

实验十

Hollow Man

实验名称：

文件系统观察

实验目的：

1. 学习和掌握文件系统的基本概念
2. 学习对文件和文件系统的观察和操作
3. 学习和使用文件系统的权限控制

实验时间

3 学时

预备知识：

1. 基本命令

命令名	主要选项	功能说明
ls	-a, -l, -i	列出指定文件
stat		显示文件系统信息
cd		切换目录
pwd		报告当前路径
touch	-a, -m	创建新文件
mv	-i, -u	移动
cp	-a, -i, -l, -s, -R	复制
rm	-i, -r	删除文件
mkdir	-p	创建目录
rmdir		删除目录(空目录)
ln	-s	建立链接
find	-type, -name, -ctime	查找文件
locate		快速查找文件
grep	-i, -l, -r, -v, -n	查找文件内容
chmod	-R	添加、删除、指派文件或目录的权限
chown		改变文件属主
chgrp		改变文件组
umask		查看、设置权限掩码
mkfs	-t	创建文件系统
mke2fs	-j, -b, -i	创建 ext2/ext3 文件系统
mount	-t, -o, -a	挂载文件系统
umount		卸载文件系统
df	-i, -h, -k, -a	提供硬盘及其分区、其它驱动器在文件系统上的装入位置以及它们所占用的空间大小等信息。
du	-c, -h	提供关于文件和目录所占空间的信息
fsck		检查文件系统

2. 文件类型

类型	说明
普通文件	一组连续的数据用一个名称表示
目录	实施了分级文件系统的结构
设备文件	要访问硬件的每个程序都必须通过对应的设备文件来访问硬件
链接	对存储在文件系统中其他点的文件的引用
套接字	通过文件系统实施两个本地运行的进程之间的数据交换
FIFO	在进程之间交换数据

3. 文件和目录的权限保护

每个文件和目录都具有一定的访问权限。指派的权限决定给定用户的访问级别。权限的指派分为三个级别：

用户(u,owner)：指派给文件或目录所有者的权限决定了所有者的访问级别。

组(group)：为组指派的权限确定了组成员对文件或目录的访问级别。

其他(other)：指派给该实体的权限用于已鉴定的用户，这些用户本身不是组的成员但已和文件或目录相关联。

可以对文件或目录指派以下三种权限：

读 (r)：该权限允许读取文件并列出目录内容。

写 (w)：该权限允许修改文件。还允许在目录内创建或删除文件。

执行 (x)：该权限允许执行文件。还允许访问目录。

可以使用命令 `ls -l` 显示当前目录中的内容以及指派的对每个文件或子目录的访问权限。

例如，输入 `ls -l` 显示 `myfile.txt` 的权限如下：

```
Owner Group Others
[ root@RCE tmp ]# ls -l
total 40
-rw-r--r-- 1 root root 12304 Dec 12 03:44 install.log
-rw-r--r-- 1 root root 0 Dec 12 03:17 install.log.syslog
-rwxr-xr-- 1 root root 6 Dec 12 14:55 myfile.txt
drwx----- 2 root root 4096 Dec 12 03:32 orbit-root
-rw-rw-rw- 1 apache apache 6 Dec 12 11:48 rcq-runner.pid
-rw----- 1 apache apache 6255 Dec 12 11:53 sess_51e5c8d48f41b4a3d0e
ef964966f8775
-rw----- 1 root root 0 Dec 12 11:48 session_mm_apache0.sem
-rw----- 1 root root 0 Dec 12 11:48 session_mm_cgi0.sem
drwxr-xr-x 8 201 201 4096 Dec 12 10:51 vmware-tools-distrib
[ root@RCE tmp ]#
```

每个文件和目录都指定有数字权限值。该值有3位数字。第一位数字表示指派给文件或目录所有者的权限。第二位数字表示指派给和文件及目录相关的组的权限。第三位数字表示指派给其他用户的权限。

每位数字都是指派的以下三个值的和：读： 4；写： 2；执行： 1。

默认情况下系统以访问方式 666 创建文件，并以访问方式 777 创建目录。要修改（限制）这些默认访问方式设置，可以使用命令 `umask`。该命令将和 3 位数字值（如 022）一起使用，从默认权限中删除在 `umask` 中设置的权限。

