



### 重要

开头为句点(.) 的文件名表示隐藏文件；在使用 **ls** 和其他命令时，您无法在普通视图中看到这些文件。这不是一种安全功能。隐藏文件用于防止必要的用户配置文件让主目录凌乱不堪。许多命令仅通过特定的命令行选项处理隐藏文件，这样可以预防一个用户的配置被意外复制到其他目录或其他用户。

要防止文件内容被不当查看，需要使用文件权限。

```
[user@host ~]$ ls -R
.:
Desktop Documents Downloads Music Pictures Public Templates Videos

./Desktop:

./Documents:
thesis_chapter1.odf thesis_chapter2.odf

./Downloads:

./Music:

./Pictures:

./Public:

./Templates:

./Videos:
blockbuster1.ogg blockbuster2.ogg
[user@host ~]$
```

**cd** 命令有许多选项。其中一些非常有用，值得您尽早练习和经常使用。**cd -** 命令可更改到用户在进入当前目录之前所处的目录。以下示例演示了这一行为，在两个目录之间交替，这在处理一系列类似任务时很有用。

```
[user@host ~]$ cd Videos
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd /home/user/Documents
[user@host Documents]$ pwd
/home/user/Documents
[user@host Documents]$ cd -
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd -
[user@host Documents]$ pwd
/home/user/Documents
[user@host Documents]$ cd -
[user@host Videos]$ pwd
```

```
/home/user/Videos  
[user@host Videos]$ cd  
[user@host ~]$
```

**cd ..** 命令使用 .. 隐藏目录上移一个级别，进入其父目录，而不必知道确切的父目录名称。其他隐藏目录(.) 可为当前位置是来源或目标参数的命令指定当前目录，以此免除键入目录绝对路径名的必要。

```
[user@host Videos]$ pwd  
/home/user/Videos  
[user@host Videos]$ cd .  
[user@host Videos]$ pwd  
/home/user/Videos  
[user@host Videos]$ cd ..  
[user@host ~]$ pwd  
/home/user  
[user@host ~]$ cd ..  
[user@host home]$ pwd  
/home  
[user@host home]$ cd ..  
[user@host /]$ pwd  
/  
[user@host /]$ cd  
[user@host ~]$ pwd  
/home/user  
[user@host ~]$
```



## 参考文献

**info libc 'file name resolution'** (GNU C 库参考手册)

· 第 11.2.2 节：文件名称解析

**bash(7)、cd(1)、ls(1)、pwd(1)、unicode(1) 和 utf-8(7) man page**

## UTF-8 和 Unicode

<http://www.utf-8.com/>

## ► 小测验

### 通过名称指定文件

选择以下问题的正确答案：

- 1. 假设当前用户的当前工作目录为 /tmp，主目录为 /home/user，哪一个命令用于返回到其主目录？
- a. **cd**
  - b. **cd ..**
  - c. **cd .**
  - d. **cd \***
  - e. **cd /home**
- 2. 哪一个命令可显示当前位置的绝对路径名？
- a. **cd**
  - b. **pwd**
  - c. **ls ~**
  - d. **ls -d**
- 3. 哪一个命令会始终返回到在进入当前工作目录前所使用的工作目录？
- a. **cd -**
  - b. **cd -p**
  - c. **cd ~**
  - d. **cd ..**
- 4. 哪一个命令会始终将工作目录从当前位置上移两个级别？
- a. **cd ~**
  - b. **cd ../**
  - c. **cd ../../..**
  - d. **cd -u2**
- 5. 哪一个命令使用长格式列出当前位置的文件，并包含隐藏的文件？
- a. **llong ~**
  - b. **ls -a**
  - c. **ls -l**
  - d. **ls -al**

► 6. 哪一个命令会始终将工作目录切换到 /bin?

- a. `cd bin`
- b. `cd /bin`
- c. `cd ~bin`
- d. `cd -bin`

► 7. 哪一个命令会始终将工作目录切换到当前位置的父目录?

- a. `cd ~`
- b. `cd ..`
- c. `cd .../..`
- d. `cd -u1`

► 8. 如果当前工作目录为 /home/student, 哪一个命令会将工作目录切换到 /tmp?

- a. `cd tmp`
- b. `cd ..`
- c. `cd .../.../tmp`
- d. `cd ~tmp`

## ► 解决方案

### 通过名称指定文件

选择以下问题的正确答案：

- 1. 假设当前用户的当前工作目录为 /tmp，主目录为 /home/user，哪一个命令用于返回到其主目录？
- a. **cd**
  - b. **cd ..**
  - c. **cd .**
  - d. **cd \***
  - e. **cd /home**
- 2. 哪一个命令可显示当前位置的绝对路径名？
- a. **cd**
  - b. **pwd**
  - c. **ls ~**
  - d. **ls -d**
- 3. 哪一个命令会始终返回到在进入当前工作目录前所使用的工作目录？
- a. **cd -**
  - b. **cd -p**
  - c. **cd ~**
  - d. **cd ..**
- 4. 哪一个命令会始终将工作目录从当前位置上移两个级别？
- a. **cd ~**
  - b. **cd ../**
  - c. **cd ../../**
  - d. **cd -u2**
- 5. 哪一个命令使用长格式列出当前位置的文件，并包含隐藏的文件？
- a. **llong ~**
  - b. **ls -a**
  - c. **ls -l**
  - d. **ls -al**

► 6. 哪一个命令会始终将工作目录切换到 /bin?

