

# 浙江大学实验报告

专业: 计算机科学与技术  
姓名: 吴同  
学号: 3170104848  
日期: 2019年7月18日

课程名称: Linux 程序设计 指导老师: 季江民 电子邮件: wutongcs@zju.edu.cn  
实验名称: Linux shell 基本命令 实验类型: 验证型 联系电话: 18888922355

## 一、实验环境

### 1. 硬件环境

实验所用的计算机为 MacBook Pro (13-inch, 2017, Four Thunderbolt 3 Ports), CPU 为 Intel Core i5 双核 3.1 GHz, 内存为 8 GB 2133 MHz LPDDR3。

### 2. 操作系统

实验所用的操作系统为 macOS Mojave 10.14.5 Beta。macOS 基于 Darwin BSD 的 Unix 内核, 符合单一 Unix 规范, 在 macOS 的终端下可以正常完成本实验的全部内容。

如果在其他 Mac 电脑中复现本实验报告, 应进行如下配置, 以完成 cd /usr/include 的操作:

```
$ xcode-select -install
$ sudo installer -pkg /Library/Developer/CommandLineTools/Packages/macOS_SDK_headers_for_macOS
_10.14.pkg -target /
```

### 3. shell

本实验所用的 shell 为 zsh。zsh 的功能比 bash 更加强大, 且可以兼容 bash, 尽管一些细节有所不同, 但本实验不受影响。

## 二、实验内容、结果及分析

### 1. 软件许可证

绝大多数软件在发布时会附带一个许可证, 就软件的使用、复制、传播、修改等方面规定许可人和被许可人的权利和义务。权利人在软件许可证所规定的权利和义务, 决定了其是否真正将源代码向公众开放。OSIA (开放源代码首创行动组织) 为经其审核认定为开源的许可证进行认证。开源软件许可证的理念与专利的理念相反, 其旨在保证能够将开源软件不断发散出去, 不会被私人占有。

每一个开源软件许可证都满足四个特征:

- 当使用人获得源代码使用或修改后, 有再次发布源代码的义务。
- 任何获得源代码的人都必须发布保证完整性的源代码。
- 接受许可证而获得源代码的任何人都可以对源代码进行修改。

- 开源软件权利人不承担瑕疵担保责任。

经过 OSI 认证的开源软件许可证已经超过 80 种。常见的开源软件许可证有 GPL、BSD、MIT、Apache、LGPL 等。

GPL 许可证具有“传染性”，只要在一个软件中使用 GPL 协议的产品，则该软件产品也必须采用 GPL 协议，即必须开源和免费。商业软件和对代码有保密要求的软件不适合将采用 GPL 协议的代码作为类库或二次开发的基础。Linux 内核采用 GPL 许可证。

LGPL 许可证允许商业软件通过类库引用的方式使用 LGPL 类库而不必开源其代码。但是如果修改采用 LGPL 许可证的代码，则所有修改的代码、涉及修改部分的额外代码和衍生的代码都必须采用 LGPL 协议。

BSD 许可证允许修改后的代码作为开源或商业软件发布。但如果再发布的产品中有源代码，则源代码应带有原来代码的 BSD 许可证；如果只以二进制形式发布，则需要在文档和版权声明中包含原来代码的 BSD 许可证。不可以用开源代码的作者名字和原来产品的名字做市场推广。

Apache 许可证与 BSD 类似，允许修改后的代码作为开源或商业软件发布。与 BSD 不同的是，Apache 在修改的每一个文件中，都必须进行说明。

MIT 许可证也与 BSD 类似，可以以各种形式对修改后的代码进行发布。无论以何种形式发布，都应在发行版中包含原许可证的声明。

开源软件许可证的实际使用是很复杂的，比如，Android 系统分为两部分，Linux 内核采用 GPL 许可证，而运行于内核以上的部分采用 Apache 许可证。Android 系统在内核与应用程序之间构建了一个 userspace，将驱动程序放在 userspace 上，而不是直接写在内核中，这样硬件驱动规避了 Linux 内核采用的 GPL 许可证的开源规定。

## 2. 基本命令

在终端中依次输入以下命令，得到输出结果。

```
$ ls
```

图 1: ls 命令运行结果截图

```
$ pwd
/Users/wutong
$ xy
zsh: command not found: xy
$ cd ..
无输出
$ pwd
/Users
$ cd
```

无输出

```
$ pwd
/Users/wutong
$ cd /usr/include
无输出
$ ls
```

```
~ $ cd /usr/include
~ $ include ls
AppleTextureEncoder.h editline      miscfs          slapi-plugin.h
AssertMacros.h    err.h            module.modulemap   spawn.h
Availability.h    errno.h          monetary.h       sqlite3.h
AvailabilityInternal.h eti.h        monitor.h        sqlite3ext.h
AvailabilityMacros.h execinfo.h     mpool.h          stab.h
Block.h           expat.h         nameser.h        standards.h
CommonCrypto      expat_external.h nc_tparm.h      stddef.h
ConditionalMacros.h fcntl.h        ncurses.h       stdint.h
Darwin.apinotes   fenv.h          ncurses_dll.h   stdio.h
MacTypes.h        ffi.h           ndbm.h          stdlib.h
NSSystemDirectories.h float.h       net.h           strhash.h
TargetConditionals.h fmtmsg.h     net-snmp.h      string.h
Xplugin.h         fnmatch.h      netdb.h          stringlist.h
__cxxabi_config.h form.h          netinet.h       strings.h
__wctype.h         fsproperties.h netinet6.h      struct.h
__ctype.h          fstab.h         netkey.h        sys.h
__locale.h         fts.h          nfs.h           sysdir.h
__regex.h          ftw.h          nl_types.h      sysexists.h
__stdio.h          get_compat.h   nlist.h          syslog.h
__types.h          gethostuuid.h notify.h        tar.h
__types.h          getopt.h       notifys_keys.h tcl.h
__wctype.h         glob.h         ntsid.h         tclDecl.h
```

图 2: /usr/include/下 ls 命令运行结果部分截图

```
$ cd
```

无输出

### 3. 帮助信息

`man`、`info`、`--help`都可用来获取命令的帮助信息。`man`显示系统手册页的内容，`info`是GNU项目开发的超文本系统，显示GNU项目开发程序说明文档，比`man`显示的内容更加详细。`--help`是命令程序自带的选项，一般较为简略，并且不是所有命令都有这一选项。图3-图6分别列出了几个命令的帮助信息的显示结果。

```
wutong — man ls — man — less + man ls — 100x24
LSC(1)          BSD General Commands Manual          LS(1)

NAME
ls -- list directory contents

SYNOPSIS
ls [-ABCFGHLOPRSTUWabcdefghijklmnopqrstuvwxyz] [file ...]

DESCRIPTION
For each operand that names a file of a type other than directory, ls displays its name as well as any requested, associated information. For each operand that names a file of type directory, ls displays the names of files contained within that directory, as well as any requested, associated information.

If no operands are given, the contents of the current directory are displayed. If more than one operand is given, non-directory operands are displayed first; directory and non-directory operands are sorted separately and in lexicographical order.

The following options are available:
-@     Display extended attribute keys and sizes in long (-l) output.
```

