci] [文件或者目录名]，其中，+、-和=分别表示增加、减少和设定。各个选项的含义如下。

- A：增加该属性后，表示文件或目录的atime将不可修改。
- s：增加该属性后，会将数据同步写入磁盘中。
- a：增加该属性后，表示只能追加不能删除，非root用户不能设定该属性。
- c：增加该属性后，表示自动压缩该文件，读取时会自动解压。

9 0 0 0 1
回到旧版
顶部

以上选项中，常用的为a和i这两个选项。下面阿铭举例说明其用法，示例命令如下：

```
# chattr +i dir2
# touch dir2/test5
touch: 无法创建"dir2/test5":权限不够
# chattr -i dir2
# touch dir2/test5
# chattr +i dir2
# rm -f dir2/test5
rm: 无法删除"dir2/test5":权限不够
```

上例中，给dir2目录增加i权限后，即使是root账户，也不能在dir2目录中创建或删除test5文件。

下面再来看看a权限的作用，示例命令如下：

```
# chattr -i dir2
# touch dir2/test6
# ls dir2
test2 test5 test6
# chattr +a dir2
# rm -f dir2/test6
rm: 无法删除"dir2/test6":不允许的操作
# touch dir2/test7
# ls dir2
test2 test5 test6 test7
```

上例中，dir2目录增加a权限后，只可以在里面创建文件，而不能删除文件。

文件同样适用以上权限，示例命令如下：

```
# chattr +a dir2/test7
# echo '11111' > dir2/test7
-bash: dir2/test7: 不允许的操作
# echo '11111' >> dir2/test7
# cat dir2/test7
11111
# chattr +i dir2/test6
# echo '11111' >> dir2/test6
-bash: test2/test3: 权限不够
# echo '11111' > dir2/test6
-bash: dir2/test6: 权限不够
# rm -f dir2/test6
rm: 无法删除"dir2/test6":权限不够
```

为：lsattr [-aR] [文件/目录名]。下面先来看看-a和-R这两个选项的含义。

- -a：类似于ls的-a选项，即连同隐藏文件一同列出。
- -R：连同子目录的数据一同列出。

这个命令的用法和ls类似，示例命令如下：

```
# lsattr dir2
---------------- dir2/test2
---------------- dir2/test5
----i----------- dir2/test6
-----a---------- dir2/test7
# lsattr -aR dir2
-----a---------- dir2/.
---------------- dir2/..
---------------- dir2/test2
---------------- dir2/test5
----i----------- dir2/test6
-----a---------- dir2/test7
```

3. set uid、set gid和sticky bit

前面介绍权限的时候，我们一直都是用3位数，其实最前面还有一位，那就是下面要讲的set uid、set gid和sticky bit。

- set uid：该权限针对二进制可执行文件，使文件在执行阶段具有文件所有者的权限。比如，passwd这个命令就具有该权限。当普通用户执行passwd命令时，可以临时获得root权限，从而可以更改密码。
- set gid：该权限可以作用在文件上（二进制可执行文件），也可以作用在目录上。当作用在文件上时，其功能和set uid一样，它会使文件在执行阶段具有文件所属组的权限。目录被设置这个权限后，任何用户在此目录下创建的文件都具有和该目录所属的组相同的组。
- sticky bit：可以理解为防删除位。文件是否可以被某用户删除，主要取决于该文件所在的目录是否对该用户具有写权限。如果没有写权限，则这个目录下的所有文件都不能删除，同时也不能添加新的文件。如果希望用户能够添加文件但不能删除该目录下其他用户的文件，则可以对父目录增加该权限。设置该权限后，就算用户对目录具有写权限，也不能删除其他用户的文件。

例如，passwd命令设置了set uid权限，而/tmp/目录则设置了sticky bit权限。下面我们来看看它们的权限，示例命令如下：

```
# ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root 27832 5月 11 2019 /usr/bin/passwd
# ls -ld /tmp/
drwxrwxrwt. 21 root 4096 12月 30
07:43 /tmp/
```

