第 3 章

文件管理与 vim 编辑器

在第 2 章中，我们初步了解了 openEuler 文件管理的一些基础知识。在这一章中，我们将深入学习 openEuler 操作系统中的文件管理。此外，本章还将介绍 vim 编辑器，vim 的使用将贯穿本书服务器配置部分的内容，学会使用 vim 对系统管理员而言至关重要。

3.1 文件管理

3.1.1 文件类型

openEuler 整个文件系统是一个树形结构，其所有的文件都位于树中，整棵树只有一个根，即根目录（/）。这里所讲的文件是一个宽泛的概念，在系统中可以理解为“一切皆文件”，也就是说普通文件、普通目录、硬件设备、程序进程、通信通道，甚至是内核的数据结构等都可以被理解成文件。总体来讲，这些都可以理解成文件，但是这些文件又被分成了不同的类型。

1. 普通文件

普通文件指的是一般意义上理解的文件，如文本文件、图片文件、MP4 视频文件等。在系统中，每个文件都有自己的属性，通过属性可以判断文件的类型。使用 ls -l 命令可以查看某个文件的属性，例如下面是查看文件 anaconda-ks.cfg 的属性的命令，该文件存放在/root 目录下，命令如下：

```
[root@server ~]# ls -l /root/anaconda-ks.cfg
-rw-------. 1 root root 1072  1月 17 06:01 /root/anaconda-ks.cfg
```

通过 ls 命令可以查看文件属性，如果属性的第 1 个符号是“-”，则表示该文件是普通文件。普通文件一般用于存放数据内容，如文本、图像、音乐等。

2. 目录文件

目录文件是表示目录的文件，也可以理解成文件夹，在目录下可以有其他文件。通过 ls -l 命令输出的目录文件的属性的第 1 个符号是 d。例如，查看 chen 目录的属性的命令如下：

```
[chen@server ~]$ ls -ld /home/chen
```

```
drwx------. 20 chen chen 4096  5月 25 17:29 /home/chen
```

为何需要目录文件？系统中的文件种类繁多，数量庞大，为了使用户方便地管理所有文件，系统需要一个良好的树形目录结构，一个目录就相当于一棵树杈，也相当于一个容器，其下又可以包含文件或目录，这样通过分级对文件进行管理。

3. 设备文件

设备文件是用于代表设备的文件，在系统中，所有的设备都被抽象成了文件，并显示在/dev 目录下，这些设备文件根据读写的粒度，又可分为块设备文件和字符设备文件。

1）块设备文件

块设备的主要特点是可以随机读写，如最常见的块设备是磁盘。块设备文件的文件属性的第 1 个符号是 b。例如，查看/dev/sda1 块设备文件的属性，命令如下：

```
[chen@server ~]$ ls -l /dev/sda1
brw-rw----. 1 root disk 8, 1  1月 17 05:23 /dev/sda1
```

2）字符设备文件

字符设备的主要特点是按顺序读写，如最常见的字符设备文件打印机和终端，它们可以接收字符流。字符设备文件的文件属性的第 1 个符号是 c。例如，查看/dev/tty5 字符设备文件的属性，命令如下：

```
[chen@server ~]$ ls -l /dev/tty5
crw--w----. 1 root tty 4, 5  1月 17 05:23 /dev/tty5
```

另外，系统中还存在一个特殊的字符设备文件/dev/null，可以理解成空设备，输出到该设备上的所有内容都会被自动丢弃。

4. 管道文件

管道文件也可以叫作 FIFO 文件，FIFO 是 First In First Out 的缩写。管道文件就是从一端流入，从另一端流出的文件，类似管道。管道文件的文件属性的第 1 个符号是 p。例如，查看 1.ref 管道文件的属性，命令如下：

```
[chen@server ~]$ ls -l /run/systemd/inhibit/1.ref
prw-------. 1 root root 0  1月 17 05:23 /run/systemd/inhibit/1.ref
```

5. 链接文件

链接文件有两种类型：软链接文件和硬链接文件。

1）软链接文件

软链接文件又叫符号链接文件，这个文件包含了另一个文件的路径名。软链接文件类似 Windows 系统中的快捷方式。在对软链接文件进行读写操作时，系统实际是对源文件进行操作，但是在删除软链接文件时，系统仅仅删除软链接文件，而不删除源文件。软链接文件的文件属性的第 1 个符号是“l”，命令如下：

```
[chen@server ~]$ ls -l /bin
```

```
lrwxrwxrwx. 1 root root 7  3月 30  2021 /bin -> usr/bin
```

2）硬链接文件

硬链接文件是已存在的一个文件的一个备份。对硬链接文件进行读写和删除操作时，结果和软链接相同，但是如果删除硬链接文件的源文件，则硬链接文件仍然存在，而且保留了原有内容，特别需要注意的是，这时系统认为它是一个普通文件，而不记得它曾是一个硬链接文件。