图 3: man ls 运行结果

```
wutong — man passwd — man — less + man passwd — 100x24
PASSWD(1)          OpenSSL          PASSWD(1)

NAME
openssl-passwd, passwd - compute password hashes

SYNOPSIS
openssl passwd [-crypt] [-1] [-apr1] [-salt string] [-in file] [-stdin] [-noverify]
[-quiet] [-table] {password}

DESCRIPTION
The passwd command computes the hash of a password typed at run-time or the hash of each password in a list. The password list is taken from the named file for option -in file, from stdin for option -stdin, or from the command line, or from the terminal otherwise. The Unix standard algorithm crypt and the MD5-based BSD password algorithm 1 and its Apache variant apr1 are available.

OPTIONS
-crypt
    Use the crypt algorithm (default).
-1
    Use the MD5 based BSD password algorithm 1.
```

图 4: man passwd 运行结果

```
wutong — info pwd — info — man + info pwd — 100x24
File: *manpages*, Node: pwd, Up: (dir)
PWD(1)          BSD General Commands Manual          PWD(1)

NAME
pwd -- return working directory name

SYNOPSIS
pwd [-L | -P]

DESCRIPTION
The pwd utility writes the absolute pathname of the current working directory to the standard output.

Some shells may provide a builtin pwd command which is similar or identical to this utility. Consult the builtin(1) manual page.

The options are as follows:
-L     Display the logical current working directory.

-----Info: (*manpages*)pwd, 91 lines --Top-----
Welcome to Info version 5.2. Type h for help, m for menu item.
```

图 5: info pwd 运行结果

```
wutong — brew --help
Example usage:
brew search [TEXT|/REGEX/]
brew info [FORMULA...]
brew install FORMULA...
brew update
brew upgrade [FORMULA...]
brew uninstall FORMULA...
brew list [FORMULA...]

Troubleshooting:
brew config
brew doctor
brew install --verbose --debug FORMULA

Contributing:
brew create [URL [--no-fetch]]
brew edit [FORMULA...]

Further help:
brew commands
brew help [COMMAND]
man brew
https://docs.brew.sh
```

图 6: brew --help 运行结果

#### 4. 系统信息

表 1 中的命令用于获取计算机系统的一些信息。

表 1: 获取系统信息的命令

命令	功能	运行结果
whoami	显示用户名	wutong
uname	显示操作系统的名称	Darwin
uname -n	显示系统域名	wutongdeMacBook-Pro.local
uname -p	显示 CPU 名称	i386

#### 5. Unix 纪元

Unix 纪元的时间起点为世界标准时间 (UTC) 1970 年 1 月 1 日 00:00:00，除去闰秒。Unix 时间为从 Unix 纪元的起点到该时刻的秒数。用 cal 命令可获取日历，例如：

```
cal 4      显示公元 4 年的日历
cal 1752    显示公元 1752 年的日历
cal 7 2012   显示公元 2012 年 7 月的日历
```

图 7 为 1752 年日历，该年共有 355 天，9 月 2 日后直接跨越到 9 月 14 日。查阅资料知，这种现象是出于历史原因。罗马教皇格里十三世在位时，当时的人发现，公元前 46 年起凯撒大帝实行的儒略历每年时间较实际天体运转快 11 分 14 秒。一千多年来，累计误差已经达到 10 天。1582 年格里十三世将该年儒略历 10 月 4 日的第二天定为格里历 10 月 15 日。当时的英皇亨利八世与教皇的关系恶劣，所以拒绝调整历法，直到 1752 年 9 月才改用格里历，当时误差已经达到 11 天。美国当时尚未独立，仍为英国殖民地，所以追溯历史时间时沿用了英国的日历。

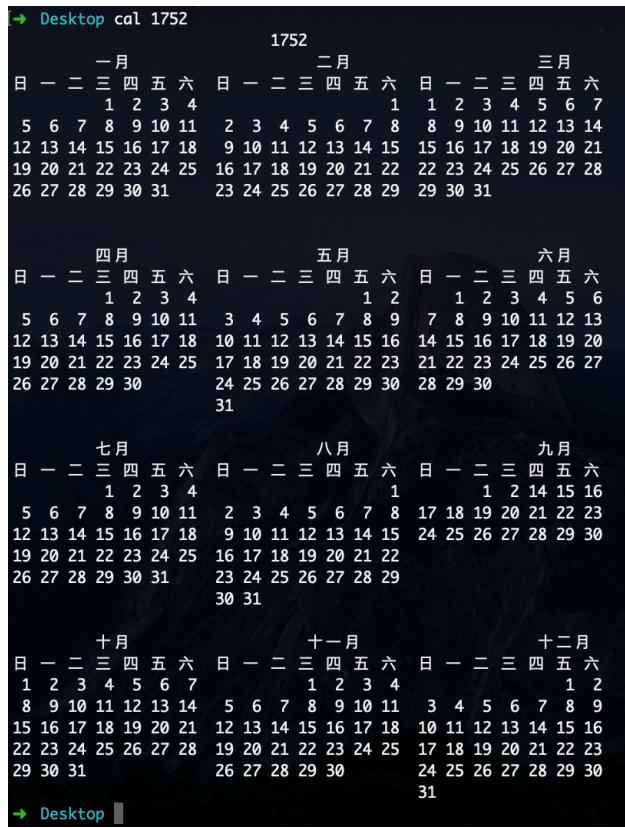


图 7: 1752 年日历

## 6. uptime

`uptime` 显示当前时间，系统运行时间，用户数和过去 1 分钟、5 分钟、10 分钟内的系统平均负载。系统平均负载是运行队列中的平均进程数。一般情况下，运行队列中每个 CPU 核心的进程数少于 4 个，则认为系统运行状态良好。如果运行队列过大，表明 CPU 负荷较大，进程等待 CPU 调度执行的时间较长。图 8 为实验时 `uptime` 的运行结果，系统于 16:29 启动，已运行 2 小时 11 分钟，当前系统负载状态良好。

```
[→ ~ uptime
16:29  up  2:11,  2 users,  load averages: 2.88 3.10 2.91
[→ ~ ]
```