此外，还有三种特殊的文件权限：

字母	编号	名称	文件	目录
t 或 T	1	粘滞位 (stick bit)	不适用	只有文件的所有者、根用户或目录所有者可以删除文件。通常应用于目录 <code>/tmp/</code> 。
s 或 S	2	SGID (SetGID)	运行程序时将进程的 组 ID 设置为文件组的 组 ID。	在此目录下创建的文件属于目录所在的 组，而不属于用户的主组。 在此目录下创建的新目录将继承 SGID 位。
s 或 S	4	SUID (SetUserID)	运行程序时将进程的 用户 ID 设置为文件所 有者的用户 ID。	不适用

4. Linux 支持的文件系统

4.1 传统的文件系统

Linux 支持的传统文件系统不将数据或元数据记入日记。

这些文件系统包括：

- **ext2**。ext2 文件系统基于 `inode`，为提高速度而设计，既高效又不容易产生文件碎片。
- **minix**。minix 文件系统比较旧，限制较多（它是首个 Linux 文件系统），但对于软盘或 RAM 磁盘，有时仍会使用此文件系统，因为 minix 极低的文件系统开销可增加数据存储量。
- **MS-DOS/VFAT**。FAT（文件分配表）是 Microsoft Windows 所使用的主文件系统。VFAT 是 FAT 的 32 位版本，包含长文件名。
- **HPFS**。HPFS（高性能文件系统）是 IBM OS/2 文件系统的原始文件系统。

4.2 日记文件系统

以下可用于 Linux 的文件系统包含日记功能：

- **ext3**。ext3 是 ext2 文件系统支持日记的版本。
- **ReiserFS**。ReiserFS 最初由 Hans Reiser 设计，该文件系统将整个磁盘分区视作一个单独的数据库表，不但存储文件元数据，而且存储文件本身。
目录、文件和文件元数据通过一种被称为“平衡树”的高效数据结构进行组织，此结构可显著提升许多应用程序的速度，尤其是那些大量使用小文件的应用程序。
- **NTFS**。NTFS（新技术文件系统）是 Windows NT 使用的文件系统。使用 Unicode 字

符集，支持长达 255 个字符的文件名。目前 Linux 系统只支持对该文件系统的读取。

■ **JFS**。此日记文件系统是 IBM 在 2001 年发布的正式产品版。

■ **XFS**。XFS 是来自 SGI 的高性能日记文件系统。它提供了系统崩溃后的快速恢复、较快的处理速度、高可伸缩性和适用的带宽。

XFS 将先进的日记技术与全面的 64 位寻址和可伸缩的结构和算法相结合。

■ **Veritas's VxFS**。VxFS 是一个商用日记文件系统，2001 年首次随 Linux 提供，经常用在 Unix 平台上。

4.3 虚拟文件系统（VFS）转换

对于用户或程序，无论使用哪种文件系统格式都无关紧要。数据始终显示相同的界面。这是通过虚拟文件系统转换（VFS，也称为虚拟文件系统）实现的。这是内核中的一个抽象层次，提供为进程定义的界面。它包含打开文件、写入文件和读取文件等功能。

5. Linux 文件系统格式

Linux 的文件系统格式的独特之处是数据和管理信息是分开的。每个文件都通过inode（索引节点或信息节点）来描述。在这些节点中，每个节点都有128个字节，包含除文件名之外有关此文件的所有信息。这些信息包括：**文件所有者、访问权限、文件大小、各种时间（修改时间、访问时间和修改inode的时间）**等详细资料，以及**指向文件数据块的链接**。

但是Inode不包含文件名。文件名包含在目录中。目录包含其它文件的信息，此信息包含文件的 inode 编号及其名称。目录可作为一张表格，在此表格中，inode编号逐行分配给文件名。

6. Linux 文件系统分区

6.1 分区类型

（Intel 平台上）每个硬盘都有一个带有4项空格的分区表。分区表中的一项可以对应于一个主分区或一个扩展分区。但只允许有一个扩展分区项。

主分区由指派给特定操作系统的一系列连续的柱面（物理磁盘区域）组成。如果只有主分区，只能使用4个分区，因为分区表中仅限4项。

扩展分区同样是一系列连续的磁盘柱面，但扩展分区可以再分为多个逻辑分区。逻辑分区不要求在分区表中有对应的项。换句话说，扩展分区是逻辑分区的容器。由于扩展分区应包括剩下的所有可用的柱面范围，所以配置扩展分区前请先配置主分区。

