

深圳大学实验报告

课程名称： 操作系统

实验项目名称： 实验四 文件系统实验

学 院： 计算机与软件学院

专 业： 计算机科学与技术

指导教师： 阮元

报告人： 刘睿辰 学号： 2018152051 班级： 数计班

实 验 时 间： 2021.6.16-2021.6.19

实验报告提交时间： 2021.6.19

一、实验目的与要求：

1. 了解 Linux 文件命令行操作命令；
2. 了解 Linux ext3 文件系统上的软硬链接。

二、方法、步骤：

1. 可以使用 Linux 或其它 Unix 类操作系统；
2. 学习该操作系统提供的文件系统的命令行接口；
3. 学习文件的软硬链接的使用。

三、实验环境

1. 硬件：桌面 PC；
2. 操作系统：Linux；
3. 实验平台：Ubuntu 18.04。

四、实验过程及内容：

1. 学习使用 Linux 文件系统提供的的 ls 、 touch、rm、cp、mv、mkdir、df 等命令（希望尽量涵盖各种满足日常编程所需操作），记录相关命令执行结果。

1) ls 指令

ls 指令作为最常见的指令，通过它，我们可以知道目录的内容，以及各种各样重要文件和目录的属性。

例如，当我们在 Linux 环境下输入 ls 的时候，会打印出当前目录的所有文件夹，如图 1 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab$ ls
com-exp1  exp1  exp2  exp3  exp4
```

图 1. ls 指令打印当前目录所有子文件夹

当然，ls 指令也可以打印别的目录的文件夹，例如如果我们想打印出 exp2 的内容，就在 ls 后面加上这个文件夹的名字，如图 2 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab$ ls exp2
Loop      Loop3.c  p2          RR-FIFO-sched.c  sched-2770
Loop1.c   Loop.c   p3          RR-FIFO.sh       sched-2771
Loop2.c   p1       RR-FIFO-sched  Run-NICE.sh      sched-2773
```

图 2. ls 指令打印文件夹内部文件信息

事实上 ls 指令后面可加的参数又很多，常见的参数如表 1 所示的参数。

选项	长选项	描述
-a	--all	列出所有文件，甚至包括文件名以圆点开头的默认会被隐藏的隐藏文件。
-d	--directory	通常，如果指定了目录名，ls 命令会列出这个目录中的内容，而不是目录本身。把这个选项与 -l 选项结合使用，可以看到所指定目录的详细信息，而不是目录中的内容。
-F	--classify	这个选项会在每个所列出的名字后面加上一个指示符。例如，如果名字是目录名，则会加上一个 '/' 字符。
-h	--human-readable	当以长格式列出时，以人们可读的格式，而不是以字节数来显示文件的大小。
-l		以长格式显示结果。
-r	--reverse	以相反的顺序来显示结果。通常，ls 命令的输出结果按照字母升序排列。
-S		命令输出结果按照文件大小来排序。
-t		按照修改时间来排序。

表 1. ls 指令常见参数

例如，我们使用 ls -l 指令，结果如图 3 所示

```

ruichen@ruichen-virtual-machine:~/桌面/oslab$ ls -l
总用量 20
drwxr-xr-x 3 ruichen ruichen 4096 4月 27 22:30 com-exp1
drwxr-xr-x 2 ruichen ruichen 4096 4月 22 12:50 exp1
drwxr-xr-x 2 ruichen ruichen 4096 5月 21 12:41 exp2
drwxr-xr-x 2 ruichen ruichen 4096 6月 5 16:56 exp3
drwxr-xr-x 2 ruichen ruichen 4096 6月 16 11:54 exp4

```

图 3. ls -l 指令输出结果

在图 3 中，第一列参数代表对于文件的访问权限。第一个字符指明文件类型。在不同类型之间，开头的“-”说明是一个普通文件，d 则表明是一个目录。第二列表示硬链接个数，第三列表示文件所有者的用户名，第四列文件所属用户组的名字。第五列表示以字节数表示的文件大小。第六列表示上次修改文件的时间和日期。最后一列就是文件名称。

