

实验五：文件系统观察

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。

答：

```
→ ~ cd ~
→ ~ pwd
/home/myself
→ ~ su root
密码: 170941350
root@myself-PC:/home/myself# cd ~
root@myself-PC:~# pwd
/root
root@myself-PC:~#
```

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。

答：ls 命令用于显示指定工作目录下之内容（列出目前工作目录所含之文件及子目录）。

```
root@myself-PC:~# ls
Desktop Documents Downloads Music Pictures Videos 模板
```

下面解释一些选项的作用：

- (1) -l：除文件名称外，亦将文件型态、权限、拥有者、文件大小等资讯详细列出。

```
root@myself-PC:~# ls -l
总用量 28
drwxr-xr-x 2 root root 4096 4月 24 22:50 Desktop
drwxr-xr-x 2 root root 4096 2月 4 01:37 Documents
drwxr-xr-x 2 root root 4096 2月 4 01:37 Downloads
drwxr-xr-x 2 root root 4096 2月 4 01:37 Music
drwxr-xr-x 2 root root 4096 2月 4 01:37 Pictures
drwxr-xr-x 2 root root 4096 2月 4 01:40 Videos
drwxr-xr-x 2 root root 4096 4月 24 22:50 模板
```

- (2) -a：显示所有文件及目录（ls 内定将文件名或目录名称开头为"."的视为隐藏档，不会列出）

```
root@myself-PC:~# ls -a
.      .bash_history  .config  Documents  .gnupg  .IntelliJ IDEA2018.2  Music  Videos
..     .bashrc       .dbus    Downloads  .gtkrc-2.0  .java  Pictures  .viminfo
.android  .cache  .gnome2-priv  gvfs  local  profile  模板
```

- (3) -li：列出文件或目录的索引节点号

```
root@myself-PC:~# ls -li
1705811 Desktop 1705813 Downloads 1705815 Pictures 1712129 模板
1705812 Documents 1705814 Music 1705818 Videos
```

可以发现，在使用 -a 选项的时候列出了"."和".."目录

其中“.”代表当前目录，可以通过 `cd` 命令测试：

```
→ ~ pwd
/home/myself
→ ~ cd .
→ ~ pwd
/home/myself
```

“..”代表上一级目录：

```
→ ~ pwd
/home/myself
→ ~ cd ..
→ /home pwd
/home
```

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。

答：

```
→ Desktop mkdir mydir
→ Desktop cd mydir
→ mydir touch myfile1
→ mydir touch myfile2
→ mydir ls
myfile1 myfile2
→ mydir cd .pwd
→ Desktop rm mydir
rm: 无法删除 'mydir': 是一个目录
→ Desktop rmdir mydir
rmdir: 删除 'mydir' 失败: 目录非空
```

经测试可以知道，`rm` 命令用于删除文件，不能删除目录，而 `rmdir` 命令用于删除空目录（目录中没有文件）

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。

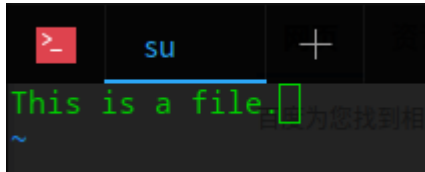
答：以 `root` 权限创建一个文件并查看其权限如下：

```
→ mydir sudo touch myfile
→ mydir ls -l
总用量 0
-rw-r--r-- 1 root root 0 6月 10 00:48 myfile
```

可以看到其默认权限为 `-rw-r--r--(644)`。

使用 `vi` 编辑该文件如下：

```
root@myself-PC:/home/myself/Desktop/mydir# vi myfile
```

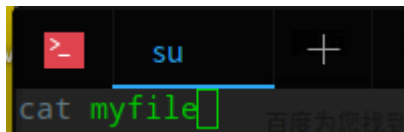


然后该文件权限改为只有用户可读，其他权限均无：

```
root@myself-PC:/home/myself/Desktop/mydir# ls -l
总用量 4
-rw-r--r-- 1 root root 16 6月 10 01:01 myfile
root@myself-PC:/home/myself/Desktop/mydir# chmod 400 myfile
root@myself-PC:/home/myself/Desktop/mydir# ls -l
总用量 4
-r----- 1 root root 16 6月 10 01:01 myfile
```

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：

```
root@myself-PC:/home/myself/Desktop/mydir# su root
root@myself-PC:/home/myself/Desktop/mydir# touch 1.sh
root@myself-PC:/home/myself/Desktop/mydir# vi 1.sh
```



将脚本文件按设置为所有用户可执行：

```
→ mydir ls -l
总用量 8
-rw-r--r-- 1 root root 11 6月 10 01:05 1.sh
-r----- 1 root root 16 6月 10 01:01 myfile
→ mydir sudo chmod 755 1.sh
→ mydir ls -l
总用量 8
-rwxr-xr-x 1 root root 11 6月 10 01:05 1.sh
-r----- 1 root root 16 6月 10 01:01 myfile
```

以 root 身份登录执行脚本，观察结果：

```
→ mydir su root
密码：
root@myself-PC:/home/myself/Desktop/mydir# ./1.sh
This is a file.
```

