

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине «Операционная система Linux»

Программирование в ОС семейства Linux

Студент

Мастылина А.А.

Группа АИ-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

Оглавление

Цель работы	3
Задание кафедры.....	4
Ход работы	5
Вывод	26

Цель работы

Изучение основных возможностей языка программирования Shell с целью автоматизации процесса администрирования системы за счет написания и использования командных файлов.

Задание кафедры

1. Используя команды ECHO, PRINTF вывести информационные сообщения на экран.
 2. Присвоить переменной A целочисленное значение. Просмотреть значение переменной A.
 3. Присвоить переменной B значение переменной A. Просмотреть значение переменной B.
 4. Присвоить переменной C значение “путь до своего каталога”. Перейти в этот каталог с использованием переменной.
 5. Присвоить переменной D значение “имя команды”, а именно, команды DATE. Выполнить эту команду, используя значение переменной.
 6. Присвоить переменной E значение “имя команды”, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной.
 7. Присвоить переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной.
- Написать скрипты, при запуске которых выполняются следующие действия:
8. Программа запрашивает значение переменной, а затем выводит значение этой переменной.
 9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной.
 10. Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) BC).,
 11. Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран.

12. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки.

13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается.

14. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно.

15. Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения.

16. Программой запрашивается год, определяется, високосный ли он.

Результат выдается на экран.

17. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.

18. В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc.

19. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение.

20. Если файл есть каталог и этот каталог можно читать, просматривается содержимое этого каталога. Если каталог отсутствует, он создается. Если файл не есть каталог, просматривается содержимое файла.

21. Анализируются атрибуты файла. Если первый файл существует и используется для чтения, а второй файл существует и используется для записи, то содержимое первого файла перенаправляется во второй файл. В случае несовпадений указанных атрибутов или отсутствия файлов на экран выдаются соответствующие сообщения (использовать а) имена файлов; б) позиционные параметры).

22. Если файл запуска программы найден, программа запускается (по выбору).

23. В качестве позиционного параметра задается файл, анализируется его размер. Если размер файла больше нуля, содержимое файла сортируется по первому столбцу по возрастанию, отсортированная информация помещается в другой файл, содержимое которого затем отображается на экране.

24. Командой TAR осуществляется сборка всех текстовых файлов текущего каталога в один архивный файл `my.tar`, после паузы просматривается содержимое файла `my.tar`, затем командой GZIP архивный файл `my.tar` сжимается.

25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных.

Ход работы

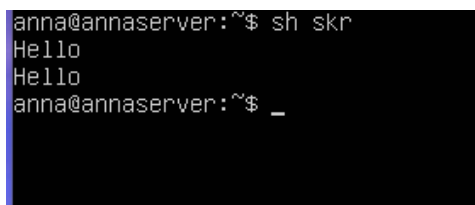
Сделаем файл `skr` исполняемым с помощью команды `chmod ugo+x skr`.

Используя команды `ECHO`, `PRINTF` выведем информационные сообщения на экран. Результат работы представлен на рисунках 1 и 2.



```
GNU nano 4.8
echo Hello
printf Hello
echo
```

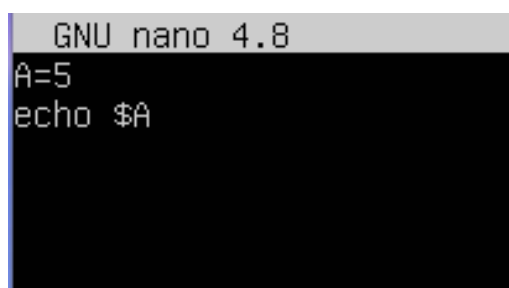
Рисунок 1 – Пример использования команд `echo` и `printf`



```
anna@annaserver:~$ sh skr
Hello
Hello
anna@annaserver:~$ _
```

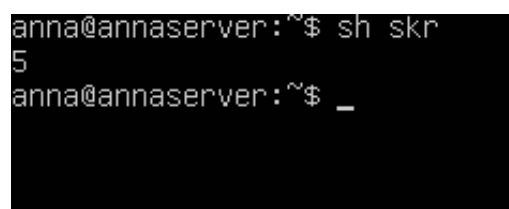
Рисунок 2 - Рисунок 2 – Результат работы после использования команд `echo` и `printf`

