



# 上海大学

SHANGHAI UNIVERSITY

## 操作系统（二）实验五报告

组	号	第 9 组
姓	名	蔡卓悦
学	号	18120482
实 验 序 号	实验五	
日	期	2020 年 1 月 24 日

## 一、实验目的

- (1) 掌握操作系统中文件分类的概念。
- (2) 了解 Linux 文件系统管理文件的基本方式和特点。
- (3) 学会使用 Linux 文件系统的命令界面和程序界面的基本要领。

## 二、实验环境

经过实验发现，Linux 和 MacOS 操作系统在本实验的操作中有较大区别。所以本实验操作使用的系统为 Ubuntu 操作系统，使用电脑为 MacBook Pro。

## 三、实验内容及其设计与实现

### 1、实验题目

运行命令界面的各命令并观察结果。

用 vi 编写 c 程序（假定程序文件名为 prog1.c）

编译程序

```
$ gcc -o prog1.o prog1.c
```

或

```
$ cc -o prog1.o prog1.c
```

运行

```
$ ./prog1.o
```

观察运行结果并讨论。

## 四、实验操作及结果

1.用 shell 命令查看 Linux 文件类型。思考：Linux 文件类型有哪些？用什么符号表示。

答：Linux 的文件类型共 7 种，它们分别是（括号内字符为 ls -l 命令输出结果中第一列内容）：

- ① (-)普通文件
- ② (d)目录文件
- ③ (p)管道文件
- ④ (l)链接文件
- ⑤ (b)块设备文件
- ⑥ (c)字符设备文件
- ⑦ (s)套接字文件

2. 用 shell 命令了解 Linux 文件系统的目录结构。

执行

```
$ cd /lib
```

```
$ ls -l|more
```

查看/lib 目录的内容，这里都是系统函数。

再查看/etc，这里都是系统设置用的配置文件；/bin 中是可执行程序；/home 下包括了每个用户主目录。

结果如下图所示:

```
caizhuoyue@caizhuoyue-VirtualBox: /etc
total 1108
drwxr-xr-x 3 root root 4096 8月 1 00:31 acpi
-rw-r--r-- 1 root root 3028 8月 1 00:27 adduser.conf
drwxr-xr-x 3 root root 4096 8月 1 00:28 alsa
drwxr-xr-x 2 root root 4096 1月 24 10:08 alternatives
-rw-r--r-- 1 root root 401 7月 17 2019 anacrontab
-rw-r--r-- 1 root root 433 10月 2 2017 apg.conf
drwxr-xr-x 5 root root 4096 8月 1 00:28 apm
drwxr-xr-x 3 root root 4096 8月 1 00:30 apparmor
drwxr-xr-x 7 root root 4096 9月 13 08:52 apparmor.d
drwxr-xr-x 4 root root 4096 8月 1 00:31 appport
-rw-r--r-- 1 root root 769 1月 19 2020 appstream.conf
drwxr-xr-x 7 root root 4096 9月 13 08:53 apt
drwxr-xr-x 3 root root 4096 8月 1 00:31 avahi
-rw-r--r-- 1 root root 2319 2月 25 2020 bash.bashrc
-rw-r--r-- 1 root root 45 1月 26 2020 bash_completion
drwxr-xr-x 2 root root 4096 8月 1 00:31 bash_completion.d
-rw-r--r-- 1 root root 367 4月 15 2020 bindresvport.blacklist
drwxr-xr-x 2 root root 4096 4月 22 2020 binfmt.d
drwxr-xr-x 2 root root 4096 8月 1 00:30 bluetooth
-rw-r----- 1 root root 33 8月 1 00:31 brlapi.key
drwxr-xr-x 7 root root 4096 8月 1 00:30 brltty
-rw-r--r-- 1 root root 26916 3月 4 2020 brltty.conf
drwxr-xr-x 3 root root 4096 8月 1 00:27 ca-certificates
-rw-r--r-- 1 root root 5714 8月 1 00:27 ca-certificates.conf
-rw-r--r-- 1 root root 5713 8月 1 00:27 ca-certificates.conf.dpkg-old
drwxr-xr-x 2 root root 4096 8月 1 00:30 calendar
drwxr-s--- 2 root dip 4096 8月 1 00:30 chatscripts
--More--
```

