

实验报告

实验名称	实验一 Linux 常用命令一		
实验教室	丹青 918	实验日期	2023 年 3 月 10 日
学 号	2021213095	姓 名	蔡浩楠
专业班级	2021 级计算机科学与技术 3 班		
指导教师	卢洋		

东北林业大学
计算机科学与技术专业

一、 实验目的

- 1、掌握 Linux 下文件和目录操作命令：cd、ls、mkdir、rmdir、rm
- 2、掌握 Linux 下文件信息显示命令：cat、more、head、tail
- 3、掌握 Linux 下文件复制、删除及移动命令：cp、mv
- 4、掌握 Linux 的文件排序命令：sort

二、 实验环境

(1) 计算机的硬件配置 PC 系列微机。 (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、实验内容及结果

1. 使用命令切换到/etc 目录，并显示当前工作目录路径

```
kai@localhost etc]$ cd /etc
kai@localhost etc]$ pwd
/etc
kai@localhost etc]$ █
```

2. 使用命令显示/home/lyj 目录下所有文件目录的详细信息，包括隐藏文件。

```
[root@localhost linux]# cd /home
[root@localhost home]# ls -a
.  ..  caihaonan  kai  linux
[root@localhost home]# cd /home
[root@localhost home]# cd /linux
bash: cd: /linux: 没有那个文件或目录
[root@localhost home]#
[root@localhost home]# cd linux
[root@localhost linux]# ls -a
.  ..  test.txt
[root@localhost linux]# █
```

3. 使用命令创建目录/home/lyj/linux，然后删除该目录。

```
[root@localhost linux]# cd /home
[root@localhost home]# cd linux
[root@localhost linux]# mkdir Linuxtest
[root@localhost linux]# ls
linuxtest  Linuxtest  test.txt
[root@localhost linux]# rmdir Linux
rmdir: 删除 "Linux" 失败: 没有那个文件或目录
[root@localhost linux]# rmdir Linuxtest
[root@localhost linux]# ls
linuxtest  test.txt
[root@localhost linux]#
```

4. 使用命令 cat 用输出重定向在/home/lyj 目录下创建文件 abc，文件内容为“Hello, Linux!”，并查看该文件的内

```
[root@localhost linux]# cat >abc
Hello linux!
[root@localhost linux]# cat abc
Hello linux!
[root@localhost linux]#
```

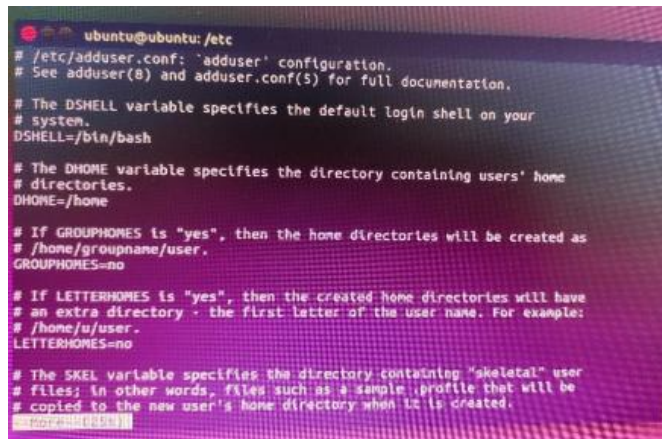
5、使用命令创建目录/home/lyj/ak，然后将/home/lyj/abc 文件 复制到该目录下，最后将该目录及其目录下的文件一起删除。

```
[zengfanmin@zengfanminindexuniji ~]$ ls
abc ak 公共 模板 视频 图片 文档 下载 音乐 桌面
[zengfanmin@zengfanminindexuniji ~]$ cp -r abc ak
[zengfanmin@zengfanminindexuniji ~]$ ls
abc ak 公共 模板 视频 图片 文档 下载 音乐 桌面
[zengfanmin@zengfanminindexuniji ~]$ cd ak
[zengfanmin@zengfanminindexuniji ak]$ ls
abc
[zengfanmin@zengfanminindexuniji ak]$ rm -i abc
rm: 是否删除普通文件 "abc"?
[zengfanmin@zengfanminindexuniji ak]$ ls
abc
[zengfanmin@zengfanminindexuniji ak]$ cd ..
[zengfanmin@zengfanminindexuniji ~]$ cd ak
[zengfanmin@zengfanminindexuniji ak]$ ls
abc
[zengfanmin@zengfanminindexuniji ak]$ rm -i abc
rm: 是否删除普通文件 "abc"? y
[zengfanmin@zengfanminindexuniji ak]$ ls
[zengfanmin@zengfanminindexuniji ak]$ cd ..
[zengfanmin@zengfanminindexuniji ~]$ rmdir ak
[zengfanmin@zengfanminindexuniji ~]$ ls
abc 公共 模板 视频 图片 文档 下载 音乐 桌面
[zengfanmin@zengfanminindexuniji ~]$
```

6、查看文件 /etc/adduser.conf 的前 3 行内容，查看文件 /etc/adduser.conf 的最后 5 行内容

```
81 # option above will be default behavior for adding
82 #ADD_EXTRA_GROUPS=1
83
84 # check user and group names also against this reg
85 #NAME_REGEX="^[a-z][-a-z0-9_]*\ $"
86
87 # use extrausers by default
88 #USE_EXTRAUSERS=1
ubuntu@ubuntu:/etc$ head -n 3 adduser.conf
# /etc/adduser.conf: 'adduser' configuration.
# See adduser(8) and adduser.conf(5) for full documentatio
ubuntu@ubuntu:/etc$ tail -n 5 adduser.conf
# check user and group names also against this regular exp
#NAME_REGEX="^[a-z][-a-z0-9_]*\ $"
# use extrausers by default
#USE_EXTRAUSERS=1
```

7、分屏查看文件 /etc/adduser.conf 的内容。

A terminal window with a dark background and light-colored text. The prompt is 'ubuntu@ubuntu: /etc'. The text shows the configuration for the 'adduser' command, including comments and variable assignments for DSHELL, DHOME, GROUPTHOMES, LETTERHOMES, and SKEL.

```
ubuntu@ubuntu: /etc
# /etc/adduser.conf: 'adduser' configuration.
# See adduser(8) and adduser.conf(5) for full documentation.

# The DSHELL variable specifies the default login shell on your
# system.
DSHELL=/bin/bash

# The DHOME variable specifies the directory containing users' home
# directories.
DHOME=/home

# If GROUPTHOMES is "yes", then the home directories will be created as
# /home/groupname/user.
GROUPTHOMES=no

# If LETTERHOMES is "yes", then the created home directories will have
# an extra directory - the first letter of the user name. For example:
# /home/u/user.
LETTERHOMES=no

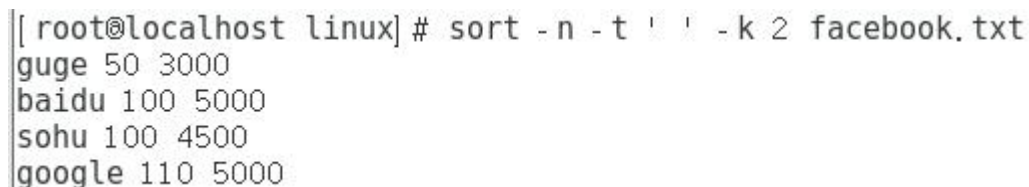
# The SKEL variable specifies the directory containing "skeletal" user
# files; in other words, files such as a sample profile that will be
# copied to the new user's home directory when it is created.
more...
```