配置扩展分区后，在扩展分区内创建多个逻辑分区。对于SCSI磁盘，逻辑分区的最大数目是15个，对于 (E)IDE 磁盘，逻辑分区的最大数目是63个。

6.2 设备和分区命名

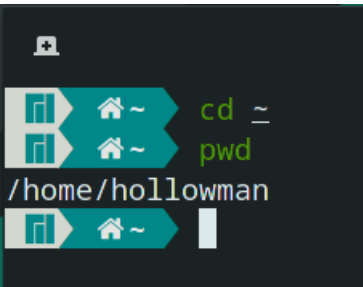
设备	名称
第一个IDE硬盘上的主设备	/dev/hda
第一个IDE硬盘上的从设备	/dev/hdb
第二个IDE硬盘上的主设备(经常是CDROM)	/dev/hdc
第二个IDE硬盘上的从设备	/dev/hdd
第一个SCSI硬盘	/dev/sda
第二个SCSI硬盘	/dev/sdb
第三个SCSI硬盘	/dev/sdc

分区	名称
第一个IDE硬盘上的第一个分区	/dev/hda1
第一个IDE硬盘上的第二个分区	/dev/hda2
第一个IDE硬盘上的第一个逻辑分区	/dev/hda5
第一个IDE硬盘上的第二个逻辑分区	/dev/hda6

实验要求:

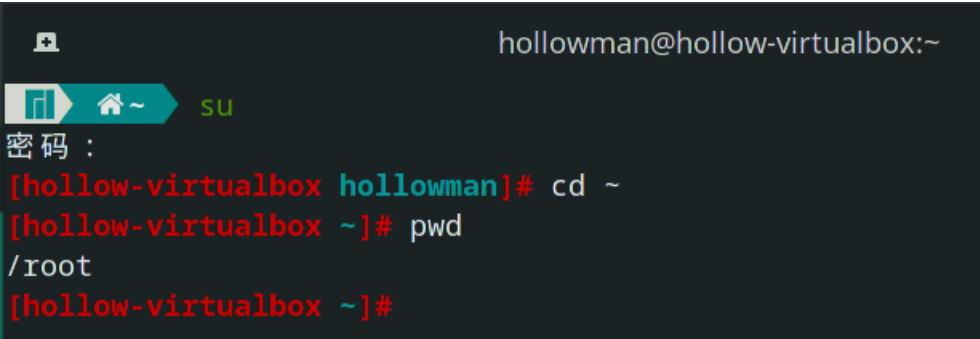
1. 分别以 root 和普通用户身份登录并进入各自的主目录，通过命令报告你的当前路径。

普通用户:



```
hollowman@hollow-virtualbox:~$ cd ~
hollowman@hollow-virtualbox:~$ pwd
/home/hollowman
```

Root 用户;



```
hollowman@hollow-virtualbox:~$ su
密码:
[hollow-virtualbox hollowman]# cd ~
[hollow-virtualbox ~]# pwd
/root
[hollow-virtualbox ~]#
```

2. 在一个目录下执行 ls 命令，验证 -l, -a, -i 选项的作用，什么时候会列出 "." 和 ".." 目录？设计一个关于使用命令的实验，验证这两个目录的含义和作用。

ls -l:

```
hollowman@hollow-virtualbox:~  
ls -l  
总用量 40  
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 公共  
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 模板  
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 视频  
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 图片  
drwxr-xr-x 4 hollowman hollowman 4096 5月 24 00:35 文档  
drwxr-xr-x 5 hollowman hollowman 4096 5月 5 01:58 下载  
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 音乐  
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 桌面  
-rw-r--r-- 1 hollowman hollowman 3717 5月 2 11:33 stylesheet.css  
-rw-r--r-- 1 hollowman hollowman 3737 5月 3 12:45 stylesheet-dark.css
```

ls -a:

```
hollowman@hollow-virtualbox:~  
ls -a  
. 下载 .config stylesheet.css .Xclients  
.. 音乐 .dir_colors stylesheet-dark.css .xinitrc  
公共 桌面 .gnupg .themes .zcompdump  
模板 .bash_logout .local .vboxclient-clipboard.pid .zhistory  
视频 .bash_profile .mozilla .vboxclient-draganddrop.pid .zshrc  
图片 .bashrc .python_history .vboxclient-seamless.pid  
文档 .cache .ssh .wget-hsts
```

ls -i:

```
hollowman@hollow-virtualbox:~  
ls -i  
1602897 公共 1602900 图片 1602899 音乐 1603446 stylesheet-dark.css  
1602896 模板 1602898 文档 1602894 桌面  
1602901 视频 1602895 下载 1603494 stylesheet.css
```

