第5章 Linux文件和目录管理上

阿铭linux
2023年10月19日 11:03

已关注

15
0
3
0
1

从这一章开始，阿铭介绍的命令会越来越多，希望你能够反复练习每一个命令的每一个选项。在Windows下，新建、复制、删除文件或者文件夹都非常简单，而Linux需要我们使用命令行进行操作。这样便增加了学习Linux系统的难度，不过不用担心，一旦能够熟练使用它们，那么你将永远也不会忘记。万事开头难，所以请大家努力吧！

5.1 绝对路径和相对路径

在Linux中，什么是一个文件的路径呢？简单地说，就是这个文件存放的地方，例如在上一章提到的/root/.ssh/authorized_keys就是一个文件的路径。只要你告诉系统某个文件的路径，系统就可以找到这个文件。

在Linux中，存在着绝对路径和相对路径。

- 绝对路径：路径的写法一定是由根目录/写起的，例如 /usr/local/mysql。
- 相对路径：路径的写法不是由根目录/写起的。例如，首先用户进入到/home，然后再进入到test ，执行的命令为

```
# cd /home
# cd test
```

此时用户所在的路径为/home/test。第一个cd命令后紧跟/home，前面有斜杠；而第二个cd命令后紧跟test，前面没有斜杠。这个test是相对于/home目录来讲的，所以称为相对路径。

5.1.1 命令cd

命令cd（change directory的简写）是用来变更用户所在目录的，如果后面什么都不跟，就会直接进入当前用户的根目录下。我们做实验用的是root账户，所以运行命令cd后，会进入root账户的根目录/root下。如果后面跟目录名，则会直接切换到指定目录下。示例命令如下：

```
# cd /tmp/
# pwd
/tmp
# cd
# pwd
/root
```

上例中，命令pwd用于显示当前所在目录。命令cd后面只能是目录名，如果跟了文件名，则会报错，例如：

```
# cd /etc/fstab
-bash: cd: /etc/fstab: 不是目录
```

因为/etc/fstab为一个文件，所以就报错了。在Linux文件系统中，有两个特殊的符号也可以表示目录。 “.”表示当前目录， “..”表示当前目录的上一级目录，示例命令如下：

```
# cd /usr/local/lib/
# pwd
```

回到旧版
顶部

```
# pwd
/usr/local/lib
# cd ..
# pwd
/usr/local
```

上例中，首先进入/usr/local/lib/目录，接着输入.，用命令pwd查看当前目录，还是在/usr/local/lib/目录下，然后输入..，则进入/usr/local/ 目录（即/usr/local/lib目录的上一级目录）。

5.1.2 命令mkdir

命令mkdir（make directory的简写）用于创建目录，这个命令在上一章中用过。该命令的格式为：mkdir [-mp] [目录名称]。其中，-m、-p为其选项。-m选项用于指定要创建目录的权限（这个选项不常用，阿铭不作重点解释）。-p选项很管用，我们先来做个试验，你就一目了然了。执行如下命令：

```
# mkdir /tmp/test/123
mkdir: 无法创建目录 '/tmp/test/123': 没有那个文件或目录
# mkdir -p /tmp/test/123
# ls /tmp/test
123
```

当我们想创建目录/tmp/test/123时，提示无法创建、/tmp/test目录不存在。在Linux中，如果它发现要创建的目录的上一级目录不存在，就会报错。为了解决这个问题，Linux设置了-p选项，这个选项可以帮我们创建一大串级联目录，并且当创建一个已经存在的目录时，不会报错。示例命令如下：

```
# ls -ld /tmp/test/123
drwxr-xr-x 2 root 6 12月 30
07:25 /tmp/test/123
# mkdir /tmp/test/123
mkdir: 无法创建目录 '/tmp/test/123': 文件已存在
# mkdir -p /tmp/test/123
# ls -ld /tmp/test/123
drwxr-xr-x 2 root 6 12月 30
07:25 /tmp/test/123
```

在上一章中阿铭已经介绍过ls命令，但并没有介绍它的-d选项。这个选项是针对目录的，通常都是和-l并用，写成-ld。它可以查看指定目录的属性，比如在本例中，它可以查看/tmp/test/123目录的创建时间，如果不加-d，则会显示该目录里面的文件和子目录的属性。