- a. `cd bin`
- b. `cd /bin`
- c. `cd ~bin`
- d. `cd -bin`

► 7. 哪一个命令会始终将工作目录切换到当前位置的父目录?

- a. `cd ~`
- b. `cd ..`
- c. `cd ./. .`
- d. `cd -u1`

► 8. 如果当前工作目录为 /home/student, 哪一个命令会将工作目录切换到 /tmp?

- a. `cd tmp`
- b. `cd ..`
- c. `cd ../../tmp`
- d. `cd ~tmp`

# 使用命令行工具管理文件

---

## 培训目标

学完本节后，您应能够创建、复制、移动和删除文件与目录。

## 命令行文件管理

要管理文件，您需要能创建、删除、复制和移动文件。此外，您还需要按逻辑关系将它们整理到目录中，这就需要您能创建、删除、复制和移动目录。

下表总结了一些最常见的文件管理命令。本节的其余部分将详细讨论这些命令的使用方式。

### 常用文件管理命令

活动	命令语法
创建目录	<code>mkdir directory</code>
复制文件	<code>cp file new-file</code>
复制目录及其内容	<code>cp -r directory new-directory</code>
移动或重命名文件或目录	<code>mv file new-file</code>
删除文件	<code>rm file</code>
删除含有文件的目录	<code>rm -r directory</code>
删除空目录	<code>rmdir directory</code>

## 创建目录

`mkdir` 命令可创建一个或多个目录或子目录。它取您要创建的目录的路径列表作为参数。

如果该目录已存在，或者您试图在一个不存在的目录中创建子目录，`mkdir` 命令将失败并出现错误。`-p`（父级）选项将为请求的目标位置创建缺失的父目录。请谨慎使用`mkdir -p` 该命令，因为拼写错误可能会创建非预期的目录，而不会生成错误消息。

在以下示例中，假设您试图在名为 **Watched** 的 **Videos** 目录中创建一个目录，但在 `mkdir` 命令中不小心遗漏了 **Videos** 中的字母“s”。

```
[user@host ~]$ mkdir Video/Watched
mkdir: cannot create directory `Video/Watched': No such file or directory
```

`mkdir` 命令失败，因为将 **Videos** 拼写错误，而且目录 **Video** 也不存在。如果您使用了 `mkdir` 命令和 `-p` 选项，此时将会创建并非您本意的目录 **Video**，并且在这个错误的目录中创建子目录 **Watched**。

正确拼写 **Videos** 父目录后，会成功地创建 **Watched** 子目录。

```
[user@host ~]$ mkdir Videos/Watched  
[user@host ~]$ ls -R Videos  
Videos/:  
blockbuster1.ogg blockbuster2.ogg Watched  
  
Videos/Watched:
```

在以下示例中，文件和目录组织在 **/home/user/Documents** 目录下。使用 **mkdir** 命令和空格分隔的目录名称列表来创建多个目录。

```
[user@host ~]$ cd Documents  
[user@host Documents]$ mkdir ProjectX ProjectY  
[user@host Documents]$ ls  
ProjectX ProjectY
```

使用 **mkdir -p** 命令和各个子目录名称的空格分隔相对路径来创建多个父目录及子目录。

```
[user@host Documents]$ mkdir -p Thesis/Chapter1 Thesis/Chapter2 Thesis/Chapter3  
[user@host Documents]$ cd  
[user@host ~]$ ls -R Videos Documents  
Documents:  
ProjectX ProjectY Thesis  
  
Documents/ProjectX:  
  
Documents/ProjectY:  
  
Documents/Thesis:  
Chapter1 Chapter2 Chapter3  
  
Documents/Thesis/Chapter1:  
  
Documents/Thesis/Chapter2:  
  
Documents/Thesis/Chapter3:  
  
Videos:  
blockbuster1.ogg blockbuster2.ogg Watched  
  
Videos/Watched:
```

最后一个 **mkdir** 命令通过一个命令创建了三个 ChapterN 子目录。**-p** 选项创建了缺少的父目录 **Thesis**。

## 复制文件

**cp** 命令可复制文件，在当前目录或指定目录中创建新文件。它也可将多个文件复制到某一目录中。

**警告**

如果目标文件已存在，则 **cp** 命令会覆盖该文件。

```
[user@host ~]$ cd Videos
[user@host Videos]$ cp blockbuster1.ogg blockbuster3.ogg
[user@host Videos]$ ls -l
total 0
-rw-rw-r--. 1 user user    0 Feb  8 16:23 blockbuster1.ogg
-rw-rw-r--. 1 user user    0 Feb  8 16:24 blockbuster2.ogg
-rw-rw-r--. 1 user user    0 Feb  8 16:34 blockbuster3.ogg
drwxrwxr-x. 2 user user 4096 Feb  8 16:05 Watched
[user@host Videos]$
```

