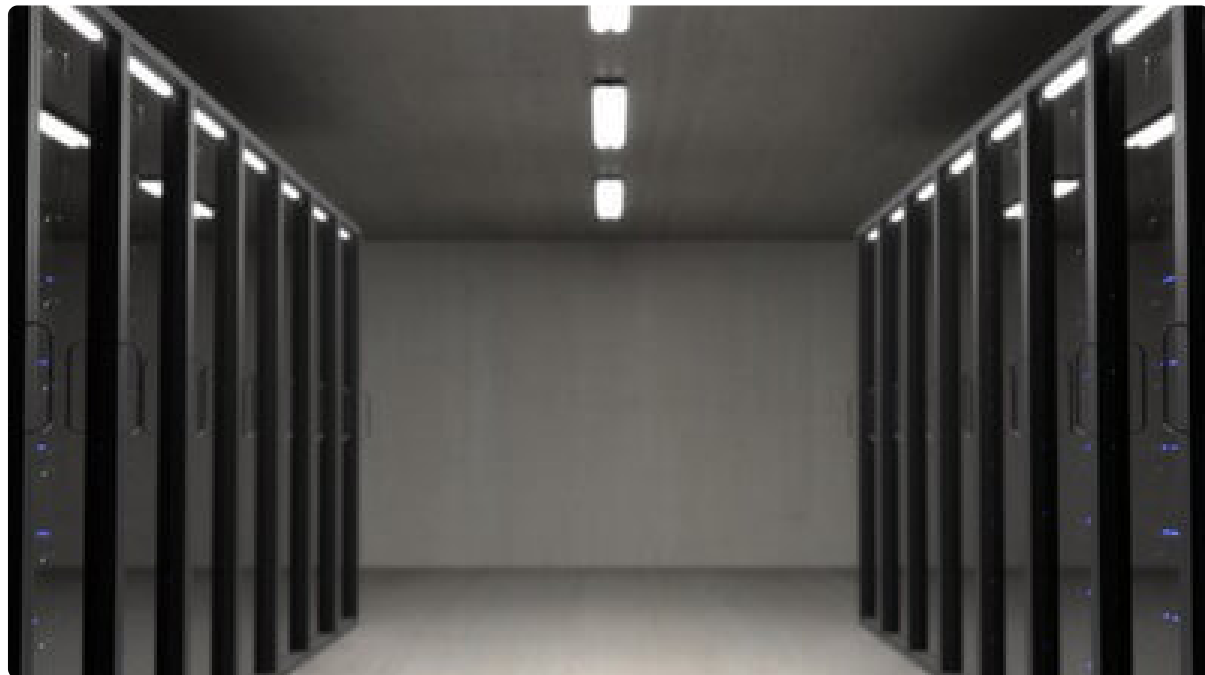


13 文件的复制，移动，删除和链接

更新时间：2019-07-04 19:58:21



“

最聪明的人是最不愿浪费时间的人。

——但丁

”

内容简介

1. 前言
2. cp 命令和 mv 命令：拷贝文件和移动文件
3. rm 命令：删除文件和目录
4. ln 命令：创建链接
5. 总结

1. 前言

上一课文件操纵，鼓掌之中，浏览和创建文件中，我们学习了如何显示文件内容、创建文件和目录。上一课的内容比较简单。

这一课的内容稍有难度，主要是链接那一块儿的知识点比较多，不过我们有大量的配图，相信会让你循序渐进地学会知识点的。

这一课我们还会学一个比较危险的命令：rm 命令。学会这个命令之后，你可别“删库跑路”啊，到时候“为师”可担待不起啊。

2. cp 命令和 mv 命令：拷贝文件和移动文件

接着来我们学习很重要的文件拷贝和移动的操作，还有文件的重命名。

cp 命令：拷贝文件或目录

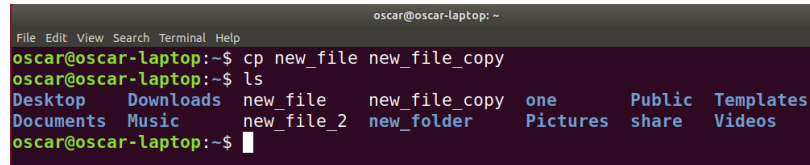
cp 是英语 copy 的缩写，表示“拷贝”。所以，聪明如你，应该已经知道了，此命令用于拷贝。