可以发现在使用命令“ls -a”（显示隐藏文件）的时候会出现.与..文件，这是在创建目录时，默认会生成两个目录项：“.”和“..”。前者的 inode 号码就是当前目录的 inode 号码，等同于当前目录的“硬链接”；后者的 inode 号码就是当前目录的父目录的 inode 号码，等同于父目录的“硬链接”。

3. 创建一个目录，并在其中创建几个文件，分别用 rm 和 rmdir 删除目录，观察有何不同。

mkdir 是建立目录,而 rmdir 是删除目录命令、rm 可以同时删除文件或目录。

```
hollowman@hollow-virtualbox:~/1
mkdir 1
cd 1
touch 2
touch 3
ls
2 3
~/1
```

```
hollowman@hollow-virtualbox:~
rm 1
rm: 无法删除 '1': 是一个目录
rmdir 1
rmdir: 删除 '1' 失败: 目录非空
rm 1/2 1/3
rmdir 1
```

4. 以 root 身份创建一个新文件，观察其默认的权限；然后用 vi 编辑该文件；将该文件权限改为只有用户可读，其他权限均无；以 root 身份创建一个脚本，该脚本使用 cat 命令在屏幕上显示前面创建文件的内容；将脚本文件按设置为所有用户可执行；分别以 root 和普通用户身份登录，执行脚本，观察结果；为 cat 文件加 SUID 权限，再重复前一步操作，观察结果，说明原因。

```
hollowman@hollow-virtualbox:/home/hollowman
su
密码:
[hollow-virtualbox hollowman]# touch file
[hollow-virtualbox hollowman]# ls
公共 视频 文档 音乐 file stylesheet-dark.css
模板 图片 下载 桌面 stylesheet.css
[hollow-virtualbox hollowman]# ls -l
总用量 40
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 公共
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 模板
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 视频
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 图片
drwxr-xr-x 4 hollowman hollowman 4096 5月 24 00:35 文档
drwxr-xr-x 5 hollowman hollowman 4096 5月 5 01:58 下载
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 音乐
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 桌面
-rw-r--r-- 1 root root 0 5月 24 09:37 file
-rw-r--r-- 1 hollowman hollowman 3717 5月 2 11:33 stylesheet.css
-rw-r--r-- 1 hollowman hollowman 3737 5月 3 12:45 stylesheet-dark.css
[hollow-virtualbox hollowman]#
```

可以看到其默认权限为-rw-r--r--(644)。

使用 vi 编辑该文件如下：


```

[hollow-virtualbox hollowman]# vi file
[hollow-virtualbox hollowman]# chmod 400 file
[hollow-virtualbox hollowman]# ls -l
总用量 44
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 公共
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 模板
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 视频
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 图片
drwxr-xr-x 4 hollowman hollowman 4096 5月 24 00:35 文档
drwxr-xr-x 5 hollowman hollowman 4096 5月 5 01:58 下载
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 音乐
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 桌面
-r----- 1 root      root        16 5月 24 09:39 file
-rw-r--r-- 1 hollowman hollowman 3717 5月 2 11:33 stylesheet.css
-rw-r--r-- 1 hollowman hollowman 3737 5月 3 12:45 stylesheet-dark.css
[hollow-virtualbox hollowman]#

```

以 root 身份创建一个脚本，该脚本使用 cat 命令在屏幕上显示前面创建文件的内容，将脚本文件按设置为所有用户可执行：

```

hollowman@hollow-virtualbox:/home/hollowman
[hollow-virtualbox hollowman]# echo "cat /home/hollowman/file" >> 1.sh
[hollow-virtualbox hollowman]# chmod 755 1.sh
[hollow-virtualbox hollowman]# ls -l
总用量 48
-rwxr-xr-x 1 root      root        25 5月 24 09:43 1.sh
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 公共
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 模板
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 视频
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 图片
drwxr-xr-x 4 hollowman hollowman 4096 5月 24 00:35 文档
drwxr-xr-x 5 hollowman hollowman 4096 5月 5 01:58 下载
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 音乐
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 桌面
-r----- 1 root      root        16 5月 24 09:39 file
-rw-r--r-- 1 hollowman hollowman 3717 5月 2 11:33 stylesheet.css
-rw-r--r-- 1 hollowman hollowman 3737 5月 3 12:45 stylesheet-dark.css
[hollow-virtualbox hollowman]#