图 1 Ubuntu 系统/etc 中的文件

```
caizhuoyue@caizhuoyue-VirtualBox: /lib
total 632
drwxr-xr-x 2 root root 4096 8月 1 00:28 accountsservice
drwxr-xr-x 2 root root 4096 8月 1 00:28 apg
drwxr-xr-x 2 root root 4096 8月 1 00:28 apparmor
drwxr-xr-x 5 root root 4096 8月 1 00:27 apt
drwxr-xr-x 3 root root 4096 8月 1 00:29 aspell
drwxr-xr-x 2 root root 4096 1月 24 10:08 bfd-plugins
drwxr-xr-x 2 root root 4096 4月 22 2020 binfmt.d
drwxr-xr-x 2 root root 4096 8月 1 00:29 bluetooth
drwxr-xr-x 2 root root 4096 8月 1 00:29 bolt
drwxr-xr-x 2 root root 4096 8月 1 00:30 brltty
-rwxr-xr-x 1 root root 684 4月 7 2020 cnf-update-db
-rwxr-xr-x 1 root root 3565 4月 7 2020 command-not-found
drwxr-xr-x 2 root root 4096 1月 24 10:08 compat-ld
drwxr-xr-x 2 root root 4096 8月 1 00:27 console-setup
lrwxrwxrwx 1 root root 21 9月 13 08:48 cpp -> /etc/alternatives/cpp
drwxr-xr-x 3 root root 4096 8月 1 00:29 crda
drwxr-xr-x 10 root root 4096 8月 1 00:29 cups
drwxr-xr-x 2 root root 4096 8月 1 00:27 dbus-1.0
drwxr-xr-x 5 root root 4096 8月 1 00:30 debug
drwxr-xr-x 3 root root 4096 8月 1 00:27 dpkg
drwxr-xr-x 2 root root 4096 8月 1 00:27 eject
drwxr-xr-x 3 root root 4096 8月 1 00:29 emacs-en-common
drwxr-xr-x 2 root root 4096 8月 1 00:30 environment.d
drwxr-xr-x 7 root root 4096 8月 1 00:29 evolution-data-server
drwxr-xr-x 2 root root 4096 8月 1 00:27 file
drwxr-xr-x 8 root root 4096 8月 1 00:29 firefox
drwxr-xr-x 5 root root 4096 8月 1 00:29 firefox-addons
--More--
```

图 2 Ubuntu 系统/lib 中的文件

```
caizhuoyue@caizhuoyue-VirtualBox:/home$ ls -l|more
total 4
drwxr-xr-x 17 caizhuoyue caizhuoyue 4096 1月 24 10:11 caizhuoyue
```