Присвоим переменной `A` целочисленное значение и посмотрим значение переменной `A` с помощью команды `echo`. Результат работы представлен на рисунках 3 и 4.



```
GNU nano 4.8
A=5
echo $A
```

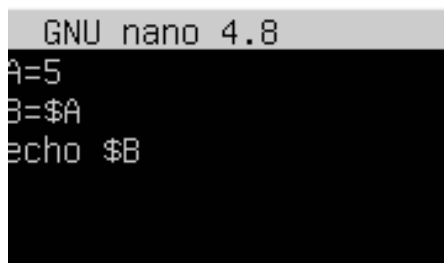
Рисунок 3 – Присвоение целочисленного значения переменной `A`



```
anna@annaserver:~$ sh skr
5
anna@annaserver:~$ _
```

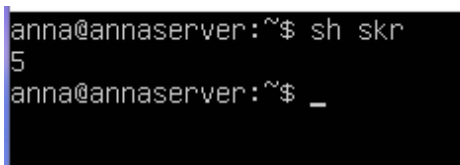
Рисунок 4 – Результат работы после присвоения значения переменной

Присвоим переменной В значение переменной А. Просмотрим значение переменной В. Результат работы представлен на рисунках 5 и 6.



```
GNU nano 4.8
A=5
B=$A
echo $B
```

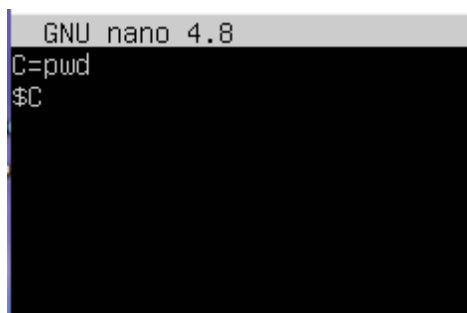
Рисунок 5 – Присвоение переменной В значение переменной А



```
anna@annaserver:~$ sh skr
5
anna@annaserver:~$ _
```

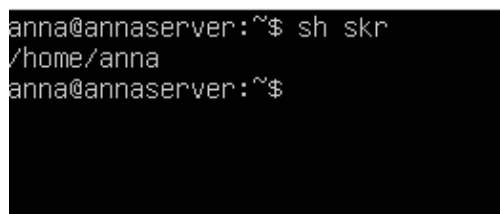
Рисунок 6 – Результат работы после присвоения переменной В значения переменной А

Присвоим переменной С значение “путь до своего каталога”. Перейдём в этот каталог с использованием переменной. Результат работы представлен на рисунках 7 и 8.



```
GNU nano 4.8
C=pwd
$C
```

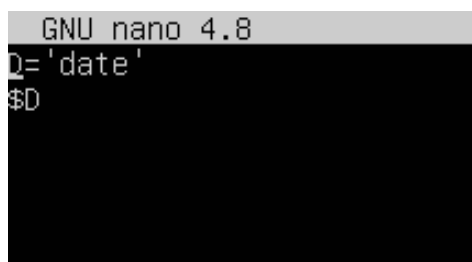
Рисунок 7 – Присвоение переменной С значение “путь до своего каталога”



```
anna@annaserver:~$ sh skr
/home/anna
anna@annaserver:~$
```

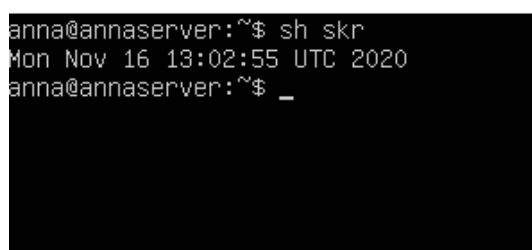
Рисунок 8 – Результат выполнения скрипта

Присвоим переменной D значение “имя команды”, а именно, команды DATE. Выполним эту команду, используя значение переменной. Результат работы представлен на рисунках 9 и 10.



```
GNU nano 4.8
D='date'
$D
```

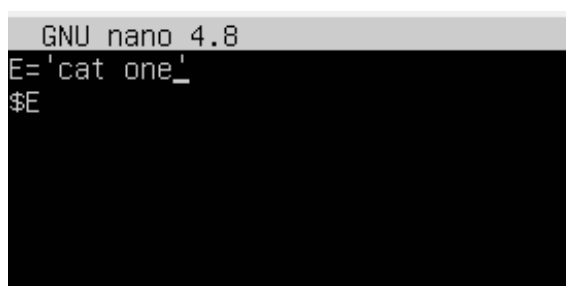
Рисунок 9 - Присвоение переменной D значение “имя команды”,
команды DATE



```
anna@annaserver:~$ sh skr
Mon Nov 16 13:02:55 UTC 2020
anna@annaserver:~$ _
```