我们再来尝试-t 参数，这个按照修改时间来进行排序，结果如图 4 所示。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab$ ls -t
exp4 exp3 exp2 com-exp1 exp1

```

图 4. ls -t 指令输出结果

我们再来尝试-S 参数，这个会按照文件大小来进行排序，如图 5 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab$ ls -S
com-exp1  exp1  exp2  exp3  exp4
```

图 5. ls -S 指令输出结果

2) touch 命令

touch 命令可以创建不存在的文件,如图 6 所示,在 exp4 这个文件夹中我们用 touch 可以新建 3 个空白文件。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ touch 1.txt 2.txt 3.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 0
-rw-r--r-- 1 ruichen ruichen 0 6月 16 21:42 1.txt
-rw-r--r-- 1 ruichen ruichen 0 6月 16 21:42 2.txt
-rw-r--r-- 1 ruichen ruichen 0 6月 16 21:42 3.txt
```

图 6. 使用 touch 来新建三个空白文件

我们使用—help 参数来打印 touch 的参数,可以看到 touch 的参数意义,如图 7 所示。

```

必选参数对长短选项同时适用。
-a          只更改访问时间
-c, --no-create 不创建任何文件
-d, --date=字符串 使用指定字符串表示时间而非当前时间
-f          (忽略)
-h, --no-dereference 会影响符号链接本身,而非符号链接所指示的目的地
              (当系统支持更改符号链接的所有者时,此选项才有用)
-m          只更改修改时间
-r, --reference=FILE use this file's times instead of current time
-t STAMP     use [[CC]YY]MMDDhhmm[.ss] instead of current time
              --time=WORD change the specified time:
              WORD is access, atime, or use: equivalent to -a
              WORD is modify or mtime: equivalent to -m
--help      显示此帮助信息并退出
--version   显示版本信息并退出

请注意, -d 和 -t 选项可接受不同的时间/日期格式。

```

图 7. touch 各参数的作用

我们看到-t 指令用于修改文件的时间戳,所以我们使用-t 指令来新建文件并规定其时间戳,如图 8 所示,我们建立文本文件 4.txt,设置其时间戳为 2021.1.1 零时零分零秒,再通过 ls 指令查看,时间戳设置成功。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ touch -t 202101010000.00 4.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 0
-rw-r--r-- 1 ruichen ruichen 0 6月 16 22:08 1.txt
-rw-r--r-- 1 ruichen ruichen 0 6月 16 22:08 2.txt
-rw-r--r-- 1 ruichen ruichen 0 6月 16 22:08 3.txt
-rw-r--r-- 1 ruichen ruichen 0 1月 1 00:00 4.txt
```

图 8. 设置时间戳成功

-t 参数不加-c 参数的话会新建一个文件,如果加上这个-c 参数的话会修改已存在的文件的时间戳。例如,我们用如图 9 所示的指令来修改 1.txt 的时间戳,再用 ls 指令查看,时间戳修改成功。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ touch -c -t 202101010000.00 1.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 0
-rw-r--r-- 1 ruichen ruichen 0 1月  1 00:00 1.txt
-rw-r--r-- 1 ruichen ruichen 0 6月 16 22:08 2.txt
-rw-r--r-- 1 ruichen ruichen 0 6月 16 22:08 3.txt
-rw-r--r-- 1 ruichen ruichen 0 1月  1 00:00 4.txt

```

图 9. 修改时间戳成功

3) rm 指令

rm 指令用于删除一个文件或者目录。

-i 参数用于删除的时候询问，如图 10 所示。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ rm -i 4.txt
rm: 是否删除普通空文件 '4.txt'? y
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 0
-rw-r--r-- 1 ruichen ruichen 0 1月  1 00:00 1.txt
-rw-r--r-- 1 ruichen ruichen 0 6月 16 22:08 2.txt
-rw-r--r-- 1 ruichen ruichen 0 6月 16 22:08 3.txt

```

图 10. rm -i 指令

-f 参数不会询问是否删除，哪怕文档设置为仅读，如图 11 所示。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ rm -f 3.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 0
-rw-r--r-- 1 ruichen ruichen 0 1月  1 00:00 1.txt
-rw-r--r-- 1 ruichen ruichen 0 6月 16 22:08 2.txt

```

图 11. rm -f 指令

-r 参数会将目录以及文档全部删除，如图 12 所示。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls
1.txt 2.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mkdir 1
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls
1 1.txt 2.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ rm 1
rm: 无法删除'1': 是一个目录
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ rm -r 1
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls
1.txt 2.txt

