

Липецкий государственный технический университет

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине «Операционная система Linux»

Программирование на SHELL. Использование командных файлов.

Студент

Титов В. А.

Группа АС-20

Руководитель

Кургасов В. В.

к.п.н.

Липецк 2022 г.

Цель работы

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

Ход работы

1. Используя команды ECHO, PRINTF вывести информационные сообщения на экран.

```
root@10:/home/vlad# echo "Hello"
Hello
root@10:/home/vlad# printf "Hello"
Helloroot@10:/home/vlad# printf "Hello\n"
Hello
root@10:/home/vlad#
```

Рис. 1 – Вывод информационного сообщения командами echo и printf

2. Присвоить переменной A целочисленное значение. Просмотреть значение переменной A.

```
vlad@10:~$ a=10
vlad@10:~$ echo $a
10
vlad@10:~$
```

Рис. 2 – Присваивание переменной A целочисленное значение и его просмотр

3. Присвоить переменной B значение переменной A. Просмотреть значение переменной B.

```
vlad@10:~$ b=$a
vlad@10:~$ echo $b
10
vlad@10:~$
```

Рис. 3 – Присваивание переменной B значение переменной A и его просмотр

4. Присвоить переменной C значение “путь до своего каталога”. Перейти в этот каталог с использованием переменной.

```
vlad@10:~$ c="/DIR"
vlad@10:~$ cd $c
vlad@10:/DIR$ _
```

Рис. 4 – Присваивание переменной C путь до каталога /DIR и переход в него с помощью этой переменной

5. Присвоить переменной D значение “имя команды”, а именно, команды DATE. Выполнить эту команду, используя значение переменной.

```
vlad@10:~$ d="date"
vlad@10:~$ $d
Чт 01 дек 2022 16:16:01 MSK
vlad@10:~$
```

Рис. 5 – Присваивание переменной D имя команды DATE и выполнение команды 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.

```
vlad@10:~$ e="cat loop"
vlad@10:~$ $e
while true; do true; done
vlad@10:~$ _
```

Рис. 6 – Присваивание переменной E команду просмотра содержимого файла и его просмотр используя значение переменной

7. Присвоить переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной.

```
vlad@10:~$ f="sort text.txt"
vlad@10:~$ $f
computer
data
debian
laptop
mouse
redhat
vlad@10:~$
```

Рис. 7 – Присваивание переменной F команду сортировки содержимого текстового файла и её выполнение используя значение переменной.

8. Программа запрашивает значение переменной, а затем выводит значение этой переменной.

```
vlad@10:~$ bash prog1
Введите значение переменной A
10
a= 10
vlad@10:~$
```

Рис. 8 – Результат выполнения программы.

9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной.

```
vlad@10:~$ bash prog2
Введите Ваше имя!
Владислав
Привет, Владислав
vlad@10:~$ _
```

Рис. 9 – Результат выполнения программы.

10. Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) BC).

```
GNU nano 5.4 prog3
#!/bin/bash

echo Введите значение a
read a
echo Введите значение b
read b

c=$( expr $a + $b )
echo Сумма: $c

u=$( expr $a \* $b )
echo Произведение: $u

d=$(bc<<<"scale=3;$a/$b")
echo Деление: $d

r=$(bc<<<"scale=3;$a-$b")
echo Разность: $r
```

Рис. 10 – Код программы для вычисления значений, введенных пользователем.

```
vlad@debian:~$ bash prog3
Введите значение a
3
Введите значение b
2
Сумма: 5
Произведение: 6
Деление: 1.500
Разность: 1
vlad@debian:~$
```

Рис. 11 – Результат выполнения команды.

11. Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран.

```

GNU nano 5.4                                prog4
#!/bin/bash

echo Введите высоту цилиндра
read h

echo Введите радиус цилиндра
read r

p=3.14

v=$(bc<<<"scale=3;$p*$r*$r*$h")
echo Объем цилиндра: $v

```

Рис. 12 – Код программы для вычисления объема цилиндра.

```

vlad@debian:~$ bash prog4
Введите высоту цилиндра
2
Введите радиус цилиндра
3
Объем цилиндра: 56.52
vlad@debian:~$ _

```

Рис. 13 – Результат выполнения команды.

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

```

GNU nano 5.4                                prog5
#!/bin/sh
echo "Имя программы: $0. Количество аргументов командной строки $#.
Значение аргументов командной строки:"
while [ ! -z $1 ]
do
echo $1
shift
done

```

Рис. 14 – Код программы для отображения имя программы, количества аргументов и значение каждого аргумента.

```

vlad@debian:~$ sh prog5 arg1 arg2
Имя программы: prog5. Количество аргументов командной строки 2.
Значение аргументов командной строки:
arg1
arg2
vlad@debian:~$ _

```

Рис. 15 – Результат выполнения команды.

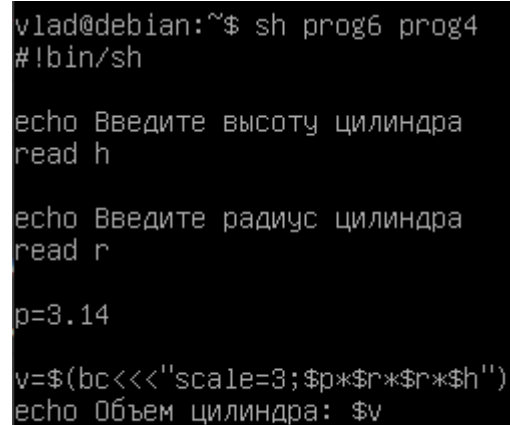
13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается.



```
GNU nano 5.4 prog6
#!/bin/sh

more $1
sleep 5s
clear
```

Рис. 16 – Код программы для отображения содержимого текстового файла и очистка экрана после паузы.



```
vlad@debian:~$ sh prog6 prog4
#!/bin/sh

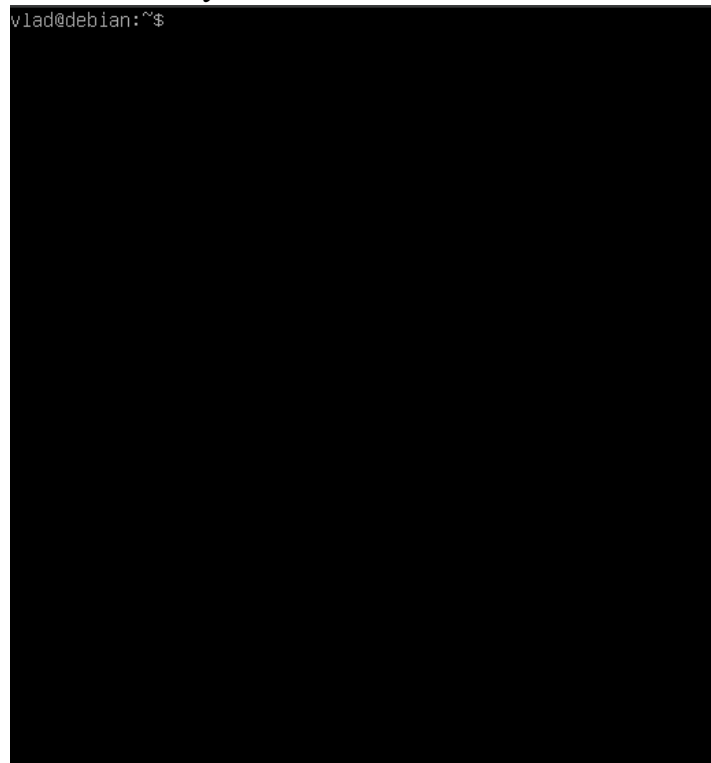
echo Введите высоту цилиндра
read h

echo Введите радиус цилиндра
read r

p=3.14

v=$(bc<<<"scale=3;$p*$r*$r*$h")
echo Объем цилиндра: $v
```

Рис. 17 – Результат №1 выполнения команды.



```
vlad@debian:~$
```

Рис. 18 – Результат №2 выполнения команды.

