

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №5

По дисциплине «ОС Linux»

Программирование на SHELL. Использование командных файлов

Студент

Чаплыгин И.С.

Группа ПИ-18

Руководитель

Доцент

Кургасов В.В.

Липецк 2020г

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.
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.

Оглавление

Цель работы	2
Задание кафедры.....	3
Выполнение работы	9
1. Вывести информационные сообщения на экран.....	9
2. Присвоить переменной целочисленное значение. Просмотреть значение переменной.....	9
3. Присвоить переменной В значение переменной А. Просмотреть значение переменной В	9
4. Присвоить переменной значение “путь до своего каталога”. Перейти в этот каталог с использованием переменной	10
5. Присвоить переменной значение “имя команды”, а именно, команды DATE. Выполнить эту команду, используя значение переменной.....	10
6. Присвоить переменной значение “имя команды”, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной	11
7. Присвоить переменной значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной.....	12
8. Программа запрашивает значение переменной, а затем выводит значение этой переменной	12
9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной.....	13
10. Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) ВС	13
11. Вычислить объем цилиндра. Исходные данные запрашиваются	

программой. Результат выводится на экран.....	14
12. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки	15
13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается.....	15
14. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно.	16
15. Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения.....	17
16. Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран	18
17. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.....	19
18. В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc	20
19. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение	21
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	22
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).....	23
22. Если файл запуска программы найден, программа запускается (по выбору).....	24
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	25
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.....	27
25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных	28
Вывод.....	29

Выполнение работы

1. Вывести информационные сообщения на экран

Для вывода сообщения на экран, напишем в файле скрипта:

echo "Number1"

Где команда `echo` отвечает за вывод сообщения на экран, а `"Number1"` – само сообщение, которое выведет команда.

```
kocmonavtik@kocmos:~/lab5$ ./num1
Number1
kocmonavtik@kocmos:~/lab5$ _
```

Рисунок 1 – Вывод сообщения на экран

2. Присвоить переменной целочисленное значение. Просмотреть значение переменной

Код скрипта:

A=25 – Присваиваем переменной `A` значение 25.

echo \$A – Выводим значение переменной.

Знак `$` позволяет обратиться к переменной, которой присвоено значение.

```
kocmonavtik@kocmos:~/lab5$ ./num2
25
kocmonavtik@kocmos:~/lab5$ _
```

Рисунок 2 – Вывод значения переменной на экран

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

Код скрипта:

A=25 – Присваивание переменной `A` значения 25

B=\$A – Присваивание переменной `B` значения `A`

echo \$B – Вывод на экран значение переменной `B`

```
kocmonavtik@kocmos:~/lab5$ ./num3
25
kocmonavtik@kocmos:~/lab5$
```

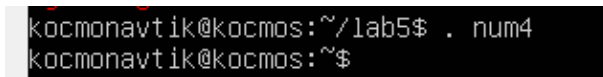
Рисунок 3 – Вывод значения переменной `B` на экран

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

Код скрипта:

C="/home/kosmonavtik" – Присваивание переменной пути до каталога в который необходимо перейти.

cd \$C – Команда перехода в каталог, который записан в переменной ***C***.



```
kosmonavtik@kocmos:~/lab5$ . num4
kosmonavtik@kocmos:~$
```

Рисунок 4 – Переход в другой каталог

Как можно заметить, использовалась немного другая команда выполнения скрипта, это связано с тем, что “./script” запускает свою версию оболочки, т.е. экземпляр оболочки, он переходит в другую директорию, но после завершения скрипта, происходит возврат в тот экземпляр shell, из которого его вызвали. А “*. script*” (альтернативный ввод - команда “source script”) позволяет выполнить команды, заложенные в скрипте в текущей оболочке.