```

图 12. rm -f 指令

特别地，如果-r 参数后面加*号，代表删除该目录下所有文件以及路径，且不可恢复，如图 13 所示。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mkdir 1 2 3
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls
1 1.txt 2 2.txt 3
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ rm -r *
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls

```

图 13. rm -r *指令

-v 参数可以配合其他参数进行使用，它会使得操作结束之后打印出执行了什么操作，如图 14 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mkdir 1
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ rm -r -v 1
removed directory '1'
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ touch 1.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ rm -i -v 1.txt
rm: 是否删除普通空文件 '1.txt'? y
已删除 '1.txt'
```

图 14. rm -v 指令

4) cp 指令

cp 指令用于复制一个源文件或文件夹到新的文件夹。

例如，我们要复制单个文件到一个另一个文件，如图 15 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cp 1.txt 1-dup.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 20
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:21 1-dup.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 1.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 2.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 3.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 4.txt
```

图 15. cp 指令的使用

如果我们想要复制文件到当前工作目录下的文件夹，则使用图 16 的指令

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mkdir 1
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cp 1.txt 1/
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cd 1
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/1$ ls
1.txt
```

图 16. 用 cp 指令复制文件到当前工作目录下的文件夹

cp 指令也有很多参数以供选择。例如，用 -r 参数可以复制整个文件夹。再刚刚的文件夹 1 中有 1.txt 这个文件，现在我们将文件夹 1 复制到 1-dup 这个文件夹，如图 17 所示，复制成功。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cp -r 1/ 1-dup
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 28
drwxr-xr-x 2 ruichen ruichen 4096 6月 16 23:25 1
drwxr-xr-x 2 ruichen ruichen 4096 6月 17 00:38 1-dup
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:21 1-dup.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 1.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 2.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 3.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 4.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cd 1-dup
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/1-dup$ ls
1.txt
```

图 17. cp -r 指令

在图 17 中我们看到文件夹 1 和文件夹 1-dup 的访问时间不一样。事实上，-p 参数可以将修改时间和访问权限也复制到新文件中，如图 18 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cp -p -r 1/ 1-dup
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 28
drwxr-xr-x 2 ruichen ruichen 4096 6月 16 23:25 1
drwxr-xr-x 2 ruichen ruichen 4096 6月 16 23:25 1-dup
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:21 1-dup.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 1.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 2.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 3.txt
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:16 4.txt
```

图 18. cp -p 指令

5) mv 指令

mv 指令用来为文件或目录改名、或将文件或目录移入其它位置。

事实上，mv 操作可以通过移动文件夹到一个不存在的文件夹来实现文件夹的改名。因为这是 mv 的默认操作，如果目的文件夹存在则进行移动；如果目的文件夹不存在则进行更名，不再新建一个文件夹用来作接收文件夹。更名操作如图 19 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mv 1 1-renamed
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls
1-dup 1-dup.txt 1-renamed 1.txt 2.txt 3.txt 4.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mv 1.txt 1-renamed.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls
1-dup 1-dup.txt 1-renamed 1-renamed.txt 2.txt 3.txt 4.txt
```

图 19. 文件夹更名

mv 操作也有很多参数可以设置。

-b: 当目标文件或目录存在时，在执行覆盖前，会为其创建一个备份，如图 20 所示，2.txt~就是原来 2.txt 的备份。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mv -b 1-renamed.txt 2.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls
1-dup 1-dup.txt 1-renamed 2.txt 2.txt~ 3.txt 4.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cat 2.txt~
This is 2.txt that is going to be copied.
```

图 20. mv -b 指令

-f: 强制覆盖。我们将 3.txt 移动到 4.txt，则不会有任何提示信息直接就会覆盖，如图 21 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mv -f 3.txt 4.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cat 4.txt
This is 3.txt that is going to be copied.
```

图 21. mv -f 指令

我们将 1-dup 文件夹移动到 1-renamed 文件夹中，如图 22 所示，在 1-renamed 文件夹中已经有了 1-dup 文件夹。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mv 1-dup/ 1-renamed
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cd 1-renamed
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/1-renamed$ ls -l
总用量 8
drwxr-xr-x 2 ruichen ruichen 4096 6月 16 23:25 1-dup
-rw-r--r-- 1 ruichen ruichen 42 6月 16 23:25 1.txt