```

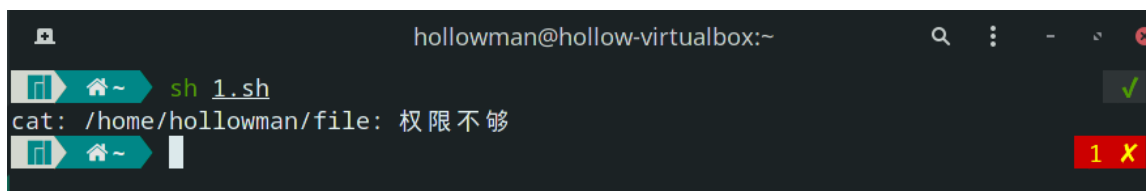
以 root 身份登录执行脚本，观察结果：

```

hollowman@hollow-virtualbox:/home/hollowman
[hollow-virtualbox hollowman]# sh 1.sh
sdfsdxcxsfdcrfd
[hollow-virtualbox hollowman]#

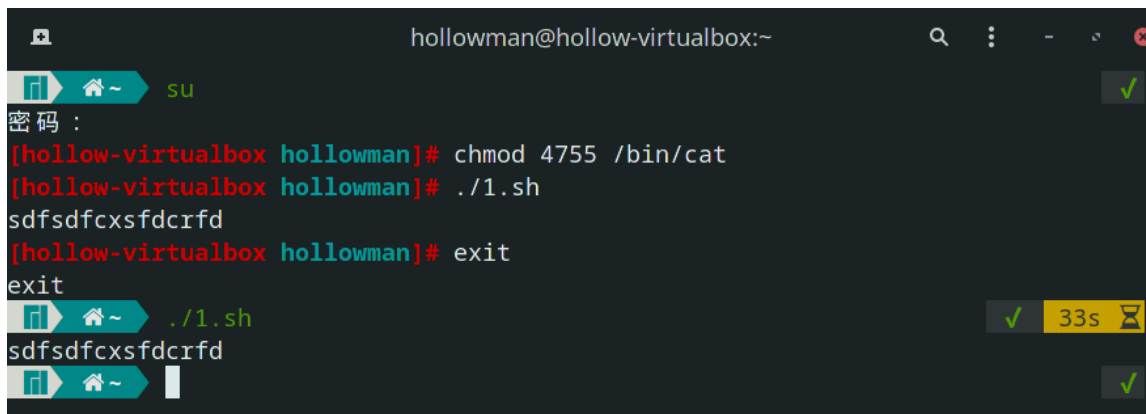
```

以普通用户身份登录执行脚本，观察结果：



```
hollowman@hollow-virtualbox:~  
sh 1.sh  
cat: /home/hollowman/file: 权限不够
```

为 cat 文件加 SUID 权限，root 身份登录执行脚本；以普通用户身份登录执行脚本，分别观察结果：



```
hollowman@hollow-virtualbox:~  
su  
密码 :  
[hollow-virtualbox hollowman]# chmod 4755 /bin/cat  
[hollow-virtualbox hollowman]# ./1.sh  
sdfsdxcxsfdcrfd  
[hollow-virtualbox hollowman]# exit  
exit  
./1.sh  
sdfsdxcxsfdcrfd
```

5. 为一个已经存在的文件分别创建多个个硬链接和多个符号链接，观察二者的不同，删除链接时又有何不同？为什么？

硬链接

一般情况下，文件名和 inode 号码是"一一对应"关系，每个 inode 号码对应一个文件名。但是，Unix/Linux 系统允许，多个文件名指向同一个 inode 号码。这意味着，可以用不同的文件名访问同样的内容；对文件内容进行修改，会影响到所有文件名；但是，删除一个文件名，不影响另一个文件名的访问。这种情况就被称为"硬链接"（hard link）。

ln 命令可以创建硬链接：

ln 源文件 目标文件

运行上面这条命令以后，源文件与目标文件的 inode 号码相同，都指向同一个 inode。inode 信息中有一项叫做"链接数"，记录指向该 inode 的文件名总数，这时就会增加 1。反过来，删除一个文件名，就会使得 inode 节点中的"链接数"减 1。当这个值减到 0，表明没有文件名指向这个 inode，系统就会回收这个 inode 号码，以及其所对应 block 区域。

