实验题目：Shell 编程

姓名：王波　　学号：19122557　　实验日期：2021 年 10 月 28 日

Shell 编程

一、实验环境

Vmware centOS 7

二、实验目的

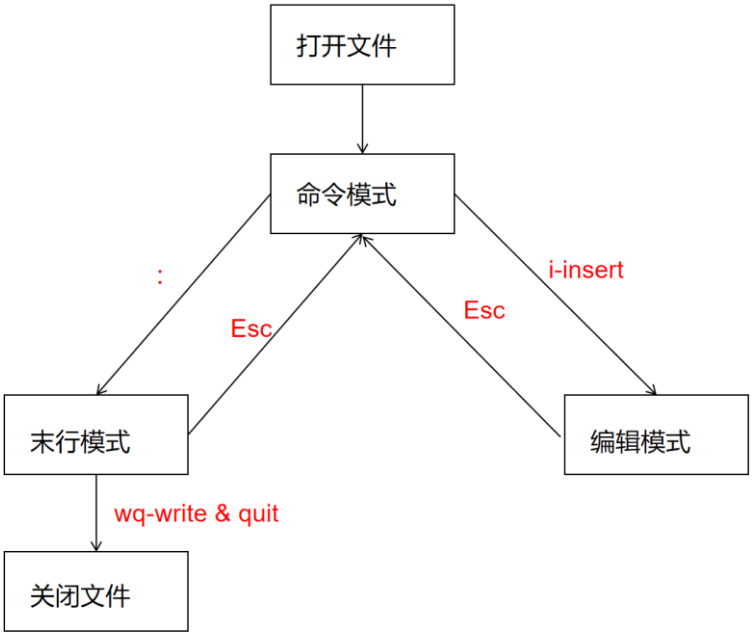
（1）掌握 vi 的三种工作方式，熟悉 vi 编辑程序的使用；
（2）学习 Shell 程序设计方法，掌握编程要领。

三、实验内容

（1）学习使用 vi 编辑程序
（2）编写 Shell 程序；
（3）将程序文件设置为可执行文件（用 chmod 命令）；
（4）在命令行方式中运行 Shell 程序。

四、实验步骤

1. 按本《实验指导》第三部分的内容，熟悉 vi 的三种工作方式，熟悉使用各种编辑功能。思考：试一试 vi 的三种工作方式各用在何时？用什么命令进入插入方式？怎样退出插入方式？文件怎样存盘？注意存盘后的提示信息。



①命令模式：

（1）在该模式中，可以输入命令来执行许多种功能；
（2）打开文件首先进入命令模式，它是使用 vim 编辑器的入口
②末行模式：
（1）将文件保存或退出 vi，也可以设置编辑环境，如寻找字符串、列出行号等；
（2）末行模式是 vim 编辑器对的出口，要退出 vim，必须要在末行模式下。
③编辑模式：可以对文本进行编辑操作。

2. 创建和执行 Shell 程序

用前面介绍的 Vi 或其他文本编辑器编写 Shell 程序，并将文件以文本文件方式保存在相应的目录中。 用 chmod 将文件的权限设置为可执行模式，如若文件名为 shdemo.h,则命令如下：

$ chmod 755 shdemo.h

(文件主可读、写、执行，同组人和其他人可读和执行)

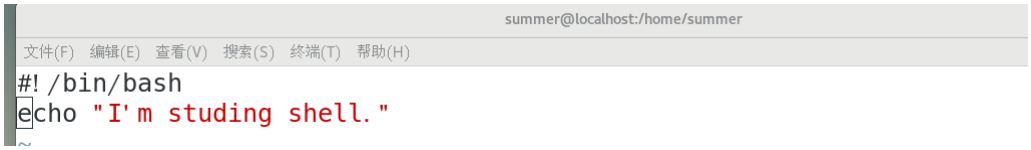
在提示符后执行 Shell 程序：

$ shdemo.h （直接键入程序文件名执行）

或 $ sh shdemo.h （执行 Shell 程序）

或 $ .shdemo.h （没有设置权限时可用点号引导）

【代码】



【运行结果】

```
[root@localhost summer]# vim prog2.h
[root@localhost summer]# chmod 755 prog2.h
[root@localhost summer]# sh prog2.h
I'm studing shell.
```

3. 用 vi 编写《实验指导》“第四部分 Shell 程序设计”中的例 1，练习内部变量和位置参数的用法。

用 chmod 将文件的权限设置为可执行模式，并在提示符后键入命令行：

$./prog1. 或 $sh prog1.　　#有一个参数

屏幕显示: Name not provided

在提示符后键入命令行：

$./prog1.h Theodore

屏幕显示: Your name is Theodore　#引用$1 参数的效果

【代码】

```
#Name display program
if [ $# == 0 ]
then
  echo "Name not provided."
else
  echo "Your name is " $1
fi
```

【运行结果】

```
[root@localhost summer]# ./prog3.h
Name not provided.
[root@localhost summer]# ./prog3.h Sherry
Your name is  Sherry
```