```

图 22. 用 mv 指令移动文件夹

6) mkdir 指令

mkdir 指令用于创建文件夹。之前的 touch 指令可以用于创建不存在的文件，mkdir 指令用于创建不存在的文件夹。

我们创建一个名为 1 的文件夹，并将 4 个 txt 文件移动进去，再进入查看，结果如图 23 所示，文件夹创建成功。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mkdir 1
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mv 1.txt 2.txt 3.txt 4.txt 1
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cd 1
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/1$ ls
1.txt 2.txt 3.txt 4.txt

```

图 23. mkdir 指令

mkdir 也可以带参数 -p，判断文件夹是否存在。当然，mkdir 也可以创建子文件夹。当父文件夹不存在而此时没带 -p 参数的时候 Linux 会报错，但是加上 -p 参数之后会自动生成这样一个父文件夹，如图 24 所示。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mkdir 2/2
mkdir: 无法创建目录"2/2": 没有那个文件或目录
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mkdir -p 2/2
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cd 2
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/2$ ls
2

```

图 24. mkdir -p 指令

有时候在 mkdir 的过程中，我们想知道文件内部是什么样子的，所以我们希望有一个形象化的表示方法。tree 指令可以帮助我们打印一棵树，父节点为父文件夹，子树为子文件夹。我们先在文件夹 1/1 中创建两个子文件，然后用 tree 来进行打印，如图 25 所示，非常直观地展示了文件夹内部结构，便于我们管理。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ tree
.
├── 1
│   ├── 1
│   │   ├── 11.txt
│   │   └── 22.txt
│   ├── 1.txt
│   ├── 2.txt
│   ├── 3.txt
│   └── 4.txt
└── 2
    └── 2