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

```
GNU nano 5.4 prog7 *
#!/bin/sh

for f in *.txt
do
printf"\n\tфайл $f\n\n"

more $f

done
```

Рис. 19 – Код программы для отображения содержимого текстовых файлов текущего каталога поэкранно.

```
v=$(bc<<<"scale=3;$r*$r*$r*$h")
echo Объем цилиндра: $v

        файл prog5

#!/bin/sh
echo "Имя программы: $0. Количество аргументов командной строки $#.
Значение аргументов командной строки:"
while [ ! -z $1 ]
do
echo $1
shift
done

        файл prog6

#!/bin/sh

more $1
sleep 5s
clear

        файл prog7

#!/bin/sh

for f in *
do
echo "\n\tфайл $f\n\n"

more $f

done
vlad@debian:~$
```

Рис. 20 – Результат выполнение программы.

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


```

GNU nano 5.4                                prog8
#!/bin/sh

echo -n "Введите число от 1 до 10: "

read s

if [ $s -lt 1 ] || [ $s -gt 10 ]
then
    echo "Введено не верное значение"
else
    echo "Введено верное значение"
fi

```

Рис. 21 – Код программы для сравнения вводимого числа пользователем с допустимым значением.

```

vlad@debian:~$ sh prog8
Введите число от 1 до 10: 3
Введено верное значение
vlad@debian:~$ sh prog8
Введите число от 1 до 10: 12
Введено не верное значение
vlad@debian:~$ _

```

Рис. 22 – Результат выполнения программы.

16. Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран.

```

GNU nano 5.4                                prog9
#!/bin/sh

echo "Введите год: "

read g

c=$((g%4))

if [ $c = 0 ]
then
    echo "Високосный год"
else
    echo "Не високосный год"
fi_

```

Рис. 23 – Код программы для определения високосного года.

```

vlad@debian:~$ sh prog9
Введите год:
1999
Не високосный год
vlad@debian:~$ sh prog9
Введите год:
2000
Високосный год
vlad@debian:~$

```

Рис. 24 – Результат выполнения программы.

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

```
GNU nano 5.4 prog10
#!/bin/sh

echo "Введите значения переменной A"
read a

echo "Введите значение переменной B"
read b

read -p "Задайте границы диапазона данных через пробел: " c d

if [ $a -gt $c ] && [ $b -gt $c ]
then
    while [ $a -lt $d ] && [ $b -lt $d ]
    do
        a=$((a+1))
        b=$((b+1))
        echo "a = $a, b = $b"
    done
else
    echo "a = $a, b = $b""Не попали в заданный диапазон"
fi
```

Рис. 25 – Код программы для введения двух переменных и диапазона, после чего переменные в указанном диапазоне инкрементируются.

```
vlad@debian:~$ sh prog10
Введите значения переменной A
2
Введите значение переменной B
3
Задайте границы диапазона данных через пробел: 1 10
a = 3, b = 4
a = 4, b = 5
a = 5, b = 6
a = 6, b = 7
a = 7, b = 8
a = 8, b = 9
a = 9, b = 10
vlad@debian:~$ _
```

Рис. 26 – Результат выполнения программы.