这里顺便说一下目录文件的"链接数"。创建目录时，默认会生成两个目录项："."和"..". 前者的 inode 号码就是当前目录的 inode 号码，等同于当前目录的"硬链接"；后者的 inode 号码就是当前目录的父目录的 inode 号码，等同于父目录的"硬链接"。所以，任何一个目录的"硬链接"总数，总是等于 2 加上它的子目录总数（含隐藏目录），这里的 2 是父目录对其

的“硬链接”和当前目录下的“硬链接”。

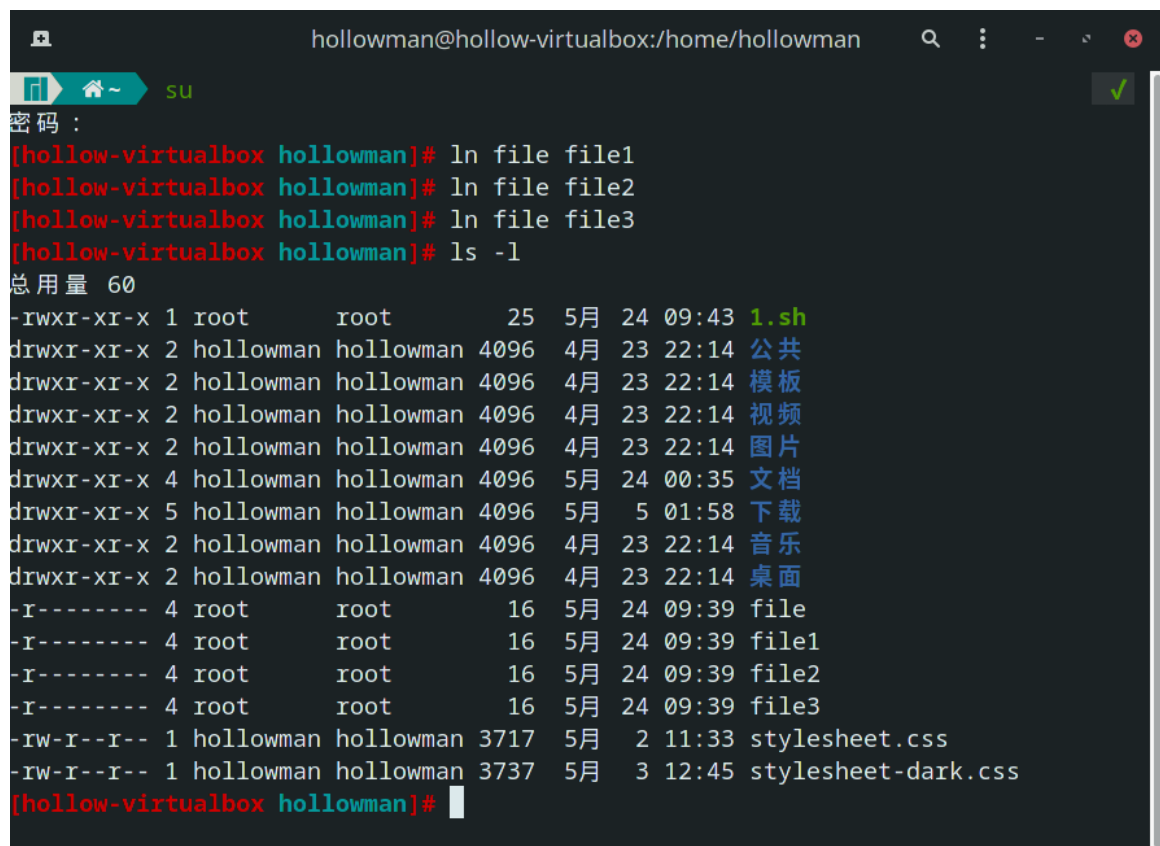
软链接

除了硬链接以外，还有一种特殊情况。文件 A 和文件 B 的 inode 号码虽然不一样，但是文件 A 的内容是文件 B 的路径。读取文件 A 时，系统会自动将访问者导向文件 B。因此，无论打开哪一个文件，最终读取的都是文件 B。这时，文件 A 就称为文件 B 的“软链接”（soft link）或者“符号链接”（symbolic link）。