Рисунок 10 – Результат выполнения скрипта

Присвоим переменной E значение “имя команды”, а именно, команды просмотра содержимого файла, посмотрим содержимое переменной. Выполним эту команду, используя значение переменной. Результат работы представлен на рисунках 11 и 12.



```
GNU nano 4.8
E='cat one_'
$E
```

Рисунок 11 - Присвоение переменной E значение “имя команды”, а
именно, команды просмотра содержимого файла

```
anna@annaserver:~$ sh skr
q
w
e
r
anna@annaserver:~$ _
```

Рисунок 12 – Результат выполнения скрипта

Присвоим переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполним эту команду, используя значение переменной. Результат работы представлен на рисунках 13 и 14.

```
GNU nano 4.8
F='sort one'
$F
```

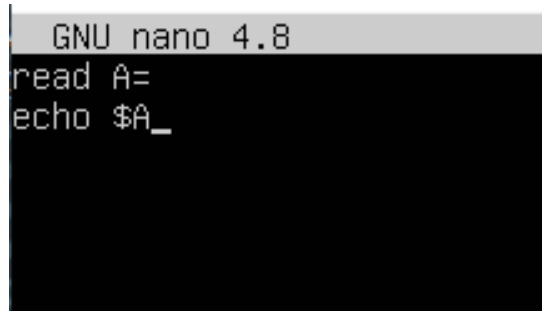
Рисунок 13 - Присвоение переменной F значение “имя команды”, а именно сортировки содержимого текстового файла

```
anna@annaserver:~$ sh skr
e
q
r
w
anna@annaserver:~$ cat one
q
w
e
r
anna@annaserver:~$
```

Рисунок 14 – Результат выполнения скрипта

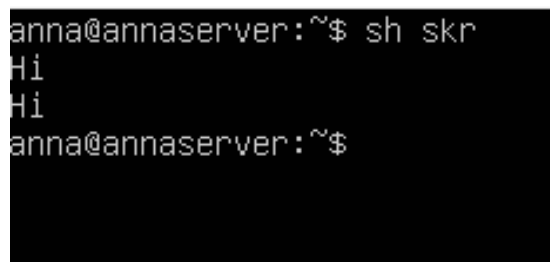
Следующим заданием нужно написать скрипты, при запуске которых выполняются следующие действия:

Программа запрашивает значение переменной с помощью команды `read`, а затем выводит значение этой переменной с помощью команды `echo`. Результат работы представлен на рисунках 15 и 16.



```
GNU nano 4.8
read A=
echo $A_
```


Рисунок 15 – Запрос значения переменной и вывод его на экран



```
anna@annaserver:~$ sh skr
Hi
Hi
anna@annaserver:~$
```

Рисунок 16 – Результат выполнения скрипта

Программа запрашивает имя пользователя с помощью команды `read`, затем здоровается с ним, используя значение введенной переменной. Результат работы представлен на рисунках 17 и 18.



```
GNU nano 4.8
read A=
echo Hello $A
```

Рисунок 17 – Запросим имя пользователя с помощью команды `read`,
затем поздороваемся с ним

```
anna@annaserver:~$ sh skr
Anna
Hello Anna
anna@annaserver:~$ _
```

Рисунок 18 – Результат выполнения скрипта

Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) BC). Результат работы представлен на рисунках 19, 20, 21 и 22.

```
GNU nano 4.8
echo A=
read A
echo B=
read B
sum=$(expr $A + $B)
razn=$(expr $A - $B)
umn=$(expr $A \* $B)
del=$(expr $A / $B)
echo sum=$sum razn=$razn umn=$umn del=$del
```

Рисунок 19 - Вычисления с использованием команды EXPR

```
anna@annaserver:~$ sh skr
A=
4
B=
2
sum=6 razn=2 umn=8 del=2
anna@annaserver:~$
```

Рисунок 20 – Результат выполнения скрипта

```
GNU nano 4.8
echo A=
read A
echo B=
read B
echo "$A + $B" |bc
echo "$A - $B" |bc
echo "$A * $B" |bc
echo "$A / $B" |bc_
```

