Operating System Lab 4 : UNIX Shell 实用命令

实 验 报 告

刘子涵　518021910690

【1】实验目的

熟悉和掌握 Unix 的 Shell 一些实用命令，正则表达式和过滤器程序的使用。

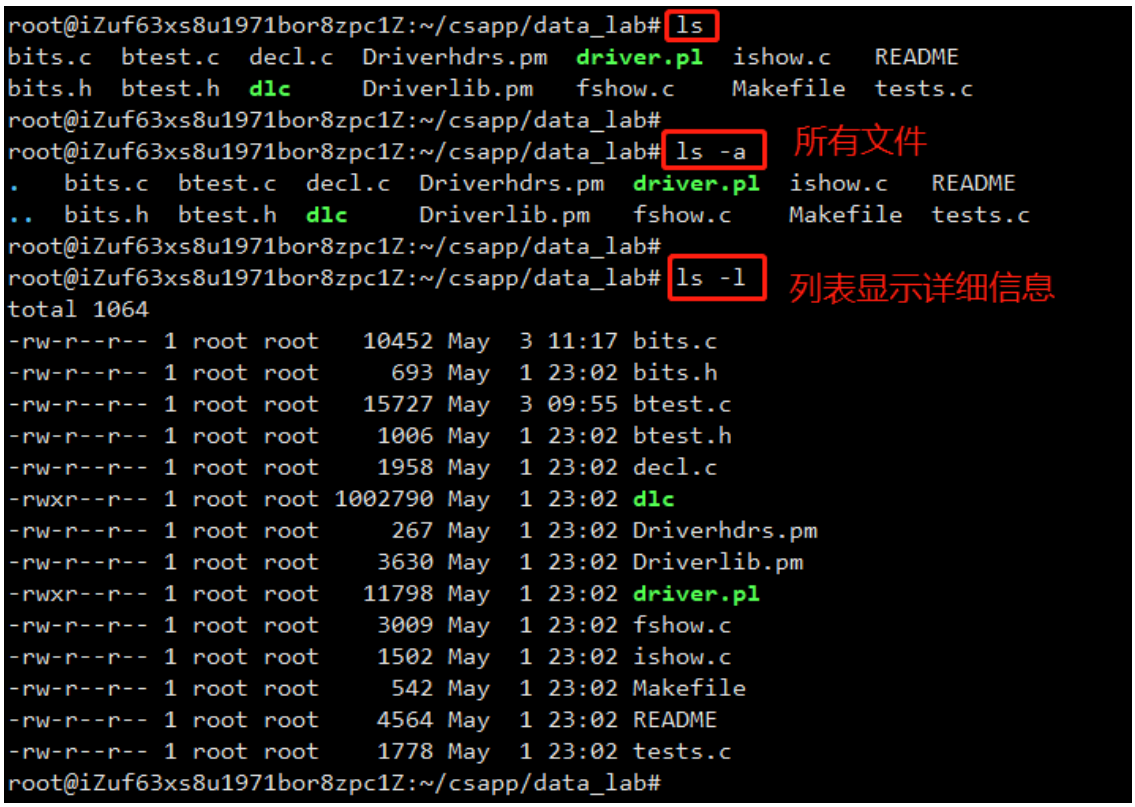
【2】实验环境

- 操作系统：Ubuntu SMP 20.04 x86_64
- Linux Kernel 版本：5.4.0
- GCC 版本：9.3.0

【3】实验题目与解答

1. **练习 `ls`、`ln`、`ln -s`、`diff`、`file`、`chown`、`chmod`、`head`、`tail`、`ps`、`od` 等命令**

   ① 使用 `ls` 命令列出当前目录（默认）的所有文件，`-a` 参数列出所有文件，`-l` 参数以列表形式列出文件的详细属性。截图如下：



   ② 使用 `ln src dst` 为源文件 `src` 建立硬链接（hard link），实则新建一个 dentry，文件名为 `dst`，是一个普通文件，和 `src` 指向同一个内存 inode。使用 `-s` 参数建立软链接/符号链接（soft/symbolic link），此时目标文件 `dst` 是一个 link 型文件，存储 `src` 的绝对路径，类似 Windows 的快捷方式，访问速度比硬链接稍慢些。截图如下：