4. 进一步修改上一题中的程序，要求显示参数个数、程序名字，并逐个显示参数。

【代码】

```
#name display program
if [ $# ]
then
echo "number of parameters:" $#
echo "Your file name is "$0
echo $*
else echo "Name is not provided"
fi
```

【运行结果】

```
[root@localhost summer]# ./prog4.h summer spring winter autumn
number of parameters: 4
Your file name is ./prog4.h
summer spring winter autumn
```

5. 修改例 1 程序，用 read 命令接受键盘输入。若没有输入显示第一种提示，否则第二种提示。

【代码】

```
#!/bin/bash
#name display program
read name
if [ $name ]
then
echo "Your name is " $name
else
echo "Name is not provided"
fi
```

【运行结果】

有输入：

```
[root@localhost summer]# ./prog5.h
summer
Your name is  summer
```

没有输入：

```
[root@localhost summer]# ./prog5.h

Name is not provided
```

6. 用 vi 编写《实验指导》“第四部分 Shell 程序设计”中的例 2、例 3，练习字符串比较运算符、数据比较运算符和文件运算符的用法，观察运行结果。

【代码 1】

```
#!/bin/bash
string1="The first one"
string2="The second one"
if [ string1 == string2 ]
then
  echo "string1 equal to strng2"
else
  echo "string2 not equal to string2"
fi
if [ string1 ]
then
  echo "string1 is not empty"
else
  echo "string1 is empty"
fi
if [ -n string2 ]
then
  echo "string2 has a length greater than zero"
else
  echo "string2 has a length equal to zero"
fi
```

【运行结果 1】

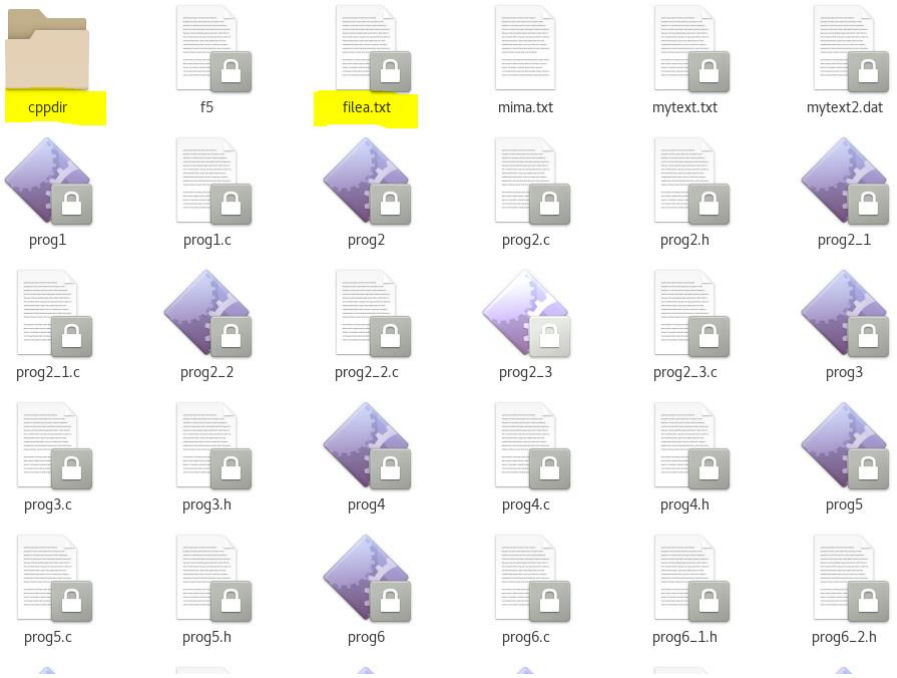
```
[root@localhost summer]# ./prog6_1.h
string2 not equal to string2
string1 is not empty
string2 has a length greater than zero
```

【代码 2】

```
#!/bin/bash
if [ -d cppdir ]
then
  echo "cppdir is a directory"
else
  echo "cppdir is not a directory"
fi

if [ -f filea.txt ]
then
  echo "filea is a regular file"
else
  echo "filea is not a regular file"
fi

if [ -r filea.txt ]
then
  echo "filea has reaad permission"
else
  echo "filea does not have read permission"
fi

if [ -w filea.txt ]
then
  echo "filea has write permission"
else
  echo "filea does not have write permission"
fi
```

【运行结果 2】

设置文件夹 cppdir 和文件 filea.txt：




```
[root@localhost summer]# ./prog6_2.h
cppdir is a directory
filea is a regular file
filea has reaad permission
filea has write permission
```

7. 修改例 2 程序，使在程序运行中能随机输入字符串，然后进行字符串比较。

【代码】

```
#!/bin/bash
read string1
read string2
if [  $string1 == $string2 ]
then
  echo "string1 equal to strng2"
else
  echo "string1 not equal to string2"
fi
```

【运行结果】

① 输入两个不相同的字符串：

```
[root@localhost summer]# ./prog7.h
summer
winter
string1 not equal to string2
```