Рисунок 21 - Вычисления с использованием команды BC

```
anna@annaserver:~$ sh skr
A=
5
B=
2
7
3
10
2
anna@annaserver:~$ _
```

Рисунок 22 – Результат выполнения скрипта

Вычислим объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран. Результат работы представлен на рисунках 23 и 24.

```
GNU nano 4.8
echo S
read S
echo h
read h
echo "$S * $h" |bc
```

Рисунок 23 – Вычисление объёма цилиндра

```
anna@annaserver:~$ sh skr
S
3
h
2
6
anna@annaserver:~$
```

Рисунок 24 – Результат выполнения скрипта

Используя позиционные параметры, отобразим имя программы, количество аргументов командной строки, значение каждого аргумента командной строки. \$0 – имя скрипта (название файла), \$1 – первый параметр. \$#- данная переменная содержит количество параметров \$@ или \$*- захват всех параметров, переданных скрипту (\$@ - параметры разбиты на слова, \$* - параметры представлены в виде одного слова). Результат работы представлен на рисунках 25 и 26.

```

GNU nano 4.8
#!/bin/bash
echo "Name program $0, kolv arg cmd $#"
```

```

for argum in $@
do
echo $argum
done
```

Рисунок 25 – Скрипт для отображения имени программы, количества аргументов командной строки и значений каждого аргумента командной строки

```

anna@annaserver:~$ ./skr one 1 2 3
Name program ./skr, kolv arg cmd 4
one
1
2
3
anna@annaserver:~$
```

Рисунок 26 – Результат выполнения скрипта

Используя позиционный параметр, отобразим содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается. Результат работы представлен на рисунках 27 и 28.

```

GNU nano 4.8
#!/bin/bash
if [ $# -eq 1 ]
then
if [ -s $1 ]
then
echo $(cat $1)
sleep 10
clear
else
echo "file not found"
fi
else
echo "kolv param != 1"
fi
```

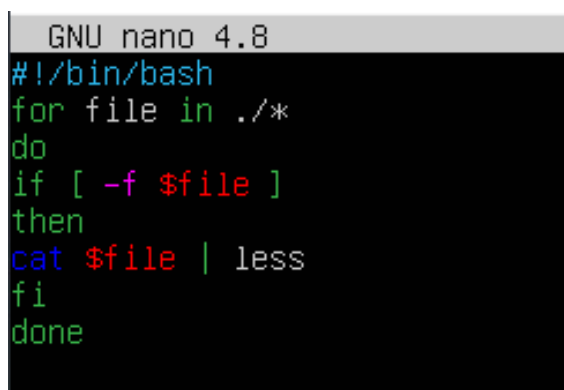
Рисунок 27 – Скрипт для отображения текстового файла

```

anna@annaserver:~$ sh skr one
q w e r
```

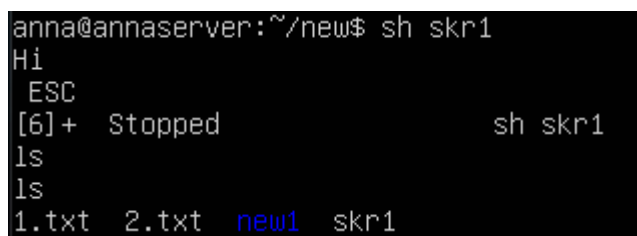
Рисунок 28 – Результат выполнения скрипта

Используя оператор FOR, отобразим содержимое текстовых файлов текущего каталога поэкранно. Результат работы представлен на рисунках 29 и 30.



```
GNU nano 4.8
#!/bin/bash
for file in ./*
do
if [ -f $file ]
then
cat $file | less
fi
done
```

Рисунок 29 – Скрипт для отображения содержимого текстовых файлов текущего каталога поэкранно



```
anna@annaserver:~/new$ sh skr1
Hi
ESC
[6]+  Stopped                  sh skr1
ls
ls
1.txt  2.txt  new1  skr1
```

Рисунок 30 – Результат выполнения скрипта

Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения. Результат работы представлен на рисунках 31 и 32.