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls
file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# cat file1
hello world
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ln file1 file2
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ln -s file1 file3
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 8
-rw-r--r-- 2 root root 12 Jun  3 22:15 file1
-rw-r--r-- 2 root root 12 Jun  3 22:15 file2
lrwxrwxrwx 1 root root  5 Jun  3 22:17 file3 -> file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# cat file2
hello world
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# cat file3
hello world
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

硬链接

软链接

硬链接：文件属性相同

软链接：链接文件，指向 file1

cat 查看内容一致

③ 使用 `diff` 命令以编辑指令的形式逐行显示两个文件的差异。截图如下：

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# cat file1
hello world
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# cat file4
cyber security
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# cat file2
hello world
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# diff file1 file4
1c1
< hello world
---
> cyber security
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# diff file1 file2
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

两文件不同，逐行显示差异

两文件相同，不输出

④ 使用 `file` 命令推测文件类型。截图如下：

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls
app.json  file1  file2  file3  file4  manage.py  test  test.c  update.sh
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# file app.json
app.json: JSON data
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# file file1
file1: ASCII text
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# file file3
file3: symbolic link to file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# file manage.py
manage.py: Python script, ASCII text executable
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# file test
test: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linke
d, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=83d116f4686acdbad659a
a3087885c4dbc323b02, for GNU/Linux 3.2.0, not stripped
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# file test.c
test.c: C source, ASCII text
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# file update.sh
update.sh: Bourne-Again shell script, ASCII text executable
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

⑤ 使用 `chown` 命令改变文件属主，`chmod` 命令改变文件存取方式，形式为 `chmod 模式 文件/目录`，其中模式可以为八进制数字，也可以为符号表示。截图如下：

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 0
-rw-r--r-- 1 root root 0 Jun  3 23:10 file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# chown testuser:testgroup file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 0
-rw-r--r-- 1 testuser testgroup 0 Jun  3 23:10 file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# chown root file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 0
-rw-r--r-- 1 root testgroup 0 Jun  3 23:10 file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

修改所属 用户:组

修改所属用户

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 0
-rw-r--r-- 1 root root 0 Jun  3 23:10 file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# chmod 764 file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 0
-rwxrw-r-- 1 root root 0 Jun  3 23:10 file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# chmod u-x file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 0
-rw-rw-r-- 1 root root 0 Jun  3 23:10 file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# chmod a+x file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 0
-rwxrwxr-x 1 root root 0 Jun  3 23:10 file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

所属用户可读、可写、可执行；组用户可读、可写；其他用户可读

去掉所属用户的执行权限

增加所有用户的执行权限

⑥ 使用 `head` 和 `tail` 命令显示文件的头部和尾部。截图如下：

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# head file1
a
b
c
d
e
f
g
h
i
j
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# head -5 file1
a
b
c
d
e
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# tail -5 file1
v
w
x
y
z
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

默认显示头10行

显示头5行

显示尾5行

⑦ 使用 `ps` 命令（process state）查看进程状态。不带参数表示自己终端上启动进程的基本信息，带参数 `-ef` 表示以长列表形式列出所有运行进程的详细信息。截图如下：

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ps
    PID TTY          TIME CMD
 469056 pts/4    00:00:00 bash
 469085 pts/4    00:00:00 ps
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Apr17 ?        00:00:34 /sbin/init noibrs
root           2       0  0 Apr17 ?        00:00:00 [kthreadd]
root           3       2  0 Apr17 ?        00:00:00 [rcu_gp]
root           4       2  0 Apr17 ?        00:00:00 [rcu_par_gp]
root           6       2  0 Apr17 ?        00:00:00 [kworker/0:0H-kblockd]
```

⑧ 使用 `od` 命令（octal dump）查看二进制文件的内容。部分参数如下：

`-b` ：八进制，单字节

`-c` ：可打印字符和转义字符（以 ASCII 格式显示）

`-d` ：无符号十进制，两字节

`-o` ：八进制，两字节

`-x` ：十六进制，两字节

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# od -b test > test-octal-1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# head -5 test-octal-1
0000000 177 105 114 106 002 001 001 000 000 000 000 000 000 000 000 000
0000020 003 000 076 000 001 000 000 000 140 020 000 000 000 000 000 000
0000040 100 000 000 000 000 000 000 000 170 071 000 000 000 000 000 000
0000060 000 000 000 000 100 000 070 000 015 000 100 000 037 000 036 000
0000100 006 000 000 000 004 000 000 000 100 000 000 000 000 000 000 000
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# od -c test > test-ascii
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# head -5 test-ascii
0000000 177   E   L   F 002 001 001  \0  \0  \0  \0  \0  \0  \0  \0
0000020 003  \0   >  \0 001  \0  \0  \0   ` 020  \0  \0  \0  \0  \0  \0
0000040   @  \0  \0  \0  \0  \0  \0  \0   x   9  \0  \0  \0  \0  \0  \0
0000060  \0  \0  \0  \0   @  \0   8  \0  \r  \0   @  \0 037  \0 036  \0
0000100 006  \0  \0  \0 004  \0  \0  \0   @  \0  \0  \0  \0  \0  \0  \0
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# od -d test > test-decimal-2
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# head -5 test-decimal-2
0000000 17791 17996   258     1     0     0     0     0
0000020     3    62     1     0  4192     0     0     0
0000040    64     0     0     0 14712     0     0     0
0000060     0     0    64    56    13    64    31    30
0000100     6     0     4     0    64     0     0     0
```