图 8: uptime 命令运行结果

## 7. 命令功能描述

通过整理 man 和 info 的信息, 表 2列举了 Linux 常用的命令或系统调用及库函数的功能。一些命令和库函数重名, 此表列出其作为命令的功能和用法。

表 2: 常见命令功能简述

命令	功能简述	实例
touch	改变文件的访问和修改时间, 如果文件不存在, 则创建文件。	touch -t 192608171200.23 test.txt
cp	复制文件。	cp -v test.txt test
mv	重命名文件或移动文件。	mv test.sh test
rm	删除文件。	rm -r test
mkdir	创建目录。	mkdir test
who	显示当前登录的用户。	who am i
ls	列出目录中的内容。	ls -al /System/Library
cd	改变当前的工作目录。	cd /bin
pwd	显示当前的绝对路径。	pwd
open	打开文件, 效果等同于在图形界面中双击图标。	open -a /Applications/Microsoft\ Word.app test.docx
read	从标准输入中读取字符串, 并赋值给指定的 shell 变量。	read studentID studentName
write	向其他用户发送消息。	write username
close	系统调用。删除文件描述符, 如果删除的是对该资源的最后一个引用, 则释放相应内存。	int close(int fildes);
pipe	系统调用。创建管道, 参数中提供两个文件描述符。	int pipe(int filedes[2]);
socket	系统调用。创建一个 socket, 返回 socket 描述符。	int socket(int domain, int type, int protocol);
mkfifo	创建一个 fifo, 相当于管道。	mkfifo fifo
system	标准库函数。向 shell 传递一个命令, 返回值为 shell 的退出值。	int system(const char *command);
printf	格式化输出。	printf "%d\n" studentID

## 8. vi 编辑器

在 shell 内输入 vi firscript，打开 vim 编辑器。按 <A> 键进入编辑模式，依次输入三行文字，此时屏幕如图 9 所示。按 <Esc> 键进入命令模式，输入:wq，即保存并退出。为了在 shell 中直接运行该脚本程序，先输入 chmod a+x firscript 为文件所有者授予执行权限，再输入./firscript 运行脚本。由图 10 运行结果知，当前的工作目录为 /Users/wutong/Desktop，有 18 个文件。有两个用户在使用计算机系统，用户名均为 wutong，tty 分别为 console 和 ttys001，ttys001 为当前正在使用的终端。

```
-- INSERT --
```

图 9: insert 模式下的 vim

```
Desktop ./firscript
total 392
drwxr-xr-x  3 wutong  staff      96  5 19 22:21 $RECYCLE.BIN
drwx-----@ 18 wutong  staff     576  7 13 11:05 .
drwxr-xr-x@ 86 wutong  staff    2752  7 13 11:13 ..
-rw-r--r--@  1 wutong  staff   6148  7 13 11:05 .DS_Store
-rw-r--r--  1 wutong  staff     48  6  8  2018 .bash_profile
drwxr-xr-x  3 wutong  staff     96  5 30  2018 .du..kPAb27IH4l
drwxr-xr-x  8 wutong  staff    256  5  2 23:19 .idea
drwxr-xr-x  3 wutong  staff     96  6 30  06:45 .ipynb_checkpoints
-rw xr-xr-x  1 wutong  staff  18944  1  8  2019 Thumbs.db
-rw xr-xr-x  1 wutong  staff   8480  7 12 23:59 a.out
-rw r--r--  1 wutong  staff    282  5 19 22:19 desktop.ini
-rw xr-xr-x  1 wutong  staff     16  7 13 11:05 firscript
-rw r--r--@  1 wutong  staff  129536  7 13 10:50 project1 shell命令 .doc
drwxr-xr-x  2 wutong  staff     64  7 12 21:48 test
-rw r--r--  1 wutong  staff    116  7 12 23:59 test.c
-rw r--r--@  1 wutong  staff  11979  7 12 22:07 test.docx
-rw r--r--  1 wutong  staff      0  7 12 23:59 test.py
-rw r--r--  1 wutong  staff      0  8 17  1926 test.txt
wutong  console Jul 12 14:18
wutong  ttys001 Jul 13 11:12
/Users/wutong/Desktop
```

图 10: 脚本 firscript 的运行结果

## 9. 建立目录树

在主目录下建立 lab 目录, 用于存放实验所用文件。输入如下命令, 建立目录树:

```
$ mkdir -p {temp,professional/{courses/{major/{cs213,cs381/{notes,labs,programs},cs475},general},societies/{ieee,acm}},personal/{funstuff,taxes}}
```

输入 \$ tree 命令查看该目录树:

```
lab tree
.
├── personal
│   ├── funstuff
│   └── taxes
├── professional
│   ├── courses
│   │   ├── general
│   │   └── major
│   │       ├── cs213
│   │       ├── cs381
│   │       │   ├── labs
│   │       │   ├── notes
│   │       │   └── programs
│   │       └── cs475
│   └── societies
│       ├── acm
│       └── ieee
└── temp

17 directories, 0 files
```

图 11: 建立的目录树结构

## 10. cd professional/courses

三种获得主目录绝对路径的命令:

- \$ echo \$HOME
- \$ printf "\$HOME\n"
- \$ cd && pwd && cd - > /dev/null

命令输出均为:

/Users/wutong

acm 目录的绝对路径为:

/Users/wutong/lab/professional/societies/acm

两个相对路径为:

./societies/acm

~/lab/professional/societies/acm

执行 cd major/cs381/labs 命令后, 输入以下命令以获得当前目录的绝对路径:

\$ pwd

命令输出为:

/Users/wutong/lab/professional/courses/major/cs381/labs

## 11. 隐藏文件

输入以下命令列出主目录下的隐藏文件:

```
ls -d *
```

主目录下的隐藏文件如图 12 所示:

```
~ ls -d *
.DDPReview      .config          .mailcap        .ssh
.DS_Store       .cups            .matplotlib   .subversion
.IdentityService .dotfiles      .mime.types    .thumbnails
.Rapp.history   .eclipse         .mono          .tooling
.ShadowsocksX-NG .emacs.d      .mume          .vim
.Trash          .gitconfig      .mysql_history .viminfo
.Xauthority     .idlerc         .oh-my-zsh    .vimrc
.adobe          .ipython        .oracle_jre_usage .vscode
.android        .jupyter        .profile       .p2
.bash_history   .jssc           .purpose.swp  .vscode-cpptools
.bash_profile   .jupyter        .python_history .wget-hsts
.bash_profile.pysave .kindle      .lldb          .zsh_history
.bash_sessions  .lldb           .qqbot-tmp    .zshrc
.cache          .local          .rstudio-desktop
```