5.1.3 命令rmdir

命令rmdir（remove directory的简写）用于删除空目录，后面可以是一个目录，也可以是多个目录（用空格分隔）。该命令只能删除目录，不能删除文件，所以阿铭一般不用它，而改用命令rm（remove的简写），这个命令既可以删除目录，又可以删除文件，将在下一节中介绍。rmdir有和mkdir具有相同的选项-p，它同样可以级联删除一大串目录，但在级联的目录中，如果某一个目录里还有目录或者文件时，这个命令就不好用了。我们先来看看命令rmdir的用法，示例命令如下：

15 0 3 0 1
回到旧版
顶部

```
# rmdir /tmp/test/
rmdir: 删除 '/tmp/test/' 失败: 目录非空
# rmdir /tmp/test/123
# ls /tmp/test
#
```

在上例中，命令rmdir只能删除空目录，即使加上-p选项也只能删除一串空目录。可见，这个命令有很大的局限性，偶尔用一下还可以。

5.1.4 命令rm

命令rm是最常用的，它也有很多选项。你可以通过命令man rm来获得它的详细帮助信息。这里，阿铭只介绍最常用的两个选项。

-r：删除目录用的选项，类似于rmdir，但可以删除非空目录。下面阿铭先创建一连串的目录，然后尝试删除它们。示例命令如下：

```
# mkdir -p /tmp/test/123
# rm -r /tmp/test/123
rm:是否删除目录 '/tmp/test/123'? y
```

和rmdir不同的是，使用rm –r命令删除目录时，会询问是否删除，如果输入“y”则会删除，如果输入“n”则不删除。另外，rm -r命令能删除非空目录。

-f：表示强制删除。它不再询问是否删除，而是直接删除。如果后面跟一个不存在的文件或者目录，则不会报错。下面阿铭尝试删除一个不存在的目录，示例命令如下：

```
# rm /tmp/test/123/123
rm: 无法删除 '/tmp/test/123/123': 没有那个文件或目录
# rm -f /tmp/test/123/123
```

上例中，/tmp/test/123/123这个目录是不存在的，但加上-f选项后，就不会报错。但如果要删除一个存在的目录时，即使加上-f选项也会报错。所以，使用命令rm删除目录时，一定要加-r选项。请对比下面的示例命令和上面的示例命令的区别：

```
# rm -f /tmp/test/123
rm: 无法删除 '/tmp/test/123': 是一个目录
# rm -rf /tmp/test/123
```

关于rm命令，阿铭使用最多的是-rf选项，这样删除文件或目录比较方便。但请大家千万要注意，rm -rf命令后面不能加“/”，否则它会把你的系统文件全部删除，这是非常危险的！

5.2 环境变量PATH

在讲环境变量之前，阿铭先介绍一下命令which，它用于查找某个命令的绝对路径。示例命令如下：

```
# which rmdir
/usr/bin/rmdir
# which rm
alias rm='rm -i'
```

15 0 3 0 1
回到旧版
顶部

```
alias ls='ls --color=auto'
/usr/bin/ls
```

其中rm和ls是两个特殊的命令，在上例中我们使用alias命令做了别名。我们用的rm实际上是rm -i，加上-i选项后，删除文件或者命令时都会询问是否确定要删除，这样做比较安全。命令alias可以设置命令或文件的别名，阿铭会在10.1.3节中详细介绍。命令which阿铭不常使用，平时只用来查询某个命令的绝对路径。

在上面的示例中，用which查到rm命令的绝对路径为/usr/bin/rm。那么你是否会问：“为什么我们使用命令时，只是直接打出了命令，而没有使用这些命令的绝对路径呢？”这是环境变量PATH在起作用。请输入如下命令：

```
# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
```

这里的echo用来输出$PATH的值。PATH前面的$是变量的前缀符号，这些知识点将会在第10章中详细介绍。

因为/bin目录在PATH的设定中，所以自然可以找到ls。但值得注意的是，由于PATH里没有/root目录，如果你将ls移到/root目录下，当执行ls命令时，系统自然就找不到可执行文件了，它会提示command not found!。示例命令如下：

```
# mv /usr/bin/ls /root/
# ls
-bash: /usr/bin/ls: 没有那个文件或目录
```