cp 命令不仅可以拷贝单个文件，还可以拷贝多个文件，也可以拷贝目录。

如果我们要拷贝上一课创建的 new_file 文件怎么做呢？很简单：

```
cp new_file new_file_copy
```

第一个文件 new_file 是已经存在的文件，也就是被拷贝的文件；第二个文件 new_file_copy 是需要创建的文件，是 new_file 的副本，内容一模一样。



```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ cp new_file new_file_copy  
oscar@oscar-laptop:~$ ls  
Desktop  Downloads  new_file      new_file_copy  one      Public  Templates  
Documents Music      new_file_2    new_folder     Pictures  share   Videos  
oscar@oscar-laptop:~$
```

我们可以看到，我们用 cp 命令在当前目录下创建了 new_file 的副本 new_file_copy。

复制文件到另一个目录

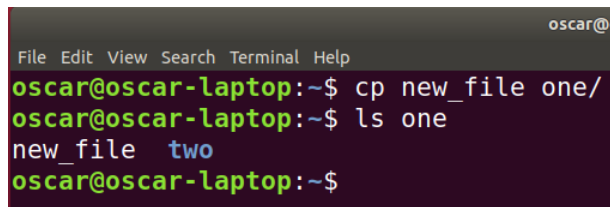
上面的例子中，我们是在当前目录下拷贝了文件 new_file，并生成了它的副本 new_file_copy。

当然，我们并不一定要在同一个目录下拷贝。我们也可以把文件拷贝到其它目录。

只需要把 cp 命令的第二个参数换成目录名。

```
cp new_file one/
```

上面的命令就把 new_file 这个文件拷贝到了 one 这个目录中。

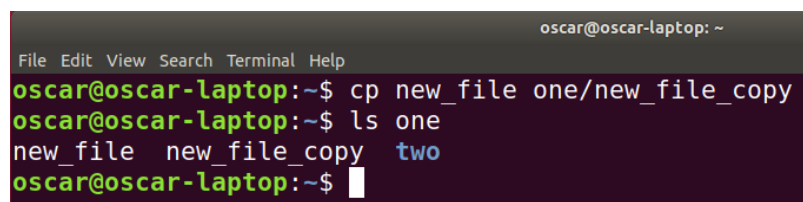


```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ cp new_file one/  
oscar@oscar-laptop:~$ ls one  
new_file  two  
oscar@oscar-laptop:~$
```

原先我们的 one 目录下只有 two 这个子目录，现在多了一个文件 new_file，它的内容和 one 目录的上层目录中的 new_file 是一样的。

如果你想拷贝文件到其它目录的同时，不具有相同名字，那么可以这样做：

```
cp new_file one/new_file_copy
```



```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ cp new_file one/new_file_copy  
oscar@oscar-laptop:~$ ls one  
new_file  new_file_copy  two  
oscar@oscar-laptop:~$
```

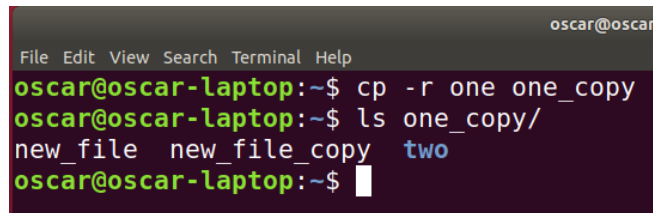
看到了吗，one 这个目录里多了一个 new_file_copy 的文件，它的内容和 one 目录的上层目录中的 new_file 是一样的。

拷贝目录

要拷贝目录，只要在 `cp` 命令之后加上 `-r` 或者 `-R` 参数（大写和小写作用是一样的，都表示 recursive，也就是“递归的”）。拷贝的时候，目录中的所有内容（子目录和文件）都会被拷贝。