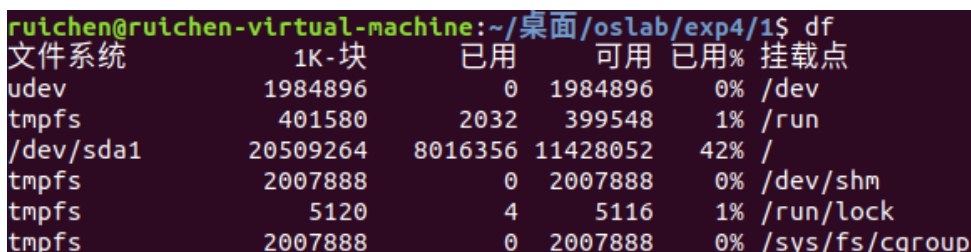
```

图 25. tree 指令打印文件夹内部结构

7) df 指令

df 指令，即将 disk free，用于显示目前在 Linux 系统上的文件系统磁盘使用情况统计。

显示文件系统的磁盘使用情况统计，如图 26 所示。



文件系统	1K-块	已用	可用	已用%	挂载点
udev	1984896	0	1984896	0%	/dev
tmpfs	401580	2032	399548	1%	/run
/dev/sda1	20509264	8016356	11428052	42%	/
tmpfs	2007888	0	2007888	0%	/dev/shm
tmpfs	5120	4	5116	1%	/run/lock
tmpfs	2007888	0	2007888	0%	/sys/fs/cgroup

图 26. df 指令显示文件系统的磁盘使用情况

第一列指定文件系统的名称，第二列指定一个特定的文件系统 1K-块 1K 是 1024 字节为单位的总内存。已用和可用列分别指定已经使用和可以使用的内存量。使用列指定使用的内存的百分比，而最后一列指定安装在指定的文件系统的挂载点。

df 指令也有很多参数。例如，-h 参数用于产生可读的格式 df 命令的输出，如图 27 所示。



文件系统	容量	已用	可用	已用%	挂载点
udev	1.9G	0	1.9G	0%	/dev
tmpfs	393M	2.0M	391M	1%	/run
/dev/sda1	20G	7.7G	11G	42%	/
tmpfs	2.0G	0	2.0G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	2.0G	0	2.0G	0%	/sys/fs/cgroup
tmpfs	393M	16K	393M	1%	/run/user/121
vmhgfs-fuse	118G	102G	17G	87%	/mnt/hgfs
tmpfs	393M	36K	393M	1%	/run/user/1000

图 27. df -h 指令

我们可以看到输出显示的数字形式的千兆字节、兆字节，这使得输出容易阅读和理解。

2. 学习 Linux 文件系统中关于文件硬链接和软链接的概念和相关操作命令,创建软硬链接各一个，给出实验证据表明它们分别是软硬连接。

在学习硬链接和软链接之前我们先来了解 inode 的概念。在 Linux 的文件系统中，保存在磁盘分区中的文件不管是什么类型都给它分配一个编号，这个编号被称之为索引节点号（Node Index），也就是常说的 inode。inode 与文件名关联，下与用户数据库（data block）关联。

硬链接：又称实体链接，指文件名与索引节点号（即 inode 号）的链接（创建一个新的文件,该文件使用 stat 命令查看时，links 显示的是 1），索引节点号（inode 号）可以对应一个或多个文件名，并且这些文件名可以在同一或不同目录。

由于硬链接是直接将文件名与索引节点号（即 inode 号）链接，因此硬链接存在以下几个特点：

- 1) 文件有相同的 **inode** 号及 **data block**, 这使得修改其中一个硬链接文件属性或文件数据时, 其他硬链接文件都会发生相应修改;
- 2) 只能对已存在的文件进行创建;
- 3) 不能跨文件系统 (即分区) 进行创建;
- 4) 不能对目录文件进行创建;
- 5) 删除其中一个硬链接文件时, 不会对其他硬链接文件产生影响

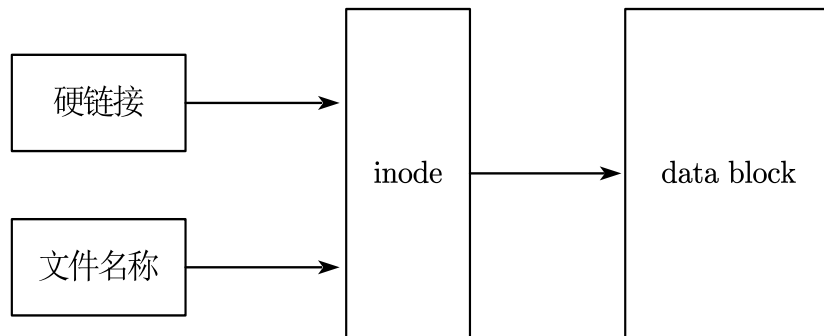


图 28. 硬链接原理

软链接: 有着自己的索引节点号 (即 **inode**) 以及用户数据块 (**data block**), 但用户数据块 (**data block**) 中包含的是另一个文件的位置信息。

由于软链接有着自己的索引节点号 (即 **inode** 号) 以及用户数据块 (**data block**), 因此没有硬链接的诸多限制, 它的特性如下:

- 1) 软链接有自己的文件属性、**inode** 和 **data block**, 但是编辑文件其实就是编辑源文件;
- 2) 可以对不存在的文件或目录进行创建;
- 3) 可以跨文件系统 (即分区) 进行创建, 使用 **ln** 命令跨文件系统创建时, 源文件必须是绝对路径, 否则为死链接;
- 4) 可以对文件或目录文件进行创建;
- 5) 删除软链接并不影响源文件, 但源文件被删除, 则相关软链接文件变为死链接 (**dangling link**), 若源文件 (原地址原文件名) 重新被创建, 则死链接恢复为正常软链接。

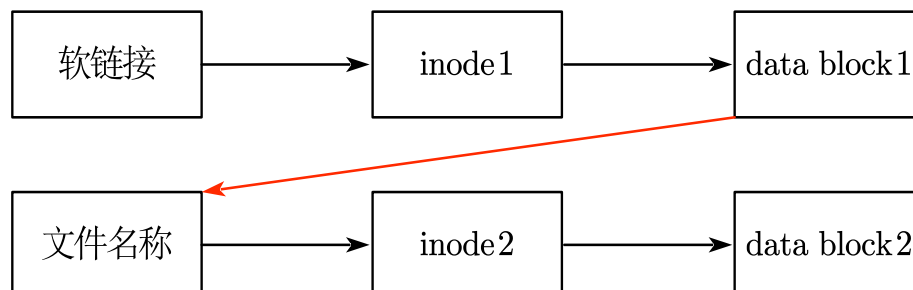


图 29. 软链接原理

接下来我们学习在 **Linux** 系统下建立软硬链接。

在 **Linux** 环境下, 创建链接的方法为 **ln** 指令或 **link**, 我们首先编辑 **1.txt**, 然后用 **stat**

指令查看其状态，可以看到其 inode 以及硬链接个数，如图 30 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ stat 1.txt
 文件：1.txt
 大小：16          块：8          IO 块：4096   普通文件
设备：801h/2049d   Inode：132691    硬链接：1
权限：(0644/-rw-r--r--)  Uid：( 1000/ ruichen)  Gid：( 1000/ ruichen)
最近访问：2021-06-17 21:15:37.834876120 +0800
最近更改：2021-06-17 21:15:37.834876120 +0800
最近改动：2021-06-17 21:15:37.834876120 +0800
创建时间：-
```