Список логических операторов, которые используются для if|then|else:

- z — строка пуста
- n — строка не пуста
- =, (==) — строки равны
- != — строки неравны
- eq — равно
- ne — неравно
- lt,(<) — меньше
- le,(<=) — меньше или равно

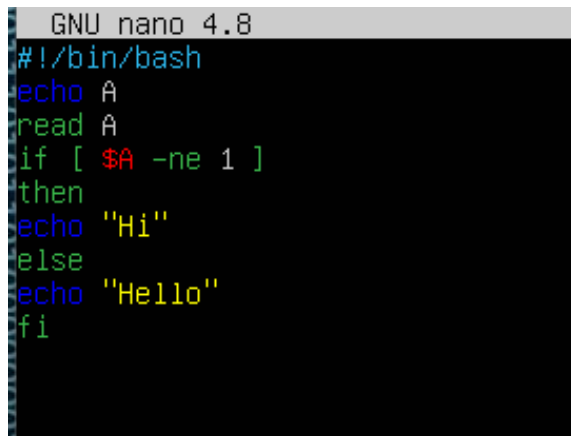
-gt,(>) — больше

-ge,(>=) — больше или равно

! — отрицание логического выражения

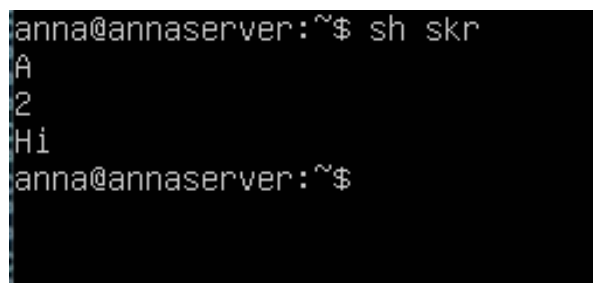
-a,(&&) — логическое «И»

-o,(||) — логическое «ИЛИ»



```
GNU nano 4.8
#!/bin/bash
echo A
read A
if [ $A -ne 1 ]
then
echo "Hi"
else
echo "Hello"
fi
```

Рисунок 31 – Ввод числа и сравнение его с допустимым значением

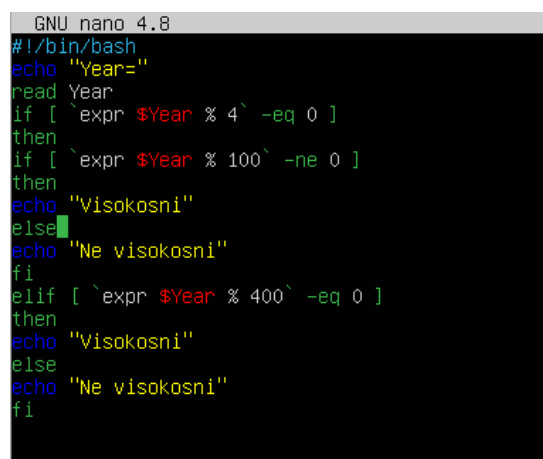


```
anna@annaserver:~$ sh skr
A
2
Hi
anna@annaserver:~$
```

Рисунок 32 – Результат выполнения скрипта

Программой запрашивается год, определяется, високосный ли он.

Результат выдается на экран. Результат работы представлен на рисунках 33 и 34.



```
GNU nano 4.8
#!/bin/bash
echo "Year="
read Year
if [ `expr $Year % 4` -eq 0 ]
then
if [ `expr $Year % 100` -ne 0 ]
then
echo "Visokosni"
else
echo "Ne visokosni"
fi
elif [ `expr $Year % 400` -eq 0 ]
then
echo "Visokosni"
else
echo "Ne visokosni"
fi
```

Рисунок 33 – Определение года (високосный/ не високосный)


```

anna@annaserver:~$ sh skr
Year=
2020
Visokosni
anna@annaserver:~$ sh skr
Year=
2021
Ne visokosni
anna@annaserver:~$ sh skr
Year=
1900
Ne visokosni
anna@annaserver:~$ _

```

Рисунок 34 – Результат выполнения скрипта

Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются. Результат работы представлен на рисунках 35 и 36.

```

#!/bin/bash
echo "A="
read A
echo "B="
read B
echo "c="
read c
echo "d="
read d
while [ $c -gt $A ] && [ $c -gt $B ] && [ $d -gt $A ] && [ $d -gt $B ]
do
A=$(expr $A + 1)
B=$(expr $B + 1)
done
echo "$A"
echo "$B"

```

Рисунок 35 – Пример скрипта ввода целочисленных значений двух переменных и ввода диапазона данных, пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.

```

anna@annaserver:~$ sh skr
A=
4
B=
5
c
7
d
8
6
7
anna@annaserver:~$

```

Рисунок 36 – Результат выполнения скрипта

В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc. Результат работы представлен на рисунках 37 и 38.