之前，我们创建了一个目录 `one`，现在我们将它拷贝为 `one_copy` 看看：

```
cp -r one one_copy
```



```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ cp -r one one_copy
oscar@oscar-laptop:~$ ls one_copy/
new_file  new_file_copy  two
oscar@oscar-laptop:~$
```

看到了吗？`one_copy` 目录中的内容和 `one` 里面一模一样。

使用通配符 *

*号（星号）是很常用的正则表达式符号，被称为“通配符”，顾名思义就是百搭，可以替代任意字符串。

所以如果我们用如下命令：

```
cp *.txt folder
```

那么就会把当前目录下所有 `txt` 文件拷贝到 `folder` 这个子目录当中。

又如：

```
cp ha* folder
```

那么就会把当前目录下凡是以 `ha` 开头的文件都拷贝到 `folder` 目录中。

通配符是极为强大的，如果用得好，可以大大提高效率。是不是觉得比在 Windows 下用鼠标拷贝文件快捷很多呢？

mv 命令：移动文件

`mv` 是英语 `move` 的缩写，表示“移动”。`mv` 命令有两个功能：

- 移动文件（或目录）
- 重命名文件（或目录）

移动文件

与 `cp` 命令用法类似，不同的是 `cp` 命令会复制当前文件，而 `mv` 命令则是单纯的移动，并不会制作副本。

所以，`cp` 命令就好比 Windows 中的复制+粘贴，而 `mv` 命令就好比 Windows 中的剪切+粘贴。

```
mv new_file_2 one
```

以上命令将 new_file_2 这个文件移动到 one 这个目录，使得原先存在于家目录的 new_file_2 文件不存在了。

看到与 cp 的不同的吗？之前我们用 cp 命令的时候，原文件还是在的，只是把副本移动到了其它目录。

```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ ls  
Desktop  Music  new_file_copy  one_copy  share  
Documents new_file  new_folder  Pictures  Templates  
Downloads new_file_2  one  Public  Videos  
oscar@oscar-laptop:~$ mv new_file_2 one  
oscar@oscar-laptop:~$ ls  
Desktop  Downloads  new_file  new_folder  one_copy  Public  Templates  
Documents Music  new_file_copy  one  Pictures  share  Videos  
oscar@oscar-laptop:~$ ls one  
new_file  new_file_2  new_file_copy  two  
oscar@oscar-laptop:~$
```

用 mv 命令来移动目录则很简单，不需要额外的参数，就跟移动文件一样：

```
mv new_folder one
```

以上命令将 new_folder 这个目录（包括其下的子目录和文件）移动到 one 这个目录中。

```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ ls  
Desktop  Downloads  new_file  new_folder  one_copy  Public  Templates  
Documents Music  new_file_copy  one  Pictures  share  Videos  
oscar@oscar-laptop:~$ mv new_folder one  
oscar@oscar-laptop:~$ ls  
Desktop  Downloads  new_file  one  Pictures  share  Videos  
Documents Music  new_file_copy  one_copy  Public  Templates  
oscar@oscar-laptop:~$ ls one  
new_file  new_file_2  new_file_copy  new_folder  two  
oscar@oscar-laptop:~$
```

当然，我们也可以使用通配符：

```
mv *.txt one
```

以上命令是将当前目录下所有 txt 文件移动到 one 这个目录中。

除了移动文件，mv 命令还可以用于重命名文件。

事实上，Linux 中没有一个专门的命令用于重命名文件。

之所以 mv 命令可以重命名文件，其实还是归因于它的机制：移动文件。经过 mv 移动之后，原始文件变成了新的名字的文件，文件内容是不变的，这不就相当于重命名了吗？

```
mv new_file renamed_file
```

以上命令会将 new_file 重命名为 renamed_file:

```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ ls  
Desktop  Downloads  new_file  one  Pictures  share  Videos  
Documents Music  new_file_copy  one_copy  Public  Templates  
oscar@oscar-laptop:~$ mv new_file renamed_file  
oscar@oscar-laptop:~$ ls  
Desktop  Downloads  new_file_copy  one_copy  Public  share  Videos  
Documents Music  one  Pictures  renamed_file  Templates  
oscar@oscar-laptop:~$
```