8、使用命令 cat 用输出重定向在/home/lyj 目录下创建文件 facebook.txt，文件内容为：

A terminal window with a dark background and light-colored text. The prompt is '[root@localhost linux]'. The user enters 'cat >facebook.txt' and then types the content of the file. Then they enter 'cat facebook.txt' to view the content. Finally, they enter 'sort facebook.txt' to sort the content.

```
[root@localhost linux] # cat >facebook.txt
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
[root@localhost linux] # cat facebook.txt
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
[root@localhost linux] # sort facebook.txt
baidu 100 5000
google 110 5000
guge 50 3000
sohu 100 4500
```

9、第一列为公司名称，第2列为公司人数，第3列为员工平均工资。利用 sort 命令完成下列排序：（1）按公司字母顺序排序（2）按公司人数排序。（3）按公司人数排序，人数相同的按照员工平均工资升序排序（4）按员工工资降序排序，如工资相同，则按公司人数升序排序（5）从公司英文名称的第2个字母开始进行排序。

A terminal window with a dark background and light-colored text. The prompt is '[root@localhost linux]'. The user enters a complex sort command: 'sort -n -t ' ' -k 2 facebook.txt'. The output shows the file content sorted by the second column (company count) in ascending order.

```
[root@localhost linux] # sort -n -t ' ' -k 2 facebook.txt
guge 50 3000
baidu 100 5000
sohu 100 4500
google 110 5000
```

```
[root@localhost linux]# sort -n -t ' ' -k 2 -k 3 facebook.txt
guge 50 3000
sohu 100 4500
baidu 100 5000
google 110 5000
```

```
[root@localhost linux]# sort -n -t ' ' -k 2 -k 3 facebook.txt
guge 50 3000
sohu 100 4500
baidu 100 5000
google 110 5000
```

```
baidu 100 5000
sohu 100 4500
google 110 5000
guge 50 3000
```

四、实验过程分析与讨论

实验过程中不太熟悉命令，会在一些比较简单的地方犯错。比如用 `cat` 命令的操作方法，用 `rm -i` 交互式删除文件需要按 `y` 才能成功删除。

五、指导教师意见

指导教师签字：卢洋

2023 年 3 月 10 日

实验报告

实验名称	实验二 Linux 常用命令（二）		
实验教室	丹青 918	实验日期	2023 年 3 月 16 日
学 号	2021213095	姓 名	蔡浩楠
专业班级	2021 级计算机科学与技术 3 班		
指导教师	卢洋		

东北林业大学
计算机科学与技术专业

一、 实验目的

1. 掌握 Linux 下查找文件和统计文件行数、字数和字节数命令：
find、wc;
2. 掌握 Linux 下文件打包命令：tar;
3. 掌握 Linux 下符号链接命令和文件比较命令：ln、comm、diff;
4. 掌握 Linux 的文件权限管理命令：chmod。

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、实验内容及结果

1、查找指定文件

(1) 在用户目录下新建目录 baz，在 baz 下新建文件 qux，并写下任意几行内容；

```
[kai@localhost ~]$ mkdir baz
[kai@localhost ~]$ ls
baz  公共  模板  视频  图片  文档  下载  音乐  桌面
[kai@localhost ~]$ cd baz
[kai@localhost baz]$ vim qux
[kai@localhost baz]$ ls
qux
[kai@localhost baz]$ cat >qux <<EOF
> HELLO Linux!
> karly
> what is your name?
~ EOF
```

(2) 在用户目录下查找文件 qux，并显示该文件位置信息；

```
[kai@localhost baz]$ ls
qux
[kai@localhost baz]$ cd ..
[kai@localhost ~]$ find -name qux
./baz/qux
[kai@localhost ~]$ ls -i -F qux
ls: 无法访问 qux: 没有那个文件或目录
[kai@localhost ~]$ cd baz
[kai@localhost baz]$ ls -i -F qux
68784757 qux
[kai@localhost baz]$ █
```

(3) 统计文件 qux 中所包含内容的行数、字数和字节数；

```
[kai@localhost baz]$ wc -l qux
3 qux
[kai@localhost baz]$ wc -m qux
38 qux
[kai@localhost baz]$
[kai@localhost baz]$ wc -c qux
38 qux
[kai@localhost baz]$ wc qux
3 7 38 qux
[kai@localhost baz]$
```

(4) 在用户目录下查找文件 qux，并删除该文件；

```
[kai@localhost baz]$ cd
[kai@localhost ~]$ ls
baz 公共 模板 视频 图片 文档 下载 音乐 桌面
[kai@localhost ~]$ cd baz
[kai@localhost baz]$ find -name qux -exec rm -rf {} \;
```

(5) 查看文件夹 baz 内容，看一下是否删除了文件 qux。

```
kai@localhost ~]$ cd baz
kai@localhost baz]$ ls
kai@localhost baz]$ █
```

2、文件打包

(1) 在用户目录下新建文件夹 path1，在 path1 下新建文件 file1 和 file2；

```
[kai@localhost ~]$
[kai@localhost ~]$ mkdir path1
[kai@localhost ~]$ cd path1
bash: cd: path1: 没有那个文件或目录
[kai@localhost ~]$ cd path1
[kai@localhost path1]$ touch file1
[kai@localhost path1]$ touch file2
[kai@localhost path1]$ ls
file1  file2
```

(2) 在用户目录下新建文件夹 path2，在 path2 下新建文件 file3；

```
[kai@localhost path1]$ cd
[kai@localhost ~]$ mkdir path2
[kai@localhost ~]$ cd path2
[kai@localhost path2]$ touch file3
[kai@localhost path2]$ ls
file3
[kai@localhost path2]$ █
```

(3) 在用户目录下新建文件 file4；

```
[kai@localhost ~]$ tar -cvf package.tar file4 path1
file4
path1/
path1/file1
path1/file2
```

