

**Липецкий государственный технический университет**

**Факультет автоматизации и информатики**

**Кафедра автоматизированных систем управления**

**Лабораторная работа № 5**

**По дисциплине «OS Linux»**

**Программирование на “SHELL”**

Студент

Бахмутский М.В.

Группа АС-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

## 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

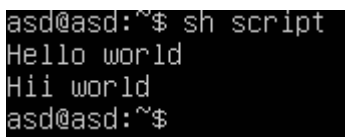
### 1 Задание 1

Скрипт для задания 1 показан на рисунке 1, через `echo` выведем “Hello word”, а через `printf` “Hii world”. Выполнение данного скрипта показано на рисунке 2.



```
GNU nano 2.9.3
echo Hello world
printf '%b\n' 'Hii world'
```

Рисунок 1 – Скрипт для задания 1

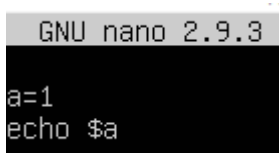


```
asd@asd:~$ sh script
Hello world
Hii world
asd@asd:~$
```

Рисунок 2 – Выполнение задания 1

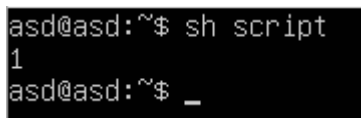
### 2 Задание 2

Скрипт для задания 2 показан на рисунке 3, для присваивания переменной значение воспользуемся знаком равенства. Выполнение данного скрипта показано на рисунке 4.



```
GNU nano 2.9.3
a=1
echo $a
```

Рисунок 3 – Скрипт для задания 2

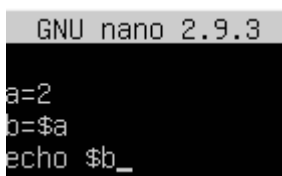


```
asd@asd:~$ sh script
1
asd@asd:~$ _
```

Рисунок 4 – Выполнение задания 2

### 3 Задание 3

Скрипт для задания 3 показан на рисунке 5, для присваивания одной переменной значение другой воспользуемся знаком равенства и укажем переменную через знак `$`. Выполнение данного скрипта показано на рисунке 6.



```
GNU nano 2.9.3
a=2
b=$a
echo $b_
```

### Рисунок 5 – Скрипт для задания 3

```
asd@asd:~$ sh script
2
asd@asd:~$ _
```

### Рисунок 6 – Выполнение задания 3

#### 4 Задание 4

Скрипт для задания 4 показан на рисунке 7, для присваивания переменной значение пути т.е. строки помести необходимое выражение в одинарные кавычки и приравняем его к необходимой переменной. Выполнение данного скрипта показано на рисунке 8.

```
GNU nano 2.9.3
c='/home/asd'
cd $c
```

### Рисунок 7 – Скрипт для задания 3

```
asd@asd:~$ cd /home
asd@asd:/home$ . asd/script
asd@asd:~$
```

### Рисунок 8 – Выполнение задания 3

#### 5 Задание 5

Скрипт для задания 5 показан на рисунке 9, для присваивания переменной значение даты поместим функцию date в скобки и поставим перед ними знак \$. Выполнение данного скрипта показано на рисунке 10.

```
GNU nano 2.9.3
d=$(date)
echo $d
```

### Рисунок 9 – Скрипт для задания 5

```
asd@asd:~$ sh script
Fri Nov 27 18:09:33 UTC 2020
asd@asd:~$
```

### Рисунок 10 – Выполнение задания 5

#### 6 Задание 6

Скрипт для задания 6 показан на рисунке 11, для присваивания переменной команды необходимо записать команду в переменную как строку т.е.

заклучить в одинарные кавычки, а для выполнения команды необходимо обратиться к переменной. Выполнение данного скрипта показано на рисунке 12.

```
GNU nano 2.9.3
e='cat script'
$e
```

Рисунок 11 – Скрипт для задания 6

```
asd@asd:~$ sh script
e='cat script'
$e
asd@asd:~$
```

Рисунок 12 – Выполнение задания 6

## 7 Задание 7

Скрипт для задания 7 показан на рисунке 13, для присваивания переменной команды необходимо записать команду в переменную как строку т.е. заключить в одинарные кавычки, а для выполнения команды необходимо обратиться к переменной и приписать файл для сортировки. Выполнение данного скрипта показано на рисунке 14.

```
GNU nano 2.9.3
f='sort'
$f test.txt
```