希望大家好好练习 `cp` 和 `mv` 这两个命令，因为这两个命令真的很常用。

好了，经过了这一系列 `cp` 和 `mv` 的操作，现在我们的家目录已经有点凌乱了。是时候做一些清理工作了，下面掌声有请我们的 `rm` 命令。

3. `rm` 命令：删除文件和目录

`rm` 是英语 `remove` 的缩写，表示“移除”，这个命令就是用来删除东西的。

`rm` 命令可不好惹。

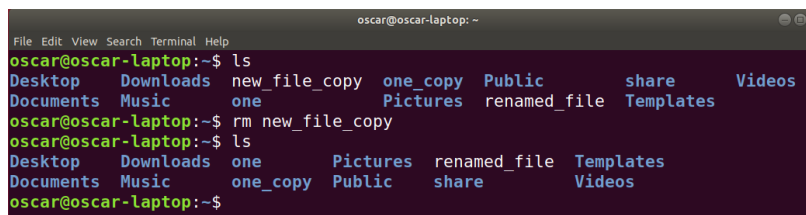
为什么说它不好惹呢？因为在终端中没有回收站或垃圾箱。如果用 `rm` 命令删除了文件，那可没后悔药吃，一般比较难恢复（还是有办法可以恢复的，只是挺麻烦）。

所以 `rm` 命令用起来虽然很过瘾，但是还需谨慎啊。

`rm` 命令可以删除一个文件、多个文件、目录，甚至你的整个 Linux 系统（如果你愿意的话）。

```
rm new_file_copy
```

以上命令删除当前目录下的 `new_file_copy` 这个文件。

A terminal window titled 'oscar@oscar-laptop: ~' showing the execution of the `rm` command. The user first runs `ls`, listing files: Desktop, Downloads, new_file_copy, one_copy, Public, share, Videos, Documents, Music, one, Pictures, renamed_file, Templates. Then the user runs `rm new_file_copy`. After running `ls` again, the file `new_file_copy` is no longer in the list. The terminal output is as follows:

```
oscar@oscar-laptop:~$ ls
Desktop  Downloads  new_file_copy  one_copy  Public  share  Videos
Documents Music      one           Pictures  renamed_file Templates
oscar@oscar-laptop:~$ rm new_file_copy
oscar@oscar-laptop:~$ ls
Desktop  Downloads  one  Pictures  renamed_file  Templates
Documents Music   one_copy  Public  share        Videos
oscar@oscar-laptop:~$
```

我们也可以同时删除多个文件，只要用空格隔开每个文件即可。例如：

```
rm file1 file2 file3
```

-i 参数：向用户确认是否删除

保险起见，用 `rm` 命令删除文件时，可以加上 `-i` 参数。这样对于每一个要删除的文件，终端都会询问我们是否确定删除。`i` 是英语 `inform` 的缩写，表示“告知，通知”。

有两种回答：

- `y`：是英语 `yes` 的缩写，表示“是”。那么回车确认后，文件就删除了；
- `n`：是英语 `no` 的缩写，表示“否”。那么回车确认后，文件不会删除。

例如：

```
rm -i renamed_file
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~
oscar@oscar-laptop:~$ ls
Desktop  Downloads  one      Pictures  renamed_file  Templates
Documents Music      one_copy Public    share         Videos
oscar@oscar-laptop:~$ rm -i renamed_file
rm: remove regular empty file 'renamed_file'? n
oscar@oscar-laptop:~$ ls
Desktop  Downloads  one      Pictures  renamed_file  Templates
Documents Music      one_copy Public    share         Videos
oscar@oscar-laptop:~$
```