图 3 Ubuntu 系统/home 中的文件

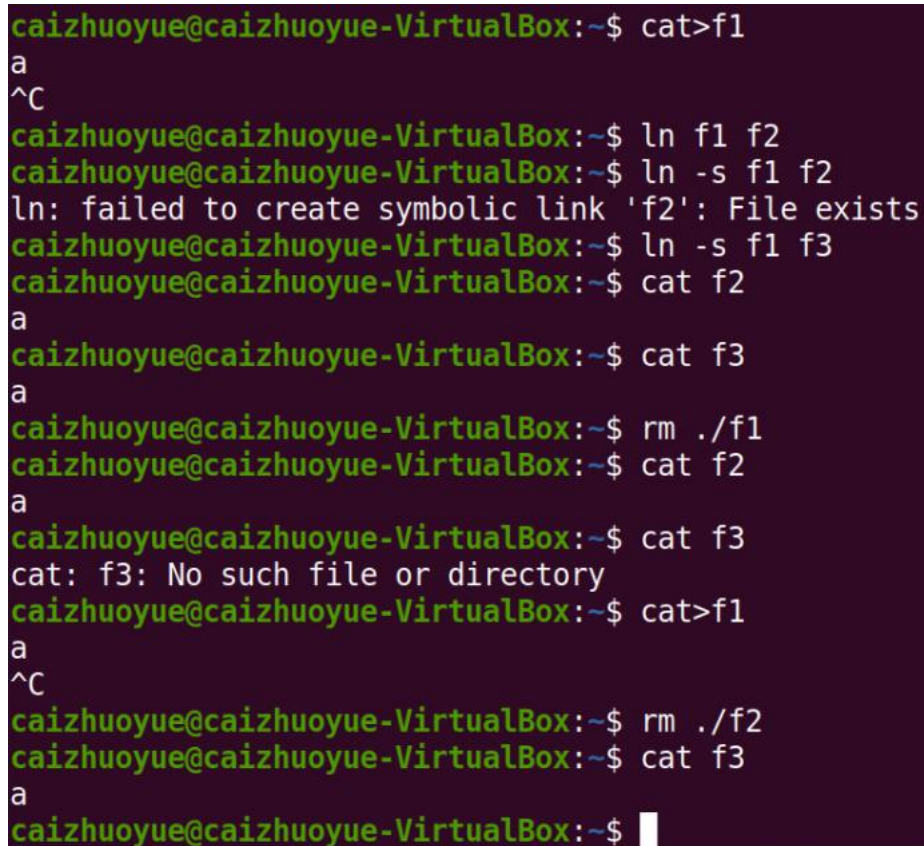


3. 用命令分别建立硬链接文件和符号链接文件。通过 `ls -il` 命令所示的 inode、链接计数观察它们的区别。

找找其他目录中的文件，如：`/home/zzl/mytest.c` 执行

`$ ln /home/zzl/mytest.c myt.c` （建立硬链接文件）

`$ ln -s /home/zzl/mytest.c myt2.c` （建立符号链接文件）

A terminal window with a dark purple background and green text. The prompt is 'caizhuoyue@caizhuoyue-VirtualBox:~\$'. The user enters 'cat>f1', types 'a', and presses Ctrl-C (^C). Then they enter 'ln f1 f2', followed by 'ln -s f1 f2', which results in an error: 'ln: failed to create symbolic link 'f2': File exists'. They then enter 'ln -s f1 f3' and 'cat f2', which outputs 'a'. Next, they enter 'cat f3', which outputs 'a'. Then they enter 'rm ./f1' and 'cat f2', which outputs 'a'. They then enter 'cat f3', which results in an error: 'cat: f3: No such file or directory'. Finally, they enter 'cat>f1', type 'a', press Ctrl-C (^C), enter 'rm ./f2', and 'cat f3', which outputs 'a'. The prompt is now 'caizhuoyue@caizhuoyue-VirtualBox:~\$' with a cursor.

```
caizhuoyue@caizhuoyue-VirtualBox:~$ cat>f1
a
^C
caizhuoyue@caizhuoyue-VirtualBox:~$ ln f1 f2
caizhuoyue@caizhuoyue-VirtualBox:~$ ln -s f1 f2
ln: failed to create symbolic link 'f2': File exists
caizhuoyue@caizhuoyue-VirtualBox:~$ ln -s f1 f3
caizhuoyue@caizhuoyue-VirtualBox:~$ cat f2
a
caizhuoyue@caizhuoyue-VirtualBox:~$ cat f3
a
caizhuoyue@caizhuoyue-VirtualBox:~$ rm ./f1
caizhuoyue@caizhuoyue-VirtualBox:~$ cat f2
a
caizhuoyue@caizhuoyue-VirtualBox:~$ cat f3
cat: f3: No such file or directory
caizhuoyue@caizhuoyue-VirtualBox:~$ cat>f1
a
^C
caizhuoyue@caizhuoyue-VirtualBox:~$ rm ./f2
caizhuoyue@caizhuoyue-VirtualBox:~$ cat f3
a
caizhuoyue@caizhuoyue-VirtualBox:~$
```