命令mv（move的简写）用于移动目录或者文件，它还有重命名的作用（这个将在4.2.2节中介绍）。那么，该如何解决上面的这种问题呢？有两种方法，一种方法是直接将/root这个路径加入到$PATH当中，命令如下：

```
# PATH=$PATH:/root
# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin:/root
# ls
anaconda-ks.cfg ls
```

另一种方法是使用绝对路径，命令如下：

```
# /root/ls
anaconda-ks.cfg ls
```

为了不影响系统使用，建议将ls文件还原，命令如下：

```
# mv /root/ls /usr/bin/
```

5.2.1 命令cp

cp是copy（即复制）的简写，该命令的格式为：cp [选项] [来源文件] [目的文件]。例如，我想把test1复制成test2，可以写为cp test1 test2。下面介绍命令cp的几个常用选项。

-r：如果要复制一个目录，必须加-r选项，否则不能复制，这类似于rm命令。示例命令如下：

```
# mkdir 123
```

15 0 3 0 1
回到旧版
顶部

```
# cp -r 123 456
# ls –ld 123 456
drwxr-xr-x 2 root 6 12月 30 07:35 123
drwxr-xr-x 2 root 6 12月 30 07:36 456
```

-i：这是安全选项，如果遇到一个已存在的文件，会询问是否覆盖，这也与rm命令类似。在RedHat/Rocky系统中，使用的cp命令其实是cp –i，我们可以通过which命令查看，具体如下：

```
# which cp
alias cp='cp -i'
/bin/cp
```

为了更形象地说明-i 选项的作用，我们来做一个简单的小试验，命令如下：

```
# cd 123
# ls
# touch 111
# touch 222
# cp -i 111 222
cp：是否覆盖 '222'？ n
# echo 'abc' > 111
# echo 'def' > 2
22
# cat 111 222
abc
def
# /bin/cp 111 222
# cat 111
abc
# cat 222
abc
```

上例中，touch可以解释为：如果有这个文件，则会改变该文件的访问时间；如果没有这个文件，就会创建这个文件。前面说过，echo命令用于打印，这里echo的内容abc和def并没有显示在屏幕上，而是分别写入了文件“111”和“222”。起写入作用的就是符号“>”，这在Linux中叫做重定向，即把前面产生的输出写入到后面的文件中。而cat命令则用于读一个文件，并把读出的内容打印到当前屏幕上。（重定向将在第11章中介绍，cat命令将在5.3.1节中详细介绍，这里你只要明白它们的含义即可。）

5.2.2 命令mv

mv是move的简写，该命令的格式为：mv [选项] [源文件或目录] [目标文件或目录]。该命令有如下几种情况。

- 目标文件是目录，但该目录不存在。
- 目标文件是目录，且该目录存在。
- 目标文件是文件，且该文件不存在。
- 目标文件是文件，但该文件存在。

15 0 3 0 1
回到旧版
顶部

命名为给定的目标文件名。

当目标文件是文件时，其存在与否，执行后的结果也是不一样的。如果该文件存在，则会询问是否覆盖。如果该文件不存在，则会把源文件重命名为给定的目标文件名。

下面我们来做几个小试验，示例命令如下：

```
# mkdir /tmp/test_mv
# cd /tmp/test_mv
# mkdir dira dirb
# ls
dira dirb
# mv dira dirc
# ls
dirb dirc
```

上例中，首先阿铭创建了一个实验用的目录/tmp/test_mv，然后进入到该目录下进行实验，这样做的目的是保持目录和文件简洁，后面的实验以此类推。这里，目标文件是目录dirc，并且dirc不存在，相当于把目录dira重命名为dirc。

下例中，目标文件是目录dirb，且dirb存在，则会把目录dirc移动到目录dirb里：

```
# mv dirc dirb
# ls
dirb
# ls dirb
dirc
```

下例中，mv filed filee的目标文件是文件filee且这个文件不存在，相当于把文件filed重命名为filee。mv filee dirb命令则将更名后的文件filee移动到目录dirb里。

```
# touch filed
# ls
dirb filed
# mv filed filee
# ls
dirb filee
# mv filee dirb
# ls
dirb
# ls dirb
dirc filee
```