上图中，我输入了 `n`，再回车，那么 `rm` 命令就不生效，文件没有被删除。如果我输入的是 `y`，那么文件就会被删除。

-f 参数：慎用，不会询问是否删除，强制删除

如果在 `rm` 命令后加上 `-f` 参数，那么终端不会询问用户是否确定删除文件，不论如何，文件会立刻被强制删除。

`f` 是英语 `force` 的缩写，表示“强迫，强制”。

```
rm -f file
```

以上命令会强制删除 `file` 文件。

-r 参数：递归地删除

`r` 是英语 `recursive` 的缩写，表示“递归的”。所以使用 `-r` 参数，可以使 `rm` 命令删除目录，并且递归地删除其包含的子目录和文件。

这个命令也挺危险的，用得不好可能你的子目录和文件都没了。

```
rm -r one
```

以上命令会删除 `one` 这个目录，包括其子目录和文件。

其实，也存在一个命令 `rmdir`，看着和 `rm -r` 挺像的。但是这个命令有个局限性：只能删除空的目录。

rm 命令加 -r 和 -f 参数：极为危险！

前方高能预警！

为什么说 `rm -rf` 命令极为危险呢？因为不凑巧的话，你可能毁了整个操作系统。

接下来，我们要给大家演示一个很可怕的命令，一个令人闻风哭泣的命令，一个“未成年人免入”的命令，一个“草木为之含悲，风云因而变色”的命令。

是的，那就是：

千万不要这样做！
`=> rm -rf /*` 或者 `rm -rf /`

请读者千万不要抱着试试看的心态，复制这条命令，然后用 `root` 身份运行这条命令。因为这条命令会删除你的整个 Linux 系统，如果你的 Windows 系统也挂载在 Linux 下，那么也会把你的 Windows 系统删了。所以，千万不要玩火！

这条命令可谓是 Linux 中头号一等危险的命令，历史上不少公司因为这个命令造成过惨重的损失。

我们把这条命令分解开来分析：

- `rm`：rm 命令，这个没问题吧，删除命令么；
- `-r`：递归删除；
- `-f`：不询问，强制删除；
- `/`：系统的根目录。后面可以不加通配符 `*`，也可以加。

所以整个命令的意思很明确：强制递归删除根目录下所有文件！

但你要问了：“既然有这么危险的命令，那么为什么 Linux 的开发允许这样的命令存在呢？”

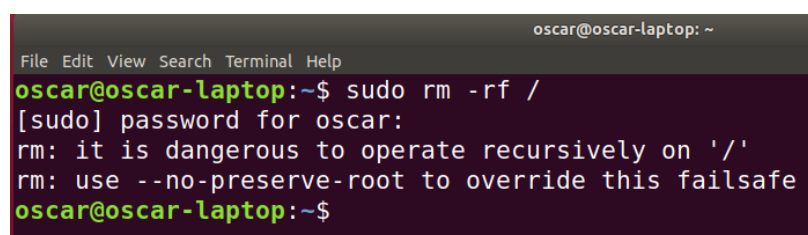
这是因为，不一定是谁都可以运行这条命令的，比如我目前是以 `oscar` 这个用户登录的。`oscar` 是普通用户，是不具备执行这条命令的权限。但如果是系统的超级用户，大管家 `root` 的话，执行这条命令是完全没问题的。

所以，这也是为什么我们说初学者尽量不要登录为 `root`。

下一课我们会讲用户和权限，到时候会有更深理解。

最后提醒一次：不管何种情况，千万不要运行这条命令：`rm -rf /`！

当然了，Ubuntu 系统中，如果你用 `root` 权限运行 `rm -rf /` 也是没问题的，因为它已经有了保护机制，如果你运行，会看到以下信息：



```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ sudo rm -rf /  
[sudo] password for oscar:  
rm: it is dangerous to operate recursively on '/'  
rm: use --no-preserve-root to override this failsafe  
oscar@oscar-laptop:~$
```

可以看到，什么也没有发生，终端显示了两条信息：

rm: it is dangerous to operate recursively on '/'