② 输入两个相同的字符串：

```
[root@localhost summer]# ./prog7.h
summer
summer
string1 equal to strng2
```

8. 修改例 3 程序，使在程序运行中能随机输入文件名，然后进行文件属性判断。

【代码】

```
#!/bin/bash
read name
if [ -d $name ]
then
  echo $name "is a directory"
else
  echo $name  "is not a directory"
fi

if [ -f $name ]
then
  echo $name "is a regular file"
else
  echo $name "is not a regular file"
fi

if [ -r $name ]
then
  echo $name "has reaad permission"
else
  echo $name "does not have read permission"
fi

if [ -w filea.txt ]
then
  echo "filea has write permission"
else
  echo "filea does not have write permission"
fi
```

【运行结果】

输入文件 filea.txt：

```
[root@localhost summer]# ./prog8.h
filea.txt
filea.txt is not a directory
filea.txt is a regular file
filea.txt has reaad permission
filea has write permission
```

9. 用 vi 编写《实验指导》“第四部分 Shell 程序设计”中的例 4、例 5、例 6、例 7，掌握控制语句的用法，观察运行结果。

【代码 1】

```
#!/bin/bash
for filename in `ls`
do
  cp $filename backup/$filename
  if [ $? -ne 0 ]
  then
     echo "copy $filename failed"
  fi
done
```

【运行结果 1】

```
[root@localhost summer]# ./prog9_1.h
cp: 略过目录"backup"
copy backup failed
cp: 略过目录"cppdir"
copy cppdir failed
cp: 略过目录"test"
copy test failed
cp: 略过目录"公共"
copy 公共 failed
cp: 略过目录"模板"
copy 模板 failed
cp: 略过目录"视频"
copy 视频 failed
cp: 略过目录"图片"
copy 图片 failed
cp: 略过目录"文档"
copy 文档 failed
cp: 略过目录"下载"
copy 下载 failed
cp: 略过目录"音乐"
copy 音乐 failed
cp: 略过目录"桌面"
copy 桌面 failed
```

```
[root@localhost summer]# ls
backup        prog11.c     prog2_2     prog4.h     prog7_1.c  test2          图片
cppdir        prog12       prog2_2.c   prog5       prog7.c    test2.c        文档
f5            prog12.c     prog2_3     prog5.c     prog7.c~   test3          下载
filea.txt     prog1.c      prog2_3.c   prog5.h     prog7.h    test3.c        音乐
mima.txt      prog2        prog2.c     prog6       prog8      test.c         桌面
mytext2.dat   prog20       prog2.h     prog6_1.h   prog8.c    testshadow
mytext.txt    prog20_1     prog3       prog6_2.h   prog8.h    testshadow.c
prog1         prog20_1.c   prog3.c     prog6.c     prog9      text.txt
prog10        prog20.c     prog3.h     prog7       prog9_1.h  公共
prog10.c      prog2_1      prog4       prog7_      prog9.c    模板
prog11        prog2_1.c    prog4.c     prog7_1     test       视频
[root@localhost summer]# cd backup
[root@localhost backup]# ls
f5            prog11.c     prog2_1     prog3.c   prog6_1.h  prog7.h    test3
filea.txt     prog12       prog2_1.c   prog3.h   prog6_2.h  prog8      test3.c
mima.txt      prog12.c     prog2_2     prog4     prog6.c    prog8.c    test.c
mytext2.dat   prog1.c      prog2_2.c   prog4.c   prog7      prog8.h    testshadow
mytext.txt    prog2        prog2_3     prog4.h   prog7_     prog9      testshadow.c
prog1         prog20       prog2_3.c   prog5     prog7_1    prog9_1.h  text.txt
prog10        prog20_1     prog2.c     prog5.c   prog7_1.c  prog9.c
prog10.c      prog20_1.c   prog2.h     prog5.h   prog7.c    test2
prog11        prog20.c     prog3       prog6     prog7.c~   test2.c
```

【代码 2】

```
#!/bin/bash
loopcount=0
result=0
while [ $loopcount -lt 10 ]
do
  let loopcount+=1
  let result=$result+$loopcount*2
done
echo "result is $result"
~
```

【运行结果 2】

```
[root@localhost summer]# ./prog9_2.h
result is 110
```

【代码 3】

```
#!/bin/bash
loopcount=0
result=0
until [ $loopcount -ge 10 ]
do
  ((loopcount++))
  let result=result+$loopcount*2
done
echo "result is $result"
```

【运行结果 3】

```
[root@localhost summer]# ./prog9_3.h
result is 110
```

【代码 4】

```
#!/bin/bash
select item in continue finish
do
  if [ $item = "finish" ]
  then
    break
  fi
done
```

【运行结果 4】

```
[root@localhost summer]# ./prog9_4.h
1) continue
2) finish
#? 1
#? 2
```