在通过一个命令复制多个文件时，最后一个参数必须为目录。复制的文件在新的目录中保留其原有名称。如果目标目录中存在具有相同名称的文件，则会覆盖现有文件。默认情况下，**cp** 不复制目录，而会忽略它们。

下例中列出了两个目录，**Thesis** 和 **ProjectX**。只有最后参数 **ProjectX** 可以有效作为目标。**Thesis** 目录被忽略。

```
[user@host Videos]$ cd .. /Documents
[user@host Documents]$ cp thesis_chapter1.odf thesis_chapter2.odf Thesis ProjectX
cp: omitting directory `Thesis'
[user@host Documents]$ ls Thesis ProjectX
ProjectX:
thesis_chapter1.odf  thesis_chapter2.odf

Thesis:
Chapter1  Chapter2  Chapter3
```

在第一个 **cp** 命令中，**Thesis** 目录未能被复制，但 **thesis\_chapter1.odf** 和 **thesis\_chapter2.odf** 文件复制成功。

如果想向当前工作目录中复制一个文件，您可以使用特殊目录 **.**：

```
[user@host ~]$ cp /etc/hostname .
[user@host ~]$ cat hostname
host.example.com
[user@host ~]$
```

使用 **-r**（递归）选项运行复制命令，将 **Thesis** 目录及其内容复制到 **ProjectX** 目录。

```
[user@host Documents]$ cp -r Thesis ProjectX
[user@host Documents]$ ls -R ProjectX
ProjectX:
Thesis  thesis_chapter1.odf  thesis_chapter2.odf

ProjectX/Thesis:
Chapter1  Chapter2  Chapter3
```

```
ProjectX/Thesis/Chapter1:
```

```
ProjectX/Thesis/Chapter2:  
thesis_chapter2.odf
```

```
ProjectX/Thesis/Chapter3:
```

## 移动文件

**mv** 命令可将文件从一个位置移动到另一个位置。如果您将文件的绝对路径视为它的全名，那么移动文件实际上和重命名文件一样。文件内容保持不变。

使用 **mv** 命令重命名文件。

```
[user@host Videos]$ cd ..\Documents  
[user@host Documents]$ ls -l thesis*  
-rw-rw-r--. 1 user user 0 Feb 6 21:16 thesis_chapter1.odf  
-rw-rw-r--. 1 user user 0 Feb 6 21:16 thesis_chapter2.odf  
[user@host Documents]$ mv thesis_chapter2.odf thesis_chapter2_reviewed.odf  
[user@host Documents]$ ls -l thesis*  
-rw-rw-r--. 1 user user 0 Feb 6 21:16 thesis_chapter1.odf  
-rw-rw-r--. 1 user user 0 Feb 6 21:16 thesis_chapter2_reviewed.odf
```

使用 **mv** 命令将文件移动到另一个目录。

```
[user@host Documents]$ ls Thesis/Chapter1  
[user@host Documents]$  
[user@host Documents]$ mv thesis_chapter1.odf Thesis/Chapter1  
[user@host Documents]$ ls Thesis/Chapter1  
thesis_chapter1.odf  
[user@host Documents]$ ls -l thesis*  
-rw-rw-r--. 1 user user 0 Feb 6 21:16 thesis_chapter2_reviewed.odf
```

## 删除文件和目录

**rm** 命令可删除文件。默认情况下，除非添加了 **-r** 或 **--recursive** 选项，否则 **rm** 不会删除包含文件的目录。



### 重要

没有命令行取消删除功能，也没有可从中恢复暂存删除文件的“垃圾箱”。

在删除文件或目录之前，最好先验证您的当前工作目录。

```
[user@host Documents]$ pwd  
/home/student/Documents
```

使用 **rm** 命令从工作目录中删除单个文件。

```
[user@host Documents]$ ls -l thesis*
-rw-rw-r--. 1 user user 0 Feb  6 21:16 thesis_chapter2_reviewed.odf
[user@host Documents]$ rm thesis_chapter2_reviewed.odf
[user@host Documents]$ ls -l thesis*
ls: cannot access 'thesis*': No such file or directory
```

如果您试图使用 **rm** 命令来删除目录，但不使用 **-r** 选项，命令将失败。

```
[user@host Documents]$ rm Thesis/Chapter1
rm: cannot remove `Thesis/Chapter1': Is a directory
```

使用 **rm -r** 命令删除子目录及其内容。

```
[user@host Documents]$ ls -R Thesis
Thesis/:
Chapter1 Chapter2 Chapter3

Thesis/Chapter1:
thesis_chapter1.odf

Thesis/Chapter2:
thesis_chapter2.odf

Thesis/Chapter3:
[user@host Documents]$ rm -r Thesis/Chapter1
[user@host Documents]$ ls -l Thesis
total 8
drwxrwxr-x. 2 user user 4096 Feb 11 12:47 Chapter2
drwxrwxr-x. 2 user user 4096 Feb 11 12:48 Chapter3
```