上面这句英语意思是：“在根目录 / 上递归地运行 `rm` 命令是危险的”。

rm: use --no-preserve-root to override this failsafe

上面这句英语意思是：“如果你坚持要在根目录 / 上递归地运行 `rm` 命令，那么请加上 `--no-preserve-root` 来取消保护”。

也就是说，如果你用这句命令：

```
sudo rm -rf --no-preserve-root /
```

那么，就会递归删除根目录下所有文件。请你也别运行上面这句命令。

所以，这也是我们喜欢 Ubuntu 系统的原因之一，不怕你不小心运行了这条可怕的命令：`rm -rf /`。

当然了，你可不要先进入根目录：

```
cd /
```

然后再用下面命令：

```
sudo rm -rf *
```

因为我在为了测试创建的某个虚拟机里用以上命令尝试过，没有保护，会直接递归删除系统文件，然后就几乎啥都没了。

不过没关系，那个虚拟机本来就是我来测试 `rm` 命令的，把那个虚拟机文件删除即可。

VirtualBox 的好处就是，你可以创建无数个虚拟机，想怎么玩就怎么玩。

所以，Ubuntu 系统对下面的命令是没有防护措施的：

```
cd /  
sudo rm -rf *
```

总结：`rm -rf` 命令是很危险的，用之前请先搞清楚自己所在的目录，以及后面所加的路径。以免铸成大错！

4. ln 命令：创建链接

虽然说 `ln` 这个命令不是特别常用，比之前的 `cp`、`mv`、`rm` 等使用频率要低，但是这个命令很有用。

`ln` 是 `link` 的缩写，在英语中表示“链接”。所以 `ln` 命令用于在文件之间创建链接。

说起链接可能你比较陌生，那么为了简单起见，我们用一个你比较熟悉的词好了：快捷方式。

虽然 Linux 的链接比起 Windows 的快捷方式要更复杂一些。但是性质是类似的。

事实上，Linux 下有两种链接类型：

- Physical link：物理链接或硬链接
- Symbolic link：符号链接或软链接

为了区分这两种链接类型的不同，我们首先来谈一谈像 Linux 这样的操作系统中，文件在硬盘上是如何存放的。

好啦，不要做苦瓜脸啦，只不过讲一下操作系统的一些原理，对于我们更好地理解知识点是很有帮助的。

文件的存储

在硬盘上存储时，大致来说（请注意我用了“大致来说”），每个文件有两部分：

- 文件名
- 文件内容

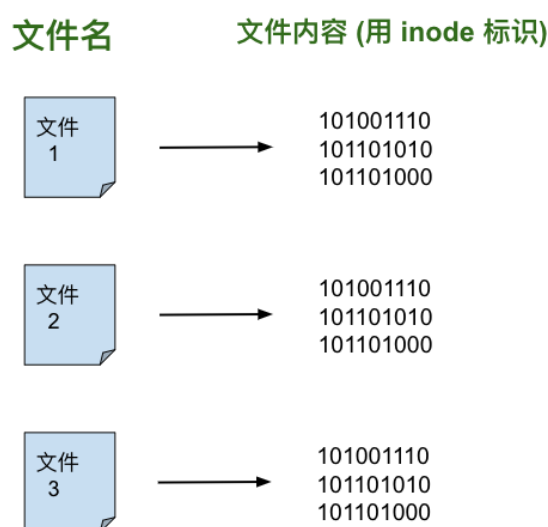
文件名的列表是储存在硬盘的其它地方的，和文件内容分开存放，这样方便 Linux 管理。