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

```
GNU nano 5.4 prog11
#!/bin/sh

pass="123"

if [ $1 = $pass ]
then
    ls -la /etc | more
fi
```

Рис. 27 – Код программы для вывода скрытых файлов каталога /etc.

```

итого 716
drwxr-xr-x 74 root root 4096 дек 2 21:06 .
drwxr-xr-x 18 root root 4096 дек 1 21:34 ..
-rw-r--r-- 1 root root 2981 дек 1 21:14 adduser.conf
-rw-r--r-- 1 root root 44 дек 1 21:46 adjtime
drwxr-xr-x 2 root root 4096 дек 1 21:40 alternatives
-rw-r--r-- 1 root root 401 фев 6 2021 anacrontab
-rw-r--r-- 1 root root 4185 июл 29 2019 analog.cfg
drwxr-xr-x 8 root root 4096 дек 1 21:40 apache2
drwxr-xr-x 2 root root 4096 дек 1 21:14 apparmor
drwxr-xr-x 7 root root 4096 дек 1 21:40 apparmor.d
drwxr-xr-x 8 root root 4096 дек 1 21:46 apt
drwxr-xr-x 2 root root 4096 дек 1 21:40 avahi
-rw-r--r-- 1 root root 1994 мар 27 2022 bash.bashrc
-rw-r--r-- 1 root root 45 янв 25 2020 bash_completion
-rw-r--r-- 1 root root 367 июл 29 2019 bindresvport.blacklist
drwxr-xr-x 2 root root 4096 авг 7 16:25 binfmt.d
drwxr-xr-x 2 root root 4096 дек 1 21:40 bluetooth
drwxr-xr-x 3 root root 4096 дек 1 21:39 ca-certificates
-rw-r--r-- 1 root root 5662 дек 1 21:40 ca-certificates.conf
drwxr-xr-x 2 root root 4096 дек 1 21:14 console-setup
drwxr-xr-x 2 root root 4096 дек 1 21:40 cron.d
drwxr-xr-x 2 root root 4096 дек 1 21:40 cron.daily
drwxr-xr-x 2 root root 4096 дек 1 21:14 cron.hourly
drwxr-xr-x 2 root root 4096 дек 1 21:40 cron.monthly
-rw-r--r-- 1 root root 1042 фев 23 2021 crontab
drwxr-xr-x 2 root root 4096 дек 1 21:40 cron.weekly
drwxr-xr-x 4 root root 4096 дек 1 21:39 dbus-1
-rw-r--r-- 1 root root 2969 июн 10 2021 debconf.conf
-rw-r--r-- 1 root root 5 сен 3 15:10 debian_version
drwxr-xr-x 3 root root 4096 дек 1 21:40 default
-rw-r--r-- 1 root root 604 июн 26 2016 deluser.conf
drwxr-xr-x 4 root root 4096 дек 1 21:34 dhcp
drwxr-xr-x 2 root root 4096 дек 1 21:40 dictionaries-common
drwxr-xr-x 2 root root 4096 дек 1 21:34 discover.conf.d
-rw-r--r-- 1 root root 346 янв 15 2018 discover-modprobe.conf
--More--

```

Рис. 28 – Результат выполнения программы.

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

```

GNU nano 5.4 prog12
#!/bin/sh

if [ -f $1 ]
then
    more $1
else
    echo "Файла не существует"
fi

```

Рис. 29 – Код программы для проверки существующего файла, если такой файл существует, то вывод осуществляется путем вывода содержимого.

```

vlad@debian:~$ sh prog12 prog4
#!/bin/sh

echo Введите высоту цилиндра
read h

echo Введите радиус цилиндра
read r

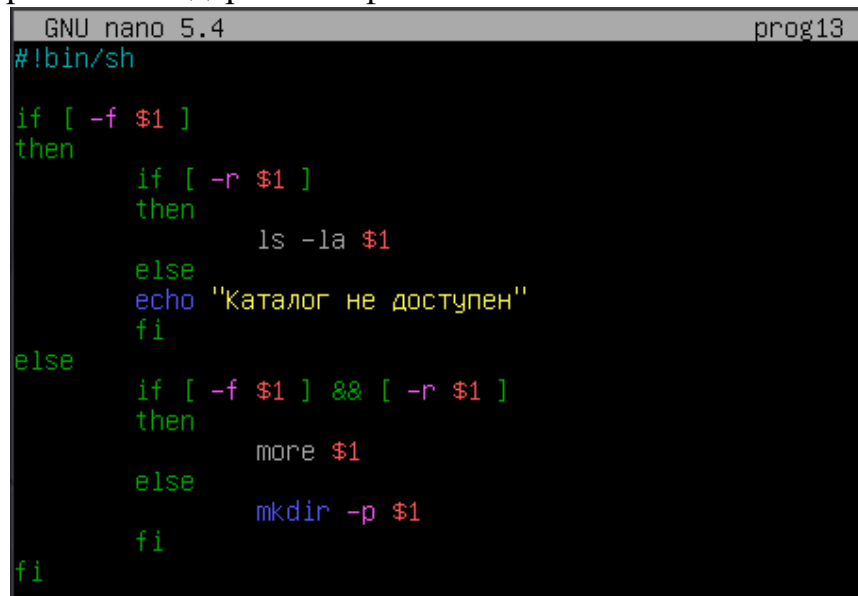
p=3.14

v=$(bc<<<"scale=3;$p*$r*$r*$h")
echo Объем цилиндра: $v
vlad@debian:~$