10. 用 vi 编写《实验指导》“第四部分 Shell 程序设计”中的例 8 及例 9 掌握条件语句的用法，函数的用法，观察运行结果。

【代码 1】

```
#!/bin/bash
case $1 in
  01|1) echo “Month is January”;;
  02|2) echo “Month is February”;;
  03|3) echo “Month is March”;;
  04|4) echo “Month is April”;;
  05|5) echo “Month is May”;;
  06|6) echo “Month is June”;;
  07|7) echo “Month is July”;;
  08|8) echo “Month is August”;;
  09|9) echo “Month is September”;;
  010|10) echo “Month is October”;;
  011|11) echo “Month is November”;;
  012|12) echo “Month is December”;;
  *) echo “Invalid parameter”;;
esac
```

【运行结果 1】

```
[root@localhost summer]# ./prog10_1.h 1
“Month is January”
```

【代码 2】

```
#!/bin/bash
displaymonth(){
case $1 in
  01|1) echo “Month is January”;;
  02|2) echo “Month is February”;;
  03|3) echo “Month is March”;;
  04|4) echo “Month is April”;;
  05|5) echo “Month is May”;;
  06|6) echo “Month is June”;;
  07|7) echo “Month is July”;;
  08|8) echo “Month is August”;;
  09|9) echo “Month is September”;;
  010|10) echo “Month is October”;;
  011|11) echo “Month is November”;;
  012|12) echo “Month is December”;;
  *) echo “Invalid parameter”;;
esac
}
displaymonth 8
displaymonth 12
```

【运行结果 2】

```
[root@localhost summer]# ./prog10_2.h
“Month is August”
“Month is December”
```

11. 编程，在屏幕上显示用户主目录名（HOME）、命令搜索路径（PATH），并显示由位置参数指定的文件的类型和操作权限。

【代码】

```
#!/bin/bash
echo "home is $HOME"
echo "path is $PATH"
if [ $# -ne 0 ]
then
  ls -ld $1
fi
```

【运行结果】

```
[root@localhost summer]# ./prog11.h filea.txt
home is /root
path is /usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/summer/.local/bin:/hom
e/summer/bin
-rw-r--r--. 1 root root 0 10月  22 10:17 filea.txt
```

思考：到此为止你对 Shell 有所认识了吧？怎么样？自己再编两个程序：

① 做个批处理程序，体会一下批处理概念。

【代码】

```
#!/bin/bash
for FRUIT in apple banana pear;
 do
        echo "I like $FRUIT"
    done
```

【运行结果】

```
[root@localhost summer]# ./prog12.h
I like apple
I like banana
I like pear
```

② 做个菜单，显示系统环境参数。将此程序设置为人人可用。

【代码】

```
#!/bin/bash
select cho in env finish
do
  if [ $cho = "finish" ]
  then
    break
  else
    $cho
  fi
done
```