注意：为什么我上面要用“大致来说”呢？因为此处我们简化了描述，其实每个文件有三部分：

- 文件名
- 权限
- 文件内容

我们这里简化地将文件分为两部分：文件名和文件内容，因为我们不想把事情复杂化。我们想要理解两种链接类型的区别，暂时只要知道这些就够了。

每个文件的文件内容被分配到一个标示号码，就是 `inode`。因此每个文件名都绑定到它的文件内容（用 `inode` 标识），原理如下图：



理解了这点就可以学习下面的内容了，暂时我们并不需要钻研太深。当然了，有兴趣的话，你也可以百度 / Google 一下 `inode`，看一些资料，加深理解。

下面我们学习如何创建硬链接和软链接。

创建硬链接

比起软链接，硬链接的使用几率小很多，但我们还是要学习一下，毕竟可能会用到。

硬链接的原理：使链接的两个文件共享同样的文件内容，也就是同样的 `inode`。

所以一旦文件 1 和文件 2 之间有了硬链接，那么你修改文件 1 或文件 2，其实修改的是相同的一块儿内容。只不过我们可以用两个文件名来获取到文件内容。

硬链接有一个缺陷：只能创建指向文件的硬链接，不能创建指向目录的硬链接。但软链接可以指向文件或目录。

当然了，事实上，通过一些参数的修改，也可以创建指向目录的硬链接，但是比较复杂，这里不再详述。所以对于目录的链接，我们一般都是用软链接。

为了演示硬链接和软链接的操作，我们在家目录下新建一个目录吧：

```
cd
mkdir test
cd test
```

并且用 touch 命令创建一个新的空白文件：

```
touch file1
```

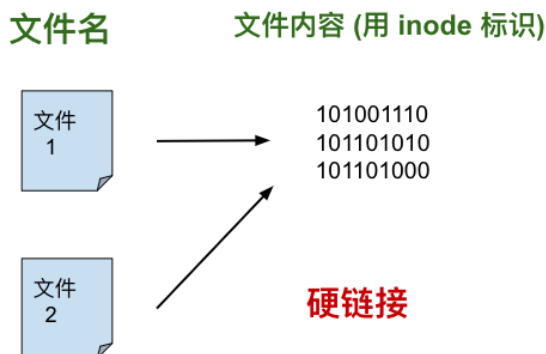
我们接着来创建一个文件 file2，使之成为 file1 的硬链接。

要创建硬链接，直接用 ln 命令，不加任何参数：

```
ln file1 file2
```

以上命令创建了 file1 的一个硬链接 file2。

硬链接原理图：



上图中，我们看到，file2 是新生成的硬链接，它指向 file1 的文件内容，也就是说它们共享相同的文件内容，也就是拥有同一个 inode。

我们用 ls -l 命令看一下，仿佛看不出什么端倪。因为目前看来，并没什么因素使我们相信这两个文件指向同一块文件内容（同一个 inode）。

但是我们可以用 ls -i 命令查看一下（-i 参数可以显示文件的 inode）。我们可以看到 file1 和 file2 的 inode 是一样的，都是 655571。当然你的电脑上应该和我不一样，是其它数字吧。

如果我们用 rm file2 来删除 file2，那么对 file1 没什么影响。如果我们用 rm file1 来删除 file1，对 file2 也没什么影响。所以，对于硬链接来说，删除任意一方的文件，共同指向的文件内容并不会从硬盘上被删除。

只有既删除 file1 又删除 file2，它们共同指向的文件内容才会消失，也就是那个 inode 才会被删去。

```
oscar@oscar-laptop: ~/test
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ mkdir test
oscar@oscar-laptop:~$ cd test
oscar@oscar-laptop:~/test$ touch file1
oscar@oscar-laptop:~/test$ ls
file1
oscar@oscar-laptop:~/test$ ln file1 file2
oscar@oscar-laptop:~/test$ ls -l
total 0
-rw-r--r-- 2 oscar oscar 0 May  3 22:32 file1
-rw-r--r-- 2 oscar oscar 0 May  3 22:32 file2
oscar@oscar-laptop:~/test$ ls -li
655571 file1 655571 file2
oscar@oscar-laptop:~/test$ rm file2
oscar@oscar-laptop:~/test$ ls -l
total 0
-rw-r--r-- 1 oscar oscar 0 May  3 22:32 file1
oscar@oscar-laptop:~/test$
```

我们用 `ls -li` 命令查看文件信息的时候，第二列的那个 2，其实是表示拥有相同 inode 号的文件数。不难理解，因为它们指向相同的文件内容，所以共享一个 inode。

这个第二列的数字，对于普通文件，一般来说是 1，因为不同文件 inode 不同嘛。对于目录来说，这第二列的数字标明目录内所含文件数目。

创建软链接

其实，软链接才是真正像我们在 Windows 下的快捷方式的，其原理很相似。

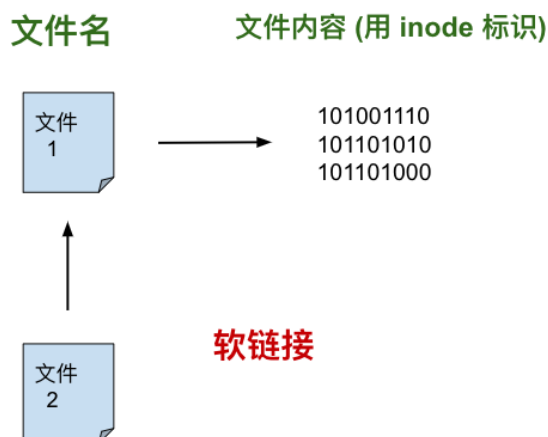
创建硬链接时 `ln` 命令不带任何参数，但是要创建软链接需要加上 `-s` 参数。s 是 symbolic（符号的）的缩写。