图 12: 主目录下的隐藏文件

## 12. inode

ls 的 -i 可用于查看文件的 inode 号。输入以下命令获得根目录、主目录、~/lab/temp、~/lab/professional、~/lab/personal 的 inode 号:

```
ls -id1 {/,~,~/lab,~/lab/temp,~/lab/professional,~/lab/personal}
```

命令结果如图 13 所示。

```
[~] ls -id1 {/,~,~/lab,~/lab/temp,~/lab/professional,~/lab/personal}
2
2
636744 /Users/wutong
8670402616 /Users/wutong/lab
8670402742 /Users/wutong/lab/personal
8670402729 /Users/wutong/lab/professional
8670402728 /Users/wutong/lab/temp
```

图 13: 所查询目录的 inode 号

## 13. 文件类型

输入以下命令，创建并编辑 lab1 文件:

```
$ vim lab1
```

输入以下命令，查看 lab1 文件类型:

```
$ ls -l lab1
```

命令输出如下:

```
-rw-r--r-- 1 wutong staff 151 7 13 16:58 lab1
```

该文件为普通文件。

输入以下命令，查看 lab1 文件内容类型:

```
$ file lab1
```

命令输出如下:

```
lab1: ASCII text
```

该文件为 ASCII 文本文件。

#### 14. 查找头文件

输入以下命令, 查找/usr/include 下以 t 开头的头文件:

```
$ ls t*.h
```

命令输出如下:

```
[→ include ls t*.h
tar.h          tclTomMathDecls.h  tgmath.h      tkDecls.h      tzfile.h
tcl.h          term.h          tic.h        tkIntXlibDecls.h
tclDecls.h    term_entry.h   time.h       tkMacOSX.h
tclPlatDecls.h termcap.h     timeconv.h  tkPlatDecls.h
tclTomMath.h   termios.h     tk.h         ttyent.h]
```

图 14: 列出以 t 开头的头文件

#### 15. 创建文件

创建三个不同大小的文件:

```
$ man cat > mediumFile
```

```
$ man bash >largeFile
```

```
$ vim smallFile
```

输入以下命令查看所创建文件的情况:

```
$ ls -l *File
```

输出结果如图所示。

```
[→ lab ls -l *File nelson ECE 3.81 nelson@tn.abc.org III.111.6666
-rw-r--r-- 1 wutong staff 329778 7 13 17:08 largeFile
-rw-r--r-- 1 wutong staff 3876 7 13 17:08 mediumFile pk.xyz.org 111.111.7777
-rw-r--r-- 1 wutong staff 952 7 13 17:18 smallFile xyz.org 111.111.0000]
```

图 15: 创建的三个大小不同的文件

#### 16. 输出指定行

显示 largeFile 文件开始的 12 行:

```
$ head -n 12 largeFile
```

```
[→ lab head -n 12 largeFile
BASH(1)

NAME
  bash - GNU Bourne-Again SHell
SYNOPSIS
  bash [options] [file]
COPYRIGHT
  Bash is Copyright (C) 1989-2005 by the Free Software Foundation, Inc.
```

图 16: 显示 largeFile 文件开始的 12 行

显示 smallFile 文件的最后 5 行:

```
$ tail -n 5 smallFile
```

```
→ lab tail -n 5 smallFile
John Lee EE 3.64 jlee@j.lee.com 111.111.2222
Sunil Raj ECE 3.86 raj@sr.cs.edu 111.111.3333
Charles Right EECS 3.31 right@cr.abc.edu 111.111.4444
Diane Rover ECE 3.87 rover@dr.xyz.edu 111.111.5555
Aziz Inan EECS 3.75 ainan@ai.abc.edu 111.111.1111
```

图 17: 显示 smallFile 文件的最后 5 行

显示 smallFile 文件从第 6 行到结束:

```
$ tail -n +6 smallFile
```

```
→ lab tail -n +6 smallFile
Sam Chu ECE 3.68 chu@sam.ab.com 111.222.5555
Arun Roy SS 3.86 roy@ss.arts.edu 111.222.8888
Rick Marsh CS 2.34 marsh@a.b.org 111.222.6666
James Adam CS 2.77 jadam@a.b.org 111.222.7777
Art Pohm ECE 4.00 pohm@ap.a.org 111.222.9999
John Clark ECE 2.68 clark@xyz.ab.com 111.111.5555
Nabeel Ali EE 3.56 ali@ee.eng.edu 111.111.8888
Tom Nelson ECE 3.81 nelson@tn.abc.org 111.111.6666
Pat King SS 3.77 king@pk.xyz.org 111.111.7777
Jake Zulu CS 3.00 zulu@jz.sa.org 111.111.9999
John Lee EE 3.64 jlee@j.lee.com 111.111.2222
Sunil Raj ECE 3.86 raj@sr.cs.edu 111.111.3333
Charles Right EECS 3.31 right@cr.abc.edu 111.111.4444
Diane Rover ECE 3.87 rover@dr.xyz.edu 111.111.5555
Aziz Inan EECS 3.75 ainan@ai.abc.edu 111.111.1111
```

图 18: 显示 smallFile 文件从第 6 行到结束

## 17. 复制文件

整个过程的会话如下:

```
$ cp smallFile dataFile
$ ls -l
```

```
→ lab cp smallFile dataFile
→ lab ls -l
total 672
-rw-r--r-- 1 wutong staff 952 7 13 17:40 dataFile
-rw-r--r-- 1 wutong staff 329778 7 13 17:08 largeFile
-rw-r--r-- 1 wutong staff 3876 7 13 17:08 mediumFile
drwxr-xr-x 4 wutong staff 128 7 13 12:36 personal
drwxr-xr-x 4 wutong staff 128 7 13 12:36 professional
-rw-r--r-- 1 wutong staff 952 7 13 17:18 smallFile
drwxr-xr-x 2 wutong staff 64 7 13 12:36 temp
```

图 19: 复制 smallFile

dataFile 和 smallFile 的修改时间不同, 可以用 touch 命令强行修改 dataFile 的修改时间。

```
$ ls -l
```

```
→ lab ls -i
8670667175 dataFile    8670659330 mediumFile   8670402729 professional 8670402728 temp
8670659368 largeFile   8670402742 personal    8670661102 smallFile
```

图 20: 查看 inode 号

两个文件的 inode 号不同。

```
$ mv dataFile newDataFile
$ ls -i
```

```
→ lab mv dataFile newDataFile
→ lab ls -i
8670659368 largeFile   8670667175 newDataFile  8670402729 professional 8670402728 temp
8670659330 mediumFile  8670402742 personal    8670661102 smallFile
```

图 21: 重命名 dataFile

dataFile 和 newDataFile 的 inode 号相同, 因为 mv 执行重命名的操作, 不改变 inode 的信息。

```
$ mv newDataFile /tmp
$ ls -i /tmp/newDataFile
```

```
→ lab mv newDataFile /tmp
→ lab ls -i /tmp/newDataFile
8670667175 /tmp/newDataFile
```