图 30. 未进行硬链接的时候的文件状态

然后创建 1.txt 的硬链接，如图 31 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ link 1.txt 1-hardlink
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 8
-rw-r--r-- 2 ruichen ruichen 16 6月 17 21:15 1-hardlink
-rw-r--r-- 2 ruichen ruichen 16 6月 17 21:15 1.txt
```

图 31. 创建 1.txt 的硬链接

接下来我们用 stat 指令来查看这两个文件的状态，如图 32 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ stat 1.txt 1-hardlink
 文件：1.txt
 大小：16          块：8          IO 块：4096   普通文件
设备：801h/2049d   Inode：132691    硬链接：2
权限：(0644/-rw-r--r--)  Uid：( 1000/ ruichen)  Gid：( 1000/ ruichen)
最近访问：2021-06-17 21:15:37.834876120 +0800
最近更改：2021-06-17 21:15:37.834876120 +0800
最近改动：2021-06-17 21:16:17.910279745 +0800
创建时间：-
 文件：1-hardlink
 大小：16          块：8          IO 块：4096   普通文件
设备：801h/2049d   Inode：132691    硬链接：2
权限：(0644/-rw-r--r--)  Uid：( 1000/ ruichen)  Gid：( 1000/ ruichen)
最近访问：2021-06-17 21:15:37.834876120 +0800
最近更改：2021-06-17 21:15:37.834876120 +0800
最近改动：2021-06-17 21:16:17.910279745 +0800
创建时间：-
```

图 32. 查看文件及其硬链接状态

从图 32 中我们看出原文件和它的链接的硬链接都是 2，硬链接文件格式为普通文件，且 inode 的值都是 132691，表示二者共用了一个 inode。而如图 33 所示，两份文件的内容是完全相同的。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cat 1.txt 1-hardlink
This is 1.txt.

This is 1.txt.
```

图 33. 原文件和硬链接的内容完全一致

尽管硬链接建立方便，但由于不能对文件夹来作硬链接，且不可以在不同文件系统的文件间建立链接。所以有时候我们需要来作软链接。软链接克服了硬链接的不足，没有任何文件系统的限制，任何用户可以创建指向目录的符号链接。因而现在更为广泛使用，它具有更大的灵活性，甚至可以跨越不同机器、不同网络对文件进行链

接。

软链接不能用 link 指令，link 指令只能用于硬链接，而 ln 指令后面可以跟参数。软链接构建我们用 ln -s 指令，如图 34 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ln -s 1.txt 1-softlink
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 8
-rw-r--r-- 2 ruichen ruichen 16 6月 17 21:15 1-hardlink
lrwxrwxrwx 1 ruichen ruichen 5 6月 17 21:51 1-softlink -> 1.txt
-rw-r--r-- 2 ruichen ruichen 16 6月 17 21:15 1.txt
```