(4) 在用户目录下对文件夹 path1 和 file4 进行打包，生成文件 package.tar；

```
[ kai@localhost ~]$ cd
[ kai@localhost ~]$ tar -rf package.tar path2
[ kai@localhost ~]$ tar -tf package.tar
file4
path1/
path1/file1
path1/file2
path2/
path2/file3
[ kai@localhost ~]$
```

(5) 查看包 package.tar 的内容;

```
[ kai@localhost ~]$ cat package.tar
file40000664000175000017500000000000014432673524010100 0ustar kaikaipath1/00007
750001750000175000000000000014432673057010202 5ustar kaikaipath1/file10000664000
17500001750000000000000014432673051011105 0ustar kaikaipath1/file2000066400017500
00175000000000000014432673057011114 0ustar kaikai[ kai@localhost ~]$
```

(6) 向包 package.tar 里添加文件夹 path2 的内容

```
[ kai@localhost ~]$ cd
[ kai@localhost ~]$ tar -rf package.tar path2
[ kai@localhost ~]$ tar -tf package.tar
file4
path1/
path1/file1
path1/file2
path2/
path2/file3
[ kai@localhost ~]$
```

(7) 将包 package.tar 复制到用户目录下的新建文件夹 path3 中;

```
[ kai@localhost ~]$ mkdir path3
[ kai@localhost ~]$ cp package.tar path3
[ kai@localhost ~]$ cd path3
[ kai@localhost path3]$ ls
package.tar
[ kai@localhost path3]$
```

(8) 进入path3文件夹, 并还原包 package.tar 的内容。

```
file4
path1/
path1/file2
path1/file1
path2/
path2/file3
```

4. 符号链接内容

(1) 新建文件 foo.txt, 内容为 123 。

```
[kai@localhost path3]$ echo 123 > foo.txt
[kai@localhost path3]$ ls
foo.txt  package.tar
[kai@localhost path3]$ cat foo.txt
123
[kai@localhost path3]$
```

(2) 建立foo.txt 的硬链接文件 bar.txt, 并比较 bar.txt 的内容和 foo.txt 是否相同, 要求用comm 或 diff 命令;

```
[kai@localhost ~]$ cd
[kai@localhost ~]$ echo 123 > foo.txt
[kai@localhost ~]$ ls
baz      foo.txt      path1  path3  模板  图片  下载  桌面
file4    package.tar  path2  公共  视频  文档  音乐
[kai@localhost ~]$ cat foo.txt
123
[kai@localhost ~]$
```

(3) 查看 foo.txt 和 bar.txt 的 i 节点号 (inode) 是否相同;

```
[kai@localhost ~]$ ls -i -F foo.txt
102993096 foo.txt
[kai@localhost ~]$ ls -i -F bar.txt
102993096 bar.txt
[kai@localhost ~]$
```

(4) 修改 bar.txt 的内容为 abc, 然后通过命令判断 foo.txt 与 bar.txt 是否相同;

```
[kai@localhost ~]$ diff foo.txt bar.txt
[kai@localhost ~]$ comm foo.txt bar.txt
      abc
[kai@localhost ~]$ cat foo.txt
abc
```

(5) 删除 foo.txt文件, 然后查看 bar.txt文件的 inode 号及内容;

```
[kai@localhost ~]$ rm foo.txt
[kai@localhost ~]$ ls -i -F bar.txt
102993096 bar.txt
[kai@localhost ~]$ cat bar.txt
abc
[kai@localhost ~]$
```

(6) 创建文件 bar.txt 的符号链接文件 baz.txt, 然后查看 bar.txt 和 baz.txt 的 inode 号, 并观察两者是否相同, 比较 bar.txt 和 baz.txt 的文

件内容是否相同；

```
[kai@localhost ~]$ ln -s bar.txt baz.txt
[kai@localhost ~]$ ls
bar.txt  baz.txt  package.tar  path2  公共  视频  文档  音乐
baz      file4    path1        path3  模板  图片  下载  桌面
[kai@localhost ~]$ ls -i -F bar.txt
102993096 bar.txt
[kai@localhost ~]$ ls -i -F baz.txt
102520272 baz.txt@
[kai@localhost ~]$ comm baz.txt bar.txt
      abc
[kai@localhost ~]$
```

(7) 删除 bar.txt，查看文件 baz.txt，观察系统给出什么提示信息。

```
[kai@localhost ~]$ rm bar.txt
[kai@localhost ~]$ cat baz.txt
cat: baz.txt: 没有那个文件或目录
[kai@localhost ~]$
```

5. 权限管理

(1) 新建文件 qux.txt；

```
[kai@localhost ~]$ touch qux.txt
[kai@localhost ~]$ ls
baz      file4    path1  path3  公共  视频  文档  音乐
baz.txt  package.tar  path2  qux.txt  模板  图片  下载  桌面
[kai@localhost ~]$ chmod a+x qux.txt
[kai@localhost ~]$ ls -al qux.txt
-rwxrwxr-x. 1 kai kai 0 5月 23 11:42 qux.txt
[kai@localhost ~]$
```

(2) 为文件 qux.txt 增加执行权限（所有用户都可以执行）。

```
[kai@localhost ~]$ chmod a+x qux.txt
[kai@localhost ~]$ ls -al qux.txt
-rwxrwxr-x. 1 kai kai 0 5月 23 11:42 qux.txt
[kai@localhost ~]$
```

四、实验过程分析与讨论 对于很多命令的不熟悉导致做实验需要花大量的时间，所以 Linux 这门课需要不断地练习来达到熟能生巧的目的。往往一个命令拥有很多的参数，能够实现各种各样的功能。其中 ln 命令生成硬链接时是使当前文件指向该文件的索引号，这样删除另一个文件，通过硬链接仍然可以访问到以前的数据。而生成软连接时实际上是生成了一个包含另一文件的位置信息的文本文件，当源文件删除，该文件便不可访问内容。

五、指导教师意见

指导教师签字：卢洋

2023 年 3 月 16 日

实验报告

实验名称	实验三 vim 编辑器及 gcc 编译器的使用		
实验教室	丹青 918	实验日期	2023 年 3 月 23 日
学 号	2021213095	姓 名	蔡浩楠
专业班级	2021 级计算机科学与技术 3 班		
指导教师	卢洋		

东北林业大学

计算机科学与技术专业

一、 实验目的 掌握 vim 编辑器及 gcc 编译器的使用方法。

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、实验内容及结果

(1) 在用户目录下新建一个目录，命名为 workspace1;

```
[kai@localhost ~]$ mkdir workspace1
[kai@localhost ~]$ ls
workspace1 公共 模板 视频 图片 文档 下载 音乐 桌面
```