可以发现，passwd显示的是rws而非传统的rwx，用数字表示为4755。/tmp/显示的rwt而非rwx，用数字表示为1777。那么，这个4和1是如何计算出来的呢？当有特殊权限时，第一位数字可以是0、1（--t）、2（-s-）、3（-st）、4（s--）、5（s-t）、6（ss-）或7（sst）。再回过头来看passwd，它是s--，所以是4；而/tmp/是--t，所以是1。

9 0 0 0 1
回到旧版
顶部

filename。同理，想设置set gid权限的命令为chmod g+s dirname，设置sticky bit权限的命令为chmod o+t dirname。

有时候，你可能会发现set_uid上的权限为大写的S，而不是小写的s，比如rwS，这是因为该文件没有x权限所致，不管是大写的S还是小写的s，都表示它存在set_uid或者set_gid权限，同理sticky bit也一样。

5.7 在Linux下搜索文件

在Windows下有一个搜索工具，可以让我们很快找到文件，这很有用。然而在Linux下，搜索功能更加强大。

5.7.1 用which命令查找可执行文件的绝对路径

前面已经用过which命令，但需要注意的是，which只能用来查找PATH环境变量中出现的路径下的可执行文件。这个命令比较常用，有时我们不知道某个命令的绝对路径，用which查找就很容易知道了。例如，查找vi和cat的绝对路径，示例命令如下：

```
# which vi
/usr/bin/vi
# which cat
/usr/bin/cat
```

5.7.2 用whereis命令查找文件

whereis命令通过预先生成的一个文件列表库查找与给出的文件名相关的文件，其格式为whereis [-bms] [文件名称]，其中各选项的含义如下所示。

- -b：只查找二进制文件。
- -m：只查找帮助文件（在man目录下的文件）。
- -s：只查找源代码文件。

例如，用whereis查看ls的示例命令如下：

```
# whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
```

可以看到，这里找到了2个文件。这个命令类似于模糊查找，只要文件名包含“ls”字符，就会列出来。此外，whereis 命令阿铭很少用到。

5.7.3 用locate命令查找文件

locate命令类似于whereis，也是通过查找预先生成的文件列表库来告诉用户要查找的文件在哪里，后面直接跟文件名。如果你的Linux没有这个命令，请安装mlocate软件包，安装命令如下：

```
# yum install -y mlocate
# locate passwd
locate: 无法执行 stat ()
`/var/lib/mlocate/mlocate.db': 没有那个文件或目录
```

安装好mlocate软件包后，初次运行locate命令会报错，这是因为系统还没有生成那个文件列表库。可以使用updatedb命令立即生成（或更新）这个库。如果你的服务器上正执行着重要的业务，那么最好不要去运行这个命令，因为一旦运行，服务器的压力会增大。默认情况下，这个数据库每周更新一次。如果使用locate命令搜索一个文件，而该文件正好是在两次更新时间段内创建的，那肯定

9 0 0 0 1
回到旧版
顶部

locate所搜索到的文件列表，不管是目录名还是文件名，只要包含我们要搜索的关键词，都会列出来，所以locate不适合精准搜索。这个命令阿铭也不常用。

5.7.4 使用find搜索文件

find这个搜索工具是阿铭用得最多的一个，请务必熟记，其格式为：find [路径] [参数] 。下面介绍阿铭常用的几个参数。

- -atime +n/-n：表示访问或执行时间大于或小于n天的文件。
- -ctime +n/-n：表示写入、更改inode属性（如更改所有者、权限或者链接）的时间大于或小于n天的文件。
- -mtime +n/-n：表示写入时间大于或小于n天的文件，该参数用得最多。

下面我们先来做个简单的试验，示例命令如下：

```
# find /tmp/4_6/ -mtime -1
/tmp/4_6/
/tmp/4_6/test1
/tmp/4_6/dir2
/tmp/4_6/dir2/test2
/tmp/4_6/dir2/test5
/tmp/4_6/dir2/test6
/tmp/4_6/dir2/test7
/tmp/4_6/dir3
/tmp/4_6/dir3/test3
/tmp/4_6/dir4
/tmp/4_6/test4
/tmp/4_6/test5
```