图 34. ln -s 指令构建软链接

我们用 stat 指令来查看原文件和其软链接的状态信息，如图 35 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ stat 1.txt 1-softlink
文件: 1.txt
大小: 16          块: 8          IO 块: 4096   普通文件
设备: 801h/2049d  Inode: 132691   硬链接: 2
权限: (0644/-rw-r--r--)  Uid: ( 1000/ ruichen)  Gid: ( 1000/ ruichen)
最近访问: 2021-06-17 21:18:11.932420296 +0800
最近更改: 2021-06-17 21:15:37.834876120 +0800
最近改动: 2021-06-17 21:16:17.910279745 +0800
创建时间: -
文件: 1-softlink -> 1.txt
大小: 5          块: 0          IO 块: 4096   符号链接
设备: 801h/2049d  Inode: 132695   硬链接: 1
权限: (0777/lrwxrwxrwx)  Uid: ( 1000/ ruichen)  Gid: ( 1000/ ruichen)
最近访问: 2021-06-17 21:51:50.283269195 +0800
最近更改: 2021-06-17 21:51:50.279269215 +0800
最近改动: 2021-06-17 21:51:50.279269215 +0800
创建时间: -
```

图 35.查看文件及其软链接状态

从图 35 中看出，原文件的硬链接数并没有发生变化，软链接个数也是 1，证明没有新的硬链接产生。我们可以看出软链接的文件格式为符号链接，且 inode 和原文件的 inode 不相等，说明原文件和其软链接并不共用一个 inode。此外，两份文件大小并不相等，一个是 16B，另一个是 5B，也就是“1.txt”这 5 个字符的大小，代表软链接内容并不是原文件的内容而是原文件的名称。

而如图 36 所示，我们也可以给文件夹来构建软链接，这是软链接相对于硬链接的优势。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mkdir 2
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ln -s 2 2-softlink
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls
1-hardlink 1-softlink 1.txt 2 2-softlink
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ ls -l
总用量 12
-rw-r--r-- 2 ruichen ruichen 16 6月 17 21:15 1-hardlink
lrwxrwxrwx 1 ruichen ruichen 5 6月 17 21:51 1-softlink -> 1.txt
-rw-r--r-- 2 ruichen ruichen 16 6月 17 21:15 1.txt
drwxr-xr-x 2 ruichen ruichen 4096 6月 17 22:01 2
lrwxrwxrwx 1 ruichen ruichen 1 6月 17 22:01 2-softlink -> 2
```

图 36. 给文件夹添加软链接，硬链接并不能加在文件夹上

但是软链接也有其弊端。因为链接文件包含有原文件的路径信息，所以当原文件从

一个目录下移到其他目录中，再访问链接文件，系统就找不到了。所以当我们删除了原文件之后，软链接随之失效，如图 37 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cat 1-softlink
This is 1.txt.
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ rm 1.txt
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cat 1-softlink
cat: 1-softlink: 没有那个文件或目录
```

图 37. 删除原文件之后软链接随之失效

但是删除了 1.txt 之后硬链接依然有效，如图 38 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cat 1-hardlink
This is 1.txt.
```

图 38. 删除原文件之后硬链接并未失效

3. 构建如图 39 所示的文件树：

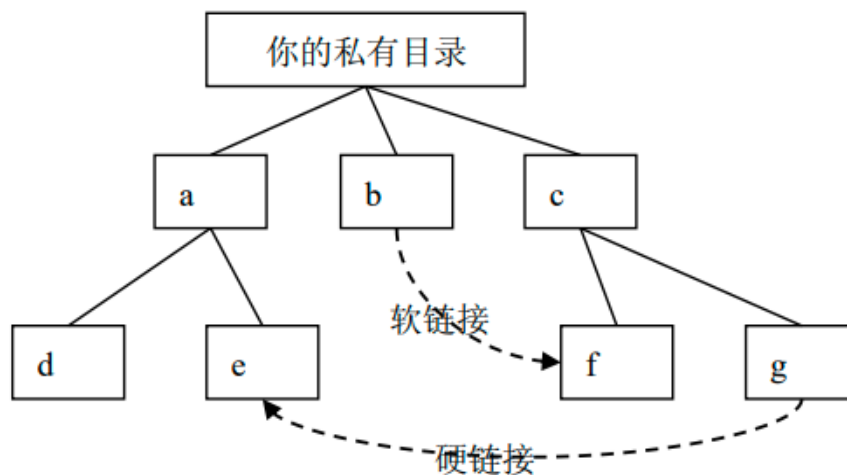


图 39. 目标文件树

分析文件树，可知 b 和 g 都是链接，其中 g 是硬链接，b 是软链接。所以使用 touch 指令和 mkdir 指令建树的时候不要建立 b 和 g，且注意 e 有硬链接，所以 e 不应该是文件夹而是一份文件。所以我们规定第三层子树都是单个文件，第一第二层为文件夹，建树过程如图 40 所示。

```
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ mkdir private
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ cd private
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private$ mkdir a c
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private$ cd a
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private/a$ touch d e
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private/a$ cd ..
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private$ ls
a c
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private$ cd c
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private/c$ touch f
```