Рисунок 13 – Скрипт для задания 7

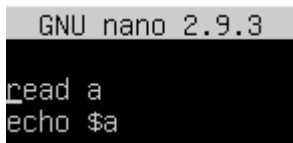
```
asd@asd:~$ sh script
1
2
2
3
3
5
6
6
7
9
asd@asd:~$ _
```

Рисунок 14 – Выполнение задания 7

## 8 Задание 8

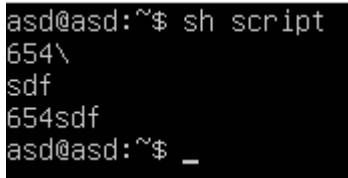
Скрипт для задания 8 показан на рисунке 15, для чтения текста из терминала воспользуемся командой `read` с выбранной переменной и выведем ее

значения с помощью команды `echo`. Выполнение данного скрипта показано на рисунке 16.



```
GNU nano 2.9.3
read a
echo $a
```

Рисунок 15 – Скрипт для задания 8

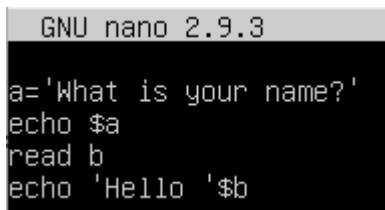


```
asd@asd:~$ sh script
654\
sdf
654sdf
asd@asd:~$ _
```

Рисунок 16 – Выполнение задания 8

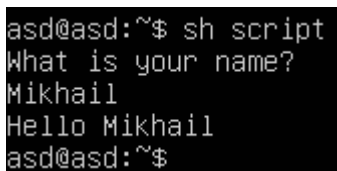
## 9 Задание 9

Скрипт для задания 9 показан на рисунке 17, для вывода команды на экран воспользуемся командой `echo`, а для чтения воспользуемся командой `read`. Выполнение данного скрипта показано на рисунке 18.



```
GNU nano 2.9.3
a='What is your name?'
echo $a
read b
echo 'Hello '$b
```

Рисунок 17 – Скрипт для задания 9



```
asd@asd:~$ sh script
What is your name?
Mikhail
Hello Mikhail
asd@asd:~$
```

Рисунок 18 – Выполнение задания 9

## 10 Задание 10

Скрипт для задания 10 показан на рисунках 19 и 21, для решения арифметических операций можно воспользоваться двумя методами. Один из методов это воспользоваться `expr`, для перед арифметической операцией напишем `expr` и заключим это в скобки и поставим спереди знак `$`. А другой способ заключается в том, что решение арифметической операцией происходит в



команде echo с припиской в конце |bc. Выполнение данного скрипта показано на рисунке 20 и 22.

```
read a
read b
sum=$(expr $a + $b)
raz=$(expr $a - $b)
umn=$(expr $a \* $b)
del=$(expr $a / $b)
echo $sum,$raz,$umn,$del
```

Рисунок 19 – Скрипт для задания 10.1

```
asd@asd:~$ sh script
5
8
13,-3,40,0
asd@asd:~$
```

Рисунок 20 – Выполнение задания 10.1

```
GNU nano 2.9.3
read a
read b
echo $a + $b|bc
echo $a - $b|bc
echo $a \* $b|bc
echo $a / $b|bc
```

Рисунок 21 – Скрипт для задания 10.2

```
asd@asd:~$ sh script
4
2
6
2
8
2
asd@asd:~$
```

Рисунок 22 – Выполнение задания 10.2

## 11 Задание 11

Скрипт для задания 11 показан на рисунке 23. Выполнение данного скрипта показано на рисунке 24.

```
GNU nano 2.9.3
read a
read b
R=$(( $a*$a ))
echo $R
pi='3.14'
V=$(echo "$pi * $R * $b" | bc)
echo $V
```

Рисунок 23 – Скрипт для задания 11

```
asd@asd:~$ sh script
2
2
4
25.12
asd@asd:~$
```

Рисунок 24 – Выполнение задания 11

## 12 Задание 12

Скрипт для задания 12 показан на рисунке 25, имя программы в позиционных параметрах находятся на 0 ячейке, а все остальные с 1 позиции по n-ую, обратиться к ним можно через знак \$, а узнать их количество можно через \$#. Для перебора позиционных параметров воспользуемся циклом for, который перебирается все значения в позиционных параметрах, его область видимости заключается с do до done. Выполнение данного скрипта показано на рисунке 26.

```
GNU nano 2.9.3
echo name $0, count arg $#
for arg in $@
do
echo $arg
done
```

Рисунок 25 – Скрипт для задания 12

```
asd@asd:~$ sh script 1 2 3 4
name script, count arg 4
1
2
3
4
asd@asd:~$ _
```