3.1.2 目录的创建与删除

1. 创建目录

创建目录是系统管理中常见的操作，使用 mkdir 命令可以在系统中创建空目录，mkdir 可以理解成英文 make directory 的缩写，系统中的命令一般是一些英文单词或词组的缩写，了解其含义有助于学习者记忆。

mkdir 命令语法格式：

```
mkdir [选项] [目录]
```

mkdir 命令的常用选项及含义见表 3-1。

表 3-1 mkdir 命令的常用选项及含义

选　项	含　义
-m <权限>	对新创建的目录设置权限，类似 chmod，而不是 a=rwx 减 umask
-v	每次创建新目录都显示信息
-p	需要时创建目标目录的上层目录，但即使这些目录已存在也不当作错误处理

下面通过一些示例说明 mkdir 命令的应用。

【例 3-1】 使用 mkdir 命令在当前用户的家目录下创建如图 3-1 所示的目录结构。

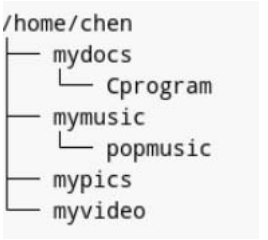

图 3-1 本题中要创建的目录结构示意图

（1）mkdir 命令可以一次建立一个或多个目录。如在用户 chen 的家目录下建立 mydocs 和 mypics 两个目录，命令如下：

```
[chen@server ~]$ mkdir mydocs mypics
//一次创建了两个目录。这两个目录使用的是相对路径写法
```

或者执行如下命令：

```
[chen@server ~]$ mkdir /home/chen/mydocs /home/chen/mypics
//一次创建了两个目录。这两个目录使用的是绝对路径写法
```

这里，以上两个命令是等价的，第 1 个命令默认为在当前目录上创建，第 2 个命令采用到了绝对路径。

用户 chen 在使用 mkdir 创建目录时，默认目录的权限是 775，因此，mypics 和 mydocs 目录的权限都是 775，可以通过如下命令查看文件的属性。

```
[chen@server ~]$ ll -d mydocs mypics
drwxrwxr-x. 3 chen chen 4.0K  5月 22 08:06 mydocs
drwxrwxr-x. 2 chen chen 4.0K  5月 22 07:56 mypics
```

（2）创建目录 Cprogram，命令如下：

```
[chen@server ~]$ mkdir ~/mydocs/Cprogram
//说明：命令中的~/mydocs/Cprogram 也可以简写成 mydocs/Cprogram
```

由于在用户 chen 的家目录中 mydocs 已经存在，因此这条命令是合法的。

（3）创建 popmusic 目录。

使用 mkdir 创建目录时需要注意其父目录是否存在，当用户运行下面这条命令时，mkdir 会提示错误。

```
[chen@server ~]$ mkdir ~/mymusic/popmusic
mkdir: 无法创建目录“/home/chen/mymusic/popmusic”：没有那个文件或目录
```

这是因为在当前用户的家目录中并不存在目录 mymusic，所以无法在其下创建 popmusic 目录，对于这个问题，可以通过下面这两种方法解决。

第 1 种方法是先用 mkdir 创建 mymusic 目录，再用 mkdir 创建 popmusic 目录，执行两条 mkdir 命令。这种方案如果路径中目录层次较多，则需要多次执行 mkdir 命令，过程有点烦琐。

第 2 种方法是使用 mkdir 的-p 选项，命令如下：

```
[chen@server ~]$ mkdir -p ~/mymusic/popmusic
//mkdir 会首先创建 mymusic 目录，然后创建 popmusic，即路径中所有不存在目录将被逐一创建
```

在需要创建一个完整的目录结构时，mkdir 命令的-p 选项非常有用。

（4）创建目录 myvideo，其权限为 777。使用-m 选项指定目录的权限，命令如下：

```
[chen@server ~]$ mkdir -m 777 myvideo
[chen@server ~]$ ll -d myvideo
drwxrwxrwx. 2 chen chen 4.0K  5月 23 08:25 myvideos
```

2. 删除目录

使用 rmdir 命令可以删除目录，rmdir 可以理解成英文 remove directory 的缩写，rmdir 命令不能删除非空目录。

rmdir 命令的语法格式：

```
rmdir [目录]
```

【例 3-2】 rmdir 命令的应用。

（1）删除一个空目录。

```
[chen@server ~]$ mkdir film
//在当前目录下新建一个目录 film
[chen@server ~]$ rmdir film
//删除该目录 film
```

（2）rmdir 只能删除空目录，不能直接删除非空目录。下面命令执行时会提示错误。

```
[chen@server ~]$ mkdir -p videos/mtv
//在当前目录下新建目录 videos，并在 videos 下新建目录 mtv
[chen@server ~]$ rmdir videos
rmdir: 删除 'videos' 失败: 目录非空
//删除目录 videos 时失败
```

那怎么办呢？针对这个例子，mtv 是个空目录，先删除 mtv 目录使 videos 成为空目录，再来删除 videos，命令如下：

```
[chen@server ~]$ rmdir videos/mtv
[chen@server ~]$ rmdir videos
//命令执行成功，videos 目录被删除
```