【运行结果】

```
[root@localhost summer]# ./prog13.h
1) env
2) finish
#? 1
XDG_VTNR=1
SSH_AGENT_PID=2009
XDG_SESSION_ID=1
HOSTNAME=localhost.localdomain
IMSETTINGS_INTEGRATE_DESKTOP=yes
VTE_VERSION=5204
TERM=xterm-256color
SHELL=/bin/bash
XDG_MENU_PREFIX=gnome-
HISTSIZE=1000
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/cffe4613_898e_4c7b_8c95_a4750e907d84
IMSETTINGS_MODULE=none
USER=summer
LS_COLORS=rs=0:di=38;5;27:ln=38;5;51:mh=44;38;5;15:pi=40;38;5;11:so=38;5;13:do=38;5;5:bd=48;5;232;38;5;11:cd=48
;5;232;38;5;3:or=48;5;232;38;5;9:mi=05;48;5;232;38;5;15:su=48;5;196;38;5;15:sg=48;5;11;38;5;16:ca=48;5;196;38;5
;226:tw=48;5;10;38;5;16:ow=48;5;10;38;5;21:st=48;5;21;38;5;15:ex=38;5;34:*.tar=38;5;9:*.tgz=38;5;9:*.arc=38;5;9
:*.arj=38;5;9:*.taz=38;5;9:*.lha=38;5;9:*.lz4=38;5;9:*.lzh=38;5;9:*.lzma=38;5;9:*.tlz=38;5;9:*.txz=38;5;9:*.tzo
=38;5;9:*.t7z=38;5;9:*.zip=38;5;9:*.z=38;5;9:*.Z=38;5;9:*.dz=38;5;9:*.gz=38;5;9:*.lrz=38;5;9:*.lz=38;5;9:*.lzo=
38;5;9:*.xz=38;5;9:*.bz2=38;5;9:*.bz=38;5;9:*.tbz=38;5;9:*.tbz2=38;5;9:*.tz=38;5;9:*.deb=38;5;9:*.rpm=38;5;9:*.
jar=38;5;9:*.war=38;5;9:*.ear=38;5;9:*.sar=38;5;9:*.rar=38;5;9:*.alz=38;5;9:*.ace=38;5;9:*.zoo=38;5;9:*.cpio=38
;5;9:*.7z=38;5;9:*.rz=38;5;9:*.cab=38;5;9:*.jpg=38;5;13:*.jpeg=38;5;13:*.gif=38;5;13:*.bmp=38;5;13:*.pbm=38;5;1
3:*.pgm=38;5;13:*.ppm=38;5;13:*.tga=38;5;13:*.xbm=38;5;13:*.xpm=38;5;13:*.tif=38;5;13:*.tiff=38;5;13:*.png=38;5
;13:*.svg=38;5;13:*.svgz=38;5;13:*.mng=38;5;13:*.pcx=38;5;13:*.mov=38;5;13:*.mpg=38;5;13:*.mpeg=38;5;13:*.m2v=3
8;5;13:*.mkv=38;5;13:*.webm=38;5;13:*.ogm=38;5;13:*.mp4=38;5;13:*.m4v=38;5;13:*.mp4v=38;5;13:*.vob=38;5;13:*.qt
=38;5;13:*.nuv=38;5;13:*.wmv=38;5;13:*.asf=38;5;13:*.rm=38;5;13:*.rmvb=38;5;13:*.flc=38;5;13:*.avi=38;5;13:*.fl
i=38;5;13:*.flv=38;5;13:*.gl=38;5;13:*.dl=38;5;13:*.xcf=38;5;13:*.xwd=38;5;13:*.yuv=38;5;13:*.cgm=38;5;13:*.emf
=38;5;13:*.axv=38;5;13:*.anx=38;5;13:*.ogv=38;5;13:*.ogx=38;5;13:*.aac=38;5;45:*.au=38;5;45:*.flac=38;5;45:*.mi
d=38;5;45:*.midi=38;5;45:*.mka=38;5;45:*.mp3=38;5;45:*.mpc=38;5;45:*.ogg=38;5;45:*.ra=38;5;45:*.wav=38;5;45:*.a
xa=38;5;45:*.oga=38;5;45:*.spx=38;5;45:*.xspf=38;5;45:
GNOME_TERMINAL_SERVICE=:1.104
```

```
d=38;5;45:*.midi=38;5;45:*.mka=38;5;45:*.mp3=38;5;45:*.mpc=38;5;45:*.ogg=38;5;45:*.ra=38;5;45:*.wav=38;5;45:*.a
xa=38;5;45:*.oga=38;5;45:*.spx=38;5;45:*.xspf=38;5;45:
GNOME_TERMINAL_SERVICE=:1.104
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
USERNAME=summer
SESSION_MANAGER=local/unix:@/tmp/.ICE-unix/1881,unix/unix:/tmp/.ICE-unix/1881
GNOME_SHELL_SESSION_MODE=classic
PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/summer/.local/bin:/home/summer/bin
MAIL=/var/spool/mail/summer
DESKTOP_SESSION=gnome-classic
QT_IM_MODULE=ibus
XDG_SESSION_TYPE=x11
PWD=/home/summer
XMODIFIERS=@im=ibus
LANG=zh_CN.UTF-8
GDM_LANG=zh_CN.UTF-8
GDMSESSION=gnome-classic
HISTCONTROL=ignoredups
XDG_SEAT=seat0
HOME=/root
SHLVL=4
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
XDG_SESSION_DESKTOP=gnome-classic
LOGNAME=summer
XDG_DATA_DIRS=/home/summer/.local/share/flatpak/exports/share/:/var/lib/flatpak/exports/share/:/usr/local/share
/:/usr/share/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-OUM7mISLXA,guid=6389e90513ffefd63ef2331b6172aad7
LESSOPEN=||/usr/bin/lesspipe.sh %s
WINDOWPATH=1
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=GNOME-Classic:GNOME
COLORTERM=truecolor
XAUTHORITY=/root/.xauth5uSTTD
_=/usr/bin/env
#? 2
[root@localhost summer]#
```

五. 讨论

1. Linux 的 Shell 有什么特点?

（1）shell 概述

Shell 是用 C 语言编写的程序，它是用户使用 Linux 的桥梁。Shell 既是一种命令语言，又是一种程序设计语言。简单来说 Shell 是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问

操作系统内核的服务。也可以说，linux 中的 shell 就是 linux 内核的一个外层保护工具，负责完成用户与内核之间的交互。

（2）shell 的优缺点

优点方面：

1）shell 的语法和结构比较简单，易于掌握，学习和使用比较方便；

2）shell 是解释型语言，运行之前不需要编译；

3）程序开发的效率非常高，依赖于功能强大的命令可以迅速地完成开发任务。

4）把已有命令进行适当组合构成新的命令；

5）提供了文件名扩展字符（通配符，如* 、 ?、 [ ]），使得用单一的字符串可以匹配多个文件名，省去键入一长串文件名的麻烦；

6）可以直接使用 Shell 的内置命令,而不需创建新的进程，如 Shell 中提供的 cd、echo、exit、pwd、kill 等命令。为防止因某些 Shell 不支持这类命令而出现麻烦，许多命令都提供了对应的二进制代码，从而也可以在新进程中运行；