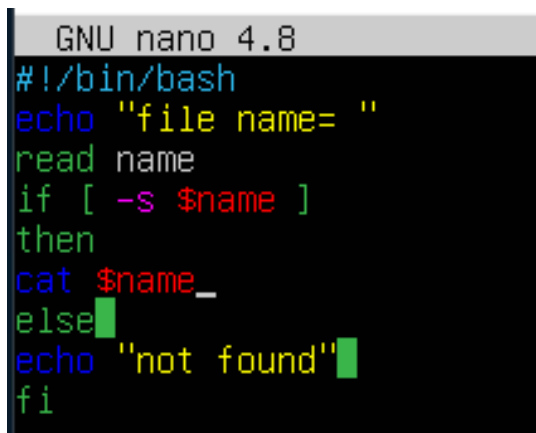
```
GNU nano 4.8
#!/bin/bash
echo "passw= "
echo passw
pass = "genuvo21"
if [ $passw = $pass ]
then
ls -al /etc | less
else "error"
fi
```

Рисунок 37 – Пример скрипта постраничного отображения содержимого каталога /etc

```
total 796
drwxr-xr-x 93 root root      4096 Oct 28 14:42 .
drwxr-xr-x 20 root root      4096 Oct 29 13:05 ..
-rw-r----- 1 root root         0 Jul 31 16:28 .pwd.lock
drwxr-xr-x  3 root root      4096 Jul 31 16:29 NetworkManager
drwxr-xr-x  2 root root      4096 Oct 23 07:50 PackageKit
drwxr-xr-x  4 root root      4096 Jul 31 16:29 X11
-rw-r--r--  1 root root     3028 Jul 31 16:28 adduser.conf
drwxr-xr-x  2 root root      4096 Jul 31 16:29 alternatives
drwxr-xr-x  3 root root      4096 Jul 31 16:29 apparmor
drwxr-xr-x  7 root root      4096 Jul 31 16:29 apparmor.d
drwxr-xr-x  3 root root      4096 Oct 23 07:52 apport
drwxr-xr-x  7 root root      4096 Oct 23 07:36 apt
-rw-r----- 1 root daemon    144 Nov 12 2018 at.deny
-rw-r--r--  1 root root     2319 Feb 25 2020 bash.bashrc
-rw-r--r--  1 root root        45 Jan 26 2020 bash_completion
drwxr-xr-x  2 root root      4096 Oct 23 07:52 bash_completion.d
-rw-r--r--  1 root root       367 Apr 14 2020 bindresvport.blacklist
drwxr-xr-x  2 root root      4096 Apr 22 2020 binfmt.d
drwxr-xr-x  2 root root      4096 Jul 31 16:29 byobu
drwxr-xr-x  3 root root      4096 Jul 31 16:28 ca-certificates
-rw-r--r--  1 root root     5714 Jul 31 16:29 ca-certificates.conf
-rw-r--r--  1 root root     5713 Jul 31 16:28 ca-certificates.conf.dpkg-old
drwxr-xr-x  2 root root      4096 Jul 31 16:29 calendar
drwxr-xr-x  4 root root      4096 Oct 23 07:47 cloud
drwxr-xr-x  2 root root      4096 Oct 23 07:57 console-setup
drwxr-xr-x  2 root root      4096 Jul 31 16:29 cron.d
drwxr-xr-x  2 root root      4096 Oct 23 07:52 cron.daily
drwxr-xr-x  2 root root      4096 Jul 31 16:28 cron.hourly
drwxr-xr-x  2 root root      4096 Jul 31 16:28 cron.monthly
drwxr-xr-x  2 root root      4096 Jul 31 16:30 cron.weekly
-rw-r--r--  1 root root     1042 Feb 13 2020 crontab
drwxr-xr-x  2 root root      4096 Oct 23 07:52 cryptsetup-initramfs
-rw-r--r--  1 root root        54 Jul 31 16:29 crypttab
drwxr-xr-x  4 root root      4096 Jul 31 16:28 dbus-1
drwxr-xr-x  3 root root      4096 Jul 31 16:29 dconf
:
```