这意味着，文件 A 依赖于文件 B 而存在，如果删除了文件 B，打开文件 A 就会报错：“No such file or directory”。这是软链接与硬链接最大的不同：文件 A 指向文件 B 的文件名，而不是文件 B 的 inode 号码，文件 B 的 inode“链接数”不会因此发生变化。

ln -s 命令可以创建软链接。

ln -s 源文件或目录 目标文件或目录



```
hollowman@hollow-virtualbox:/home/hollowman
su
密码：
[hollow-virtualbox hollowman]# ln file file1
[hollow-virtualbox hollowman]# ln file file2
[hollow-virtualbox hollowman]# ln file file3
[hollow-virtualbox hollowman]# ls -l
总用量 60
-rwxr-xr-x 1 root      root      25  5月 24 09:43 1.sh
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 公共
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 模板
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 视频
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 图片
drwxr-xr-x 4 hollowman hollowman 4096 5月 24 00:35 文档
drwxr-xr-x 5 hollowman hollowman 4096 5月  5 01:58 下载
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 音乐
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 桌面
-r----- 4 root      root       16  5月 24 09:39 file
-r----- 4 root      root       16  5月 24 09:39 file1
-r----- 4 root      root       16  5月 24 09:39 file2
-r----- 4 root      root       16  5月 24 09:39 file3
-rw-r--r-- 1 hollowman hollowman 3717  5月  2 11:33 stylesheet.css
-rw-r--r-- 1 hollowman hollowman 3737  5月  3 12:45 stylesheet-dark.css
[hollow-virtualbox hollowman]#
```

删除 file3 链接数-1:

```
hollowman@hollow-virtualbox:/home/hollowman

[hollow-virtualbox hollowman]# rm file3
[hollow-virtualbox hollowman]# ls -l
总用量 52
-rwxr-xr-x 1 root      root      25  5月 24 09:43 1.sh
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 公共
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 模板
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 视频
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 图片
drwxr-xr-x 4 hollowman hollowman 4096 5月 24 00:35 文档
drwxr-xr-x 5 hollowman hollowman 4096 5月  5 01:58 下载
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 音乐
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 桌面
-r----- 1 root      root        0  5月 24 09:51 file
-r----- 2 root      root        16  5月 24 09:39 file1
-r----- 2 root      root        16  5月 24 09:39 file2
-rw-r--r-- 1 hollowman hollowman 3717 5月  2 11:33 stylesheet.css
-rw-r--r-- 1 hollowman hollowman 3737 5月  3 12:45 stylesheet-dark.css
[hollow-virtualbox hollowman]#
```

创建删除软连接，链接数不变：

```
hollowman@hollow-virtualbox:/home/hollowman

[hollow-virtualbox hollowman]# ln -s file file3
[hollow-virtualbox hollowman]# ln -s file file4
[hollow-virtualbox hollowman]# ls -l
总用量 52
-rwxr-xr-x 1 root      root      25  5月 24 09:43 1.sh
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 公共
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 模板
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 视频
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 图片
drwxr-xr-x 4 hollowman hollowman 4096 5月 24 00:35 文档
drwxr-xr-x 5 hollowman hollowman 4096 5月  5 01:58 下载
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 音乐
drwxr-xr-x 2 hollowman hollowman 4096 4月 23 22:14 桌面
-r----- 1 root      root        0  5月 24 09:51 file
-r----- 2 root      root        16  5月 24 09:39 file1
-r----- 2 root      root        16  5月 24 09:39 file2
lrwxrwxrwx 1 root      root        4  5月 24 09:53 file3 -> file
lrwxrwxrwx 1 root      root        4  5月 24 09:53 file4 -> file
-rw-r--r-- 1 hollowman hollowman 3717 5月  2 11:33 stylesheet.css
-rw-r--r-- 1 hollowman hollowman 3737 5月  3 12:45 stylesheet-dark.css
[hollow-virtualbox hollowman]#
```