缺点方面：

1）弱类型语言，对变量定义的查检不是很严格；

2）效率上存在劣势，开发产能优于运行的性能；

3）I/O 性能不高；

4）一些细节上的不足：向 awk 传入 shell 定义的变量时，表达式比较复杂，且对 shell 的变量是只能读取而不能修改。

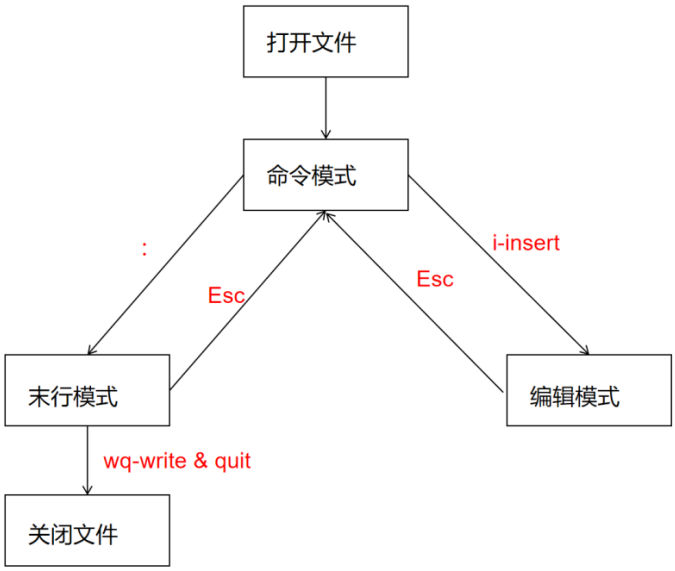
2. 怎样进行 Shell 编程？如何运行？有什么条件？

shell 程序就是一个包含若干行 shell 或者 linux 命令的文件，像编写高级语言的程序一样，编写一个 shell 程序需要一个文本编辑器，如 vi 和 vim 等。

在文本编辑环境下，依据 shell 的语法规则，输入一些 shell/linux 命令行，形成一个完整的程序文件。

用 chmod 将文件的权限设置为可执行模式，如若文件名为 shdemo.h,则命令如下：$chmod 755 shdemo.h（文件主可读、写、执行），同组人和其他人可读和执行）在提示符后执行 Shell 程序：$shdemo.h（直接键入程序文件名执行）或$ sh shdemo.h（执行 Shell 程序）或$.shdemo.h（没有设置权限时可用点号引导）。

3. vi 编辑程序有几种工作方式？查找有关的详细资料，熟练掌握屏幕编辑方式、转移命令方式以及末行命令的操作。学习搜索、替换字符、字和行，行的复制、移动，以及在 vi 中执行 Shell 命令的方式。



①命令模式：

（1）在该模式中，可以输入命令来执行许多种功能；

（2）打开文件首先进入命令模式，它是使用 vim 编辑器的入口

②末行模式：

（1）将文件保存或退出 vi，也可以设置编辑环境，如寻找字符串、列出行号等；

（2）末行模式是 vim 编辑器对的出口，要退出 vim，必须要在末行模式下。

③编辑模式：可以对文本进行编辑操作。

4. 编写一个具有以下功能的 Shell 程序。

【代码】

```
#!/bin/bash
ls -o>filedir.txt
mkdir testdir2
cp $(find *.c) testdir2
cd testdir2
for filename in `ls`
do
  chmod 311 $filename
done
cd
cd /home/summer
ls -o testdir2>>filedir.txt
id >>filedir.txt
less filedir.txt
~
```