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# od -o test > test-octal-2
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# head -5 test-octal-2
0000000 042577 043114 000402 000001 000000 000000 000000 000000
0000020 000003 000076 000001 000000 010140 000000 000000 000000
0000040 000100 000000 000000 000000 034570 000000 000000 000000
0000060 000000 000000 000100 000070 000015 000100 000037 000036
0000100 000006 000000 000004 000000 000100 000000 000000 000000
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# od -x test > test-hexadecimal-2
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# head -5 test-hexadecimal-2
0000000 457f 464c 0102 0001 0000 0000 0000 0000
0000020 0003 003e 0001 0000 1060 0000 0000 0000
0000040 0040 0000 0000 0000 3978 0000 0000 0000
0000060 0000 0000 0040 0038 000d 0040 001f 001e
0000100 0006 0000 0004 0000 0040 0000 0000 0000
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

- **假定当前目录中没有 `ls.save` 文件，先后键入 `ls -l` 和 `ls -l > ls.save` 两个命令。请将第一个 `ls` 命令的输出和第二个命令的输出文件 `ls.save` 中的内容进行对比，有何不同？解释。**

  第一个 `ls` 命令输出显示的文件所占数据块总数为 28，而第二个 `ls` 命令输出文件 `ls.save` 中内容显示文件所占数据块总数仍为 28，仔细观察可发现 `ls.save` 所占字节数为 0。

  原因是：Shell 命令行解释器首先解析 `ls -l > ls.save` 命令，解释器知道该命令会写一个 `ls.save` 文件，而当前目录并没有该文件，所以会调用 open 系统调用创建一个空文件（占 0 字节）。然后，Shell 命令行解释器首先执行 `ls -l` 命令，然后将标准输出重定向到 `ls.save` 文件，此时才真正确定 `ls.save` 文件的大小，但 `ls.save` 中显示的大小是输出重定向前刚创建后的大小，即 0 字节。再次 `ls -l` 命令，可以得到 `ls.save` 的当前大小，为 203 字节，所占磁盘数据块增加到 32。使用 `ls -s1` 可具体查看每个文件所占数据块的数目。

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 28
-rw-r--r-- 1 root root    52 Jun  4 08:07 file1
-rwxr-xr-x 1 root root 16696 Jun  4 08:21 test
-rw-r--r-- 1 root root    82 Jun  4 08:21 test.c
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l > ls.save
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# cat ls.save
total 28
-rw-r--r-- 1 root root    52 Jun  4 08:07 file1
-rw-r--r-- 1 root root     0 Jun  4 08:52 ls.save
-rwxr-xr-x 1 root root 16696 Jun  4 08:21 test
-rw-r--r-- 1 root root    82 Jun  4 08:21 test.c
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -l
total 32
-rw-r--r-- 1 root root    52 Jun  4 08:07 file1
-rw-r--r-- 1 root root   203 Jun  4 08:52 ls.save
-rwxr-xr-x 1 root root 16696 Jun  4 08:21 test
-rw-r--r-- 1 root root    82 Jun  4 08:21 test.c
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls -s1
total 32
 4 file1
 4 ls.save
20 test
 4 test.c
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