图 4 软硬链接操作

**思考：建立硬链接文件和建立符号链接文件有什么区别，体现在哪里？**

答：符号链接也称为软链接，它是包含在文件中的路径名。当系统遇到符号链接时它沿着符号链接提供的路径名前行，然后继续沿着符号链接后面的任何其余路径前行。如果路径名以一个/开始，则系统返回到/（“根”）目录，并从该目录开始沿着路径前行。如果路径名未以/开始，则系统返回到前一级目录，并从那个目录开始沿着符号链接中的路径名前行，硬链接则通过索引节点来进行链接。

#### 4.按照附录 1 中代码编程查看指定文件的 inode 结果

```
joe→ Desktop ▷ ./getinfo.o f1

File:f1
1)On device(major/minor):1 16,inode number:7693519
2)Type:0100000 Permission:00644
3)Owner id:501 Group id:20
4)Number of hard links:1
5)Size:2
6)Last access:Sun Jan 24 09:20:21 2021
   Last modify inode:Sun Jan 24 09:20:21 2021
7)Datablocks:4096 , 8
joe→ Desktop ▷
```

图 5 查看 f1 的 inode 结果

5. 修改父进程创建子进程的程序，用显示程序段、数据段地址的方法，说明子进程继承父进程的所有资源。再用父进程创建子进程，子进程调用其它程序的方法进一步证明执行其它程序时，程序段发生的变化。按照附录 2 中代码编程查看结果。

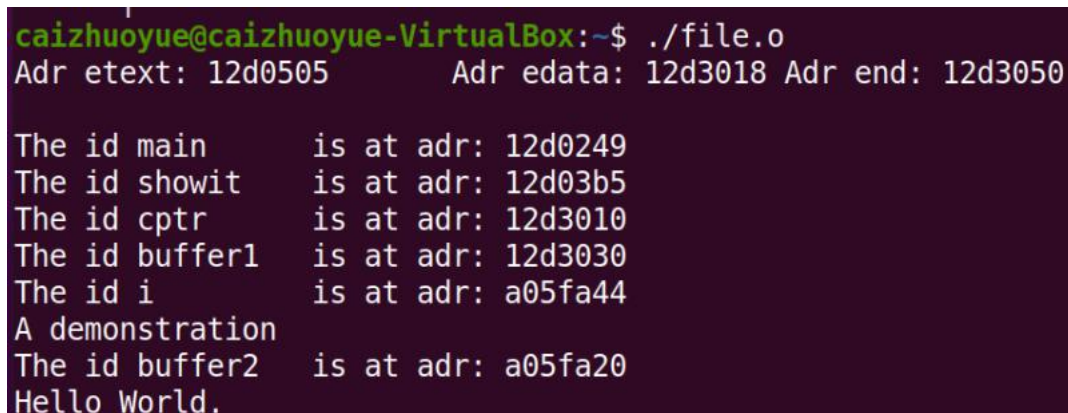
结果如下图所示:

```
caizhuoyue@caizhuoyue-VirtualBox:~$ ./process.o
etext:90b80205 edata:90b83010 end:%6
```

图 6 父子进程运行结果

## 6.按照附录 3 中代码编写涉及流文件的程序查看结果。

结果如下图所示:



```
caizhuoyue@caizhuoyue-VirtualBox:~$ ./file.o
Adr etext: 12d0505      Adr edata: 12d3018 Adr end: 12d3050

The id main      is at adr: 12d0249
The id showit    is at adr: 12d03b5
The id cptr      is at adr: 12d3010
The id buffer1   is at adr: 12d3030
The id i         is at adr: a05fa44
A demonstration
The id buffer2   is at adr: a05fa20
Hello World.
```