【运行结果】

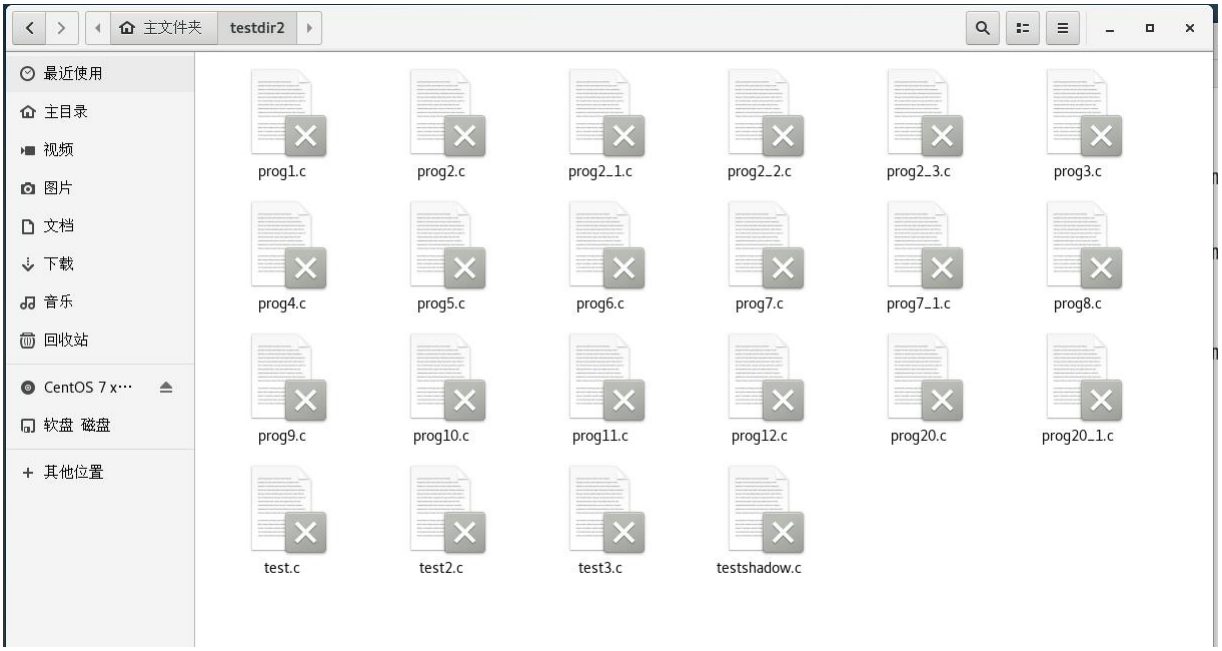
（1）把当前目录下的文件目录信息输出到文件 filedir.txt 中；

```
总用量 460
drwxr-xr-x. 2 summer  4096 10月 22 20:21 backup
drwxr-xr-x. 2 summer     6 10月 22 10:16 cppdir
-rw-r--r--. 1 root      13 10月 14 10:41 f5
-rw-r--r--. 1 root       0 10月 22 10:17 filea.txt
-rw-r--r--. 1 root       0 10月 22 21:47 filedir.txt
-rw-rw-r--. 1 summer   136 7月   7 15:53 mima.txt
-rwxr-x--x. 2 winter    24 9月   9 11:04 mytext2.dat
-rwxr-x--x. 2 winter    24 9月   9 11:04 mytext.txt
-rwxr-xr-x. 1 root    8880 9月  29 21:04 prog1
-rwxr-xr-x. 1 root   11032 10月  7 22:22 prog10
-rwxr-xr-x. 1 root     517 10月 22 11:25 prog10_1.h
-rwxr-xr-x. 1 root     566 10月 22 11:28 prog10_2.h
-rw-r--r--. 1 root    2200 10月  7 22:22 prog10.c
-rwxr-xr-x. 1 root   10720 10月  7 22:28 prog11
-rw-r--r--. 1 root    1732 10月  7 22:31 prog11.c
-rwxr-xr-x. 1 root      90 10月 22 21:41 prog11.h
-rwxr-xr-x. 1 root   10272 10月  8 20:04 prog12
-rw-r--r--. 1 root     896 10月  8 20:04 prog12.c
-rwxr-xr-x. 1 root      89 10月 22 21:39 prog12.h
-rwxr-xr-x. 1 root     108 10月 22 21:31 prog13.h
-rwxr-xr-x. 1 root     205 10月 22 21:47 prog14.h
-rw-r--r--. 1 root    1201 9月  29 21:15 prog1.c
-rwxr-xr-x. 1 root    8568 9月  29 20:22 prog2
-rwxr-xr-x. 1 root    9976 10月 20 23:37 prog20
-rwxr-xr-x. 1 root   10048 10月 19 11:39 prog20_1
-rw-r--r--. 1 root     907 10月 19 11:39 prog20_1.c
-rw-r--r--. 1 root     883 10月 21 00:03 prog20.c
-rwxr-xr-x. 1 root    8664 9月  30 12:46 prog2_1
-rw-r--r--. 1 root     551 9月  30 12:46 prog2_1.c
-rwxr-xr-x. 1 root    8768 10月  1 15:51 prog2_2
-rw-r--r--. 1 root     522 10月  1 15:51 prog2_2.c
-rwxr-xr-x. 1 root    8616 10月  2 20:07 prog2_3
-rw-r--r--. 1 root     633 10月  2 19:48 prog2_3.c
-rw-r--r--. 1 root     341 9月  29 20:41 prog2.c
-rwxr-xr-x. 1 root      38 10月 22 09:16 prog2.h
-rwxr-xr-x. 1 root    8760 10月  2 20:25 prog3
-rw-r--r--. 1 root    1261 10月  2 20:40 prog3.c
-rwxr-xr-x. 1 root     104 10月 22 09:22 prog3.h
-rwxr-xr-x. 1 root    8640 10月  4 19:42 prog4
-rw-r--r--. 1 root     537 10月  5 18:59 prog4.c
```