图 22: 移动 newDataFile

newDataFile 的 inode 号未发生变化, 因为 mv 不改变 inode 信息。

## 18. 显示字节数

输入以下命令获得 smallFile、mediumFile、largeFile、/tmp/newDataFile 四个文件的字节数、字数和行数:

```
$ wc {smallFile,mediumFile,largeFile,/tmp/newDataFile}
```

输出结果如下:

```
→ lab wc {smallFile,mediumFile,largeFile,/tmp/newDataFile}
 20      120      952 smallFile
 95      485     3876 mediumFile
 4890    37094   329778 largeFile
 20      120      952 /tmp/newDataFile
 5025    37819   335558 total
```

图 23: wc 命令输出结果

以下命令也可以获得文件的字节数:

```
$ ls -l {smallFile,mediumFile,largeFile,/tmp/newDataFile} | awk '{print 5,9}'
```

输出结果如下:

```
[→ lab ls -l {smallFile,mediumFile,largeFile,/tmp/newDataFile} | awk '{print $5,$9}'  
952 /tmp/newDataFile  
329778 largeFile  
3876 mediumFile  
952 smallFile
```

图 24: 输出文件字节数

## 19. 按后缀名搜索

输入以下命令查找所有以.html, .htm, .c 结尾的文件, 其中 find 命令中如果带有通配符, 必须要加双引号, 这样可以把字符串传给 find, 由 find 进行通配符展开, 否则 shell 会直接对通配符进行展开。如果要进行多条件查找, 可加入-a(and), -o(or), !(not)。

```
$ find . -name "*.html" -o -name "*.htm" -o -name "*.c"
```

输出结果如下:

```
[→ lab find . -name "*.*html" -o -name "*.*htm" -o -name "*.*c"  
./test2.htm  
./test0.htm  
./test1.htm  
./test2.html  
./test0.c  
./a.c  
./a.htm  
./test1.c  
./test1.html  
./a.html  
./test2.c  
./test0.html
```

图 25: 搜索 html 文件和 C 程序文件

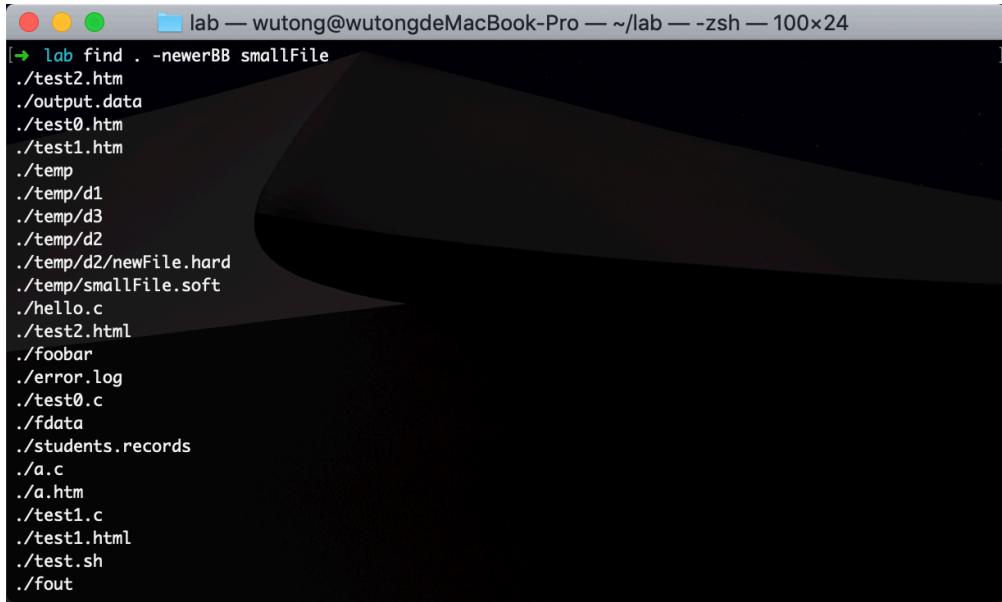
## 20. 按时间搜索

### 方法一

输入以下命令查找创建时间晚于 smallFile 的文件:

```
$ find . -newerBB smallFile
```

输出结果如下:



The screenshot shows a terminal window titled 'lab' with the command 'find . -newerBB smallFile' entered. The output lists numerous files and directories, including '.test2.htm', './output.data', './test0.htm', './test1.htm', './temp', './temp/d1', './temp/d3', './temp/d2', './temp/d2/newFile.hard', './temp/smallFile.soft', './hello.c', './test2.html', './foobar', './error.log', './test0.c', './fdata', './students.records', './a.c', './a.htm', './test1.c', './test1.html', './test.sh', and './fout'. The terminal is running on a MacBook Pro with a resolution of 100x24.

图 26: 创建时间在 smallFile 之后的部分文件

## 方法二

在 macOS 系统中, GetFileInfo 命令可以获取文件的创建时间, 例如:

```
$ GetFileInfo -d test.c
```

通过以下命令将 GetFileInfo -d 的输出转换成时间戳:

```
$ date -jf "%m/%d/%Y %H:%M:%S" "$(GetFileInfo -d smallFile)" +%s
```

依次输入以下命令, 显示创建时间在 smallFile 之后的文件:

```
$ d=$(date -jf "%m/%d/%Y %H:%M:%S" "$(GetFileInfo -d smallFile)" +%s)
```

```
$ for file in $(ls)
```

```
for> do
```

```
for> temp=$(date -jf "%m/%d/%Y %H:%M:%S" "$(GetFileInfo -d $file)" +%s)
```

```
for> if test ${temp} -gt ${d}
```

```
for if> then
```

```
for then> echo $file
```

```
for then> fi
```

```
for> done
```

以上命令可以编写为脚本。运行结果如下:

```

lab — wutong@wutongdeMacBook-Pro — ~/lab — -zsh — 100x24
lab d=$(date -jf "%m/%d/%Y %H:%M:%S" "$(GetFileInfo -d smallFile)" +%s)
for file in $(ls)
for> do
for> if test ${temp} -gt ${d}
for if> then
for then> echo $file
for then> fi
for> done
a.c
a.htm
a.html
error.log
fdata
foobar
foobar.path
fout
hello.c
largeFile
mediumFile
output.data
personal
professional
smallFile
students.records