因此，在使用 rmdir 删除一个目录之前，首先要将该目录下的文件和子目录删除。删除文件需要用到 rm 命令。rm 命令后面再讲解，而且 rm 也可以删除目录，甚至比 rmdir 还要“效率高”。基于这个原因，在实际应用中，很多人倾向于使用 rm 删除目录。

（3）删除在例 3-1 中创建的所有目录。这里不再赘述，读者可自行完成。

3.1.3 创建空文件

touch 命令可以创建空文件或更改文件的创建日期和时间。不过在修改文件的时间属性时，用户必须是文件的所有者，或拥有写文件的访问权限。

touch 命令的语法格式：

```
touch [选项] [文件]
```

touch 命令的选项及含义见表 3-2。

表 3-2　touch 命令的选项及含义

选　项	含　义
-a	只更改访问时间
-m	只更改修改时间
-r	使用指定文件的时间属性，而非当前时间
-c	不创建新文件
-d	使用指定字符串表示时间而非当前时间
-t	使用[[CC]YY]MMDDhhmm[.ss]格式的时间戳而非当前时间

【例 3-3】 touch 命令的应用。

（1）创建单个空文件 euler.txt，命令如下：

```
[chen@server ~]$ touch euler.txt
//创建 file1.txt
[chen@server ~]$ ll euler.txt
-rw-rw-r--. 1 chen chen 0  5 月 23 09:16 file.txt
//使用 ll 命令查看文件的属性
```

（2）批量创建空文件。

批量创建 5 个空文件，命令如下：

```
[chen@server ~]$ touch testfile{1..5}
[chen@server ~]$ ll testfile?
-rw-rw-r--. 1 chen chen 0  5 月 23 09:20 testfile1
-rw-rw-r--. 1 chen chen 0  5 月 23 09:20 testfile2
-rw-rw-r--. 1 chen chen 0  5 月 23 09:20 testfile3
-rw-rw-r--. 1 chen chen 0  5 月 23 09:20 testfile4
-rw-rw-r--. 1 chen chen 0  5 月 23 09:20 testfile5
```

若要批量生成 10 个文件，并想输出 01 和 02 这样的字样，则可以使用{01..10}，命令如下：

```
[chen@server ~]$ touch testfile{01..10}
[chen@server ~]$ ll testfile??
-rw-rw-r--. 1 chen chen 0  8 月  7 09:32 testfile01
-rw-rw-r--. 1 chen chen 0  8 月  7 09:32 testfile02
-rw-rw-r--. 1 chen chen 0  8 月  7 09:32 testfile03
……
-rw-rw-r--. 1 chen chen 0  8 月  7 09:32 testfile10
```

（3）使用 touch 命令改变或更新 euler.txt 文件的访问时间。

首先使用 stat 命令查看 euler.txt 文件的时间戳，命令如下：

```
[chen@server ~]$ stat euler.txt
```

```
文件：euler.txt
大小：0             块：0         IO 块：4096   普通空文件
设备：fd00h/64768d     Inode：274180      硬链接：1
权限：(0664/-rw-rw-r--)  Uid：( 1001/    chen)   Gid：( 1001/    chen)
环境：unconfined_u:object_r:user_home_t:s0
最近访问：2022-05-23 09:16:38.959182752 +0800
最近更改：2022-05-23 09:16:38.959182752 +0800
最近改动：2022-05-23 09:38:30.264431891 +0800
创建时间：2022-05-23 09:16:38.959182752 +0800
//查看时间戳
```

接着，执行 touch 命令修改 euler.txt 文件的访问时间，命令如下：

```
[chen@server ~]$ touch -a euler.txt
```

最后，再次执行 stat 命令查看 euler.txt 文件的时间戳，验证该文件的访问时间是否已更新，命令如下：

```
[chen@server ~]$ stat euler.txt
文件：euler.txt
大小：0             块：0         IO 块：4096   普通空文件
设备：fd00h/64768d     Inode：274180      硬链接：1
权限：(0664/-rw-rw-r--)  Uid：( 1001/    chen)   Gid：( 1001/    chen)
环境：unconfined_u:object_r:user_home_t:s0
最近访问：2022-05-23 09:45:10.892021912 +0800
最近更改：2022-05-23 09:16:38.959182752 +0800
最近改动：2022-05-23 09:45:10.892021912 +0800
创建时间：2022-05-23 09:16:38.959182752 +0800
//访问时间已更新
```

（4）将文件 euler.txt 的访问时间和修改时间设定为指定时间，命令如下：

```
[chen@server ~]$ touch -c -t 202505301230 euler.txt
//将 euler.txt 文件的时间修改为 202505301230，即 2025 年 5 月 30 日 12 点 30 分
//上述时间按照格式 YYYYMMDDHHmm 书写。若 YYYY 省略，则表示使用当前年份
[chen@server ~]$ stat euler.txt
文件：euler.txt
大小：0             块：0         IO 块：4096   普通空文件
设备：fd00h/64768d     Inode：274180      硬链接：1
权限：(0664/-rw-rw-r--)  Uid：( 1001/    chen)   Gid：( 1001/    chen)
环境：unconfined_u:object_r:user_home_t:s0
最近访问：2025-05-30 12:30:00.000000000 +0800
最近更改：2025-05-30 12:30:00.000000000 +0800
最近改动：2022-05-23 10:43:57.128831407 +0800
创建时间：2022-05-23 09:16:38.959182752 +0800
//访问时间和修改时间已更新
```