Рисунок 26 – Выполнение задания 12

## 13 Задание 13

Скрипт для задания 13 показан на рисунке 27, для паузы воспользуемся командой `sleep`, а для очистки экрана командой `clear`. Выполнение данного скрипта показано на рисунке 28.

```
GNU nano 2.9.3
echo $(cat $1)
sleep 5
clear
```

Рисунок 27 – Скрипт для задания 13

```
asd@asd:~$ sh script test.txt
5 3 2 6 1 2 3 6 9 7
_
```

Рисунок 28 – Выполнение задания 13

#### 14 Задание 14

Скрипт для задания 14 показан на рисунке 29, для отображения содержимого текстовых файлов текущего каталога найдем их названия и поместим все эти файлы в лист с помощью команды `find` и выведем их содержимое на экран с помощью команды `cat`. Выполнение данного скрипта показано на рисунке 30.

```
GNU nano 2.9.3
list=$(find . -maxdepth 1 -type f -name "*.txt")
for f in $list
do
cat $f
done
```

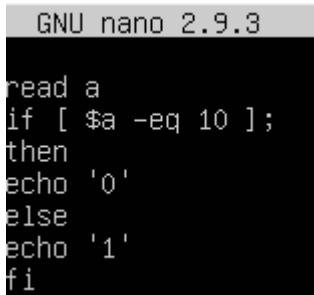
Рисунок 29 – Скрипт для задания 14

```
asd@asd:~$ sh script
dfg
sdf sdf
eryerue
5
sdfsefs
5
3
2
6
1
2
3
6
9
7
asd@asd:~$
```

## Рисунок 30 – Выполнение задания 14

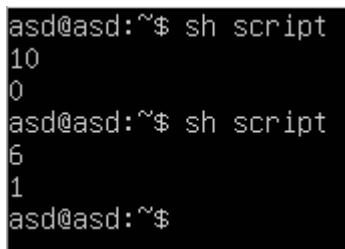
### 15 Задание 15

Скрипт для задания 15 показан на рисунке 31, для сравнения переменной с допустимым числом используется команда `if` в квадратные скобки помечаем логическую операцию, область видимости заключена с `then` до `fi`, для отслеживания исключений используем `else`. Выполнение данного скрипта показано на рисунке 32.



```
GNU nano 2.9.3
read a
if [ $a -eq 10 ];
then
echo '0'
else
echo '1'
fi
```

## Рисунок 31 – Скрипт для задания 15



```
asd@asd:~$ sh script
10
0
asd@asd:~$ sh script
6
1
asd@asd:~$
```

## Рисунок 32 – Выполнение задания 15

### 16 Задание 16

Скрипт для задания 16 показан на рисунке 32.

Описание скрипта:

1. запись из терминала в переменную `a`
2. проверка является ли остаток от деления на 4 не нулем
3. если да то выводим на экран что год является не високосным
4. иначе проверяем является ли остаток от деления на 100 нулем и является ли остаток от деления на 400 не нулем
5. если проверка правильна выводим на экран что год не високосный
6. иначе выводим что год високосный

Выполнение данного скрипта показано на рисунке 34.

```

GNU nano 2.9.3 script
read a
if [ `expr $a % 4` -ne 0 ]
then
echo normal year
else
if [ `expr $a % 100` -eq 0 -a `expr $a % 400` -ne 0 ]
then
echo normal year
else
echo leap year
fi
fi

```

Рисунок 33 – Скрипт для задания 16

```

asd@asd:~$ sh script
2000
leap year
asd@asd:~$ sh script
2001
normal year
asd@asd:~$

```

Рисунок 34 – Выполнение задания 16

### 17 Задание 17

Скрипт для задания 17 показан на рисунке 35, воспользуемся циклом `while`, который повторяет действия пока условие верно, условие заключено в квадратные скобки. Выполнение данного скрипта показано на рисунке 36.

```

read a
read b
read first
read second
while [ $first -gt $a ] && [ $first -gt $b ] && [ $second -gt $a ] && [ $second -gt $b ]
do
a=$(expr $a + 1)
b=$(expr $b + 1)
done
echo $a,$b

```

Рисунок 35 – Скрипт для задания 17

```

asd@asd:~$ sh script
2
3
10
15
9,10
asd@asd:~$ _

```

Рисунок 36 – Выполнение задания 17

## 18 Задание 18

Скрипт для задания 18 показан на рисунке 37. Выполнение данного скрипта показано на рисунке 38.

```
GNU nano 2.9.3
read a
if [ $a -eq $1 ]
then
ls -al /etc
else
echo error
fi
```