(2) 进入目录 workspace1

```
kai@localhost ~]$ cd workspace1
kai@localhost workspace1]$ ls
kai@localhost workspace1]$ vim
kai@localhost workspace1]$ vim test.c
kai@localhost workspace1]$ vim test.c
kai@localhost workspace1]$ █
```

(3) 在 workspace1 下用vim 编辑器新建一个 c 语言程序文件， 文件名为 test.c， 内容为：

```
#include <stdio.h>
int main()
{
    printf("hello world!\n");
    return 0;
}
~
~
~
~
~
```

(4) 保存 test.c 的内容， 并退出；

```
#include <stdio.h>
int main()
{
    printf("hello world!\n");
    return 0;
}
~
~
~
~
~
```

(5) 编译 test.c文件，生成可执行文件 test，并执行，查看 执行结果。

```
[kai@localhost workspace1]$ gcc test.c -o test
[kai@localhost workspace1]$ ./test
hello world!
[kai@localhost workspace1]$
```


2. vim 编辑器的详细使用:

(1) 在用户目录下创建一个名为 workspace2 的目录;

```
[kai@localhost workspace1]$ cd
[kai@localhost ~]$ mkdir workspace2
[kai@localhost ~]$ ls
workspace1 workspace2 公共 模板 视频 图片 文档 下载 音乐 桌
fo [kai@localhost ~]$
```

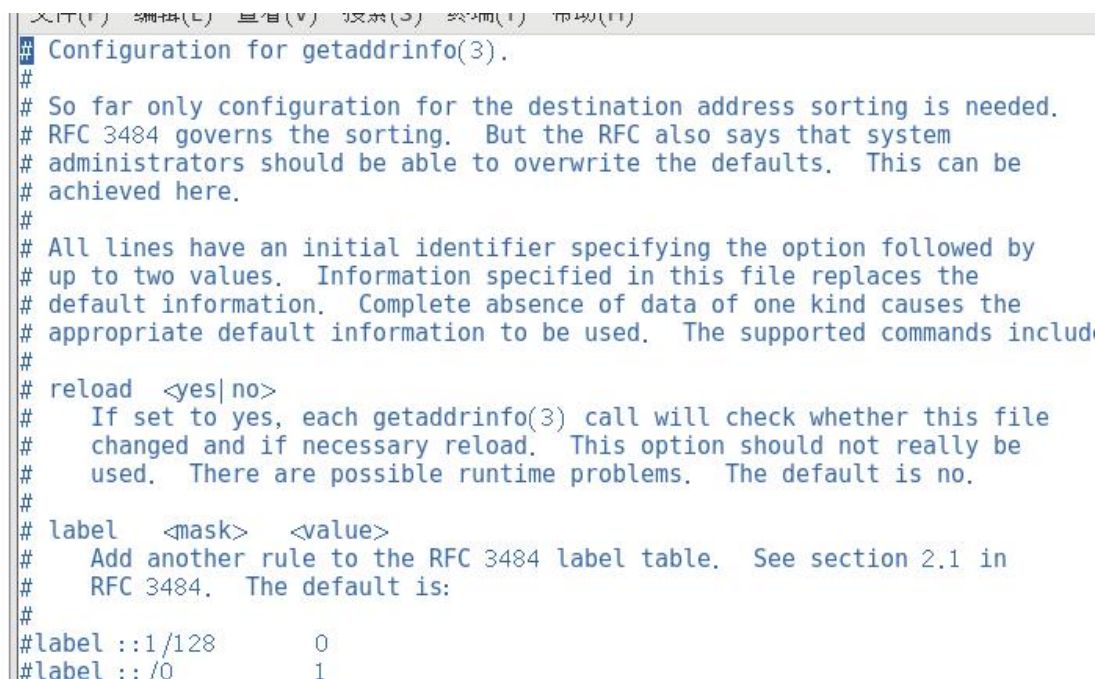
(2) 进入workspace2目录;

```
kai@localhost ~]$ cd workspace2
kai@localhost workspace2]$
```

(3) 使用以下命令: 将文件/etc/gai.conf 的内容复制到当前 目录下的新建文件 gai.conf 中

```
[kai@localhost workspace2]$ cat /usr/share/doc/glibc-common-2.17/gai.conf > ./gai.conf
[kai@localhost workspace2]$ ls
gai.conf
```

(4) 使用vim 编辑当前目录下的 gai.conf;



```
Configuration for getaddrinfo(3).
#
# So far only configuration for the destination address sorting is needed.
# RFC 3484 governs the sorting. But the RFC also says that system
# administrators should be able to overwrite the defaults. This can be
# achieved here.
#
# All lines have an initial identifier specifying the option followed by
# up to two values. Information specified in this file replaces the
# default information. Complete absence of data of one kind causes the
# appropriate default information to be used. The supported commands includ
#
# reload <yes|no>
#   If set to yes, each getaddrinfo(3) call will check whether this file
#   changed and if necessary reload. This option should not really be
#   used. There are possible runtime problems. The default is no.
#
# label <mask> <value>
#   Add another rule to the RFC 3484 label table. See section 2.1 in
#   RFC 3484. The default is:
#
#label ::1/128 0
#label ::/0 1
```

(5) 将光标移到第 18行;

```

11 # appropriate address information to be used. The upper 32 bits
12 #
13 # reload <yes|no>
14 #   If set to yes, each getaddrinfo(3) call will check whether t
15 #   changed and if necessary reload. This option should not be
16 #   used. There are possible runtime problems. The default is
17 #
18 # label <mask> <value>
19 #   Add another rule to the RFC 3484 label table. See section 2
20 #   RFC 3484. The default is:
21 #
22 #label ::1/128      0
23 #label ::1/128      1

```

(6). 复制该行内容;

```

11 # appropriate address information to be used. The upper 32 bits
12 #
13 # reload <yes|no>
14 #   If set to yes, each getaddrinfo(3) call will check whether t
15 #   changed and if necessary reload. This option should not be
16 #   used. There are possible runtime problems. The default is
17 #
18 # label <mask> <value>
19 #   Add another rule to the RFC 3484 label table. See section 2
20 #   RFC 3484. The default is:
21 #
22 #label ::1/128      0
23 #label ::1/128      1

```

7 将光标移到最后一行行首;

```

49 #precedence ::/96      20
50 #precedence ::ffff:0:0/96  10
51 #
52 #   For sites which prefer IPv4 connections change the last line to
53 #
54 #precedence ::ffff:0:0/96  100
55 #
56 #
57 # scopeev4 <mask> <value>
58 #   Add another rule to the RFC 6724 scope table for IPv4 addresses.
59 #   By default the scope IDs described in section 3.2 in RFC 6724 are
60 #   used. Changing these defaults should hardly ever be necessary.
61 #   The defaults are equivalent to:
62 #
63 #scopeev4 ::ffff:169.254.0.0/112  2
64 #scopeev4 <mask> <value>ev4 ::ffff:127.0.0.0/104  2
65 #scopeev4 ::ffff:0:0.0.0/96      14