3.1.4　文件及目录的复制与删除

1. 文件及目录的复制

使用 cp 命令可以复制文件或目录，cp 是英文 copy 的缩写。

cp 命令的语法格式如下：

```
cp [选项] [源文件|目录] [目标文件|目录]
```

cp 命令的常用选项及含义见表 3-3。

表 3-3　cp 命令的常用选项及含义

选项	含　义
-i	在覆盖目标文件之前将给出提示信息，要求用户确认
-r	如果要复制的源是一个目录，则将递归复制该目录及其子目录内的所有内容。此时的目标必须是一个目录
-p	复制时保持指定的属性（默认包括所有权、权限、时间戳等）

【例 3-4】 cp 命令的应用。

（1）准备工作——创建一些文件和目录，命令如下：

```
[chen@server ~]$ touch myreport
//在当前目录下创建名为 myreport 的源文件，命令 touch myreport 使用了相对路径
//绝对路径写法为 touch /home/chen/myreport
[chen@server ~]$ mkdir bak
//在当前目录下创建名为 bak 的目录，用于文件的备份
[chen@server ~]$ touch ./bak/report00
//在 bak 目录下，创建名为 report00 的文件
```

（2）将文件 myreport 复制到 bak 目录下，命令如下：

```
[chen@server ~]$ cp myreport ./bak/
//将源文件 myreport 不改名而直接复制到/home/chen/bak/目录下
[chen@server ~]$ ls bak
myreport
//通过 ls 命令查看 bak 目录可知 myreport 文件复制成功
```

如果复制的目标目录已经存在和源文件同名的文件，则默认会直接覆盖。若希望在执行复制操作时，当发现同名文件时能提示是否覆盖，cp 命令则需要加上-i 选项。下面再次将 myreport 文件复制到 bak 目录下，命令如下：

```
[chen@server ~]$ cp -i myreport ./bak/
cp：是否覆盖'bak/myreport'？ <--此处由用户在键盘输入 y 或者 n
//若用户输入 y，则复制并覆盖原来的同名文件；若用户输入 n，则不执行复制操作
```

注意：建议将文件复制到目录时，直接使用 cp -i，避免无意中覆盖同名文件。

（3）改名复制。

① 将 myreport 文件复制到 bak 目录中，并改名为 myreport0524。此时目标是文件，myreport0524 文件事先并不存在，因此可执行正常复制操作，命令如下：

```
[chen@server ~]$ cp myreport ./bak/myreport0524
[chen@server ~]$ ls bak
myreport  myreport0524
//bak 目录下已经存在两个文件，其中 myreport0524 中的数字 0524 一般是备份日期
```

② 将 myreport 文件复制到 bak 目录中，并改名为 report00。此时目标是文件，但是 bak 目录中已经存在名为 report00 的文件，若直接执行下面的命令：

```
[chen@server ~]$ cp myreport ./bak/report00
//不推荐这么写
```

原有的 report00 文件将在没有任何提示的情况下，直接被覆盖了。怎么办？解决方法仍然是加-i 选项，命令如下：

```
[chen@server ~]$ cp -i myreport ./bak/report00
```

注意：建议改名复制时，使用 cp -i，避免无意中覆盖已存在的文件。

（4）复制目录——将目录 bak 复制至/tmp 目录下，命令如下：

```
[chen@server ~]$ cp -r ./bak /tmp
[chen@server ~]$ ls /tmp/bak
myreport  myreport0524
//查看可知整个 bak 目录已经被复制到了/tmp 目录下
```

（5）保留源文件属性复制。例如在做文件备份或者日志备份时，若希望能够保留源文件的属性，包括所有者和所属群组及时间，执行 cp 命令时就要加上-p 选项。

切换至 root 用户，使用 root 用户执行 cp 命令，命令如下：

```
[chen@server ~]$ su -
//使用 su 命令切换至 root 用户
[root@server ~]#cp /home/chen/myreport /root/
//cp 命令源文件为/home/chen/myreport，目标文件为/root/myreport
//为了方便读者理解，此处全部使用文件的绝对路径
[root@server ~]# ll /home/chen/myreport
-rw-rw-r--. 1 chen chen 6  5月 24 09:10 /home/chen/myreport
//查看源文件/home/chen/myreport 的属性
[root@server ~]# ll /root/myreport
-rw-r--r--. 1 root root 6  5月 24 09:10 /root/myreport
//查看目标文件/root/myreport 的属性
```

显然复制后，目标文件的属性和源文件的属性不同。

下面改为执行 cp -p 命令，命令如下：

```
[root@server ~]# cp -p /home/chen/myreport /root/
cp：是否覆盖'/root/myreport'？y
//将之前/root目录下的myreport文件覆盖
[root@server ~]# ll /home/chen/myreport
-rw-rw-r--. 1 chen chen 6  5月 24 09:10 /home/chen/myreport
//查看源文件/home/chen/myreport的属性
[root@server ~]# ll /root/myreport
-rw-rw-r--. 1 chen chen 6  5月 24 09:10 /root/myreport
//查看目标文件/root/myreport的属性
```