图 40. 建立文件树（1）

接下来我们构建两个链接，如图 41 所示。


```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private/c$ link ../a/e g
ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private/c$ ln -s f ../b

```

图 41. 建立文件树（2）

我们用 tree 指令来查看此时文件系统的结构，如图 42 所示。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4$ tree private
private
├── a
│   ├── d
│   └── e
├── b -> f
└── c
    ├── f
    └── g

```

图 42. 文件系统结构

再来看硬链接 e 和 g 的关系，如图 43 所示。两份文件的 inode 的值是一样的（都是 262236），且两份文件的硬链接的个数都是 2。

```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private$ stat a/e c/g
 文件：a/e
 大小：0                块：0                IO 块：4096    普通空文件
设备：801h/2049d      Inode：262236          硬链接：2
权限：(0644/-rw-r--r--) Uid：( 1000/ ruichen)  Gid：( 1000/ ruichen)
最近访问：2021-06-17 22:33:54.030315050 +0800
最近更改：2021-06-17 22:33:54.030315050 +0800
最近改动：2021-06-17 22:38:17.942073448 +0800
创建时间：-
 文件：c/g
 大小：0                块：0                IO 块：4096    普通空文件
设备：801h/2049d      Inode：262236          硬链接：2
权限：(0644/-rw-r--r--) Uid：( 1000/ ruichen)  Gid：( 1000/ ruichen)
最近访问：2021-06-17 22:33:54.030315050 +0800
最近更改：2021-06-17 22:33:54.030315050 +0800
最近改动：2021-06-17 22:38:17.942073448 +0800
创建时间：-

```

图 43. 文件 e 及其硬链接 g

再来看软链接 f 和 b 的关系，如图 44 所示。两份文件的 inode 的值是不一样的（一个是 262237，一个是 262238），且两份文件的硬链接的个数都是 1。


```

ruichen@ruichen-virtual-machine:~/桌面/oslab/exp4/private$ stat c/f b
 文件：c/f
 大小：0                块：0                IO 块：4096      普通空文件
设备：801h/2049d        Inode：262237        硬链接：1
权限：(0644/-rw-r--r--)  Uid：( 1000/ ruichen)  Gid：( 1000/ ruichen)
最近访问：2021-06-17 22:34:44.786287505 +0800
最近更改：2021-06-17 22:34:44.786287505 +0800
最近改动：2021-06-17 22:34:44.786287505 +0800
创建时间：-
 文件：b -> f
 大小：1                块：0                IO 块：4096      符号链接
设备：801h/2049d        Inode：262238        硬链接：1
权限：(0777/lrwxrwxrwx)  Uid：( 1000/ ruichen)  Gid：( 1000/ ruichen)
最近访问：2021-06-17 22:39:34.250314436 +0800
最近更改：2021-06-17 22:39:24.262265017 +0800
最近改动：2021-06-17 22:39:24.262265017 +0800
创建时间：-

```

图 44. 文件 f 及其软链接 b

在图 44 中，软链接的大小为 1B，也就是原文件名称的大小，且 b 的文件格式为符号链接。

五、实验体会

本次实验学习了 Linux 的命令行的相关指令 ls、touch、rm、cp、mv、mkdir、df。此外通过本次实验学习了硬链接与软链接的概念，通过实际实验对比了软链接与硬链接的区别。最后一部分我们通过 Linux 的指令建立了如图 39 所示的文件系统。

本次实验内容虽然不多，但是比较重要，因为文件管理系统非常重要，在日常的操作中文件的管理非常关键，掌握文件管理的方法非常重要。

指导教师批阅意见：

成绩评定：

指导教师签字：
年 月 日

备注：