15
0
3
0
1
回到旧版
顶部

5.3 几个与文档相关的命令

5.3.1 命令cat

命令cat（它并不是某个单词的简写，大家可以通过man cat命令查看它的解释）是比较常用的一个命令，用于查看一个文件的内容并将其显示在屏幕上。cat后面可以不加任何选项，直接跟文件名。下面阿铭介绍它的两个常用选项。

-n：查看文件时，把行号也显示到屏幕上。示例命令如下（当前目录依然在/tmp/test_mv）：

```
# echo '111111111' >
dirb/filee
# echo '222222222' >>
dirb/filee
# cat dirb/filee
111111111
222222222
# cat -n dirb/filee
1 111111111
2 222222222
```

上例中出现了符号>>，它跟前面介绍的符号>类似，其作用也是重定向，即把前面的内容输入到后面的文件中，但符号>>是“追加”的意思。当使用符号>时，如果文件中有内容，则会删除文件中原有的内容，而使用符号>>则不会删除原有的内容。

-A：显示所有的内容，包括特殊字符。示例命令如下：

```
# cat -A dirb/filee
111111111$
222222222$
```

上例中，若不加-A选项，那么每行后面的$符号是看不到的。

5.3.2 命令tac

和命令cat一样，命令 tac（正好是命令cat的反序写法）也是把文件的内容显示在屏幕上，只不过是先显示最后一行，然后显示倒数第二行，最后才显示第一行。我们使用命令tac来查看刚才创建的文件dirb/filee，显示的结果和命令cat正好是反序，如下所示：

```
# tac dirb/filee
222222222
111111111
```

5.3.3 命令more

命令more也用于查看一个文件的内容，后面直接跟文件名。当文件内容太多，一屏不能全部显示时，用命令cat肯定是看不了前面的内容，这时可以使用命令more。当看完一屏后，按空格键可以继续看下一屏，看完所有内容后就会退出，按Ctrl+B可以向上翻屏，按Ctrl+F向下翻屏（同空格）。如果你想提前退出，按q键即可。

15
0
3
0
1
回到旧版
顶部

命令less的作用相命令more一样，后面直接跟文件名，但命令less比more功能要多一些。按空格键可以翻页，按j键可以向下移动（按一下就向下移动一行），按k键可以向上移动。在使用more和less查看某个文件时，你可以按一下/键，并输入一个字符串（如root），然后回车，这样就可以查找这个字符串了。如果是查找多个该字符串，可以按n键显示下一个。另外，也可以用?键替代/键来搜索字符串，唯一不同的是，/是在当前行向下搜索，而?是在当前行向上搜索。

15
0
3
0
1

5.3.5 命令head

命令head用于显示文件的前10行，后面直接跟文件名。如果加-n选项，则显示文件的前几行，示例命令如下：

```
# 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
# head -n 1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
# head -n2 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

大家请注意，选项-n后有无空格均可。另外，也可以省略字母n，-后面直接跟数字， 如下：

```
# head -2 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

5.3.6 命令tail

和命令head类似，命令tail用于显示文件的最后10行，后面直接跟文件名。如果加-n选项，则显示文件的最后几行，示例命令如下：

```
# tail /etc/passwd
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd
daemon:/dev/null:/sbin/nologin
polkitd:x:998:996:User for polkitd:/:/sbin/nologin
```

```
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
chrony:x:995:992::/var/lib/chrony:/sbin/nologin
# tail -n2 /etc/passwd
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
chrony:x:995:992::/var/lib/chrony:/sbin/nologin
# tail -2 /etc/passwd
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
chrony:x:995:992::/var/lib/chrony:/sbin/nologin
```

同样，-n后面有无空格均可，且字母n也可以省略。

另外，命令tail的-f选项也常用，它可以动态显示文件的最后10行。如果文件内容在不断增加，使用-f选项非常方便和直观。比如tail -f /var/log/messages可以动态、实时地查看文件/var/log/ messages中的内容。