显然，这次复制，目标文件保留了源文件的属性。

最后，恢复 Shell 环境，命令如下：

```
[root@server ~]# su – chen
[chen@server ~]$
//恢复到普通用户chen登录
```

读者注意，cp 命令在复制文件或目录时还要受到文件或目录的权限的制约，有时，由于当前用户权限不足，不能完整地复制目录。

【例 3-5】 cp 命令的应用（权限问题）。

系统的/etc 目录是一个非常重要的目录，因此希望使用 cp 命令为该目录做一个备份。以普通用户 chen 身份登录系统，操作如下。

（1）创建 backup 目录，命令如下：

```
[chen@server ~]$ mkdir backup
```

（2）使用 cp -r 命令将/etc 完整地复制到刚刚新建的 backup 目录中。

```
[chen@server ~]$ cp -r /etc ./backup/
cp: 无法打开'/etc/crypttab' 读取数据: 权限不够
cp: 无法访问 '/etc/pki/CA/private': 权限不够
cp: 无法访问 '/etc/pki/rsyslog': 权限不够
cp: 无法打开'/etc/at.allow' 读取数据: 权限不够
cp: 无法打开'/etc/sysconfig/ip6tables' 读取数据: 权限不够
…
```

显然，以普通用户 chen 身份做的/etc 目录的备份是一个不完全备份。关于用户对文件或目录的访问权限的知识，可参看第 4 章内容。

2. 文件及目录的删除

使用 rm 命令可以删除系统中的文件或目录。rm 命令的语法格式：

```
rm [选项] [文件|目录]
```

rm 命令的选项及含义见表 3-4。

表 3-4 rm 命令的选项及含义

选　项	含　义
-f	强制删除。忽略不存在的文件，也不给出提示信息
-r	递归删除目录及目录中的内容
-i	在删除前必须确认

特别注意的是若用户没有相应的操作权限，则 rm 命令将执行失败。

【例 3-6】 rm 命令的应用。

（1）准备工作——创建一些目录和文件，命令如下：

```
[chen@server ~]$ touch linux.txt
//在当前目录下创建 linux.txt 文件
[chen@server ~]$ mkdir test
//创建目录/home/chen/test
[chen@server ~]$ touch test/sort.c
//在 test 目录下创建文件 sort.c
[chen@server ~]$ touch test/link.c
//在 test 目录下创建文件 link.c
[chen@server ~]$ touch test/1.log
//在 test 目录下创建一个日志文件 1.log
[chen@server ~]$ mkdir test/mydocs
//创建 test 目录的子目录 mydocs，此处使用的仍然是相对路径
[chen@server ~]$ touch test/mydocs/doc{1,2}
//在 mydocs 目录下创建两个文件：doc1 和 doc2
```

（2）删除 linux.txt 文件，命令如下：

```
[chen@server ~]$ rm linux.txt
//rm 命令可一次删除一个文件
```

删除 test 目录下所有的后缀为 c 的文件，命令如下：

```
[chen@server ~]$ rm test/*.c
//删除/home/chen/test 目录下所有的后椄为 c 的文件
//rm 命令可一次删除多个文件
```

从刚才的例子中，读者可以看出 rm 命令在删除文件时没有任何提示。通过 rm 删除的文件不会被放入“回收站”，而是将永远从系统中消失（某些恢复软件可能能找回一些文件）。

（3）删除日志文件 1.log。

为了避免误操作，在删除重要文件（如日志文件）时，需要使用-i 选项，这样在删除文件前系统会给出提示，并等待用户确认，命令如下：

```
[chen@server ~]$ rm -i test/1.log
rm：是否删除普通空文件 'test/1.log'？<-- y
```

```
//根据提示，输入 y 并按 Enter 键才执行删除操作；输入 n 并按 Enter 键表示不删除
```

（4）删除 test 目录，需使用-r 选项，命令如下：

```
[chen@server ~]$ rm -r test
//删除 test 目录，并将 test 目录下的所有子目录和文件全部删除
[chen@server ~]$ ls -d  test
ls: 无法访问 'test': 没有那个文件或目录
//验证 test 目录是否已经被删除
```

至此，在本例（1）中创建的目录和文件已经全部被删除了。

注意：使用 rm 命令一定要慎重。以 root 身份执行 rm 命令时更要格外谨慎。在删除一个文件前，应认真评估后果。不要随便使用 rm -rf 这样的选项，因为-r 为删除目录，-f 为不询问就直接删除，因此若后续的目录名或文件名写错，则可能造成误删。一般而言，被误删的内容多数情况下是无法挽回的。

（5）使用 rm 命令将 3.1.3 节中例 3-3 touch 命令的应用中创建的所有文件删除，命令如下：

```
[chen@server ~]$ rm euler.txt
[chen@server ~]$ rm testfile?
[chen@server ~]$ rm testfile??
```