```

8. 粘贴复制行的内容;

```

49 #precedence :: /96 20
50 #precedence :: ffff:0:0/96 10
51 #
52 # For sites which prefer IPv4 connections change the last line to
53 #
54 #precedence :: ffff:0:0/96 100
55 #
56 #
57 # scopev4 <mask> <value>
58 # Add another rule to the RFC 6724 scope table for IPv4 addresses.
59 # By default the scope IDs described in section 3.2 in RFC 6724 are
60 # used. Changing these defaults should hardly ever be necessary.
61 # The defaults are equivalent to:
62 #
63 #scopev4 :: ffff:169.254.0.0/112 2
64 #scopev4 <mask> <value>pev4 :: ffff:127.0.0.0/104 2
65 #scopev4 :: ffff:0.0.0.0/96 14

```

9. 撤销第 8 步的动作

```

56 #
57 # scopev4 <mask> <value>
58 # Add another rule to the RFC 6724 scope table for IPv4 ad
59 # By default the scope IDs described in section 3.2 in RFC
60 # used. Changing these defaults should hardly ever be nec
61 # The defaults are equivalent to:
62 #
63 #scopev4 :: ffff:169.254.0.0/112 2
64 #scopev4 :: ffff:127.0.0.0/104 2
65 #scopev4 :: ffff:0.0.0.0/96 14
1 行发生改变 ; before #1 42 seconds ago

```

10. 存盘但不退出;

```

61 # The defaults are equivalent to:
62 #
63 #scopev4 :: ffff:169.254.0.0/112 2
64 #scopev4 :: ffff:127.0.0.0/104 2
65 #scopev4 :: ffff:0.0.0.0/96 14
:w

```

11. 将光标移到首行;

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```

1 # Configuration for getaddrinfo(3).
2 #
3 # So far only configuration for the destination address
4 # RFC 3484 governs the sorting. But the RFC also says
5 # administrators should be able to overwrite the default
6 # achieved here.
7 #
8 # All lines have an initial identifier specifying the c
9 # up to two values. Information specified in this file
10 # default information. Complete absence of data of one
11 # appropriate default information to be used. The supp
12 #
13 #

```


12. 插入模式下输入"Hello, this is vim world!"

```
62 #
63 #scopev4 :: ffff:169.254.0.0/112 2
64 #scopev4 :: ffff:127.0.0.0/104 2
65 #scopev4 :: ffff:0.0.0.0/96 14
66 #Hello, this is vim world!
67 #
-- 插入 --
```

13. 删除字符串"this";

位

```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助
43 # Add another rule to the RFC 3484
44 # and 10.3 in RFC 3484. The default
45 #
46 #precedence :: 1/128 50
47 #precedence :: /0 40
48 #precedence 2002:: /16 30
49 #precedence :: /96 20
50 #precedence :: ffff:0:0/96 10
51 #
52 # For sites which prefer IPv4 con
53 #
54 #precedence :: ffff:0:0/96 100
55
56 #
57 # scopev4 <mask> <value>
58 # Add another rule to the RFC 6724
59 # By default the scope IDs describ
60 # used. Changing these defaults :
61 # The defaults are equivalent to:
62 #
63 #scopev4 :: ffff:169.254.0.0/112 2
64 #scopev4 :: ffff:127.0.0.0/104 2
65 #scopev4 :: ffff:0.0.0.0/96 14
66 #Hello, is vim world!
67 #
-- 插入 --
```

14. 强制退出 vim, 不存盘

```
49 #precedence :: /96      20
50 #precedence :: ffff:0:0/96 10
51 #
52 #   For sites which prefer IPv4 connections change the last line
53 #
54 #precedence :: ffff:0:0/96 100
55 #
56 #
57 # scopev4 <mask> <value>
58 #   Add another rule to the RFC 6724 scope table for IPv4 address
59 #   By default the scope IDs described in section 3.2 in RFC 6724
60 #   used. Changing these defaults should hardly ever be necessary
61 #   The defaults are equivalent to:
62 #
63 #scopev4 :: ffff:169.254.0.0/112 2
64 #scopev4 :: ffff:127.0.0.0/104 2
65 #scopev4 :: ffff:0.0.0.0/96 14
66 #Hello, is vim world!
67 #
:q!
```

四、实验过程分析与讨论

Vim 的使用操作很多很杂，但整体上来讲难度并不大，只要跟着笔记和资料都是能轻松完成的。只要能熟悉这些操作，就能多快好省的完成编辑工作。

五、指导教师意见

指导教师签字：卢洋

2023 年 3 月 23 日

实验报告

实验名称	实验四 用户和用户组管理		
实验教室	丹青 922	实验日期	2023 年 3 月 30 日
学 号	2021213095	姓 名	蔡浩楠
专业班级	2021 级计算机科学与技术 3 班		
指导教师	卢洋		

东北林业大学

计算机科学与技术专业

一、 实验目的

1. 掌握用户管理命令，包括命令 `useradd`、`usermod`、`userdel`、`newusers`；
2. 掌握用户组管理命令，包括命令 `groupadd`、`groupdel`、`groupmod`、`gpasswd`；
3. 掌握用户和用户组维护命令，包括命令 `passwd`、`su`、`sudo`。

二、实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、实验内容及结果

1. 创建一个名为 foo，描述信息为 bar，登录 shell 为/bin/sh，家目录为 /home/foo 的用户，并设置登陆口令为 123456；

```
root@localhost kai] # useradd -d /home/foo -s /bin/sh -g users -m foo
root@localhost kai] # usermod -c bar foo
root@localhost kai] # tail -1 /etc/passwd
oo: x:1001:100: bar: /home/foo: /bin/sh
root@localhost kai] #
root@localhost kai] # usermod -p 123456 foo
root@localhost kai] # tail -1 /etc/passwd
oo: x:1001:100: bar: /home/foo: /bin/sh
root@localhost kai] #
```

2. 使用命令从 root 用户切换到用户 foo，修改 foo 的 UID 为 2000，其 shell 类型为/bin/csh

```
[root@localhost kai] # usermod -u 2000 -s /bin/csh foo
usermod: 无改变
[root@localhost kai] # su foo
[foo@localhost kai] $
```

3. 从用户 foo 切换到 root；

```
[foo@localhost kai] $ exit
exit
[root@localhost kai] #
```

4. 删除 foo 用户，并在删除该用户的同时一并删除其家目录；