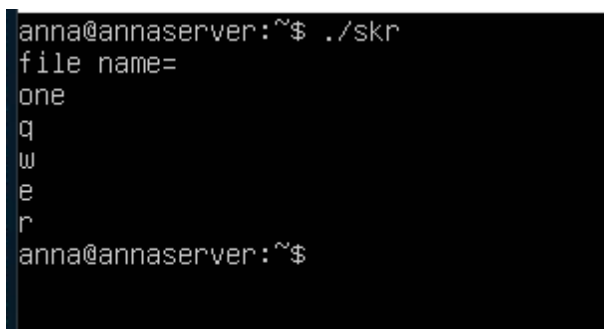
Рисунок 38 – Результат выполнения скрипта

Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение. Результат работы представлен на рисунках 39 и 40.



```
GNU nano 4.8
#!/bin/bash
echo "file name= "
read name
if [ -s $name ]
then
cat $name_
else
echo "not found"
fi
```

Рисунок 39 – Проверка файла на наличие и вывод содержимого этого файла на экран



```
anna@annaserver:~$ ./skr
file name=
one
q
w
e
r
anna@annaserver:~$
```

Рисунок 40 – Результат выполнения скрипта

Если файл есть каталог и этот каталог можно читать, просматривается содержимое этого каталога. Если каталог отсутствует, он создается. Если файл не есть каталог, просматривается содержимое файла. Результат работы представлен на рисунках 41 и 42.

```

GNU nano 4.8
#!/bin/bash
echo "file name= "
read name
if [ -e $name ]
then
if [ -d $name ]
then
if [ -r $name ]
then
ls $name
fi
else
cat $name
fi
else
mkdir $name
fi

```

Рисунок 41 – Пример скрипта

```

anna@annaserver:~$ ./skr
file name=
one
q
w
e
r
anna@annaserver:~$ ./skr
file name=
new
1.txt 2.txt new1 skr1
anna@annaserver:~$ ./skr
file name=
newd
anna@annaserver:~$ ls
chan channel loop.sh loop2.sh new newd one res ress skr t.txt text1 two txtone u.txt
anna@annaserver:~$

```

Рисунок 42 – Результат выполнения скрипта

Анализируются атрибуты файла. Если первый файл существует и используется для чтения, а второй файл существует и используется для записи, то содержимое первого файла перенаправляется во второй файл. В случае несовпадений указанных атрибутов или отсутствия файлов на экран выдаются соответствующие сообщения (использовать а) имена файлов; б) позиционные параметры). Результат работы представлен на рисунках 43, 44, 45 и 46.

```

GNU nano 4.8                                skr
#!/bin/bash
echo "first file name= "
read first
echo "second file name="
read second
if [ -e $first ]
then
if [ -r $first ]
then
if [ -e $second ]
then
if [ -w $second ]
then
cat $first > $second
else
echo "The file is not writing"
fi
else
echo "file not found"
fi
else
echo "The file is not readable"
fi
else
echo "file not found "
fi

```

Рисунок 43 – Пример скрипта задания под буквой «а»

```

anna@annaserver:~$ cat txtone
anna@annaserver:~$ cat one
q
w
e
r
anna@annaserver:~$ ./skr
first file name=
one
second file name=
txtone
anna@annaserver:~$ cat txt one
cat: txt: No such file or directory
q
w
e
r
anna@annaserver:~$ cat txtone
q
w
e
r
anna@annaserver:~$ ./skr
first file name=
one
second file name=
hj
file not found
anna@annaserver:~$ ls
chan channel loop.sh loop2.sh new newd one res ress skr t.txt text1 two txtone u.txt
anna@annaserver:~$ _

```

Рисунок 44 – Результат выполнения скрипта

```

GNU nano 4.8                                skr
#!/bin/bash
echo "first file name= "
read first
echo "second file name="
read second
if [ -e $1 ]
then
  if [ -r $1 ]
  then
    if [ -e $2 ]
    then
      if [ -w $2 ]
      then
        cat $2 > $2_
      else
        echo "The file is not writing"
      fi
    else
      echo "file not found"
    fi
  else
    echo "The file is not readable"
  fi
else
  echo "file not found "
fi

```

Рисунок 45 – Пример скрипта задания под буквой «б»

```

anna@annaserver:~$ cat txtone
anna@annaserver:~$ cat one
q
w
e
r
anna@annaserver:~$ ./skr
first file name=
one
second file name=
txtone
anna@annaserver:~$ cat txtone
q
w
e
r
anna@annaserver:~$ _