- **比较 `ls`、`ls *`、`ls .`、`ls .*`、`ls .?*`，这些命令输出有何不同？**

`ls` 列出当前目录下的所有文件（不包括隐藏文件），`ls` 可以跟一个路径参数，表示列出相应路径下的所有文件，这个参数支持正则表达式。所以：

`ls *` 列出当前目录及其所有子目录的文件，这里包括 `ls` 和 `ls folder`；

`ls .` 列出当前目录下所有文件（不包括隐藏文件），同 `ls`；

`ls .*` 列出至少零个 `.` 的路径下所有文件，这里包括 `.` 和 `..`；

`ls .?*` 列出以 `.` 开头，后面匹配至少一个字符的路径下所有文件，这里只包括 `..`。（如果有 `.ssh` 这样的文件，也符合要求，也会列出）

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls
file1  folder  test  test.c
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls *
file1  test  test.c

folder:
file2
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls .
file1  folder  test  test.c
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls .*
.:
file1  folder  test  test.c

..:
lab1  lab2  lab3  lab4
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls .?*
lab1  lab2  lab3  lab4
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

- **显示某一文件从 n1 行到 n2 行的内容。**

  比如，显示从 6 到 10 行的内容：

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# head -10 file1 | tail -5
f
g
h
i
j
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

- **以下两条命令：**
  - `mv file1 file2`
  - `cp file1 file2; rm file1`

  的执行效果是否总是相同的？仔细推敲并验证。

  相同。第一条命令是直接将 file1 移动到 file2，如果在同一目录下则为重命名 file1 为 file2。第二条命令是将 file1 复制到 file2，然后删除 file1，本质上是将 file1 移动到了 file2。

```
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls
file1  folder  test  test.c
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# mv file1 file2
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls
file2  folder  test  test.c
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# mv file2 file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# cp file1 file2; rm file1
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4# ls
file2  folder  test  test.c
root@iZuf63xs8u1971bor8zpc1Z:~/os/lab4#
```