(1)软连接可以跨文件系统，硬连接不可以。

(2)硬连接不管有多少个，都指向的是同一个 i 节点，结点连接数会增加， 只要结点的连接
数不是 0,文件就一直存在，不管你删除的是源文件还是连 接的文件。只要有一个存在，文

件就存在。当你修改源文件或者连接文件任何一个的时候，其他的文件都会做同步的修改。软链接不直接使用 i 节点号 作为文件指针,而是使用文件路径名作为指针。所以删除连接文件 对源文件 无影响，但是删除源文件，连接文件就会找不到要指向的文件。软链接有自己的 inode,并在磁盘上有一小片空间存放路径名。

(3) 软连接可以对一个不存在的文件名进行连接。

(4) 软连接可以对目录进行连接。

6. 报告你当前使用的系统已经挂载了那些文件系统，挂载点、文件系统类型和对应设备文件以及设备和分区分别是什么？硬盘的当前使用情况（数据及索引节点）。

```
hollowman@hollow-virtualbox:/home/hollowman
[hollow-virtualbox hollowman]# mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sys on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
dev on /dev type devtmpfs (rw,nosuid,relatime,size=1997532k,nr_inodes=499383,mode=755,inode64)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755,inode64)
efivarfs on /sys/firmware/efi/efivars type efivarfs (rw,nosuid,nodev,noexec,relatime)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
/dev/sda2 on / type ext4 (rw,noatime)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,size=4096k,nr_inodes=1024,mode=755,inode64)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
```

```
hollowman@hollow-virtualbox:/home/hollowman

[hollow-virtualbox hollowman]# df -ihka
文件系统      Inodes 已用 (I) 可用 (I) 已用 (I)% 挂载点
proc           0         0         0         - /proc
sys            0         0         0         - /sys
dev          499383       458  498925       1% /dev
run          501830       689  501141       1% /run
efivarfs        0         0         0         - /sys/firmware/efi/efivars
devpts          0         0         0         - /dev/pts
/dev/sda2     1949696   391281 1558415      21% /
securityfs      0         0         0         - /sys/kernel/security
tmpfs          501830         1  501829       1% /dev/shm
tmpfs           1024        18   1006        2% /sys/fs/cgroup
cgroup2          0         0         0         - /sys/fs/cgroup/unified
cgroup           0         0         0         - /sys/fs/cgroup/systemd
pstore           0         0         0         - /sys/fs/pstore
none            0         0         0         - /sys/fs/bpf
cgroup           0         0         0         - /sys/fs/cgroup/freezer
cgroup           0         0         0         - /sys/fs/cgroup/cpu,cpuacct
cgroup           0         0         0         - /sys/fs/cgroup/blkio
cgroup           0         0         0         - /sys/fs/cgroup/memory
cgroup           0         0         0         - /sys/fs/cgroup/cpuset
cgroup           0         0         0         - /sys/fs/cgroup/hugetlb
cgroup           0         0         0         - /sys/fs/cgroup/perf_event
cgroup           0         0         0         - /sys/fs/cgroup/net_cls,net_prio
cgroup           0         0         0         - /sys/fs/cgroup/pids
cgroup           0         0         0         - /sys/fs/cgroup/devices
cgroup           0         0         0         - /sys/fs/cgroup/rdma
systemd-1        0         0         0         - /proc/sys/fs/binfmt_misc
mqueue           0         0         0         - /dev/mqueue
hugetlbfs        0         0         0         - /dev/hugepages
debugfs          0         0         0         - /sys/kernel/debug

hollowman@hollow-virtualbox:/home/hollowman

[hollow-virtualbox hollowman]# fdisk -l
Disk /dev/sda: 30 GiB, 32212254720 字节, 62914560 个扇区
磁盘型号 : VBOX HARDDISK
单元 : 扇区 / 1 * 512 = 512 字节
扇区大小 (逻辑/物理) : 512 字节 / 512 字节
I/O 大小 (最小/最佳) : 512 字节 / 512 字节
磁盘标签类型 : gpt
磁盘标识符 : 1499452A-8C42-5546-AC2A-FC0DC90CC0B2

设备      起点      末尾      扇区      大小  类型
/dev/sda1   4096    618495    614400    300M  EFI 系统
/dev/sda2 618496 62910539 62292044  29.7G  Linux 文件系统

[hollow-virtualbox hollowman]#
```