3.1.5　文件及目录的移动与重命名

mv 命令可以实现文件及目录的移动与重命名，即 mv 命令有两大类功能：

（1）将源文件或目录（可以是多个）移动到目标目录。若目标目录中已有与源文件或目录同名的文件，则该同名文件会被覆盖，所有的源文件都会被移至目标目录中。所有移到目标目录下的文件都将保留以前的文件名。

（2）对源文件或目录（只能是一个）进行重命名。如果源文件或目录和目标文件或目标目录在同一目录下，则 mv 命令的作用为重命名。

mv 命令的语法格式：

```
mv [选项] [源文件|目录] [目标文件|目录]
```

mv 命令的常用选项及含义见表 3-5。

表 3-5　mv 命令的常用选项及含义

选　项	含　义
-i	覆盖前询问
-f	覆盖前不询问
-n	不覆盖已存在的文件
-u	只有在源文件比目标文件新，或者目标文件不存在时才进行移动
-T	将目标文件视作普通文件处理

【例 3-7】 mv 命令的应用。

（1）准备工作。

```
[chen@server ~]$ touch testfile
[chen@server ~]$ touch myfile{1，2}
//在当前目录下创建 testfile、myfile1 和 myfile2 共 3 个文件
[chen@server ~]$ mkdir data mydocs
//创建目录/home/chen/data 和/home/chen/mydocs
[chen@server ~]$ cp myfile? mydocs/
//将 myfile1 和 myfile2 复制到 test 目录下
```

到此，完成了如图 3-2 所示的目录和文件的创建，接下来以此为基础，练习 mv 命令的使用。

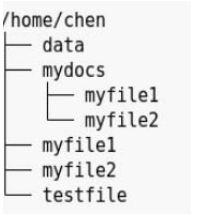

图 3-2　本题中用到的初始目录结构

（2）将源文件移动至目标目录下。

① 使用 mv 命令将 testfile 移动至 mydocs 目录下，命令如下：

```
[chen@server ~]$ mv testfile mydocs/
//将 testfile 文件从当前目录/home/chen 移动至/home/chen/mydocs 目录下
[chen@server ~]$ ls mydocs/
myfile1 myfile2 testfile
//将 testfile 移动至 mydocs 目录成功
```

② 使用 mv -i 命令将 myfile1 移动至 mydocs 目录下，命令如下：

```
[chen@server ~]$ mv -i myfile1 mydocs/
mv：是否覆盖'mydocs/myfile1'？<--y
//输入 y 表示替换，输入 n 表示不替换
//使用/home/chen/myfile1 覆盖了/home/chen/mydocs 目录下原有的 myfile1 文件
```

③ 使用 mv -u 命令将 myfile2 移动至 mydocs 目录下，命令如下：

```
[chen@server ~]$ mv -u myfile2 mydocs/
//由于源文件 myfile2 并不比 mydocs/myfile2 新，因此没有移动，也没有覆盖
```

（3）重命名文件或目录。当源文件或目录与目标文件或目标的路径一样时，mv 命令可以实现文件或目录的重命名。注意源文件或目录必须只有一个，否则会出现命名冲突。

将 mydocs 目录下的 testfile 文件重命名为 euler，命令如下：

```
[chen@server ~]$ mv mydocs/testfile mydocs/euler
[chen@server ~]$ ls mydocs
euler  myfile1  myfile2
//mydocs 目录下 testfile 不见了，取而代之的是 euler 文件，重命名成功
```

将 data 目录重命名为 mydata，命令如下：

```
[chen@server ~]$ mv data mydata
```

（4）移动目录。将 mydocs 目录移动至 mydata 目录中，命令如下：

```
[chen@server ~]$ mv mydocs mydata
[chen@server ~]$ ls mydata
mydocs
//移动成功。mydocs 目录的绝对路径为/homechen/mydata/mydocs
```

3.2　查看文件内容

3.2.1　cat 命令

使用 cat 命令可以将文件的内容输出到屏幕上，常用于查看内容不多的文本文件的内容，长文件会因滚动太快而无法阅读。

cat 命令的基本语法：

```
cat [选项] [文件]
```

cat 命令的常用选项及含义见表 3-6。

表 3-6　cat 命令的常用选项及含义

选　项	含　义
-n	给输出的所有行加上编号
--help	查看 cat 命令的帮助

【例 3-8】 使用 cat 命令查看文件内容。

使用 cat 命令显示/etc/inittab 文件的内容，命令如下：

```
[chen@server ~]$ cat -n /etc/inittab
     1  #inittab is no longer used.
     2  #
     3  #ADDING CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
     4  #
     5  #Ctrl-Alt-Delete is handled by /usr/lib/systemd/system/Ctrl-alt-del
.target
...
```

```
//此处省略部分内容
```

cat 命令还可以用来建立一个有内容的新文件，为文件追加内容或将几个文件合并为一个文件。

【例 3-9】 cat 命令的应用。

（1）使用 cat 命令创建新文件。

使用 cat 命令在当前目录下创建名为 testfile1 和 testfile2 的新文件，命令如下：

```
[chen@server ~]$ cat >testfile1
    Hi,openEuler!
    Hi,Linux!
<-- 按快捷键 Ctrl+D
//输入 hi,openEuler!后按 Enter 键，再输入 hi,Linux! 最后按快捷键 Ctrl+D 保存文件
[chen@server ~]$ cat > testfile2
    Hello,cat!
<-- 按快捷键 Ctrl+D
//请输入 hello,cat!,输入完毕按快捷键 Ctrl+D 保存文件
```