```
[root@localhost kai] # userdel -r foo
[root@localhost kai] # tail -1 /etc/passwd
tail: 无法打开 "/etc/passwd" 读取数据: 没有那个文件或目录
[root@localhost kai] # cd foo
bash: cd: foo: 没有那个文件或目录
[root@localhost kai] # ll
总用量 0
drwxrwxr-x. 2 kai kai 32 5月 23 12:17 workspace1
drwxrwxr-x. 2 kai kai 22 5月 23 14:05 workspace2
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 公共
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 模板
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 视频
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 图片
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 文档
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 下载
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 音乐
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 桌面
```

5. 使用命令 newusers 批量创建用户，并使用命令 chpasswd 为这些批量创建的用户设置密码（密码也需要批量设置），查看 /etc/passwd 文件检查用户是否创建成功；


```
[root@localhost kai] # userdel -r foo
[root@localhost kai] # tail -1 /etc/passwd
tail: 无法打开"/etc/passwd" 读取数据: 没有那个文件或目录
[root@localhost kai] # cd foo
bash: cd: foo: 没有那个文件或目录
[root@localhost kai] # ll
总用量 0
drwxrwxr-x. 2 kai kai 32 5月 23 12:17 workspace1
drwxrwxr-x. 2 kai kai 22 5月 23 14:05 workspace2
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 公共
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 模板
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 视频
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 图片
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 文档
drwxr-xr-x. 2 kai kai 6 5月 22 20:08 下载
```

```
gdm:x:121:125:Gnome Display Manager:/var/lib/gdm3:/bin/false
zfm:x:1000:1000:zfm,,,:/home/zfm:/bin/bash
2020:x:1002:1002::/home/2020:
2021:x:1003:1003::/home/2021:
2022:x:1004:1004::/home/2022:
2023:x:1005:1005::/home/2023:
2024:x:1006:1006::/home/2024:
root@ubuntu:/home#
```

6. 创建用户组 group1，并在创建时设置其 GID 为 3000；

```
root@ubuntu:/home# groupadd -g 3000 group1
root@ubuntu:/home# tail -1 /etc/group
group1:x:3000:
root@ubuntu:/home#
```

7. 在用户组 group1 中添加两个之前批量创建的用户；

```
root@ubuntu:/# usermod -G group1 2023
root@ubuntu:/# usermod -G group1 2024
usermod: user '2024' does not exist
root@ubuntu:/# usermod -G group1 2022
root@ubuntu:/# tail -1 /etc/group
group1:x:3000:2023,2022
```

8. 切换到 group1 组中的任一用户，在该用户下使用sudo 命令 查看/etc/shadow 文件，检查上述操作是否可以执行；若不能执行，修改 sudoers文件使得该用户可以查看文件/etc/shadow 的内容

```
root@ubuntu:/# vi /etc/sudoers
root@ubuntu:/# ls -al /etc/sudoers
-r--r----- 1 root root 755 Jan 17 2018 /etc/sudoers
root@ubuntu:/# chmod u+w /etc/sudoers
root@ubuntu:/# ls -al /etc/sudoers
-rw-r----- 1 root root 755 Jan 17 2018 /etc/sudoers
root@ubuntu:/# vi /etc/sudoers
```

```
cat: /etc/shadow: No such file or directory
$ sudo cat /etc/shadow
root:$6$eGqWAqnU$HIW5w0OxgSvWv.s3/yTm.9NdRM6PdMh4HX5seCRrvQknXA7jp
wZv2UWnxFWvd2CmyJ6y5Zok1:19088:0:99999:7:::
daemon*:18885:0:99999:7:::
bin*:18885:0:99999:7:::
sys*:18885:0:99999:7:::
sync*:18885:0:99999:7:::
games*:18885:0:99999:7:::
man*:18885:0:99999:7:::
lp*:18885:0:99999:7:::
mail*:18885:0:99999:7:::
news*:18885:0:99999:7:::
uucp*:18885:0:99999:7:::
proxy*:18885:0:99999:7:::
www-data*:18885:0:99999:7:::
backup*:18885:0:99999:7:::
list*:18885:0:99999:7:::
irc*:18885:0:99999:7:::
gnats*:18885:0:99999:7:::
```

四、实验过程分析与讨论

用户与组的操作比较简单，重点在于掌握有关用户与组的文件信息，以及每个字段代表的是什么意思。第一次切换到 root 用户时显示密码错误，因为第一次是随意设置的，把系统重装了才解决。

五、指导教师意见

指导教师签字：卢洋

2023 年 3 月 30 日

实验报告

实验名称	实验五 Shell 程序的创建及条件判断语句		
实验教室	丹青 922	实验日期	2023 年 4 月 6 日
学 号	2021213095	姓 名	蔡浩楠
专业班级	2021 级计算机科学与技术 3 班		
指导教师	卢洋		

东北林业大学

计算机科学与技术专业

一、实验目的

1. 掌握 Shell 程序的创建过程及 Shell 程序的执行方法；
2. 掌握 Shell 变量的定义方法，及用户定义变量、参数位置等；
3. 掌握变量表达式， 包括字符串比较、数字比较、逻辑测试、文件测试；
4. 掌握条件判断 语句，如 if 语句、case 语句。

二、实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、实验内容及结果

1. 定义变量 foo 的值为 200，并将其显示在屏幕上（终端上执行）：

```
[kai@localhost ~]$ foo=200
[kai@localhost ~]$ echo $foo
200
[kai@localhost ~]$
```

2. 定义变量 bar 的值为 100，并使用 test 命令比较其值是否大于 150，并显示 test 命令的退出码（终端上执行）

```
[kai@localhost ~]$ var=100
[kai@localhost ~]$ test $var -gt 150 && echo yes || echo no
no
[kai@localhost ~]$ echo $?
0
[kai@localhost ~]$ test $var -gt 150
[kai@localhost ~]$ echo $?
1
[kai@localhost ~]$
```

3. 创建一个 Shell 程序，其功能为显示计算机主机名（hostname）和 系统时间（date）；

```
1 #!/bin/bash
2 echo "name of this machine"
3 hostname
4 echo "system date and time"
5 date
```

```
kai@localhost ~]$ vi pr-hn-date.sh
kai@localhost ~]$ sh pr-hn-date.sh
name of this machine
localhost.localdomain
system date and time
2023年 05月 23日 星期二 15:36:41 CST
kai@localhost ~]$
```

4. 创建一个 Shell 程序，要求可以处理一个输入参数，判断该输入参数是否为水仙花数；所谓水仙花数是指一个 3 位数，该数字每位数字的 3 次幂之和等于其本身，例如：根据上述定义 153 是水仙花数。编写程序时要求首先进行输入参数个数判断，判断是否有输入参数存在：如果没有则给出提示信息；否

则给出该数是否是水仙花数。要求 对 153、124 和 370 进行测试判断