**rm -r** 命令首先遍历每个子目录，在删除每个目录之前逐一删除其中的文件。您可以使用 **rm -ri** 命令以交互方式提示确认，然后再删除。这基本上与使用 **-f** 选项相反，后者强制删除而不提示用户进行确认。

```
[user@host Documents]$ rm -ri Thesis
rm: descend into directory `Thesis'? y
rm: descend into directory `Thesis/Chapter2'? y
rm: remove regular empty file `Thesis/Chapter2/thesis_chapter2.odf'? y
rm: remove directory `Thesis/Chapter2'? y
rm: remove directory `Thesis/Chapter3'? y
rm: remove directory `Thesis'? y
[user@host Documents]$
```



### 警告

如果您同时指定了 **-i** 和 **-f** 选项，**-f** 选项将具有优先权，在 **rm** 删除文件之前，不会提示您进行确认。

在以下示例中，**rmdir** 命令仅删除空目录。就像前面的示例一样，您必须使用 **rm -r** 命令来删除包含内容的目录。

```
[user@host Documents]$ pwd  
/home/student/Documents  
[user@host Documents]$ rmdir ProjectY  
[user@host Documents]$ rmdir ProjectX  
rmdir: failed to remove `ProjectX': Directory not empty  
[user@host Documents]$ rm -r ProjectX  
[user@host Documents]$ ls -lR  
.:  
total 0  
[user@host Documents]$
```



### 注意

不带任何选项的 **rm** 命令无法删除空目录。您必须使用 **rmdir** 命令、**rm -d**（等同于 **rmdir**）或 **rm -r**。



### 参考文献

**cp(1)**、**mkdir(1)**、**mv(1)**、**rm(1)** 和 **rmdir(1)** man page

## ► 指导练习

# 使用命令行工具管理文件

在本练习中，您将创建、组织、复制和删除文件与目录。

## 成果

您应能够创建、组织、复制和删除文件与目录。

## 在你开始之前

在 workstation 上，以 student 用户身份并使用密码 student 进行登录。

在 workstation 上，运行 **lab files-manage start** 命令。此命令将运行一个起始脚本，它将确定 servera 计算机是否可从网络访问。

```
[student@workstation ~]$ lab files-manage start
```

- 1. 使用 **ssh** 命令，以 student 用户身份登录 servera。系统已配置为使用 SSH 密钥来进行身份验证，因此不需要提供密码。

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. 在 student 用户的主目录中，使用 **mkdir** 命令创建三个子目录：**Music**、**Pictures** 和 **Videos**。

```
[student@servera ~]$ mkdir Music Pictures Videos
```

- 3. 继续在 student 用户的主目录中，使用 **touch** 命令创建本实验过程中要使用的一套空白的练习文件。

- 创建六个文件，并以 **songX.mp3** 形式取名。
- 创建六个文件，并以 **snapX.jpg** 形式取名。
- 创建六个文件，并以 **filmX.avi** 形式取名。

在每一组文件中，将 X 替换为数字 1 到 6。

```
[student@servera ~]$ touch song1.mp3 song2.mp3 song3.mp3 song4.mp3  
song5.mp3 song6.mp3  
[student@servera ~]$ touch snap1.jpg snap2.jpg snap3.jpg snap4.jpg  
snap5.jpg snap6.jpg  
[student@servera ~]$ touch film1.avi film2.avi film3.avi film4.avi  
film5.avi film6.avi  
[student@servera ~]$ ls -l  
total 0
```

```
-rw-rw-r--. 1 student student 0 Feb 4 18:23 film1.avi
-rw-rw-r--. 1 student student 0 Feb 4 18:23 film2.avi
-rw-rw-r--. 1 student student 0 Feb 4 18:23 film3.avi
-rw-rw-r--. 1 student student 0 Feb 4 18:23 film4.avi
-rw-rw-r--. 1 student student 0 Feb 4 18:23 film5.avi
-rw-rw-r--. 1 student student 0 Feb 4 18:23 film6.avi
drwxrwxr-x. 2 student student 6 Feb 4 18:23 Music
drwxrwxr-x. 2 student student 6 Feb 4 18:23 Pictures
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap1.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap2.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap3.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap4.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap5.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap6.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song1.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song2.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song3.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song4.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song5.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song6.mp3
drwxrwxr-x. 2 student student 6 Feb 4 18:23 Videos
```

- 4. 继续在 **student** 用户的主目录中，将歌曲文件移动到 **Music** 子目录，将快照文件移到 **Pictures** 子目录，并将影片文件移到 **Videos** 子目录。