（2）使用 cat 命令查看刚刚创建的文件 testfile1 的内容，命令如下：

```
[chen@server ~]$ cat  testfile1
Hi,openEuler!
Hi,Linux!
```

（3）使用 cat 命令向已存在文件追加内容。

① 使用 cat 命令查看上一步创建的文件 testfile2 的内容，命令如下：

```
[chen@server ~]$ cat  testfile2
Hello,cat!
```

② 使用 cat 命令向 testfile2 中追加内容，命令如下：

```
[chen@server ~]$ cat  >> testfile2 <<EOF
> I am testing the command of cat.
> OK?
> I guess OK.
> EOF
//输入 EOF 表示输入结束并保存文件
```

③ 使用 cat 命令再次查看 testfile2 的内容，查看是否追加成功，命令如下：

```
[chen@server ~]$ cat  testfile2
Hello,cat!
I am testing the command of cat.
OK?
I guess OK.
```

```
//追加成功
```

（4）使用 cat 命令连接多个文件内容并输出到一个新文件。

下面将 testfile1 和 testfile2 这两个文件的内容连接起来后输到文件 testfile3，命令如下：

```
[chen@server ~]$ cat -n testfile1 testfile2 > testfile3
//-n 选项表示加上了行号
[chen@server ~]$ cat  testfile3
    1  Hi,openEuler!
    2  Hi,Linux!
    3  Hello,cat!
    4  I am testing the command of cat.
    5  OK?
    6  I guess OK.
```

特别注意：如果 testfile3 已经存在，则上述命令会将 testfile3 中原有的内容清空。

3.2.2　head 命令和 tail 命令

1. head 命令

使用 head 命令可以显示指定文件的前若干行内容。如果没有给出具体行的数值，则默认设置为 10 行。如果没有指定文件，则 head 会从标准输入读取。

head 命令的语法格式：

```
head [选项] [文件]
```

head 命令的常用选项及含义见表 3-7。

表 3-7　head 命令的常用选项及含义

选　　项	含　　义
-n　<K>	显示每个文件的前 *K* 行内容；如果附加“-”参数，则除了每个文件的最后 *K* 行外显示剩余的全部内容，这里 *K* 是数字
-c　<K>	显示每个文件的前 *K* 字节内容；如果附加“-”参数，则除了每个文件的最后 *K* 字节数据外显示剩余的全部内容，这里 *K* 是数字
-v	总是显示包含给定文件名的文件头

【例 3-10】 head 命令的应用。

（1）查看/etc/inittab 文件前 5 行数据内容。

```
[chen@server ~]$ head -n 5 /etc/inittab
#inittab is no longer used.
#
#ADDING CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
#
#Ctrl-Alt-Delete is handled by /usr/lib/systemd/system/Ctrl-alt-del.target
```

（2）查看/etc/passwd 文件前 100 字节数据内容。

```
[chen@server ~]$ head -c 100 /etc/inittab
#inittab is no longer used.
#
#ADDING CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
#
#C
```

2. tail 命令

使用 tail 命令可以查看文件的末尾数据，默认显示指定文件的最后 10 行。如果指定了多个文件，tail 则会在每段输出的开始添加相应文件名作为提示。如果不指定文件或文件为“-”，则从标准输入读取数据。

tail 命令的语法格式：

```
tail [选项] [文件名]
```

tail 命令的常用选项及含义见表 3-8。

表 3-8　tail 命令的常用选项及含义

选　　项	含　　义
-n <K>	显示文件最后 *K* 行内容，这里 *K* 是数字
-c <K>	显示文件的最后 *K* 字节内容，这里 *K* 是数字
-f	随文件增长即时输出新增数据

【例 3-11】 tail 命令的应用。

（1）查看/etc/inittab 文件末尾 5 行的数据内容。

```
[chen@server ~]$ tail -n 5 /etc/inittab
#To view current default target, run:
#systemctl get-default
#
#To set a default target, run:
#systemctl set-default TARGET.target
```

（2）查看/etc/inittab 文件末尾 100 字节的数据内容。

```
[chen@server ~]$ tail -c 100 /etc/inittab
un:
#systemctl get-default
#
#To set a default target, run:
#systemctl set-default TARGET.target
```

3.2.3　more 命令和 less 命令

1. more 命令

对于内容较多的文件，一屏幕显示不全，此时可用 more 命令来逐页阅读。

命令语法：

```
more [选项] [文件]
```

more 命令一次显示一屏文本，显示满之后，停下来，并在终端底部打印出“-- More --”，系统还将同时标示出已显示文本占全部文本的百分比，按空格键显示下一页内容；若要结束浏览，按 Q 键即可退出。more 命令的常用选项及含义见表 3-9。