上例中， -mtime -1表示，mtime在1天之内的文件，单位是天。而-mtime +10表示mtime在10天以上的文件。还有一种用法-mmin -10，表示mtime在10分钟内的文件。有时候，也可以不加+或者-，比如-mtime 10，这表示正好为10天，这种用法就少了。

看到这里，你可能不太理解这三个time属性，那阿铭就先介绍一下它们。 文件的access time（即atime）是在读取文件或者执行文件时更改的。文件的modified time（即mtime）是在写入文件时随文件内容的更改而更改的。文件的change time（即ctime）是在写入文件、更改所有者、权限或链接设置时随inode内容的更改而更改的。

其中，inode（索引节点）用来存放档案及目录的基本信息，包含时间信息、文档名、所有者以及所属组等。inode是Unix操作系统中的一种数据结构，其本质是结构体，在文件系统创建时生成，且个数有限。在Linux下，可以通过命令df -i来查看各个分区的inode总数以及使用情况。

因此，更改文件的内容即会更改mtime和ctime，但是文件的ctime可能会在mtime未发生任何变化时更改。例如，更改了文件的权限，但是文件内容没有变化，那么，如何获得一个文件的atime、mtime以及ctime呢？stat命令可用来列出文件的atime、ctime和mtime，示例命令如下：

```
# stat dir2/test2
文件："dir2/test2"
大小：0 块：0 IO 块：4096 普通空文件
设备：803h/2051d Inode：25689396 硬链接：1
```

9 0 0 0 1
回到旧版
顶部

最近更改：2019-12-30 07:44:10.706789647 -0500

最近改动：2019-12-30 07:45:37.978885268 -0500

创建时间：-

atime不一定在访问文件之后被修改，因为在使用ext3文件系统时，如果mount使用了noatime参数，那么就不会更新atime的信息。总之，这三个time属性值都放在了inode中。若mtime、atime被修改，那么inode就一定会改，既然inode改了，那ctime也跟着要改了。

下面阿铭继续介绍find的常用选项。

- -name filename：表示直接查找该文件名的文件，这个选项比较常用，示例命令如下：

find . -name test2 //.表示当前目录，当前目录在/tmp/4_6下面

./dir2/test2

find . -name "test*" //支持用*通配

./test1

./dir2/test2

./dir2/test5

./dir2/test6

./dir2/test7

./dir3/test3

./test4

./test5

-type filetype：表示通过文件类型查找文件。文件类型在前面已经简单介绍过，相信你已经基本了解了。filetype包含了f、b、c、d、l、s等类型，示例命令如下：

find . -type d

.

./dir2

./dir3

./dir4

5.8 Linux文件系统简介

Windows系统格式化硬盘时，会指定格式FAT或者NTFS，而Linux的文件系统格式为ext3、ext4或者xfs。早期的Linux使用ext2格式，CentOS 5默认使用ext3，CentOS 6默认使用ext4，而CentOS 7和RHEL8/Rocky8默认使用xfs格式。ext2文件系统虽然高效、稳定，但随着Linux系统在关键业务中的应用，Linux文件系统的弱点也逐渐显露出来。因为ext2文件系统不是日志文件系统，这在关键行业是一个致命的弱点。

ext3文件系统是直接从ext2文件系统发展而来的，它带有日志功能，可以跟踪记录文件系统的变化，并将变化内容写入日志。写操作首先是对日志记录文件进行操作，若整个写操作由于某种原因（如系统掉电）而中断，当系统重启时，会根据日志记录来恢复中断前的写操作，而且这个过程费时极短。目前，ext3文件

9 0 0 0 1
回到旧版
顶部

而ext4文件系统，较ext3文件系统又有很多好的特性，其中最明显的特征是ext4支持的最大文件系统容量和单个最大文件大小比ext3大了许多，二者之间的详细区别阿铭不再介绍。虽然ext4支持的单个文件大小已经达到了16TB，最大文件支持到40多亿，但依然还是有瓶颈的，xfs支持的量级要比ext4大得多。CentOS 7默认采用xfs也是必然的，还有一个原因，xfs的开发者目前受雇于Red Hat公司，ext4的开发者受雇于Google公司。