将文件从一个位置分发到多个位置时，首先更改到包含源文件的目录。使用最简单的路径语法（绝对或相对路径），到达各项文件管理任务的目标位置。

```
[student@servera ~]$ mv song1.mp3 song2.mp3 song3.mp3 song4.mp3 \
song5.mp3 song6.mp3 Music
[student@servera ~]$ mv snap1.jpg snap2.jpg snap3.jpg snap4.jpg \
snap5.jpg snap6.jpg Pictures
[student@servera ~]$ mv film1.avi film2.avi film3.avi film4.avi \
film5.avi film6.avi Videos
[student@servera ~]$ ls -l Music Pictures Videos
Music:
total 0
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song1.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song2.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song3.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song4.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song5.mp3
-rw-rw-r--. 1 student student 0 Feb 4 18:23 song6.mp3

Pictures:
total 0
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap1.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap2.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap3.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap4.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap5.jpg
-rw-rw-r--. 1 student student 0 Feb 4 18:23 snap6.jpg

Videos:
```

```
total 0
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film1.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film2.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film3.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film4.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film5.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film6.avi
```

- 5. 继续在 **student** 用户的主目录中，创建三个子目录，以便将文件整理到项目中。将这些子目录命名为 **friends**、**family** 和 **work**。使用单个命令一次性创建所有三个子目录。

您将使用这些目录把文件重新整理到项目中。

```
[student@servera ~]$ mkdir friends family work
[student@servera ~]$ ls -l
total 0
drwxrwxr-x. 2 student student 6 Feb  4 18:38 family
drwxrwxr-x. 2 student student 6 Feb  4 18:38 friends
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Music
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Pictures
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Videos
drwxrwxr-x. 2 student student 6 Feb  4 18:38 work
```

- 6. 将选定的新文件复制到项目目录 **family** 和 **friends** 中。根据需要，使用多个命令。不必像示例中那样仅使用一个命令。对于每个项目，首先更改到项目目录，然后将源文件复制到此目录中。请记住，您正在制作副本，因此在将文件复制到项目目录后，原始文件会保留在其原始位置。

- 将含有数字 1 和 2 的文件（所有类型）复制到 **friends** 子目录。
- 将含有数字 3 和 4 的文件（所有类型）复制到 **family** 子目录。

在将文件从多个位置复制到一个位置时，红帽建议您在复制文件之前更改到目标目录。使用最简单的路径语法（绝对或相对路径），到达各项文件管理任务的源位置。

```
[student@servera ~]$ cd friends
[student@servera friends]$ cp ~/Music/song1.mp3 ~/Music/song2.mp3 \
~/Pictures/snap1.jpg ~/Pictures/snap2.jpg ~/Videos/film1.avi \
~/Videos/film2.avi .
[student@servera friends]$ ls -l
total 0
-rw-rw-r--. 1 student student 0 Feb  4 18:42 film1.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:42 film2.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:42 snap1.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:42 snap2.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:42 song1.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:42 song2.mp3
[student@servera friends]$ cd ../family
[student@servera family]$ cp ~/Music/song3.mp3 ~/Music/song4.mp3 \
~/Pictures/snap3.jpg ~/Pictures/snap4.jpg ~/Videos/film3.avi \
~/Videos/film4.avi .
[student@servera family]$ ls -l
total 0
-rw-rw-r--. 1 student student 0 Feb  4 18:44 film3.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:44 film4.avi
```

```
-rw-rw-r--. 1 student student 0 Feb  4 18:44 snap3.jpg  
-rw-rw-r--. 1 student student 0 Feb  4 18:44 snap4.jpg  
-rw-rw-r--. 1 student student 0 Feb  4 18:44 song3.mp3  
-rw-rw-r--. 1 student student 0 Feb  4 18:44 song4.mp3
```

► 7. 对于您的工作项目，创建额外的副本。

```
[student@servera family]$ cd ../work  
[student@servera work]$ cp ~/Music/song5.mp3 ~/Music/song6.mp3 \  
~/Pictures/snap5.jpg ~/Pictures/snap6.jpg \  
~/Videos/film5.avi ~/Videos/film6.avi .  
[student@servera work]$ ls -l  
total 0  
-rw-rw-r--. 1 student student 0 Feb  4 18:48 film5.avi  
-rw-rw-r--. 1 student student 0 Feb  4 18:48 film6.avi  
-rw-rw-r--. 1 student student 0 Feb  4 18:48 snap5.jpg  
-rw-rw-r--. 1 student student 0 Feb  4 18:48 snap6.jpg  
-rw-rw-r--. 1 student student 0 Feb  4 18:48 song5.mp3  
-rw-rw-r--. 1 student student 0 Feb  4 18:48 song6.mp3
```

► 8. 您的项目任务现已完成，是时候清理项目了。

更改到 **student** 用户的主目录。尝试通过一个 **rmdir** 命令同时删除 **family** 和 **friends** 项目目录。

```
[student@servera work]$ cd  
[student@servera ~]$ rmdir family friends  
rmdir: failed to remove 'family': Directory not empty  
rmdir: failed to remove 'friends': Directory not empty
```

使用 **rmdir** 命令应该会失败，因为两个子目录都包含文件。

► 9. 使用 **rm -r** 命令以递归方式删除 **family** 和 **friends** 子目录及其内容。

```
[student@servera ~]$ rm -r family friends  
[student@servera ~]$ ls -l  
total 0  
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Music  
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Pictures  
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Videos  
drwxrwxr-x. 2 student student 108 Feb  4 18:48 work
```