表 3-9　more 命令的常用选项及含义

选　　项	含　　义
-p	显示下一屏之前先清屏
-c	与-p 类似
-s	将连续空白行显示为一个空白行
-d	在每屏的底部显示更友好的提示信息，而且若用户输入了一个错误命令，则显示出错信息
+<n>	文件内容从第 *n* 行开始显示，*n* 是数字
-<n>	一次显示的行数，*n* 是数字

【例 3-12】 more 命令的应用。

（1）分页显示/etc/services 文件的内容。

```
[chen@server ~]$ more -d /etc/services
…
//此处省略了显示的文件的部分内容
#Each line describes one service, and is of the form:
#
#service-name  port/protocol  [aliases ...]   [#comment]
--更多--(0%)[按空格键继续，按 q 键退出。]
//-d 选项给出了较多提示信息
```

（2）从第 50 行开始显示/etc/services 文件的内容。

```
[chen@server ~]$ more +50 /etc/services
```

（3）一次 10 行显示/etc/services 文件的内容。

```
[chen@server ~]$ more -c -10 /etc/services
//执行该命令后，先清屏，然后将以每次 10 行的方式显示文件
```

2. less 命令

使用 less 命令可以分页显示文本文件的内容。less 命令的作用与 more 命令十分相似。

不同之处在于，more 命令只能向后翻页，而 less 还可以向前翻页，即 less 命令允许回卷显示文本文件的内容。

less 命令的语法：

```
less [选项] [文件名]
```

less 命令的选项非常多，无须都记住，用到时可以查看帮助，命令如下：

```
[chen@server ~]$ less --help
```

less 命令的常用选项及含义见表 3-10。

表 3-10　less 命令的常用选项及含义

选　　项	含　　义
-e	当文件显示结束后，自动离开
-s	将连续空白行显示为一个空白行
-m	显示类似 more 命令的百分比
-N	显示行号

【例 3-13】 less 命令的应用。

（1）使用 less 命令查看/etc/services 文件的内容。

```
[chen@server ~]$ less -s /etc/services
```

（2）查看系统进程信息，使用 less 命令分页显示，并显示行号。

```
[chen@server ~]$ ps -ef|less -N
//这里使用了管道命令。ps 命令负责查看进程信息，less 命令负责分页
```

（3）使用 less 命令浏览多个文件。

```
[chen@server ~]$ less testfile1 testfile2
Hi,openEuler!
Hi,Linux!
testfile1 (file 1 of 2) (END) - Next: testfile2
//当前显示的是 testfile1 的内容，接下来
//如果用户输入:n 后按 Enter 键，则将切换至 testfile2，n 表示 next
//如果用户输入:p 后按 Enter 键，则切换至 testfile1，p 表示 previous
```

3.3　vim 编辑器

系统管理员在进行系统管理或服务器配置时，经常需要修改相关的配置文件，或对纯文本文件进行编辑，这时就需要用到 vim。vim 是系统中很多命令默认会调用的编辑器，因此管理员一定要能熟练地使用这个编辑器。

vim[①]是 vi improved 的缩写，是 vi[②]的增强版。vi 即 visual interface，它是 Linux 中的标准文本编辑器，也是 UNIX/Linux 系统中最常用的文本编辑器工具。

vim 的操作方法与 vi 基本一样，vim 在内容显示上增强了颜色支持，不同的颜色代表不同的语义，在使用上更加容易辨析和人性化。

3.3.1　vim 的启动与退出

1. 启动 vim

在系统提示符下输入 vim 及文件名称后，就会进入 vim 的工作界面，示例命令如下：

```
[chen@server ~]$ vim file
//若 file 存在，则打开该文件；若 file 不存在，则创建 file
//也可以写成 vim /home/chen/file
```

使用 vim 新建文件或编辑现有文件都会直接启动 vim。

2. 退出 vim

当编辑完文件后，准备返回 Shell 状态时，需执行退出 vim 的命令。在 vim 的命令模式下，按一下冒号键便会进入命令行模式。下面分情况讨论。

（1）直接退出 vim，命令如下：

```
:q
```

（2）保存当前文件内容后再退出 vim，命令如下：

```
:wq
```

（3）不保存文件内容，强制退出 vim，命令如下：

```
:q!
```

3.3.2　常用的 vim 工作模式

vim 的工作模式可分为 3 种：命令模式、编辑模式和命令行模式。不同工作模式下能够完成的主要操作见表 3-11。

表 3-11　vim 的工作模式

vim 工作模式	主 要 操 作
命令模式	光标的移动控制：移动到文本的某个位置 文本的编辑命令，如复制、粘贴、剪切、删除等

① vim 的作者是荷兰程序员 Bram Moolenaar。

② vi 的作者是 Bill Joy，他是前任 Sun 的首席科学家，当年在 Berkeley（伯克利）时主持开发了最早版本的 BSD。他还是 csh 的作者，并开发出了高性能的 Berkeley（伯克利）版的 TCP/IP。