Linux文件系统在Windows中是不能识别的，但在Linux系统中可以挂载Windows文件系统。Linux目前支持MS-DOS、VFAT、FAT、BSD等格式，如果你使用的是RHEL或者Rocky，那么请不要妄图挂载NTFS格式的分区到Linux下，因为它不支持NTFS。当有这方面的需求时，我们可以通过安装ntfs-3g软件包来解决这个问题。

除了ext3/ext4文件系统外，有些Linux发行版（如SUSE）默认的文件系统为ReiserFS，它在处理小于1KB的文件时的速度是ext文件系统的10倍。另外，ReiserFS空间浪费较少，它不会为一些小文件分配inode，而是打包存放在同一个磁盘块中。而ext是把它们单独存放在不同的块上。例如，块大小为4KB，那么两个100字节的文件会占用两个块，ReiserFS则只占用一个块。当然，ReiserFS也有缺点，就是每升级一个版本，都要将磁盘重新格式化一次。

5.9 Linux文件类型

前面我们简单介绍了普通文件-、目录d等文件类型，为了加深理解，阿铭将详细介绍Linux的文件类型。

5.9.1 常见文件类型

在Linux文件系统中，主要有以下几种类型的文件。

- 普通文件（regular file）：即一般类型的文件，当用命令ls -l查看某个目录时，第一个属性为“-”的文件就是普通文件。它又可分成纯文本文件（ASCII）和二进制文件（binary）。纯文本文件可以通过cat、more、less等工具直接查看内容，而二进制文件不能。例如，我们用的命令/usr/bin/ls就是一个二进制文件。
- 目录（directory）：它与Windows下的文件夹类似，只不过在Linux中我们不将其称为“文件夹”，而称为“目录”。用命令ls -l查看的第一个属性值为d的文件就是目录。
- 链接文件（link file）：用命令ls -l查看的第一个属性为l的文件就是链接文件，它类似于Windows下的快捷方式。这种文件在Linux中很常见，阿铭在日常系统运维工作中经常用到，所以你要特别留意一下这类文件。
- 设备（device）：即与系统周边相关的一些文件，通常都集中在/dev目录下。这种文件一般分为两种，一种是块（block）设备，就是一些存储数据，以提供系统存取的接口设备，简称硬盘。例如，第一块硬盘是/dev/sda1，用命令ls-l查看的第一个属性值为b的文件就是块设备。另一种是字符（character）设备，是一些串行端口的接口设备，例如键盘、鼠标等，用命令ls-l查看的第一个属性为c的文件就是字符设备。

5.9.2 Linux文件后缀名

对于“后缀名”这个概念，相信你并不陌生。在Linux系统中，文件的后缀名没有具体意义，加或者不加都无所谓。但是为了便于区分，我们习惯在定义文件名时加一个后缀名。这样当用户看到这个文件名时，就会很快知道它到底是一个什么文件，例如1.sh、2.tar.gz、my.cnf、test.zip等。

如果你首次接触这些文件，也许会很疑惑，但没关系，深入学习之后，你就会逐渐了解这些文件。阿铭所列举的几个文件名中，1.sh代表它是一个shell脚本，2.tar.gz代表它是一个压缩包，my.cnf代表它是一个配置文件，test.zip代表它是一个压缩文件。

9 0 0 0 1
回到旧版
顶部

5.9.3 Linux的链接文件

前面阿铭多次提到了“链接文件”这个概念，它分为硬链接（hard link）和软链接（symbolic link）两种。两种链接的本质区别在于inode。下面阿铭就来介绍一下这两种链接文件。

- 硬链接：当系统要读取一个文件时，会先读inode信息，然后再根据inode中的信息到块区域将数据取出来。而硬链接是直接再建立一个inode链接到文件放置的块区域，即进行硬链接时该文件内容没有任何变化，只是增加了一个指向这个文件的inode，并不会额外占用磁盘空间。硬链接有两个限制：(1)不能跨文件系统，因为不同的文件系统有不同的inode table；(2)不能链接目录。
- 软链接：与硬链接不同，软链接是建立一个独立的文件，当读取这个链接文件时，它会把读取的行为转发到该文件所链接的文件上。例如，现在有一个文件a，我们做了一个软链接文件b（只是一个链接文件，非常小），b指向了a。当读取b时，b就会把读取的动作转发到a上，这样就读取了文件a。当我们删除文件a时，链接文件b不会被删除；但如果再次读取b时，会提示无法打开文件。然而，当我们删除b时，a是不会有任何影响的。