► 10. 删除工作项目中的所有文件，但不要删除 **work** 目录本身。

```
[student@servera ~]$ cd work  
[student@servera work]$ rm song5.mp3 song6.mp3 snap5.jpg snap6.jpg \  
film5.avi film6.avi  
[student@servera work]$ ls -l  
total 0
```

- 11. 最后，从 `student` 用户的主目录，使用 `rmdir` 命令删除 `work` 目录。该命令现在应该成功完成，因为目录已为空。

```
[student@servera work]$ cd  
[student@servera ~]$ rmdir work  
[student@servera ~]$ ls -l  
total 0  
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Music  
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Pictures  
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Videos
```

- 12. 从 `servera` 退出。

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

## 完成

在 `workstation` 上，运行 `lab files-manage finish` 脚本来结束本练习。此脚本将删除本练习过程中创建的所有文件和目录。

```
[student@workstation ~]$ lab files-manage finish
```

本引导式练习到此结束。

# 制作文件间的链接

## 培训目标

学完本节后，您应能够使用硬链接和符号（或“软”）链接，使多个文件名引用同一文件。

## 管理文件间的链接

### 硬链接和软链接

可以创建指向同一文件的多个名称。有两种方法可以做到这一点：创建一个指向文件的硬链接，或创建一个指向文件的软链接（有时也称为符号链接）。每种方法都各有利弊。

### 创建硬链接

从初始名称到文件系统上的数据，每个文件都以一个硬链接开始。当创建指向文件的新硬链接时，也会创建另一个指向同一数据的名称。新的硬链接与原始文件名的作用完全相同。一经创建，新的硬链接与文件的原始名称便毫无二致。

您可以通过 **ls -l** 命令来确定某个文件是否有多个硬链接。它报告的内容之一就是每个文件的链接数，即文件所具有的硬链接数。

```
[user@host ~]$ pwd  
/home/user  
[user@host ~]$ ls -l newfile.txt  
-rw-r--r--. 1 user user 0 Mar 11 19:19 newfile.txt
```

在上面的示例中，**newfile.txt** 的链接数为 1。它正好有一个绝对路径，即 **/home/user/newfile.txt**。

您可以使用 **ln** 命令来创建一个指向现有文件的新硬链接（另一个名称）。该命令至少需要两个参数，即现有文件的路径以及要创建的硬链接的路径。

以下示例将在 **/tmp** 目录中为现有文件 **newfile.txt** 创建硬链接 **newfile-link2.txt**。

```
[user@host ~]$ ln newfile.txt /tmp/newfile-hlink2.txt  
[user@host ~]$ ls -l newfile.txt /tmp/newfile-hlink2.txt  
-rw-rw-r--. 2 user user 12 Mar 11 19:19 newfile.txt  
-rw-rw-r--. 2 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
```

如果您想知道两个文件是否为彼此的硬链接，一种方法是对 **ls** 命令使用 **-i** 选项，以列出文件的索引节点编号。如果文件位于同一文件系统上（稍后讨论），而且它们的索引节点编号相同，那么这两个文件就是指向同一数据的硬链接。

```
[user@host ~]$ ls -il newfile.txt /tmp/newfile-hlink2.txt  
8924107 -rw-rw-r--. 2 user user 12 Mar 11 19:19 newfile.txt  
8924107 -rw-rw-r--. 2 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
```

**重要**

引用同一文件的所有硬链接将具有相同的链接数、访问权限、用户和组所有权、时间戳，以及文件内容。如果使用一个硬链接更改其中的任何信息，指向同一文件的所有其他硬链接也会显示新的信息。这是因为每个硬链接都指向存储设备上的同一数据。

即使原始文件被删除，只要存在至少一个硬链接，该文件的内容就依然可用。只有删除了最后一个硬链接时，才会将数据从存储中删除。

```
[user@host ~]$ rm -f newfile.txt
[user@host ~]$ ls -l /tmp/newfile-hlink2.txt
-rw-rw-r--. 1 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
[user@host ~]$ cat /tmp/newfile-hlink2.txt
Hello World
```

## 硬链接的局限性

硬链接存在一些局限性。首先，硬链接只能用于常规文件。您不能使用 **ln** 来创建指向目录或特殊文件的硬链接。

其次，只有当两个文件都位于同一文件系统上时，才能使用硬链接。文件系统层次结构可以由多个存储设备组成。当切换到新目录时，该目录及其内容可能会存储在不同的文件系统中，具体取决于系统配置。

您可以使用 **df** 命令来列出位于不同文件系统上的目录。例如，输出可能如下所示：

```
[user@host ~]$ df
Filesystem      1K-blocks   Used   Available Use% Mounted on
devtmpfs          886788     0    886788   0% /dev
tmpfs            902108     0    902108   0% /dev/shm
tmpfs            902108   8696   893412   1% /run
tmpfs            902108     0    902108   0% /sys/fs/cgroup
/dev/mapper/rhel_rhel8--root  10258432 1630460   8627972  16% /
/dev/sda1        1038336  167128   871208  17% /boot
tmpfs           180420     0    180420   0% /run/user/1000
[user@host ~]$
```