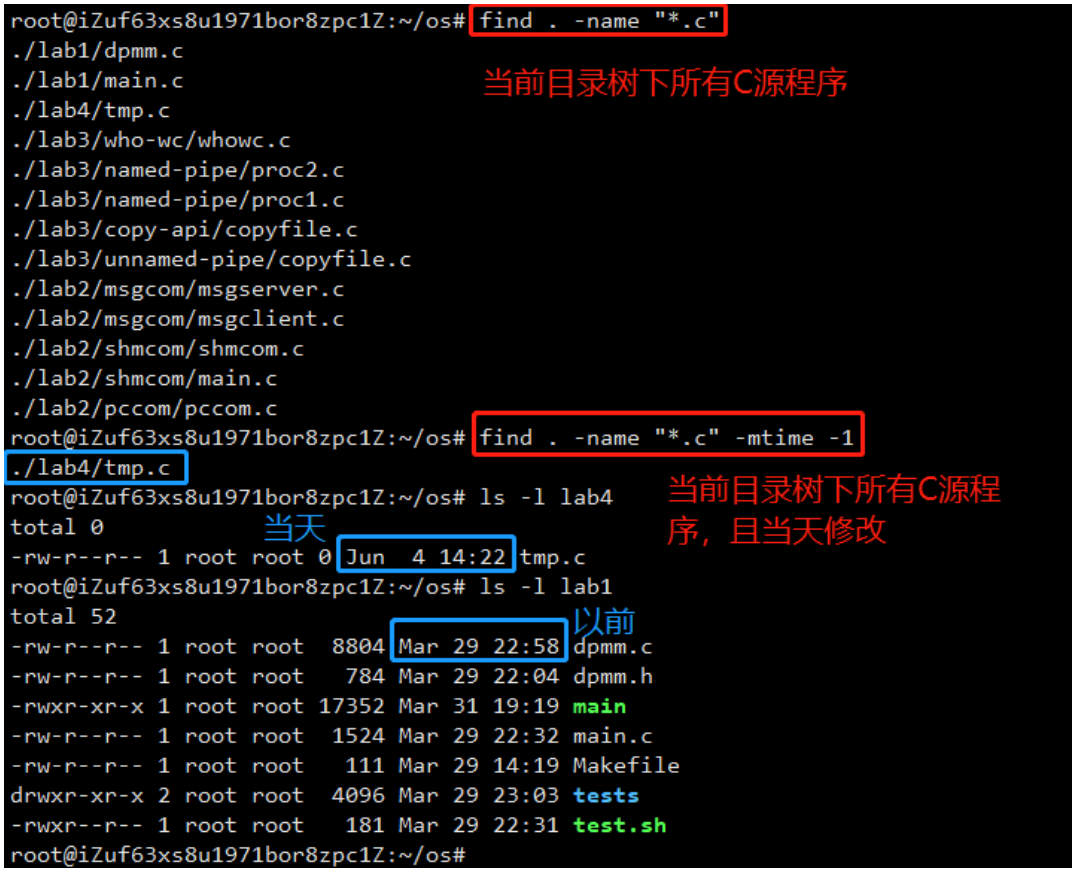
2. **练习 `find` 命令**

`find` 命令用于文件查找，`-mtime` 指定文件的修改时间，`-name` 指定文件名，`-print` 打印找到文件的带有路径的文件名，`-exec` 对找到文件要执行的 Shell 命令，`-ok` 类似 `-exec` ，执行命令前等待用户确认。