以普通用户身份登录执行脚本，观察结果：

```
→ mydir ./1.sh
cat: myfile: 权限不够
```

为 cat 文件加 SUID 权限：

```

→ mydir su root
密码:
root@myself-PC:/home/myself/Desktop/mydir# exit
exit
→ mydir rm ./cat
→ mydir su root
密码:
root@myself-PC:/home/myself/Desktop/mydir# cp /bin/cat .
root@myself-PC:/home/myself/Desktop/mydir# chmod u+s cat
root@myself-PC:/home/myself/Desktop/mydir# ls -l
总用量 44
-rwxr-xr-x 1 myself myself 13 6月 10 01:39 1.sh
-rwsr-xr-x 1 root root 35616 6月 10 01:51 cat
-r----- 1 root root 16 6月 10 01:01 myfile

```

修改 1.sh 内容如下，用当前目录的 cat 文件替换：

```

vi ./1.sh
./cat myfile
~

```

以 root 身份登录执行脚本，观察结果：

```

→ mydir su root
密码:
root@myself-PC:/home/myself/Desktop/mydir# ./1.sh
This is a file.

```

以普通用户身份登录执行脚本，观察结果：

```

→ mydir ./1.sh
This is a file.

```

5. 为一个已经存在的文件分别创建多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？

答：

创建硬链接：

```

root@myself-PC:/home/myself/Desktop/mydir# ls
myfile
root@myself-PC:/home/myself/Desktop/mydir# ln myfile hardlink1
root@myself-PC:/home/myself/Desktop/mydir# ln myfile hardlink2
root@myself-PC:/home/myself/Desktop/mydir# ls -l
总用量 12
-r----- 3 root root 16 6月 10 01:01 hardlink1
-r----- 3 root root 16 6月 10 01:01 hardlink2
-r----- 3 root root 16 6月 10 01:01 myfile

```

创建符号链接：

```

root@myself-PC:/home/myself/Desktop/mydir# ls -l
总用量 12
-r----- 3 root root 16 6月 10 01:01 hardlink1
-r----- 3 root root 16 6月 10 01:01 hardlink2
-r----- 3 root root 16 6月 10 01:01 myfile
root@myself-PC:/home/myself/Desktop/mydir# ln -s myfile softlink1
root@myself-PC:/home/myself/Desktop/mydir# ln -s myfile softlink2
root@myself-PC:/home/myself/Desktop/mydir# ls -l
总用量 12
-r----- 3 root root 16 6月 10 01:01 hardlink1
-r----- 3 root root 16 6月 10 01:01 hardlink2
-r----- 3 root root 16 6月 10 01:01 myfile
lrwxrwxrwx 1 root root 6 6月 10 09:30 softlink1 -> myfile
lrwxrwxrwx 1 root root 6 6月 10 09:30 softlink2 -> myfile

```

删除硬链接:

```

root@myself-PC:/home/myself/Desktop/mydir# ls -l
总用量 12
-r----- 3 root root 16 6月 10 01:01 hardlink1
-r----- 3 root root 16 6月 10 01:01 hardlink2
-r----- 3 root root 16 6月 10 01:01 myfile
lrwxrwxrwx 1 root root 6 6月 10 09:30 softlink1 -> myfile
lrwxrwxrwx 1 root root 6 6月 10 09:30 softlink2 -> myfile
root@myself-PC:/home/myself/Desktop/mydir# rm hardlink1
root@myself-PC:/home/myself/Desktop/mydir# ls
hardlink2  myfile  softlink1  softlink2
root@myself-PC:/home/myself/Desktop/mydir# rm -f hardlink2
root@myself-PC:/home/myself/Desktop/mydir# ls
myfile  softlink1  softlink2

```

删除符号链接:

```

root@myself-PC:/home/myself/Desktop/mydir# ls
myfile  softlink1  softlink2
root@myself-PC:/home/myself/Desktop/mydir# rm softlink1
root@myself-PC:/home/myself/Desktop/mydir# ls
myfile  softlink2
root@myself-PC:/home/myself/Desktop/mydir# rm softlink2
root@myself-PC:/home/myself/Desktop/mydir# ls
myfile