5.4 文件的所有者和所属组

一个Linux目录或者文件，都会有一个所有者和所属组。所有者是指文件的拥有者，而所属组指的是这个文件属于哪一个用户组（关于用户、用户组的概念，会在第5章中详细介绍，这里你要明白一个用户组下面会有若干个用户）。Linux这样设置文件属性的目的是为了文件的安全。

例如，test文件的所有者是user0，而test1文件的所有者是user1，那么user1很有可能是不能查看test文件的，相应地，user0也很有可能不能查看test1文件（之所以说是可能，是因为user0和user1有可能属于同一个用户组，而恰好这个用户组对这两个文件有查看权限）。

有时我们也会有这样的需求：使一个文件能同时被user0和user1查看，这怎么实现呢？这时“所属组”就派上用场了。先创建一个组users，让user0和user1同属于users组，然后建立一个文件test2，且其所属组为users，这样user0和user1都可以访问test2文件。Linux文件属性不仅规定了所有者和所属组，还规定了所有者（user）、所属组（group）以及其他用户（others）对该文件的权限。我们可以通过ls -l命令来查看这些属性，代码如下：

```
# ls -l /etc/passwd
-rw-r--r--. 1 root 1080 12月 26
08:08 /etc/passwd
```

其中，第3列和第4列的root就是所有者和所属组。

5.5 Linux文件属性

在上例中，用ls –l命令查看当前目录下的文件时，共显示了9列内容（用空格划分列），它们都代表什么含义呢?

第1列：包含该文件的类型、所有者、所属组以及其他用户对该文件的权限。第1列共11位（阿铭这里列出的是10位，没有最后一位，你可以通过ls -l /看一下，会看到最后一位是一个.），其中第1位用来描述该文件的类型。上例中我们看到的文件类型有d和-，其实除了这两种外，还有l、b、c、s等，具体描述如下所示。

- d表示该文件为目录。
- -表示该文件为普通文件。
- l表示该文件为链接文件（link file），4.9.3节中提到的软链接即为该类型，示例命令如下：

15 0 3 0 1
回到旧版
顶部

上例中，第1列第1位是l，表示该文件为链接文件，后面阿铭还会介绍它。

- b表示该文件为块设备，比如/dev/sda就是这样的文件，磁盘分区文件就是这种类型。
- c表示该文件为串行端口设备文件（又称字符设备文件），比如键盘、鼠标、打印机、tty终端等都是这样的文件。
- s表示该文件为套接字文件（socket），用于进程之间的通信，后面讲到MySQL时会用到该类型的文件。

文件类型后面的9位，每3位为一组，上例中（rc.local）均为rwx这3个参数的组合。其中，r代表可读，w代表可写，x代表可执行。前3位为所有者（user）的权限，中间3位为所属组（group）的权限，最后3位为其他非本群组用户（others）的权限。下面阿铭举例来说明一下。

假设一个文件的属性为-rwxr-xr--，它代表的意思是，该文件为普通文件，文件拥有者可读、可写且可执行，文件所属组对其可读、不可写但可执行，其他用户对其只可读。对于一个目录来讲，打开这个目录即为执行这个目录，所以任何一个目录必须要有x权限才能打开并查看该目录下的内容。例如，一个目录的属性为drwxr--r--，其所有者为root，那么除root之外的其他用户是不能打开这个目录的。

关于前面提到第1列最后1位的“.”，阿铭要特别说明一下。老版本CentOS 5 是没有这个点的，这主要是因为新版本的ls添加了Selinux或者acl的属性。如果文件或者目录使用了Selinux context的属性，这里会是一个点“.”；如果设置了acl的属性，这里会是一个加号“+”。关于Selinux和acl，阿铭不再详细介绍，你只要了解是怎么回事即可。

第2列：表示该文件占用的节点（inode），如果是目录，那这个数值与该目录下的子目录数量有关。

第3列：表示该文件的所有者。

第4列：表示该文件的所属组。

第5列：表示该文件的大小。

第6列、第7列和第8列：表示该文件最后一次被修改的时间（mtime），依次为月份、日期以及时间。

第9列：表示文件名。

15
0
3
0
1

分享至

投诉或建议

评论 1　赞与转发 15

最热 | 最新

你猜我的评论区在等待谁?

发布

阿铭linux 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 09:59　回复

回到旧版

顶部

15
0
3
0
1