- **在当前的目录树中显示当天修改的 C 源程序名。**

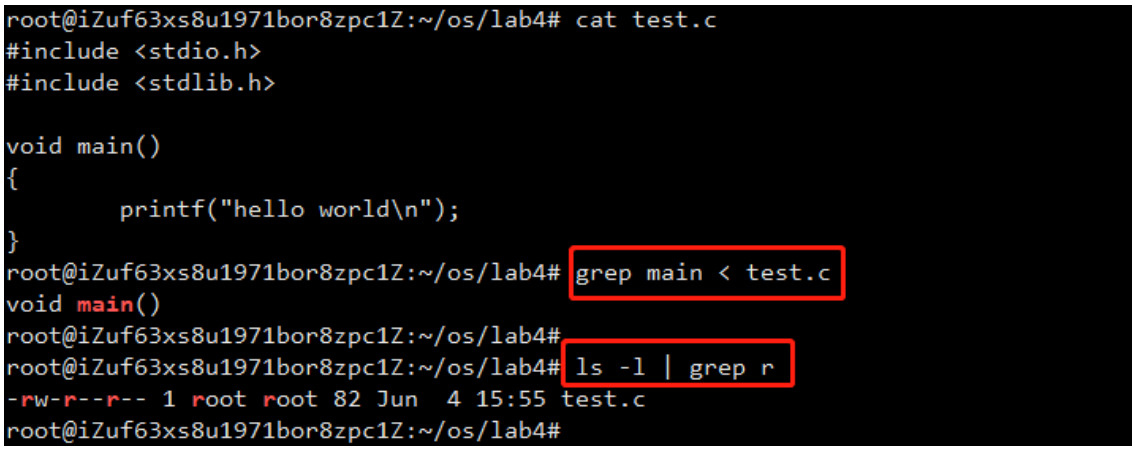
```
find . -name "*.c" -mtime -1
```

截图如下：

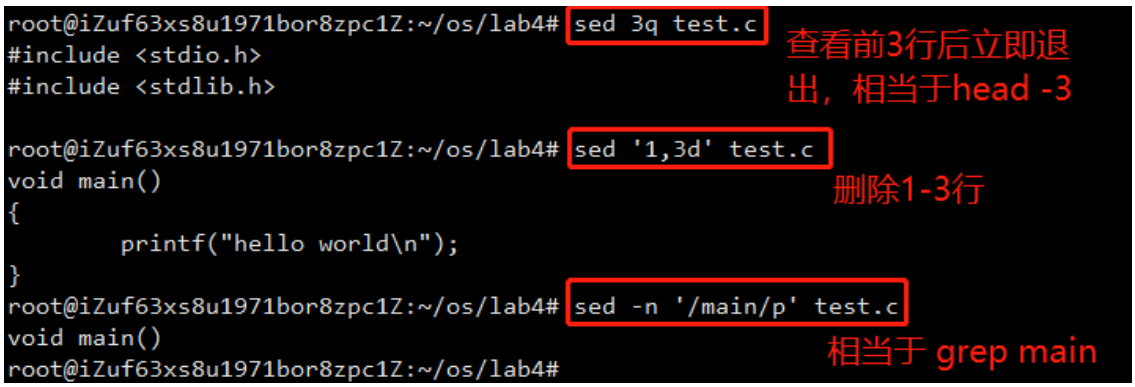


3. **掌握正则表达式的匹配规则，练习 `grep`、`sed`、`awk` 等命令。**

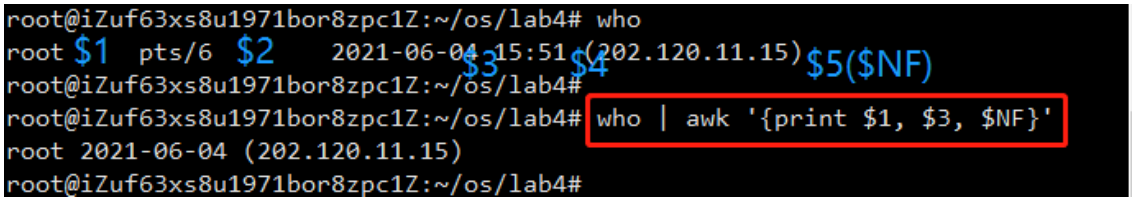
`grep` 命令可以在输入数据中查找包含指定模式的所有行，并将这些行写至标准输出，支持带通配符的字符串模式查找。举例如下：



`sed` 命令是用于过滤和转换文本的流编辑器，对于输入流（如文件）对其中的文本做基本变换。注意 `sed` 不改变源文件。举例如下：



`awk` 是一种模式查找和处理语言，支持面向字段或域的操作，是 UNIX 中功能最强的过滤器。每次从文件中读取一行，然后依次和每一个模式比较，如匹配，就执行相应的动作。举例如下：

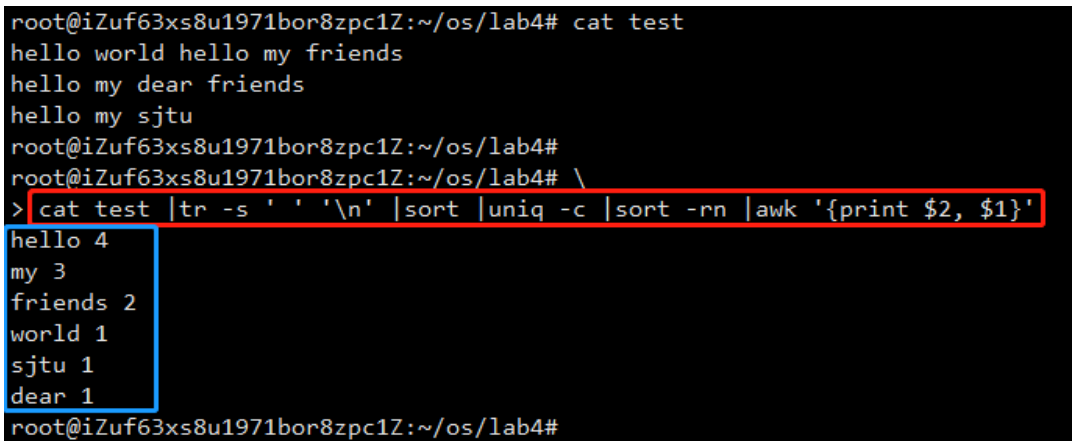


- **按相反的次序显示输入行及行号。**

  使用 `sort -r` 以相反次序显示，使用 `cat -n` 输出行号。

- **统计输入单词的频度。**

  1. 使用 `cat` 查看 `test` 文件；
  2. 使用 `tr` 将 `' '` 转换为（translate）`'\n'`；
  3. 使用 `sort` 进行排序，并使用 `uniq -c` 进行去重计数；
  4. 使用 `sort -rn` 以数值大小逆序排序；
  5. 使用 `awk` 打印结果，先打印字段 2（即单词），再打印字段 1（即数量）



【4】实验心得

本次实验主要是在 Linux Shell 命令行完成，在实践中我对许多常用的 Shell 命令更加熟悉，对正则表达式、过滤器 `grep/sed/awk` 的使用有所了解。Shell 命令的熟练使用和 Shell 程序设计对我将来的科研有着重要意义，在本次实验后我还将自主学习更多 Shell 命令的使用并熟练掌握。感谢老师课堂的讲解！