两个不同“Mounted on”目录及其子目录中的文件位于不同的文件系统上。（最为具体的匹配胜出。）因此，就本例中的系统而言，您可以在 **/var/tmp/link1** 和 **/home/user/file** 之间创建一个硬链接，因为它们都是 **/** 的子目录，而不是列表中其他任何目录的子目录。但是，您不能在 **/boot/test/badlink** 和 **/home/user/file** 之间建立硬链接，因为第一个文件位于 **/boot** 的子目录中（在“Mounted on”列表中），而第二个文件不在该子目录中。

## 创建软链接

**ln -s** 命令可创建软链接，也称为“符号链接”。软链接不是常规文件，而是指向现有文件或目录的特殊类型的文件。

软链接相比硬链接有一定的优势：

- 它们可以链接位于不同文件系统上的两个文件。
- 它们可以指向目录或特殊文件，而不仅限于常规文件。

在以下示例中，**ln -s** 命令将用于为现有文件 **/home/user/newfile-link2.txt** 创建新的软链接，并命名为 **/tmp/newfile-symlink.txt**。

```
[user@host ~]$ ln -s /home/user/newfile-link2.txt /tmp/newfile-symlink.txt
[user@host ~]$ ls -l newfile-link2.txt /tmp/newfile-symlink.txt
-rw-rw-r--. 1 user user 12 Mar 11 19:19 newfile-link2.txt
lrwxrwxrwx. 1 user user 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /home/user/
newfile-link2.txt
[user@host ~]$ cat /tmp/newfile-symlink.txt
Soft Hello World
```

在上面的示例中，**/tmp/newfile-symlink.txt** 的长列表的第一个字符是 **l** 而非 **-**。这表示该文件是软链接而不是常规文件。（**d** 表示该文件是一个目录。）

当原始常规文件被删除后，软链接依然会指向该文件，但目标已消失。指向缺失的文件的软链接称为“悬挂的软链接”。

```
[user@host ~]$ rm -f newfile-link2.txt
[user@host ~]$ ls -l /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 user user 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /home/user/
newfile-link2.txt
[user@host ~]$ cat /tmp/newfile-symlink.txt
cat: /tmp/newfile-symlink.txt: No such file or directory
```



### 重要

在上例中，悬挂的软链接有一个副作用，那就是：如果您稍后创建了一个与已删除文件同名的新文件（**/home/user/newfile-link2.txt**），那么软链接将不再“悬挂”，而是指向此新文件。

硬链接的工作方式却不是这样。如果您删除硬链接，然后使用常规工具（而不是 **ln**）创建一个同名的新文件，那么新文件将不会链接到旧文件。

有一种比较硬链接和软链接的方法可能会帮助您了解它们的工作原理：

- 硬链接是将名称指向存储设备上的数据
- 软链接则是将名称指向另一个名称，后者指向存储设备上的数据

软链接可以指向目录。而后，软链接发挥目录一样的作用。通过 **cd** 更改为软链接将使当前工作目录变为链接目录。有些工具可以跟踪您使用软链接到达当前工作目录的事实。例如，默认情况下，**cd** 将使用软链接的名称（而非实际目录的名称）来更新当前工作目录。（有一个选项 (**-P**) 会将其更新为实际目录的名称。）

在以下示例中，将创建名为 **/home/user/configfiles** 的软链接，它指向 **/etc** 目录。

```
[user@host ~]$ ln -s /etc /home/user/configfiles
[user@host ~]$ cd /home/user/configfiles
[user@host configfiles]$ pwd
/home/user/configfiles
```



### 参考文献

**ln(1)** man page

**info ln** (‘ln’：制作文件间的链接)

## ► 指导练习

# 制作文件间的链接

在本练习中，您将创建硬链接和符号链接并比较其结果。

## 成果

您应能够在文件之间创建硬链接和软链接。

## 在你开始之前

在 workstation 上，以 student 用户身份并使用密码 student 进行登录。

在 workstation 上，运行 **lab files-make start** 命令。该命令会运行一个起始脚本，它将确定 servera 主机是否可从网络访问，并在 servera 上创建文件和工作目录。

```
[student@workstation ~]$ lab files-make start
```

- 1. 使用 **ssh** 命令，以 student 用户身份登录 servera。系统已配置为使用 SSH 密钥来进行身份验证，因此不需要提供密码。

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. 为现有文件 **/home/student/files/source.file** 创建硬链接 **/home/student/backups/source.backup**。

2.1. 查看文件 **/home/student/files/source.file** 的链接数。

```
[student@servera ~]$ ls -l files/source.file  
total 4  
-rw-r--r--. 1 student student 11 Mar 5 21:19 source.file
```

2.2. 创建硬链接 **/home/student/backups/source.backup**。将它链接到文件 **/home/student/files/source.file**。

```
[student@servera ~]$ ln /home/student/files/source.file \  
/home/student/backups/source.backup
```