```

图 27: 用 GetFileInfo 命令获取创建时间在 smallFile 之后的部分文件

## 21. 按文件名搜索

输入以下命令, 查找主目录下所有包含字符串“Linux”的文件名:

```
$ find ~ -name "*Linux*"
```

输出结果如下:

```

~ find ~ -name "*Linux*"
/Users/wutong/OneDrive/Course/Linux
/Users/wutong/OneDrive/Course/Linux/课件/ch2 Linux基础.pdf
/Users/wutong/OneDrive/Course/Linux/课件/ch1-1 Linux安装.pdf
/Users/wutong/OneDrive/Linux-course
/Users/wutong/Qt/Qt Creator.app/Contents/PlugIns/libRemoteLinux.dylib
/Users/wutong/Qt/Qt Creator.app/Contents/Resources/qbs/share/qbs/modules/cpp/LinuxGCC.qbs
/Users/wutong/Qt/5.12.2/clang_64/include/QtPlatformHeaders/QLinuxFbFunctions
/Users/wutong/Library/Preferences/Parallels/Linux.dat
/Users/wutong/Library/Android/sdk/sources/android-27/android/content/pm/SELinuxUtil.java
/Users/wutong/Library/Android/sdk/sources/android-27/android/os/SELinux.java
/Users/wutong/Library/Android/sdk/sources/android-27/com/android/server/pm/SELinuxMMAC.java
/Users/wutong/Library/Caches/VisualStudio/7.0/addin-db-002/addin-data/1/MonoDevelop.Deployment.Linux
,7.3.3.maddin

```

图 28: 主目录下所有包含字符串“Linux”的文件名

## 22. 文件权限

输入以下命令获取所需信息:

```
$ ls -dl {/,/etc/passwd,/bin/df,~}
```

结果如下表:

表 3: 文件基本信息

文件	文件类型	存取权限	链接数	所有者	组	文件大小
/	目录文件	rwxrwxrwx	38	root	wheel	1216
/etc/passwd	普通文件	rw-r--r--	1	root	wheel	6804
/bin/df	普通文件	rwxr-xr-x	1	root	wheel	23392
~	目录文件	rwxr-xr-x	80	wutong	staff	2560

### 23. 设置权限

输入以下命令, 依次完成设置权限, 创建目录, 创建文件的操作, 并给出 d1, d2, d3, f1 的访问权限:

```
$ chmod 755 {temp,professional,personal} && mkdir temp/{d1,d2,d3} && touch temp/d1/f1 && ls -dl temp/{d1,d2,d3,d1/f1}
```

输出结果如下, d1、d2、d3 所有者有读、写、执行三种权限, 其他用户只有读和执行权限。f1 所有者有读写权限, 其他用户只有读权限。

```
[root@lab ~]# chmod 755 {temp,professional,personal} && mkdir temp/{d1,d2,d3} && touch temp/d1/f1 && ls -dl temp/{d1,d2,d3,d1/f1}
drwxr-xr-x 3 wutong staff 96 7 13 22:40 temp/d1
-rw-r--r-- 1 wutong staff 0 7 13 22:40 temp/d1/f1
drwxr-xr-x 2 wutong staff 64 7 13 22:40 temp/d2
drwxr-xr-x 2 wutong staff 64 7 13 22:40 temp/d3
```

图 29: d1, d2, d3, f1 的访问权限

### 24. 修改权限

整个实验的会话过程如下:

设置当前目录为主目录:

```
$ HOME=.
```

设置文件 temp 仅为执行权限:

```
$ chmod 444 temp
```

执行 \$ ls -ld temp 的输出结果:

```
dr- -r- -r- 5 wutong staff 160 7 13 22:40 temp
```

执行 \$ ls -l temp 的输出结果:

```
ls: temp: Permission denied
```

为了成功执行 ls -l temp, 所需的最小权限为 400, 则执行如下命令:

```
$ chmod 400 temp
```

再执行 ls -l temp, 没有 Permission denied 的报错。

### 25. 掩码

长列表显示文件的访问权限, 结果如下:

表 4: 文件的访问权限

umask 值	文件权限			
	f2	hello.c	greeting	d11/d21
022	rw-r--r--	rw-----	rwxr-xr-x	rwxr-xr-x
077	rw-----	rw-----	rwxr-xr-x	rwx-----

## 26. 删除目录中所有文件

执行以下命令, 删除/temp 下的所有文件和目录: \$ rm -rf temp/\*

输出结果如下:

```
[→ lab rm -rf temp/*。然后取消你输入对
zsh: sure you want to delete all 7 files in /Users/wutong/lab/temp [yn]? y]
```

图 30: 删除/temp 下的所有文件和目录

## 27. 长列表格式显示文件

输入以下命令:

```
$ mkdir temp/d{1,2,3} && cp smallFile temp/d1 && ls -il temp/d1/smallFile
```

输出结果如下:

表 5: d1/smallFile 文件属性

inode 号	访问权限	硬链接数	文件大小
8670764839	rw-----	1	952

## 28. 创建硬链接

输入以下命令创建硬链接并长列表显示:

```
$ ln .../d1/smallFile newFile.hard && ls -il newFile.hard
```

输出结果如下:

表 6: newFile.hard 文件属性

inode 号	访问权限	硬链接数	文件大小
8670764839	rw-----	2	952

smallFile 与 newFile.hard 的 inode 号相同, 所以是同一个文件。

## 29. 硬链接

输入以下命令可正常显示 smallFile 文件的内容:

```
$ cat newFile.hard
```

取消对 d1/smallFile 的读权限:

```
$ chmod a-r .../d1/smallFile
```

输入以下命令:

```
$ cat newFile.hard
```

无权限, 命令输出结果如下:

```
[→ d2 cat newFile.hard
cat: newFile.hard: Permission denied]
```

图 31: 无法查看 newFile.hard 的内容

输入以下命令, 增加对 d1/smallFile 的读权限:

```
$ chmod a+r ..d1/smallFile
```

输入 cat 命令, 可以正常查看 newFile.hard 的内容:

```
$ cat newFile.hard
```

删除 d1/smallFile 后, 再显示 newFile.hard 的内容:

```
$ rm ..d1/smallFile && cat newFile.hard
```