图 7 流文件的程序查看结果

## 五、实验结果

通过上述实验五的过程,我掌握了操作系统中文件分类的概念。同时也了解了 Linux 文件系统管理文件的基本方式和特点,并学会了使用 Linux 文件系统的命令界面和程序界面的基本要领。巩固了先前的知识并有了新的收获。

## 六、收获与体会

本次实验五是关于 Linux 文件系统的。

在上次实验中我掌握了操作系统中文件分类的概念。而在本次我更加了解了 Linux 文件系统管理文件的基本方式和特点,也学会了使用 Linux 文件系统的命令界面和程序界面的基本要领。这些实际的操作研究都是从书本中很难真正学到的知识,必须要通过实地的动手编程和分析结果才能有的体会,它们也加深了我对于操作系统方面理论知识的理解的记忆。



## 附录1

```
#include<sys/stat.h>

#include<sys/types.h>

#include<sys/sysmacros.h>

#include<stdio.h>

#include<time.h>

#include<unistd.h>

#include<string.h>

#include<errno.h>

#define TIME_STRING_LEN 50

char *time2String (time_t tm,char *buf)

{struct tm *local;

    local=localtime(&tm);

    strftime(buf,TIME_STRING_LEN,"%c",local);

    return buf;

}

int ShowFileInfo(char *file)

{struct stat buf;

    char timeBuf[TIME_STRING_LEN];

    if(lstat(file,&buf))

    {perror("lstat()error");

        return 1;

    }

    printf("\nFile:%s\n",file);

    printf("1)On device(major/minor):%d %d,inode number:%ld\n",
```

```

major(buf.st_dev),minor(buf.st_dev),buf.st_ino);

printf("2)Type:%07o\t Permission:%05o\n",

buf.st_mode & S_IFMT,buf.st_mode & ~(S_IFMT));

printf("3)Over id:%d\t Group id:%d\t \n4)Number of hard links:%d\n",

buf.st_uid,buf.st_gid,buf.st_nlink);

    printf("5)Size:%ld\t \n6)Last access:%s",buf.st_size,time2String(buf.st_atime,timeBuf));

    printf("\n  Last modify inode:%s\n\n",time2String(buf.st_atime,timeBuf));

    printf("7)Datablocks:%d , %d\t\n",buf.st_blksize,buf.st_blocks);

    return 0;

}

int main(int argc,char *argv[])

{int i,ret;

for(i=1;i<argc;i++)

{ret=ShowFileInfo(argv[i]);

    if(argc-i>1)printf("\n");

}

return ret;

}

```

## 附录2

```

#include<stdio.h>

extern int etext, edata, end;

main()

{

printf("etext:%6x \t edata:%6x \t end:%6 \n",&etext,&edata,&end);

}

```

## 附录3

```

#include<stdio.h>

#include<string.h>

#include<sys/types.h>

#include<stdlib.h>

#include<unistd.h>

#define SHW_ADR(ID,I) printf("The id %s \t is at adr:%8x\n",ID,&I);

extern int etext, edata, end;

char *cptr="Hello World.\n";

char buffer1[25];

main()

{ void showit(char *);

  int i=0;

  printf("Adr etext:%8x\t Adr edata:%8x Adr end:%8x\n\n",&etext,&edata,&end);

  SHW_ADR("main",main);

  SHW_ADR("showit",showit);

  SHW_ADR("cptr",cptr);

  SHW_ADR("buffer1",buffer1);

```

```

    SHW_ADR("i", i);

    strcpy(buffer1, "A demonstration\n");

    write(1, buffer1, strlen(buffer1)+1);

    for(; i<1; ++i)

        showit(cp1);
}

void

showit(char *p)

{
    char *buffer2;

    SHW_ADR("buffer2", buffer2);

    if ((buffer2=(char *)malloc((unsigned)(strlen(p)+1)))!=NULL)

    {
        strcpy(buffer2, p);

        printf("%s", buffer2);

        free(buffer2);
    }

    else

    {
        printf("Allocation error.\n");

        exit(1);
    }
}

```