```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
1 #!/bin/bash
2 read -p "please input the number: " num
3 numx=$num
4 num1=$((num%10))
5 num=$((num/10))
6 num2=$((num%10))
7 num=$((num/10))
8 num3=$((num%10))
9 num=$((num/10))
10 num=$((num1**3+num2**3+num3**3))
11 if test $num -eq $numx
12 then
13     echo 'this number is daffodils'
14 else
15     echo 'this number is not daffodis'
16 fi
~
~
~
[kai@localhost ~]$ vi judgenu.sh
[kai@localhost ~]$ sh judgenu.sh
please input the number:493
this number is not daffodis
[kai@localhost ~]$ sh judgenu.sh
please input the number:153
this number is daffodils
[kai@localhost ~]$
```

5. 创建一个 Shell 程序，输入3 个参数，计算 3 个输入变量的和并输出；

```
1 #!/bin/bash
2 read -p "please input the first number: " num1
3 read -p "please input the second number: " num2
4 read -p "please input the third number: " num3
5 echo "$((num1+num2+num3))"

[kai@localhost ~]$ vi sum.sh
[kai@localhost ~]$ sh sum.sh
please input the first number:10
please input the second number:15
please input the third number:36
61
[kai@localhost ~]$
```

6. 创建一个 Shell 程序，输入学生成绩，给出该成绩对应的等级：90 分以上为 A，80-90 为 B，70-80 为 C，60-70 为 D，小于 60 分为 E。要求使用实现。

```
1 #!/bin/bash
2 read -p "please input the student's grade: " gra
3 if test $gra -ge 90
4 then echo A
5 elif test $gra -ge 80
6 then echo B
7 elif test $gra -ge 70
8 then echo C
9 elif test $gra -ge 60
10 then echo D
11 else
12 echo E
13 fi
14 █
15
```

```
[kai@localhost ~]$ vi grade.sh
[kai@localhost ~]$ sh grade.sh
please input the student's grade: 96
A
[kai@localhost ~]$ sh grade.sh
please input the student's grade: 37
E
[kai@localhost ~]$
```

四、实验过程分析与讨论

运用 shell 脚本进行编程时，需要熟练地掌握 vim 的使用方法。
程序的完整性非常值得注意，使用 `$()` 的格式得到某个命令的返回值。

五、指导教师意见

指导教师签字：卢洋

2023 年 4 月 6 日

实验报告

实验名称	实验六 Shell 循环控制语句		
实验教室	丹青 922	实验日期	2023 年 4 月 25 日
学 号	2021213095	姓 名	蔡浩楠
专业班级	2021 级计算机科学与技术 3 班		
指导教师	卢洋		

东北林业大学
计算机科学与技术专业

一、实验目的

1. 熟练掌握 Shell 循环语句：for、while、until；
2. 熟练掌握 Shell 循环控制语句：break、continue。

二、实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、实验内容及结果

1. 编写一个 Shell 脚本，利用for 循环把当前目录下的所有*.c文件 复制到指定的目录中（如~/workspace）；可以事先在当前目录 下建立若干*.c文件用于测试
2. 编写 Shell 脚本，利用while 循环求前 10 个偶数之和，并输出 结果；

```
1 #!/bin/bash
2 i=2
3 sum=0
4 while (($i<=20))
5 do
6     sum=$((sum+i))
7     i=$((i+2))
8 done
9 echo $sum
```

```
[kai@localhost test]$ vi evensum.sh
[kai@localhost test]$ sh evensum.sh
110
[kai@localhost test]$
```

3. 编写 Shell 脚本，利用until 循环求 1 到 10 的平方和，并输出 结果；


```

1 #!/bin/bash
2 i=1
3 sq=0
4 until [ $i -ge 11 ]
5 do
6     temp=$((i*i))
7     sq=$((sq+temp))
8     i=$((i+1))
9 done
10 echo $sq
11 █

```

```

[kai@localhost test]$ vi quater.sh
[kai@localhost test]$ sh quater.sh
385
[kai@localhost test]$ █

```

4. 运行下列程序，并观察程序的运行结果。将程序中的——分 别替换为 break、break 2、continue、continue 2，并观察 四种情况下的实验结果

```

zfm@ubuntu:~/workspace$ sh replace.sh
a1234
b1234
c1234
d1234
zfm@ubuntu:~/workspace$ vi replace.sh
zfm@ubuntu:~/workspace$ sh replace.sh
a1234zfm@ubuntu:~/workspace$ vi replace.sh
zfm@ubuntu:~/workspace$ sh replace.sh
a1234678910
b1234678910
c1234678910
d1234678910
zfm@ubuntu:~/workspace$ vi replace.sh
zfm@ubuntu:~/workspace$ sh replace.sh
a1234b1234c1234d1234zfm@ubuntu:~/workspace$ █

```

四、实验过程分析与讨论

Until 循环条件中若为假继续循环，若为真则跳出循环，这一点是我们平常容易混淆的地方。另外在引用变量时千万不要忘记加 '\$' 符号。Break 是终止一层循环，continue 是跳出当前 循环；而 break 2 是终止两层循环，continue 2 是跳两个循环

五、指导教师意见

指导教师签字：卢洋

2023 年 4 月 25 日

实验报告

实验名称	实验七 Shell 函数		
实验教室	丹青 922	实验日期	2023 年 4 月 28 日
学 号	2021213095	姓 名	蔡浩楠
专业班级	2021 级计算机科学与技术 3 班		
指导教师	卢洋		

东北林业大学

计算机科学与技术专业

一、实验目的

1. 掌握 Shell 函数的定义方法；
2. 掌握 Shell 函数的参数传递、调用和返回值；
3. 掌握 Shell 函数的递归调用方法；
4. 理解 Shell 函数的嵌套。

二、实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、实验内容及结果

1. 编写 Shell 脚本，实现一个函数，对两个数的和进行求解，并 输出结果；

```
kai@localhost ~]$ cd workspace2
kai@localhost workspace2]$ vi funsum.sh
kai@localhost workspace2]$ sh funsum.sh
lease input the first number:6
lease input the second number:85
1
kai@localhost workspace2]$
```

```
1 #!/bin/bash
2
3 read -p "please input the first number:" a
4 read -p "please input the second number:" b
5 sum(){
6 result=$((a+b))
7 echo $result
8 }
9 sum
```

2. 编写 Shell 脚本，在脚本中定义一个递归函数，实现 n 的阶乘 的求解；