Рисунок 37 – Скрипт для задания 18

```
asd@asd:~$ sh script 1234
1234
total 800
drwxr-xr-x 91 root root      4096 Nov 27 16:37 .
drwxr-xr-x 24 root root      4096 Nov 27 16:32 ..
drwxr-xr-x  3 root root      4096 Aug  6 22:38 acpi
-rw-r--r--  1 root root     3028 Aug  6 22:35 adduser.conf
drwxr-xr-x  2 root root      4096 Aug  6 22:40 alternatives
drwxr-xr-x  3 root root      4096 Aug  6 22:38 apm
drwxr-xr-x  3 root root      4096 Aug  6 22:39 apparmor
drwxr-xr-x  9 root root      4096 Aug  6 22:40 apparmor.d
drwxr-xr-x  3 root root      4096 Nov 27 16:35 apport
drwxr-xr-x  7 root root      4096 Nov 27 16:31 apt
-rw-r----- 1 root daemon    144 Feb 20  2018 at.deny
-rw-r--r--  1 root root     2319 Apr  4  2018 bash.bashrc
-rw-r--r--  1 root root      45 Apr  2  2018 bash_completion
drwxr-xr-x  2 root root      4096 Nov 27 16:35 bash_completion.d
-rw-r--r--  1 root root     367 Jan 27  2016 bindresvport.blacklist
drwxr-xr-x  2 root root      4096 Apr 20  2018 binfmt.d
drwxr-xr-x  2 root root      4096 Aug  6 22:39 byobu
drwxr-xr-x  3 root root      4096 Aug  6 22:36 ca-certificates
-rw-r--r--  1 root root     6777 Nov 27 16:34 ca-certificates.conf
-rw-r--r--  1 root root     5986 Aug  6 22:37 ca-certificates.conf.dpkg-old
drwxr-xr-x  2 root root      4096 Aug  6 22:39 calendar
drwxr-xr-x  4 root root      4096 Nov 27 16:33 cloud
drwxr-xr-x  2 root root      4096 Aug  6 22:37 console-setup
drwxr-xr-x  2 root root      4096 Aug  6 22:39 cron.d
drwxr-xr-x  2 root root      4096 Nov 27 16:35 cron.daily
drwxr-xr-x  2 root root      4096 Aug  6 22:36 cron.hourly
drwxr-xr-x  2 root root      4096 Aug  6 22:36 cron.monthly
-rw-r--r--  1 root root      722 Nov 16  2017 crontab
drwxr-xr-x  2 root root      4096 Aug  6 22:40 cron.weekly
drwxr-xr-x  2 root root      4096 Aug  6 22:39 cryptsetup-initramfs
-rw-r--r--  1 root root      54 Aug  6 22:38 crypttab
drwxr-xr-x  4 root root      4096 Aug  6 22:36 dbus-1
-rw-r--r--  1 root root     2969 Feb 28  2018 debconf.conf
-rw-r--r--  1 root root      11 Jun 25  2017 debian_version
```

Рисунок 38 – Выполнение задания 18

## 19 Задание 19

Скрипт для задания 19 показан на рисунке 39. Выполнение данного скрипта показано на рисунке 40.

```
read a
if [ -e $a ]
then
cat $a
else
echo error
fi
```

Рисунок 39 – Скрипт для задания 19

```
asd@asd:~$ sh script
test.txt
5
3
2
6
1
2
3
6
9
7
asd@asd:~$
```

Рисунок 40 – Выполнение задания 19

## 20 Задание 20

Скрипт для задания 20 показан на рисунке 41, для проверки каталога существует ли он воспользуемся условием -e \$a, для проверки файла является ли он каталогом воспользуемся условием -d \$a, для проверки каталога на то можно ли его читать воспользуемся условием -r \$a. Выполнение данного скрипта показано на рисунке 42.

```
read a
if [ -e $a ]
then
if [ -d $a ]
then
if [ -r $a ]
then
ls $a
else
echo not available for reading
fi
else
cat $a
fi
else
mkdir $a
fi
```

Рисунок 41 – Скрипт для задания 20

```
asd@asd:~$ sh script
/home
asd
asd@asd:~$ sh script
test.txt
5
3
2
6
1
2
3
6
9
7
asd@asd:~$ sh script
/home/asd
home script test.txt tmp.txt
asd@asd:~$
```