```

Рис. 30 – Результат выполнения программы.

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.



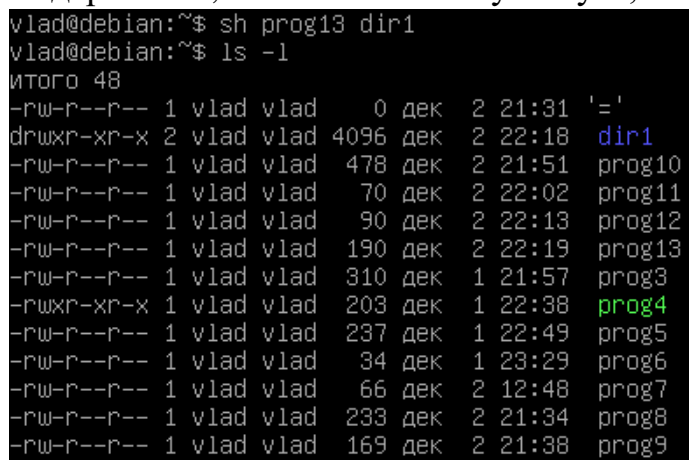
```

GNU nano 5.4                                prog13
#!/bin/sh

if [ -f $1 ]
then
    if [ -r $1 ]
    then
        ls -la $1
    else
        echo "Каталог не доступен"
    fi
else
    if [ -f $1 ] && [ -r $1 ]
    then
        more $1
    else
        mkdir -p $1
    fi
fi

```

Рис. 31 – Код программы для поиска и проверки доступа файла или каталога, если каталог или файл существует и есть к нему доступ, то выводить содержимое, если каталог отсутствует, он создается.



```

vlad@debian:~$ sh prog13 dir1
vlad@debian:~$ ls -l
итого 48
-rw-r--r-- 1 vlad vlad  0 дек  2 21:31 '='
drwxr-xr-x 2 vlad vlad 4096 дек  2 22:18 dir1
-rw-r--r-- 1 vlad vlad 478 дек  2 21:51 prog10
-rw-r--r-- 1 vlad vlad  70 дек  2 22:02 prog11
-rw-r--r-- 1 vlad vlad  90 дек  2 22:13 prog12
-rw-r--r-- 1 vlad vlad 190 дек  2 22:19 prog13
-rw-r--r-- 1 vlad vlad 310 дек  1 21:57 prog3
-rwxr-xr-x 1 vlad vlad 203 дек  1 22:38 prog4
-rw-r--r-- 1 vlad vlad 237 дек  1 22:49 prog5
-rw-r--r-- 1 vlad vlad  34 дек  1 23:29 prog6
-rw-r--r-- 1 vlad vlad  66 дек  2 12:48 prog7
-rw-r--r-- 1 vlad vlad 233 дек  2 21:34 prog8
-rw-r--r-- 1 vlad vlad 169 дек  2 21:38 prog9

```

Рис. 32 – Результат выполнения программы.