```
#!/bin/bash
echo "请输入"
read n
echo
func ()
{
    local i="$1"
    if [ "$i" -eq 0 ]; then
        result=1
    else
        let "m=i-1"
        func "$m"
        let "result=$i * $"
    fi
    return $result
}
func "$n"
echo "$n的阶乘为 : $?"
```

```
kai@localhost ~]$ vi factorrial.sh
kai@localhost ~]$ sh factorrial.sh
请输入
```

```
的阶乘为 : 6
kai@localhost ~]$
```

3. 一个 Shell 脚本的内容如下所示:

```
#!/bin/bash

function first() {
    function second() {
        function third() {
            echo "-3- here is in the third func."
        }
        echo "-2- here is in the second func."
        third
    }
    echo "-1- here is in the first func."
    second
}
echo "starting..."
first
```

```
[kai@localhost ~]$ vi test.sh
[kai@localhost ~]$ sh test.sh
starting...
-1- here is in the first func.
-2- here is in the second func.
-3- here is in the third func.
[kai@localhost ~]$
```

四、实验过程分析与讨论

Shell 函数需要通过函数名再在后面加上参数,在函数内部用\$1,\$2……的方式调用参数。

函数嵌套即在一个函数中可以定义另一个函数,如此 往复,而且不论在那一层都可以调用定义过的函数。

五、指导教师意见

指导教师签字：卢洋

2023 年 4 月 28 日

实验报告

实验名称	实验八 sed 和 awk		
实验教室	丹青 922	实验日期	2023 年 5 月 3 日
学 号	2021213095	姓 名	蔡浩楠
专业班级	2021 级计算机科学与技术 3 班		
指导教师	卢洋		

东北林业大学
计算机科学与技术专业

一、实验目的

1. 掌握 sed 基本编辑命令的使用方法；
2. 掌握 sed 与 Shell 变量的交互方法；
3. 掌握 awk 命令的使用方法；
4. 掌握 awk 与 Shell 变量的交互方法。

二、实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、实验内容及结果

1. 文件 quote.txt 的内容如下所示：

The honeysuckle band played all night long for only \$90. It was an evening of splendid music and company. Too bad the disco floor fell through at 23:10. The local nurse Miss P. Neave was in attendance.

- (1) 删除\$符号；

```
[kai@localhost ~]$ vi quote.txt
[kai@localhost ~]$ cat quote.txt
```

```
neysuckle band played all night long for only 90. It
was an evening of splendid music and company. Too bad the
disco floor fell through at 23:10. The local nurse Miss
P. Neave was in attendance.
```

```
[kai@localhost ~]$ █
```

- (2) 显示包含 music 文字的 行内容及行号；

```
[kai@localhost ~]$ sed -n '/music/{=p}' quote.txt
3
was an evening of splendid music and company. Too bad the
[kai@localhost ~]$
```

- (3) 在第 4 行后面追加内容：“hello world!”；

```
[kai@localhost ~]$ sed '$ahello world!' quote.txt

neysuckle band played all night long for only $90. It
was an evening of splendid music and company. Too bad the
disco floor fell through at 23:10. The local nurse Miss
P. Neave was in attendance.
hello world!
[kai@localhost ~]$
```

- (4) 将文本“The”替换为“Quod”；

```
[kai@localhost ~]$ sed 's/The/Quod/' quote.txt

neysuckle band played all night long for only $90. It
was an evening of splendid music and company. Too bad the
disco floor fell through at 23:10. Quod local nurse Miss
P. Neave was in attendance.
[kai@localhost ~]$
```

- (5) 将第 3 行内容修改为：“This is the third line.”；

```
[kai@localhost ~]$ sed '2d' quote.txt
```

```
was an evening of splendid music and company.Too bad the
disco floor fell through at 23:10.The local nurse Miss
P.Neave was in attendance.
```

```
[kai@localhost ~]$
```

(6) 删除第 2 行内容;

```
[kai@localhost ~]$ sed '2d' quote.txt
```

```
was an evening of splendid music and company.Too bad the
disco floor fell through at 23:10.The local nurse Miss
P.Neave was in attendance.
```

```
[kai@localhost ~]$
```

(7) 设置 Shell 变量 var 的值为 evening, 用 sed 命令查找匹配 var 变量值的行

```
[kai@localhost ~]$ var=evening'
```

```
[kai@localhost ~]$ sed -n '/'$var'/p' quote.txt
```

```
was an evening of splendid music and company.Too bad the
```

```
[kai@localhost ~]$
```

2. 文件 numbers.txt 的内容如下所示: one : two : three four : five : six

注: 每个冒号前后都有空格。试使用 awk 命令实现如下功能: 分别以空格和冒号做分隔符, 显示第 2 列的内容, 观察两者的区别;

```
[kai@localhost ~]$ awk -F " " '{print $2}' number.txt
```

```
:
```

```
:
```

```
[kai@localhost ~]$ awk -F ":" '{print $2}' number.txt
```

```
two
```

```
five
```

```
[kai@localhost ~]$
```

3. 已知文件 foo.txt 中存储的都是数字, 且每行都包含 3 个数字, 数字之前以空格作为分隔符。试找出 foo.txt 中的所有偶数进行打印, 并输出偶数的个数。要求: 判断每行的 3 个数字是否为偶数时用循环结果, 即要求程序里包含循环和分支结构。

```
[kai@localhost ~]$ vi newjudge.sh
```

```
[kai@localhost ~]$ bash newjudge.sh
```

```
even
```

```
2
```

```
46
```

```
30
```

```
number:
```

```
3
```

```
[kai@localhost ~]$
```



```

1  #!/bin/bash
2
3  cont=0
4  echo even
5  for j in `awk '{print $0}' foo.txt`
6  do
7      if [ $((j%2)) -eq 0 ]
8      then echo $j
9          cont=$((cont+1))
10     fi
11 done
12 echo number:
13 echo $cont
14
~
~
~

```

4. 脚本的内容如下所示： `#!/bin/bash read -p "enter search pattern: " pattern awk "/$pattern/" '{ nmatches++; print } END { print nmatches, "found." }' info.txt` 试运行该脚本，并理解该脚本实现的功

```

[kai@localhost ~]$ vi info.sh
[kai@localhost ~]$ sh info.sh
enter search pattern: 3
found.
[kai@localhost ~]$ █

```

```

[kai@localhost ~]$ vi info.txt
[kai@localhost ~]$ vi info.sh
[kai@localhost ~]$ sh info.sh

```

四、实验过程分析与讨论

第四问脚本实现功能是匹配当前字符在文件中出现了多少行。

Sed 和 awk 功能强大。

五、指导教师意见

指导教师签字：卢洋

2023 年 5 月 3 日