newFile.hard 的内容可以正常显示, 说明删除文件时只会删除一个硬链接, 而不影响其他的硬链接。

只有当文件所有的硬链接都被删除时, 文件所占磁盘资源才会被释放。

```
[→ d2 rm ..d1/smallFile && cat newFile.hard
FirstName LastName Major GPA Email Phone
John Doe ECE 3.54 doe@jd.home.org 111.222.3333
James Davis ECE 3.71 davis@jd.work.org 111.222.1111
Al Davis CS 2.63 davis@a.lakers.org 111.222.2222
Ahmad Rashid MBA 3.04 ahmad@mba.org 111.222.4444
Sam Chu ECE 3.68 chu@sam.ab.com 111.222.5555
Arun Roy SS 3.86 roy@ss.arts.edu 111.222.8888
Rick Marsh CS 2.34 marsh@a.b.org 111.222.6666
James Adam CS 2.77 jadam@a.b.org 111.222.7777
Art Pohm ECE 4.00 pohm@ap.a.org 111.222.9999
John Clark ECE 2.68 clark@xyz.ab.com 111.111.5555
Nabeel Ali EE 3.56 ali@ee.eng.edu 111.111.8888
Tom Nelson ECE 3.81 nelson@tn.abc.org 111.111.6666
Pat King SS 3.77 king@pk.xyz.org 111.111.7777
Jake Zulu CS 3.00 zulu@jz.sa.org 111.111.9999
John Lee EE 3.64 jlee@j.lee.com 111.111.2222
Sunil Raj ECE 3.86 raj@sr.cs.edu 111.111.3333
Charles Right EECS 3.31 right@cr.abc.edu 111.111.4444
Diane Rover ECE 3.87 rover@dr.xyz.edu 111.111.5555
Aziz Inan EECS 3.75 ainan@ai.abc.edu 111.111.1111]
```

图 32: 显示 newFile.hard 的内容

### 30. 创建软链接

输入以下命令创建 d1/smallFile 的软链接:

```
$ ln -s d1/smallFile smallFile.soft
```

长列表格式显示两个文件:

```
$ ls -il {smallFile.soft,d1/smallFile}
```

输出结果如下, smallFile.soft 和 d1/smallFile 的 inode 号和文件大小都不同, 是两个不同的文件。

```
[→ temp ls -il {smallFile.soft,d1/smallFile}
8670779338 -rw----- 1 wutong staff 952 7 14 13:17 d1/smallFile
8670779428 lrwx---- 1 wutong staff 12 7 14 13:18 smallFile.soft -> d1/smallFile]
```

图 33: 长列表显示软链接文件的内容

### 31. 软链接

取消对 smallFile 的读权限后, 无法读取 smallFile.soft 的内容:

```
[→ temp chmod a-r d1/smallFile
[→ temp cat smallFile.soft
cat: smallFile.soft: Permission denied]
```

图 34: 没有对 smallFile.soft 的读权限

增加对 smallFile 的读权限后, 可以通过 smallFile.soft 读取 smallFile 的内容:

```
[→ temp chmod a+r d1/smallFile
[→ temp cat smallFile.soft
FirstName    LastName      Major GPA Email Phone
John        Doe     ECE   3.54  doe@jd.home.org 111.222.3333
James       Davis    ECE   3.71  davis@jd.work.org 111.222.1111
Al          Davis    CS    2.63  davis@a.lakers.org 111.222.2222
Ahmad      Rashid   MBA   3.04  ahmad@mba.org 111.222.4444
Sam         Chu     ECE   3.68  chu@sam.ab.com 111.222.5555
Arun        Roy     SS    3.86  roy@ss.arts.edu 111.222.8888
Rick        Marsh    CS    2.34  marsh@ab.org 111.222.6666
James       Adam    CS    2.77  jadam@a.b.org 111.222.7777
Art         Pohm    ECE   4.00  pohm@ap.a.org 111.222.9999
John        Clark   ECE   2.68  clark@xyz.ab.com 111.111.5555
Nabeel      Ali     EE    3.56  ali@ee.eng.edu 111.111.8888
Tom         Nelson   ECE   3.81  nelson@tn.abc.org 111.111.6666
Pat         King    SS    3.77  king@pk.xyz.org 111.111.7777
Jake        Zulu    CS    3.00  zulu@jz.sa.org 111.111.9999
John        Lee     EE    3.64  jlee@j.lee.com 111.111.2222
Sunil      Raj     ECE   3.86  raj@sri.cs.edu 111.111.3333
Charles    Right   EECS  3.31  right@cr.abc.edu 111.111.4444
Diane      Rover   ECE   3.87  rover@dr.xyz.edu 111.111.5555
Aziz       Inan   EECS  3.75  ainan@ai.abc.edu 111.111.1111]
```

图 35: 可以读取 smallFile 的内容

删除 smallFile 文件后, 读取 smallFile.soft 时提示无此文件。说明软链接所指的文件删除后, 该软链接也随之失效。

```
[→ temp ls -il {smallFile.soft,d1/smallFile}
8670779338 -rw----- 1 wutong  staff  952 7 14 13:17 d1/smallFile
8670779428 lrwx---- 1 wutong  staff   12 7 14 13:18 smallFile.soft -> d1/smallFile]
```

图 36: 无 smallFile 文件

### 32. 查看进程

输入以下命令查看正在运行的总进程数, 由输出结果知共有 389 个正在运行的进程。

\$ ps -A | wc -l

输入以下命令:

\$ ps -Af | grep -E "init|zsh|ps"

输出结果如下:

表 7: 进程信息

进程	PID	父进程	父进程 PID
init	848	-zsh	96118
zsh	96118	login -pf wutong	96117
ps	1336	-zsh	96118

### 33. 后台

输入以下命令实现 5 秒后在屏幕上出现文字:

```
$ (sleep 5; echo 'Time for Lunch!')&
```

```
→ ~ (sleep 5; echo 'Time for Lunch!')&
[1] 11331
→ ~ Time for Lunch!
[1] + 11331 done      ( sleep 5; echo 'Time for Lunch!' )
```

图 37: 后台执行命令

### 34. 并发执行

并发执行 date、uname -a、who、ps:

```
$ date $ uname -a $ who $ ps
```

输出结果如下:

```
→ temp date & uname -a & who & ps
[1] 8111
[2] 8112
[3] 8113
2019年 7月 14日 星期日 15时 49分 46秒 CST
[1] 8111 done      date
Darwin wutongdeMacBook-Pro.local 18.6.0 Darwin Kernel Version 18.6.0: Mon Apr 15 21:18:10 PDT 2019;
root:xnu-4903.260.85.100.1~1RELEASE_X86_64 x86_64
    PID TTY          TIME CMD
[2] - 8112 done      uname -a
wutong  console Jul 12 14:18
wutong  ttys001 Jul 14 15:26
wutong  ttys002 Jul 14 15:27
[3] + 8113 done      who
23851 ttys000  0:00.21 /bin/zsh -l
95115 ttys001  0:00.61 -zsh
  8112 ttys002  0:00.00 (uname)
  8113 ttys002  0:00.00 (who)
  96118 ttys002  0:00.37 -zsh
```