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).

```

GNU nano 5.4                                prog14
#!/bin/sh

if [ -r $1 ]
then
    if [ -w $2 ]
    then
        cat $1 >> $2
    else
        echo "Файл $2 не существует или не доступен для записи"
    fi
else
    echo "Файл $1 не существует, либо не доступен для чтения"
fi

```

Рис. 33 – Код программы для анализа атрибута файла и записи содержимого одного в другой.

```

vlad@debian:~$ sh prog14 prog3 prog5
vlad@debian:~$ cat prog5
#!/bin/sh
echo "Имя программы: $0. Количество аргументов командной строки $#.
Значение аргументов командной строки:"
while [ ! -z $1 ]
do
    echo $1
    shift
done
#!/bin/bash

echo Введите значение a
read a

echo Введите значение b
read b

c=$( expr $a + $b )
echo Сумма: $c

u=$( expr $a \* $b )
echo Произведение: $u

d=$(bc<<<"scale=3;$a/$b")
echo Деление: $d

r=$(bc<<<"scale=3;$a-$b")
echo Разность: $r

vlad@debian:~$ _

```

Рис. 34 – Результат выполнения программы.

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

```

GNU nano 5.4                                prog15
#!/bin/bash

if [ -f $1 ]
then
    sh $1
else
    echo "Файл программы не найден"
fi

```

Рис. 35 – Код программы для запуска другой программы.

```

vlad@debian:~$ sh prog15 prog10
Введите значения переменной А
2
Введите значение переменной В
2
Задайте границы диапазона данных через пробел: 1 4
a = 3, b = 3
a = 4, b = 4
vlad@debian:~$ sh prog15 prog102
Файл программы не найден
vlad@debian:~$

```

Рис. 36 – Результат выполнения программы.

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.

```

GNU nano 5.4 prog16
#!/bin/sh

size=`stat -c %s $1`
echo размер файла = $size

if [ $size -gt 0 ]
then
    sort $1 > $2
    more $2
else
    echo Это пустой файл
fi

```

Рис. 37 – Код программы для определения размера файла и передачи отсортированных строк другому файлу.

```

vlad@debian:~$ sh prog16 text text2
размер файла = 20
1
2
3
4
4
4
4
5
6
6
vlad@debian:~$ _

```

Рис. 38 – Результат выполнения программы.

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.

```

GNU nano 5.4                                prog17
#!/bin/sh

tar -cf my.tar *.txt
sleep 2s

tar -tf my.tar
sleep 2s

gzip my.tar

```

Рис. 39 – Код программы для архивирования текстовых файлов.

```

vlad@debian:~$ ls
'='      2.txt  dir1    prog11  prog13  prog15  prog17  prog4    prog6  prog8  text
1.txt    3.txt  prog10  prog12  prog14  prog16  prog3    prog5    prog7  prog9  text2
vlad@debian:~$ sh prog17
1.txt
2.txt
3.txt
vlad@debian:~$ ls
'='      3.txt      prog10  prog13  prog16  prog4    prog7  text
1.txt    dir1      prog11  prog14  prog17  prog5    prog8  text2
2.txt    my.tar.gz prog12  prog15  prog3    prog6    prog9
vlad@debian:~$

```

Рис. 40 – Результат выполнения программы.

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

```

GNU nano 5.4                                prog18
#!/bin/sh

sum()
{
    res=`expr $1 '+' $2`
    return $res
}

sum $1 $2
echo $res

```

Рис. 41 – Код программы, суммирующий значения двух переменных с помощью функции.

```

vlad@debian:~$ sh prog18 3 5
8
vlad@debian:~$ _

```

Рис. 42 – Результат выполнения программы.

Вывод

В ходе выполнения лабораторной работы я изучил основные возможностей языка программирования Shell с целью автоматизации процесса администрирования системы за счет написания и использования командных файлов.