（2）在当前目录下建立一个子目录，目录名为 testdir2 ；

```
[root@localhost summer]# ls
backup        prog10      prog12.h    prog2_1     prog3.c   prog6_1.h  prog7.h    prog9.c     testshadow.c  桌面
cppdir        prog10_1.h  prog13.h    prog2_1.c   prog3.h   prog6_2.h  prog8      test        text.txt
f5            prog10_2.h  prog14.h    prog2_2     prog4     prog6.c    prog8.c    test2       公共
filea.txt     prog10.c    prog1.c     prog2_2.c   prog4.c   prog7      prog8.h    test2.c     模板
filedir.txt   prog11      prog2       prog2_3     prog4.h   prog7_     prog9      test3       视频
mima.txt      prog11.c    prog20      prog2_3.c   prog5     prog7_1    prog9_1.h  test3.c     图片
mytext2.dat   prog11.h    prog20_1    prog2.c     prog5.c   prog7_1.c  prog9_2.h  test.c      文档
mytext.txt    prog12      prog20_1.c  prog2.h     prog5.h   prog7.c    prog9_3.h  testdir2    下载
prog1         prog12.c    prog20.c    prog3       prog6     prog7.c~   prog9_4.h  testshadow  音乐
```

（3）把当前目录下的所有扩展名为 c 的文件以原文件名复制到子目录 testdir2 中；



（4）把子目录中的所有文件的存取权限改为不可读。（提示：用 for 循环控制语句实现，循环的控制列表用 'ls' 产生。）

```
[root@localhost testdir2]# ls -o
总用量 88
---wx---x.  1 root 2200 10月  22 21:47 prog10.c
---wx---x.  1 root 1732 10月  22 21:47 prog11.c
---wx---x.  1 root  896 10月  22 21:47 prog12.c
---wx---x.  1 root 1201 10月  22 21:47 prog1.c
---wx---x.  1 root  907 10月  22 21:47 prog20_1.c
---wx---x.  1 root  883 10月  22 21:47 prog20.c
---wx---x.  1 root  551 10月  22 21:47 prog2_1.c
---wx---x.  1 root  522 10月  22 21:47 prog2_2.c
---wx---x.  1 root  633 10月  22 21:47 prog2_3.c
---wx---x.  1 root  341 10月  22 21:47 prog2.c
---wx---x.  1 root 1261 10月  22 21:47 prog3.c
---wx---x.  1 root  537 10月  22 21:47 prog4.c
---wx---x.  1 root 1496 10月  22 21:47 prog5.c
---wx---x.  1 root  876 10月  22 21:47 prog6.c
---wx---x.  1 root  639 10月  22 21:47 prog7_1.c
---wx---x.  1 root  888 10月  22 21:47 prog7.c
---wx---x.  1 root  744 10月  22 21:47 prog8.c
---wx---x.  1 root 2995 10月  22 21:47 prog9.c
---wx---x.  1 root  790 10月  22 21:47 test2.c
---wx---x.  1 root  790 10月  22 21:47 test3.c
---wx---x.  1 root  424 10月  22 21:47 test.c
---wx---x.  1 root 1646 10月  22 21:47 testshadow.c
```

（5）在把子目录 testdir2 中所有文件的目录信息追加到文件 filedir.txt 中；

```
总用量 88
----wx----x. 1 root 2200 10月 22 21:55 prog10.c
----wx----x. 1 root 1732 10月 22 21:55 prog11.c
----wx----x. 1 root  896 10月 22 21:55 prog12.c
----wx----x. 1 root 1201 10月 22 21:55 prog1.c
----wx----x. 1 root  907 10月 22 21:55 prog20_1.c
----wx----x. 1 root  883 10月 22 21:55 prog20.c
----wx----x. 1 root  551 10月 22 21:55 prog2_1.c
----wx----x. 1 root  522 10月 22 21:55 prog2_2.c
----wx----x. 1 root  633 10月 22 21:55 prog2_3.c
----wx----x. 1 root  341 10月 22 21:55 prog2.c
----wx----x. 1 root 1261 10月 22 21:55 prog3.c
----wx----x. 1 root  537 10月 22 21:55 prog4.c
----wx----x. 1 root 1496 10月 22 21:55 prog5.c
----wx----x. 1 root  876 10月 22 21:55 prog6.c
----wx----x. 1 root  639 10月 22 21:55 prog7_1.c
----wx----x. 1 root  888 10月 22 21:55 prog7.c
----wx----x. 1 root  744 10月 22 21:55 prog8.c
----wx----x. 1 root 2995 10月 22 21:55 prog9.c
----wx----x. 1 root  790 10月 22 21:55 test2.c
----wx----x. 1 root  790 10月 22 21:55 test3.c
----wx----x. 1 root  424 10月 22 21:55 test.c
----wx----x. 1 root 1646 10月 22 21:55 testshadow.c
```

（6）把你的用户信息追加到文件 filedir.txt 中；

```
----wx----x. 1 root  790 10月 22 21:55 test2.c
----wx----x. 1 root  790 10月 22 21:55 test3.c
----wx----x. 1 root  424 10月 22 21:55 test.c
----wx----x. 1 root 1646 10月 22 21:55 testshadow.c
uid=0(root) gid=0(root) 组=0(root) 环境=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

（7）分屏显示文件 filedir.txt