```

Рисунок 46 – Результат выполнения скрипта

Если файл запуска программы найден, программа запускается (по выбору). Результат работы представлен на рисунках 47 и 48.

```

anna@annaserver:~$ ./skrr ./scr
hi
anna@annaserver:~$ _

```

Рисунок 47 – Запуск программы

```

GNU nano 4.8
#!/bin/bash
if [ $# = 1 ]
then
if [ -e $1 ] && [ -x $1 ]
then
sh $1
else
echo "error"
fi
else
echo "arv error"
fi_

```

Рисунок 48 – Результат выполнения скрипта

В качестве позиционного параметра задается файл, анализируется его размер. Если размер файла больше нуля, содержимое файла сортируется по первому столбцу по возрастанию, отсортированная информация помещается в другой файл, содержимое которого затем отображается на экране. Результат работы представлен на рисунках 49 и 50.

```

GNU nano 4.8
#!/bin/bash
if [ $# = 1 ]
then
if [ -s $1 ]
then
sort -k1 $1 >"oneone"
cat "oneone"
else
echo "error"
fi
fi

```

Рисунок 49 – Пример скрипта

```
anna@annaserver:~$ ./skr one
e
q
r
w
anna@annaserver:~$
anna@annaserver:~$ cat one
q
w
e
r
anna@annaserver:~$ _
```

Рисунок 50 – Результат выполнения скрипта

Командой TAR осуществляется сборка всех текстовых файлов текущего каталога в один архивный файл my.tar, после паузы просматривается содержимое файла my.tar, затем командой GZIP архивный файл my.tar сжимается. Результат работы представлен на рисунках 51 и 52.

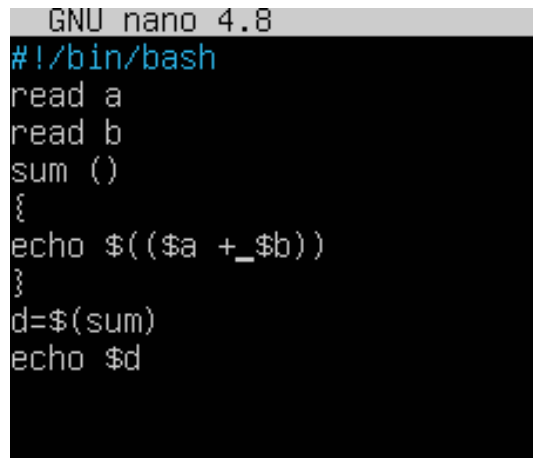
```
GNU nano 4.8
#!/bin/bash
fin=$(find . -type f -maxdepth 1)
a="archs.tar"
tar -cf $a $fin
sleep 5
tar -tf $a
gzip $fin
```

Рисунок 51 – Сборка всех текстовых файлов текущего каталога в архивный файл

```
./loop2.sh
./swp
./script.swp
./sudo_as_admin_successful
./oneone
./text1
./profile
./skr2
./ress
./bashrc
./u.txt
./t.txt
./bash_logout
./txtone
./skr.swp
./bash_history
./res
./chan
./viminfo
./loop.sh
./one
./two
./skr
anna@annaserver:~$ _
```

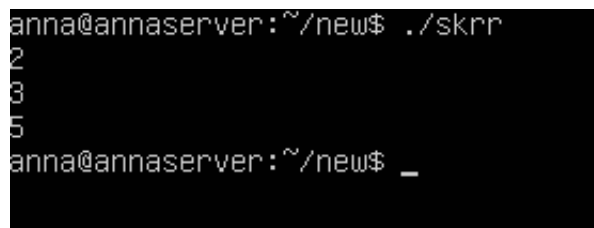
Рисунок 52 – Результат выполнения скрипта

Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных. Результат работы представлен на рисунках 53 и 54.



```
GNU nano 4.8
#!/bin/bash
read a
read b
sum ()
{
echo $((a + b))
}
d=$(sum)
echo $d
```

Рисунок 53 – Пример скрипта с использованием функции сложения



```
anna@annaserver:~/new$ ./skrr
2
3
5
anna@annaserver:~/new$ _
```

Рисунок 54 – Результат выполнения скрипта

Вывод

В ходе выполнения лабораторной работы были изучены основные возможности языка программирования Shell с целью автоматизации процесса администрирования системы за счет написания и использования командных файлов.