2.3. 验证原始的 **/home/student/files/source.file** 和新链接的文件 **/home/student/backups/source.backup** 的链接数。两个文件的链接数应该都是 2。

```
[student@servera ~]$ ls -l /home/student/files/  
-rw-r--r--. 2 student student 11 Mar 5 21:19 source.file  
[student@servera ~]$ ls -l /home/student/backups/  
-rw-r--r--. 2 student student 11 Mar 5 21:19 source.backup
```

► 3. 创建指向 servera 上的 /tmp 目录的软链接 /home/student/tempdir。

3.1. 创建软链接 /home/student/tempdir，并将它链接到 /tmp。

```
[student@servera ~]$ ln -s /tmp /home/student/tempdir
```

3.2. 使用 ls -l 命令验证新创建的软链接。

```
[student@servera ~]$ ls -l /home/student/tempdir  
lrwxrwxrwx. 1 student student 4 Mar 5 22:04 /home/student/tempdir -> /tmp
```

► 4. 从 servera 退出。

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

## 完成

在 workstation 上，运行 lab files-make finish 脚本来结束本练习。此脚本将删除练习过程中在 servera 上创建的所有文件和目录。

```
[student@workstation ~]$ lab files-make finish
```

本引导式练习到此结束。

# 使用 SHELL 扩展匹配文件名

## 培训目标

学完本节后，您应能够通过使用 Bash shell 的模式匹配功能，高效地运行影响很多文件的命令。

## 使用命令行扩展

Bash shell 有多种扩展命令行的方式，包括模式匹配、主目录扩展、字符串扩展和变量替换。其中最强大的或许是路径名称匹配功能，在过去被称为通配。Bash 通配功能通常称为“通配符”，可以使管理大量文件变得更加轻松。使用“扩展”的元字符来匹配要寻找的文件名和路径名，可以一次性针对集中的一组文件执行命令。

## 模式匹配

通配是一种 shell 命令解析操作，它将一个通配符模式扩展到一组匹配的路径名。在执行命令之前，命令行元字符由匹配列表替换。不返回匹配项的模式，将原始模式请求显示为字面上的文本。下列为常见的元字符和模式类。

### 元字符和匹配项表

模式	匹配项
*	由零个或更多字符组成任何字符串。
?	任何一个字符。
[abc...]	括起的类（位于两个方括号之间）中的任何一个字符。
[!abc...]	不在括起的类中的任何一个字符。
[^abc...]	不在括起的类中的任何一个字符。
[:alpha:]	任何字母字符。
[:lower:]	任何小写字符。
[:upper:]	任何大写字符。
[:alnum:]	任何字母字符或数字。
[:punct:]	除空格和字母数字以外的任何可打印字符。
[:digit:]	从 0 到 9 的任何单个数字。
[:space:]	任何一个空白字符。这可能包括制表符、换行符、回车符、换页符或空格。

对于接下来的几个示例，假定您已运行了以下命令来创建一些示例文件。

```
[user@host ~]$ mkdir glob; cd glob  
[user@host glob]$ touch alfa bravo charlie delta echo able baker cast dog easy  
[user@host glob]$ ls  
able alfa baker bravo cast charlie delta dog easy echo  
[user@host glob]$
```

第一个示例将使用简单模式匹配，通过星号 (\*) 和问号 (?) 字符以及一类字符来匹配其中一些文件名。

```
[user@host glob]$ ls a*  
able alfa  
[user@host glob]$ ls *a*  
able alfa baker bravo cast charlie delta easy  
[user@host glob]$ ls [ac]*  
able alfa cast charlie  
[user@host glob]$ ls ???  
able alfa cast easy echo  
[user@host glob]$ ls ????  
baker bravo delta  
[user@host glob]$
```

## 波形符扩展

波形符 (~) 可匹配当前用户的主目录。如果开始时使用斜杠 (/) 以外的字符串，shell 就会将该斜杠之前的字符串解译为用户名；如果存在匹配项，则用该用户的主目录绝对路径来替换此字符串。如果找不到匹配的用户名，则使用实际波形符加上该字符串代替。

在以下示例中，**echo** 命令用于显示波形符字符的值。**echo** 命令也可用于显示大括号和变量扩展字符等的值。

```
[user@host glob]$ ls ~root  
/root  
[user@host glob]$ ls ~user  
/home/user  
[user@host glob]$ ls ~/glob  
able alfa baker bravo cast charlie delta dog easy echo  
[user@host glob]$ echo ~/glob  
/home/user/glob  
[user@host glob]$
```

## 大括号扩展

大括号扩展用于生成任意字符串。大括号包含字符串的逗号分隔列表或顺序表达式。结果包含大括号定义之前或之后的文本。大括号扩展可以互相嵌套。此外，双句点语法(..) 可扩展成一个序列，使得 {m..p} 扩展为 m n o p。

```
[user@host glob]$ echo {Sunday,Monday,Tuesday,Wednesday}.log  
Sunday.log Monday.log Tuesday.log Wednesday.log  
[user@host glob]$ echo file{1..3}.txt  
file1.txt file2.txt file3.txt  
[user@host glob]$ echo file{a..c}.txt  
filea.txt fileb.txt filec.txt
```