图 38: 并发执行命令

### 35. 先后执行

先后执行 date、uname -a、who、ps:

```
$ date && uname -a && who && ps
```

输出结果如下:

```
[~] ~ date && uname -a && who && ps
[~] 2019年 7月14日 星期日 14时06分36秒 CST
[~] Darwin wutongdeMacBook-Pro.local 18.6.0 Darwin Kernel Version 18.6.0: Mon Apr 15 21:18:10 PDT 2019;
[~] root:xnu-4903.260.85.100.1~1/RELEASE_X86_64 x86_64
[~] wutong console Jul 12 14:18
[~] wutong ttys001 Jul 13 23:21
[~] PID TTY TIME CMD
[~] 23851 ttys000 0:00.21 /bin/zsh -l
[~] 52241 ttys001 0:01.86 -zsh
```

图 39: 先后执行命令

### 36. 子 shell

先后执行如下命令，输入输出如下：

```
$ pwd
/Users/wutong
$ bash
$ cd /usr
$ pwd
/usr
<Ctrl-D>
exit
$ pwd
/Users/wutong
```

### 37. 重定向错误信息

输入以下命令，查找 foobar 文件，错误信息重定向到/dev/null：

```
$ find /Users/wutong/lab -name foobar 2>/dev/null
```

屏幕上输出 foobar 的绝对路径。

### 38. 重定向标准输出

输入以下命令，查找 foobar 文件，绝对路径存到 foobar.path 文件中，错误信息重定向到/dev/null：

```
$ find /Users/wutong/lab -name foobar 2>/dev/null
```

显示 foobar.path 的内容：

```
[~] lab find /Users/wutong/lab -name foobar 1>foobar.path 2>/dev/null
[~] lab cat foobar.path
[~] /Users/wutong/lab/foobar
```

图 40: 重定向标准输出

### 39. cat 重定向

重定向 cat 的输入输出：

```
$ cat 0<students.records 1>output.data 2>error.log
```

运行结果如下：

```
[→ lab cat 0<students.records 1>output.data 2>error.log
[→ lab cat output.data
FirstName    LastName      Major   GPA     Email    Phone
John        Doe       ECE    3.54  doe@jd.home.org 111.222.3333
James       Davis      ECE    3.71  davis@jd.work.org 111.222.1111
Al          Davis      CS     2.63  davis@a.lakers.org 111.222.2222
Ahmad      Rashid     MBA    3.04  ahmad@mba.org 111.222.4444
Sam         Chu        ECE    3.68  chu@sam.ab.com 111.222.5555
Arun        Roy        SS     3.86  roy@ss.arts.edu 111.222.8888
Rick        Marsh      CS     2.34  marsh@a.b.org 111.222.6666
James       Adam       CS     2.77  jadam@a.b.org 111.222.7777
Art         Pohm      ECE    4.00  pohm@ap.a.org 111.222.9999
John       Clark      ECE    2.68  clark@xyz.ab.com 111.111.5555
Nabeel      Ali        EE     3.56  ali@ee.eng.edu 111.111.8888
Tom         Nelson     ECE    3.81  nelson@tn.abc.org 111.111.6666
Pat         King       SS     3.77  king@pk.xyz.org 111.111.7777
Jake        Zulu      CS     3.00  zulu@jz.sa.org 111.111.9999
John       Lee        EE     3.64  jlee@j.lee.com 111.111.2222
Sunil      Raj        ECE    3.86  raj@sr.cs.edu 111.111.3333
Charles    Right      EECS   3.31  right@cr.abc.edu 111.111.4444
Diane      Rover      ECE    3.87  rover@dr.xyz.edu 111.111.5555
Aziz       Inan      EECS   3.75  ainan@ai.abc.edu 111.111.1111
.
```

图 41: 重定向 cat 输入输出

#### 40. 重定向标准输入输出

执行如下命令, 完成重定向:

```
$ exec 0<fdta 1>fout
```

在之后的命令执行过程中, 从标准输入读入都变为从 fdta 读入, 输出到标准输出的内容都写进 fout 中。

#### 41. 管道

运行以下命令:

```
$ ls -l | wc
```

运行结果如下, 共有 36 行, 317 个单词, 1822 个字符。

图 42: 统计文件数目

```
[→ /bin ls -l | wc
36      317     1822
```

#### 42. 统计文件数

分别运行以下三个命令:

```
$ ls -l | grep ^- | wc -l
```

```
$ ls -l | grep ^d | wc -l
```

```
$ ls -l | grep ^l | wc -l
```

由输出结果知, /bin 目录下有 35 个普通文件, 没有目录文件和符号链接文件。

### 三、讨论与心得

这次实验的内容是 Linux 下的基本操作。因为我在日常使用 macOS 系统的过程中, 经常使用终端进行操作, 所以对很多命令已经有所熟悉。尽管如此, 在实验的过程中还是遇到了很多不会的问题。我

在整个实验上大约花了 50 小时，我所用的每一条命令都是经过充分的查阅资料并弄清其功能后才使用的，而不是直接将网络资料中的现成命令直接复制过来改掉文件名就做完的。

这篇实验报告中的每一句话都是我在实验过程中经过对网络资料进行整理和对实验结果进行分析后的个人见解，我力求保证实验报告内容的正确性。但由于我能力和水平的限制，难免会有疏漏和错误，有望批阅报告的老师和有机会阅读这篇报告的同仁的指正。

以下是我完成本次实验后的几点体会：

### 1. 充分发挥互联网资源的作用

实验 20 是按创建时间查找文件。我在所查阅到的资料中得知，`find -newer` 命令是根据修改时间进行查找，不符合实验的要求，而获取 Linux 文件的创建时间是一件很麻烦的事情。于是，我使用 macOS 系统的 `GetFileInfo` 命令，这一命令也无法直接用于完成实验要求，还要编写 shell 脚本。后来在一位同学的提醒下，我在 gnu.org 网站上查到了 `find` 命令的`-newer` 选项可以按创建时间查找，这样一条命令就可以完成实验要求。

### 2. 一切皆文件

通过这次实验，我加深了对 Unix “一切皆文件” 这一理念的理解。无论是普通文件还是目录甚至设备，都可以作为文件来处理。这极大地简化了命令行下操作的复杂程度。随着实验的进行，我体会到了这一理念的合理性。无论是普通的文件，还是目录或设备，与外界交互的都是二进制的字节流。在计算机组成课程上，我了解到总线的概念，不同的设备可以通过同一总线与处理器进行数据传输。Unix 操作系统的这一设计也有这样的感觉。“一切皆文件”的理念也符合“简单、集中”的 Unix 哲学。

### 3. 命令即程序

两个月前，我在学生社团的培训上，讲了一节有关命令行操作的课程。当时我就总结出了“命令即程序”这一理念。我认为我能在实验的过程中，没有在如何使用命令的问题上遇到过什么障碍，很大程度上是与理解了这一理念有关。