Рисунок 42 – Выполнение задания 20

## 21 Задание 21

Скрипт для задания 21 показан на рисунках 43 и 45, для проверки файла существует ли он воспользуемся условием `-e $a`, для проверки файла можно ли в него записать воспользуемся условием `-w $b`. Выполнение данного скрипта показано на рисунках 44 и 46.



```
echo input file1
read a
echo input file2
read b
if [ -e $a -a -e $b ]
then
if [ -r $a ]
then
if [ -w $b ]
then
echo text file1:
cat $a
echo text file2:
cat $b
echo new text file2:
cat $a > $b
cat $b
else
echo file2 not avilable for recording
fi
else
echo file1 nont available for reading
fi
else
echo no files found
fi
```

Рисунок 43 – Скрипт для задания 21.1

```
asd@asd:~$ sh script
input file1
test.txt
input file2
tmop.txt
text file1:
5
3
2
6
1
2
3
6
9
7
text file2:
dfg
sdfsd
eryerue
5
sdfsefs
new text file2:
5
3
2
6
1
2
3
6
9
7
asd@asd:~$ _
```

Рисунок 44 – Выполнение задания 21.1

```

GNU nano 2.9.3
if [ -e $1 -a -e $2 ]
then
if [ -r $1 ]
then
if [ -w $2 ]
then
echo text file1:
cat $1
echo text file2:
cat $2
echo new text file2:
cat $1 > $2
cat $2
else
echo file2 not available for recording
fi
else
echo file1 nont available for reading
fi
else
echo no files found
fi

```

Рисунок 45 – Скрипт для задания 21.2

```

asd@asd:~$ sh script test.txt tmp.txt
text file1:
5
3
2
6
1
2
3
6
9
7
text file2:
dfsdf
sdf
hgfdhfg
dfgert
new text file2:
5
3
2
6
1
2
3
6
9
7
asd@asd:~$

```

Рисунок 46 – Выполнение задания 21.2

## 22 Задание 22

Скрипт для задания 22 показан на рисунке 47, для запуска программы воспользуемся командой `exec` и путем до файла. Выполнение данного скрипта показано на рисунке 48.

```
read a
if [ -e $a -a -x $a ]
then
exec $a
else
echo error
fi_
```

Рисунок 47 – Скрипт для задания 22

```
asd@asd:~$ ./script
./testscript
4
3
7
asd@asd:~$
```

Рисунок 48 – Выполнение задания 22

## 23 Задание 23

Скрипт для задания 23 показан на рисунке 49, для перенаправления вывода необходимо > указать на название необходимого файла. Выполнение данного скрипта показано на рисунке 50.

```
GNU nano 2.9.3
if [ -s $1 ]
then
sort -k1 $1 > 'tmpfile'
cat 'tmpfile'
else
echo the file is empty
fi
```

Рисунок 49 – Скрипт для задания 23

```

asd@asd:~$ sh script test.txt
1
2
2
3
3
5
6
6
7
9
asd@asd:~$ cat tmpfile
1
2
2
3
3
5
6
6
7
9
asd@asd:~$ _

```

Рисунок 50 – Выполнение задания 23

## 24 Задание 24

Скрипт для задания 24 показан на рисунке 51, для сборки всех текстовых файлов в один архивный файл воспользуемся командой `tar -cf 'tar.tar'` и передадим название файлов из текущего каталога, для просмотра данного файла используем команду `tar -tf 'tar.tar'`, для сжатия файла используется команда `gzip 'tar.tar'`. Выполнение данного скрипта показано на рисунке 52.

```

GNU nano 2.9.3
files=$(find . -maxdepth 1 -type f)
tar -cf 'tar.tar' $files
sleep 3
tar -tf 'tar.tar'
gzip 'tar.tar'

```

Рисунок 51 – Скрипт для задания 24

```

asd@asd:~$ sh script
script: 3: script: sleep: not found
./tmpfile
./tmop.txt
./sudo_as_admin_successful
./bashrc
./tmop.txt.swp
./scrite.swp
./scripy.swp
./script
./test.txt
./profile
./bash_logout
asd@asd:~$

```

Рисунок 52 – Выполнение задания 24

## 25 Задание 25

Скрипт для задания 25 показан на рисунке 53, для определения функции после ее названия ставятся скобки и в фигурных скобках находится тело функции т.е. то, что она будет выполнять. Выполнение данного скрипта показано на рисунке 54.

```

read a
read b
dif() {
echo $a - $b |bc
}
echo $(dif)

```

Рисунок 53 – Скрипт для задания 25

```

asd@asd:~$ sh script
5
6
-1
asd@asd:~$

```

Рисунок 54 – Выполнение задания 25