```

硬链接概念

硬链接(hard link, 也称链接)就是一个文件的一个或多个文件名

硬链接建立起来后, 源文件和链接文件同步, 修改任何一方文件都会被修改

建立链接可以节省空间, 只需维护链接关系, 不需要拷贝文件

硬链接和符号链接的本质区别

硬链接可认为是一个文件拥有两个文件名;而符号链接则是系统新建一个链接文件, 此文件

指向其所要指的文件

硬链接的局限性

- 符号链接可以跨文件系统；硬链接不可以
- 符号链接可以对一个不存在的文件进行链接；硬链接不可以
- 符号链接可以对目录进行连接，硬链接不可以

符号链接克服了硬链接中的局限性。基于此，重点关注符号链接。符号链接又叫符号链接，相当于 windows 中的快捷方式。

建立符号链接

`ln -s src_file ln_file`

删除符号链接

`rm ln_file`

注：对于目录符号链接

`rm ln_dir` 是删除符号链接

`rm ln_dir/` 是删除目录 `ln_dir` 中的文件，当然源目录中的文件也会删除

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）。

答：

```
root@myself-PC:~/home/myself/Desktop/mydir# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime) [root@centos7 home]# more 2
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime) hello, this is 1.txt!
udev on /dev type devtmpfs (rw,nosuid,relatime,size=3989560k,nr_inodes=997390,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=603072k,mode=755)
/dev/sda2 on / type ext4 (rw,relatime,data=ordered)  /dev/sda2 是系统盘分区点，可以看见
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755) rw-r--r-- 2 root
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)
fstor on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
efivarfs on /sys/firmware/efi/vars type efivarfs (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/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/dma type cgroup (rw,nosuid,nodev,noexec,relatime,dma)
cgroup on /sys/fs/cgroup/bkfs type cgroup (rw,nosuid,nodev,noexec,relatime,bkfs)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=29,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=17280)
queue on /dev/queue type queue (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugetlb type hugetlbfs (rw,relatime,pagesize=2M)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
configs on /sys/kernel/config type configs (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
/dev/sda1 on /boot/efi type vfat (rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=803008k,mode=700,uid=1000,gid=1000)
gfis-fuse on /run/user/1000/gvfs type fuse.gfis-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
/dev/sda4 on /media/myself/windows 10 type fuseblk (rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other,blksize=4096,uhelper=udisks2)
/dev/sda3 on /media/myself/cfe219bf-5049-3619-9547-2fceb5dba903 type hfsplus (ro,nosuid,nodev,relatime,umask=22,uid=1000,gid=1000,nls=utf8,uhelper=udisks2)
/dev/sdb1 on /media/myself/Software type fuseblk (rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other,blksize=4096,uhelper=udisks2)
jetbrains-toolbox on /tmp/.mount_jetbraQuwaxm type fuse.jetbrains-toolbox (ro,nosuid,nodev,relatime,user_id=1000,group_id=1000)
devfuse on /run/user/1000/doc type fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
```

对应设备文件以及设备和分区索引节点）。

```

root@myself-PC:/home/myself/Desktop/mydir# df -a
文件系统 1K-块 已用 可用 已用% 挂载点
sysfs 0 0 0 - /sys
proc 0 0 0 - /proc
udev 3989560 0 3989560 0% /dev
devpts 0 0 0 - /dev/pts
tmpfs 803072 3056 800016 1% /run
/dev/sdb2 76883716 46725164 26210060 65% /
securityfs 0 0 0 - /sys/kernel/security
tmpfs 4015356 49780 3965576 2% /dev/shm
tmpfs 5120 4 5116 1% /run/lock
tmpfs 4015356 0 4015356 0% /sys/fs/cgroup
cgroup2 0 0 0 - /sys/fs/cgroup/unified
cgroup linux下的svn常用命令使用指南 0 0 0 - /sys/fs/cgroup/systemd
pstore 0 0 0 - /sys/fs/pstore
efivarfs 面试笔试动态规划问题--python篇 0 0 0 - /sys/firmware/efi/efivars
bpf 0 0 0 - /sys/fs/bpf
cgroup 阿里云飞天系统的技术架构 0 0 0 - /sys/fs/cgroup/freezer
cgroup 0 0 0 - /sys/fs/cgroup/cpu,cpuacct
cgroup python实现贪心算法 0 0 0 - /sys/fs/cgroup/pids
cgroup 0 0 0 - /sys/fs/cgroup/hugetlb
cgroup 0 0 0 - /sys/fs/cgroup/rdma
cgroup 个人分类 0 0 0 - /sys/fs/cgroup/blkio
cgroup 0 0 0 - /sys/fs/cgroup/memory
cgroup openstack 0 0 16 0 - /sys/fs/cgroup/net_cls,net_prio
cgroup openstack_nova 0 0 3 0 - /sys/fs/cgroup/cpuset
systemd-1 0 0 0 - /proc/sys/fs/binfmt_misc
mqueue ubuntu 0 0 6 0 - /dev/mqueue
debugfs centos 0 0 6 0 - /sys/kernel/debug
hugetlbfs 0 0 0 - /dev/hugepages
binfmt_misc linux 0 0 27 0 - /proc/sys/fs/binfmt_misc
configfs 0 0 0 - /sys/kernel/config
fusectl 0 0 0 - /sys/fs/fuse/connections
/dev/sda1 306008 115392 190616 38% /boot/efi
tmpfs 803068 68 803000 1% /run/user/1000
gvfsd-fuse 0 0 0 - /run/user/1000/gvfs
/dev/sda4 91282988 85412020 5870968 94% /media/myself/windows_10
/dev/sda3 158336000 115461604 42874396 73% /media/myself/cfe219bf-5049-3619-9547-2fceb5dba903
/dev/sdb1 165555344 161587288 3968056 98% /media/myself/Software
jetbrains-toolbox 2018年8月 0 0 1 0 - /tmp/.mount_jetbraQuwaxm
/dev/fuse 0 0 0 - /run/user/1000/doc

```