因为之前我们的 `test` 目录中，我们已经删除了 `file2` 这个文件。那么我们来创建 `file1` 的软链接吧，还是命名为 `file2` 好了。

```
ln -s file1 file2
```

以上命令创建了 `file1` 的软链接 `file2`。

软链接原理图：



可以看到上图中，file2 指向的不再是 file1 的文件内容（和硬链接不同），而是指向 file1 的文件名。

用 `ls -l` 命令查看一下，会发现形式和之前的硬链接不一样噢。file2 的信息是这样的：file2->file1，表示 file2 指向 file1。

file2 的文件信息里，第一列表示权限的第一个字母变成了 l，表示 link（链接）。之前硬链接的时候是没有 l 的，硬链接外表看起来就和普通文件类似。file2 的颜色是浅蓝色，也说明是链接文件。

我们用 `ls -li` 命令查看文件信息的时候，第二列的那个 1，表示拥有相同 inode 号的文件数。不难理解，因为 file2 指向 file1，它们并没有指向同一块文件内容，所以它们的 inode 号不相同。

用 `ls -i` 看一下就知道了：

file1 的 inode 号仍然是 655571。而 file2 的 inode 号是 655574，不一样。

打开 file2 查看，发现和 file1 是一样的。这不难理解，因为 file2 这个软链接只是 file1 的一个快捷方式，它指向的是 file1，所以显示的是 file1 的内容。但其实它自身的 inode 和 file1 并不一样，也就是文件内容不一样。

软链接的特点：

- 如果我们删除了 file2，没什么大不了，file1 不会受到影响。但是如果删除了 file1，那么 file2 会变成“死链接”，因为指向的文件不见了。
- 软链接可以指向目录，硬链接不行。

```
oscar@oscar-laptop: ~/test
File Edit View Search Terminal Help
oscar@oscar-laptop:~/test$ ls
file1
oscar@oscar-laptop:~/test$ ln -s file1 file2
oscar@oscar-laptop:~/test$ ls -l
total 0
-rw-r--r-- 1 oscar oscar 0 May  3 22:32 file1
lrwxrwxrwx 1 oscar oscar 5 May  3 22:40 file2 -> file1
oscar@oscar-laptop:~/test$ ls -li
655571 file1 655574 file2
oscar@oscar-laptop:~/test$ rm file1
oscar@oscar-laptop:~/test$ ls -l
total 0
lrwxrwxrwx 1 oscar oscar 5 May  3 22:40 file2 -> file1
oscar@oscar-laptop:~/test$
```

可以看到，一旦删除了 file1，那么 file2 就变成了红色，表明此软链接已经损坏。

因为 file2 指向的文件 file1 已经被删除，file2 不知道要指向哪里了。“路漫漫其修远兮，吾将上下而求索”。但“索”即不存，吾将何去何从？可怜的 file2 已经失去“人生目标”了。

无论环境如何，希望大家永远不要失去人生的目标呀。

硬链接和软链接的知识，我们大体上讲完了。也许还是有点晕，不过可以参考一些课外读物，加深理解。自己动手做做实验，就会慢慢理解了。

小结

1. `cp` 命令用于拷贝文件或目录。`mv` 命令用于移动文件或目录，也可以为文件重命名；
2. `rm` 命令用于删除文件或目录。记住：终端里可没有“回收站”，所以删除前要谨慎考虑；
3. 我们使用 `ln` 命令，可以创建指向文件的链接（类似 Windows 的快捷方式，但比快捷方式复杂）。

今天的课就到这里，一起加油吧！



12 文件操纵，鼓掌之中。浏览和
创建文件

14 用户和权限，有权就任性。用
户管理