由此看来，似乎硬链接比较安全，因为删除任何一个硬链接文件，还会有其他文件指向那个inode，既然inode存在，那文件的数据块也就存在。但由于硬链接的限制太多了（包括无法做目录的链接），所以用途上比较受限，而软链接的使用方向较广。那么，如何建立软链接和硬链接呢？这就用到了下面我们要介绍的ln（link）命令。

ln命令的格式为： ln [-s] [来源文件] [目的文件]，该命令常用的选项是-s。如果不加-s选项就是建立硬链接，加上-s选项就建立软链接。示例命令如下：

```
# mkdir /tmp/4_9
# cd /tmp/4_9
# cp /etc/fstab ./
# ll
总用量 4
-rw-r--r-- 1 root 1121 12月 30
08:03 fstab
# du -sk //du命令用来计算文件或者目录的大小，-k表示以KB为单位，这里的4，就是4KB
4 .
# ln fstab fstab-hard
# ll
总用量 8
-rw-r--r-- 2 root 1121 12月 30
08:03 fstab
-rw-r--r-- 2 root 1121 12月 30
08:03 fstab-hard
# du -sk
4 .
```

这里的ll命令等同于ls -l，请使用which命令查看一下。一开始目录下面只有一个fstab文件，目录总大小为4KB，做了硬链接后，虽然两个文件的大小都为1121B，但目录的总大小并没有变化。我们不妨先删除源文件，然后再来比较一下，示例命令如下：

9 0 0 0 1
回到旧版
顶部

总用量 4

-rw-r--r-- 1 root 1121 12月 30

08:03 fstab-hard

du -sk

4 .

上例中，删除源文件passwd后，文件大小依旧不变。这说明硬链接文件并不会复制数据块，额外占用磁盘空间。再来看硬链接的另外一个限制——不允许目录做硬链接，示例命令如下：

mkdir 123

ln 123 456

ln: "123": 不允许将硬链接指向目录

下面我们再来看看软链接的一些特性。首先建立一个测试目录456，然后复制/etc/passwd文件来做测试，再给它做一个软链接文件，示例命令如下：

mkdir 456

cd 456

cp /etc/fstab ./

ln -s fatab fstab-soft

ll

总用量 4

-rw-r--r-- 1 root 1121 12月 30

08:05 fatab

lrwxrwxrwx 1 root 6 12月 30 08:05 fatab-soft -> fatab

head -n1 fatab-soft

root:x:0:0:root:/root:/bin/bash

head -n1 fstab

root:x:0:0:root:/root:/bin/bash

rm -f fstab

head -n1 fstab-soft

head:无法打开"fstab-soft"读取数据:没有那个文件或目录

ll

总用量 0

lrwxrwxrwx 1 root 6 12月 30

08:05 fstab-soft -> fstab

上例中，如果删除源文件，则不能读取软链接文件，而且使用命令ll查看，发现颜色也有所变化。另外，目录不可以做硬链接，但可以做软链接，示例命令如下：

 # cd ..

 # ln 456 789

ln: "456": 不允许将硬链接指向目录

 # ln -s 456 789

9 0 0 0 1

回到旧版

顶部

08:06 456

lrwxrwxrwx 1 root 3 12月 30 08:07 789 -> 45

分享至

投诉或建议

9
0
0
0
1

评论 1　赞与转发 9

B站跨年阵容：孙燕姿、周深..去围观>

最热 | 最新

你猜我的评论区在等待谁?

发布

阿铭linux LV3 UP

置顶 请点“+关注”，并点赞、投币，然后看你的私信，我会送你一份价值千元的运维笔记。

2023-10-20 10:34　回复

没有更多评论