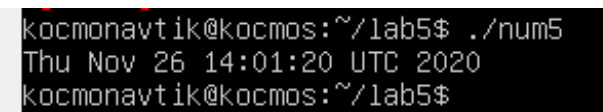
5. Присвоить переменной значение “имя команды”, а именно, команды **DATE**. Выполнить эту команду, используя значение переменной

Код скрипта:

D= "date" – присваивание переменной ***D*** имя команды “date”

echo `D` - Вывод выполнения команды, заложенной в переменную ***D***

Обратные кавычки `` позволяют присваивать переменным данные вывода системных команд. Символы, заключенные в обратные кавычки, воспринимаются интерпретатором shell как системная команда, которую необходимо выполнить.



```
kosmonavtik@kocmos:~/lab5$ ./num5
Thu Nov 26 14:01:20 UTC 2020
kosmonavtik@kocmos:~/lab5$
```

Рисунок 5 – Вывод даты на консоль

6. Присвоить переменной значение “имя команды”, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной.

Выполнить эту команду, используя значение переменной

Код скрипта:

E="ls -l" – присваивание переменной E имя команды “ls -l”

echo \$E – вывод на экран имени команды, которая присвоена переменной E

echo "`\$E`" | less – выполнение команды “ls -l” и вывод на экран

`\$E` в отличии от ***`\$E`*** позволяет вывести информацию в несколько строк, а дополнение к команде, в виде “|less” позволяет просматривать файл постранично

```
total 144
-rw-rw-r-- 1 kosmonavtik kosmonavtik 1948 Nov 25 11:33 ForNum23.txt
-rw-rw-r-- 1 kosmonavtik kosmonavtik 40 Nov 22 13:31 ForNum7.txt
-rw-rw-r-- 1 kosmonavtik kosmonavtik 1948 Nov 25 12:42 OutNum23.txt
-rw-rw-r-- 1 kosmonavtik kosmonavtik 768 Nov 25 15:43 my.tar.gz
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 15 Nov 14 07:44 num1
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 226 Nov 22 15:06 num10
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 220 Nov 22 15:33 num10Alt
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 131 Nov 22 15:54 num11
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 98 Nov 22 16:40 num12
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 77 Nov 22 16:57 num13
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 110 Nov 24 12:48 num14
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 184 Nov 24 13:26 num15
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 151 Nov 24 13:38 num16
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 145 Nov 24 13:39 num16Alt
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 341 Nov 24 14:09 num17
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 108 Nov 25 08:57 num18
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 82 Nov 25 09:34 num19
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 13 Nov 25 14:59 num2
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 97 Nov 25 09:50 num20
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 414 Nov 25 10:50 num21
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 128 Nov 25 11:09 num22
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 230 Nov 25 12:42 num23
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 68 Nov 25 15:43 num24
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 54 Nov 25 16:21 num25
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 20 Nov 14 07:45 num3
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 40 Nov 22 12:31 num4
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 31 Nov 26 14:01 num5
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 36 Nov 26 14:14 num6
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 50 Nov 24 11:53 num7
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 69 Nov 22 14:29 num7Alt
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 56 Nov 22 14:38 num8
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 69 Nov 22 14:48 num9
-rw-rw-r-- 1 kosmonavtik kosmonavtik 68 Nov 25 10:52 out.txt
-rw-rw-r-- 1 kosmonavtik kosmonavtik 40 Nov 23 19:10 output.txt
-rwxrwx-r-- 1 kosmonavtik kosmonavtik 47 Nov 25 09:31 test11
:
```

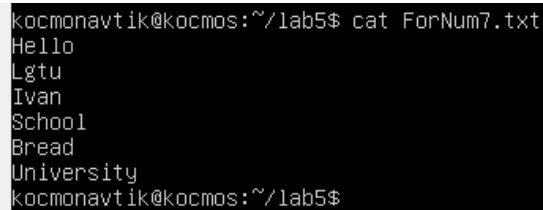
Рисунок 6 – Выполнение скрипта №6

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

Код скрипта:

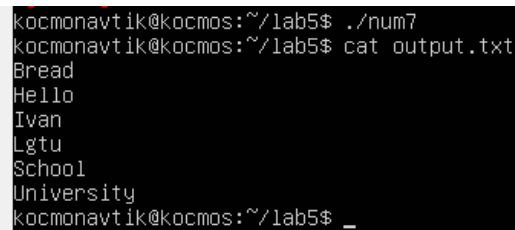
F="sort" – присваивание переменной F имя команды “sort”

`\$F ForNum7.txt >output.txt` - выполнение команды и перенаправление выполнения сортировка в файл



```
kocmonavtik@kocmos:~/lab5$ cat ForNum7.txt
Hello
Lgtu
Ivan
School
Bread
University
kocmonavtik@kocmos:~/lab5$
```

Рисунок 7 – Содержимое файла ForNum7.txt



```
kocmonavtik@kocmos:~/lab5$ ./num7
kocmonavtik@kocmos:~/lab5$ cat output.txt
Bread
Hello
Ivan
Lgtu
School
University
kocmonavtik@kocmos:~/lab5$ _
```

Рисунок 8 – Выполнение скрипта и содержимое файла output.txt

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

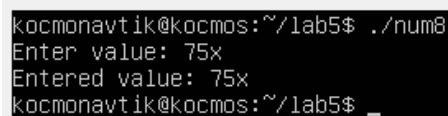
Код скрипта:

echo -n "Enter value: " – вывод обычного сообщения

read X – ожидание ввода значения X. После нажатия Enter, записанное значение присвоит переменная X.

Опция “-n” у команды “echo” позволяет после выполнения команды не переходить на новую строку.

echo "Entered value \$X" – Вывод сообщения с переменной, которую ввели.



```
kocmonavtik@kocmos:~/lab5$ ./num8
Enter value: 75x
Entered value: 75x
kocmonavtik@kocmos:~/lab5$ _
```

Рисунок 9 – Пример выполнения скрипта

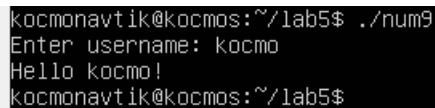
9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной.

Код скрипта:

echo -n "Enter username: "

read Name

echo "Hello \$Name!"



```
kocmonavtik@kocmos:~/lab5$ ./num9
Enter username: kocmo
Hello kocmo!
kocmonavtik@kocmos:~/lab5$
```

Рисунок 10 – Пример выполнения скрипта №9

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

а) Код скрипта:

echo -n "Enter first number: "

read X

echo -n "Enter second number: "

read Y

Z=`expr \$X + \$Y` - Присваивание переменной Z суммы введенных переменных X и Y

echo "Sum: \$Z" – Вывод сообщения с суммой чисел через значение переменной

echo "Difference: `expr \$X - \$Y`" – Нахождение разности и её вывод на консоль

echo "Multiplication: `expr \$X * \$Y`" – Произведение чисел

echo "Division: `expr \$X / \$Y`" – Деление

```
kocmonavtik@kocmos:~/lab5$ ./num10
Enter first number: 75
Enter second number: 36
Sum: 111
Difference: 39
Multiplication: 2700
Division: 2
kocmonavtik@kocmos:~/lab5$ _
```

Рисунок 11 – Вычисления с помощью “expr”

б) Код скрипта:

echo -n “Enter first number: ”

read X

echo -n “Enter second number: ”

read Y

echo “Sum: `bc <<< \$X+\$Y`” – нахождение суммы и её вывод на экран

echo “Difference: `bc <<< \$X-\$Y`” – разность

echo “Multiplication: `bc <<< \$X*\$Y`” – произведение

echo “Division: `bc <<< \$X/\$Y`” – Деление

```
kocmonavtik@kocmos:~/lab5$ ./num10Alt
Enter first number: 25
Enter second number: 12
Sum: 37
Difference: 13
Multiplication: 300
Division: 2
kocmonavtik@kocmos:~/lab5$ _
```

Рисунок 12 – Вычисления с помощью “bc”

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

Воспользуемся формулой : $V = \pi * r^2 * h$

echo -n “Cylinder height: ”

read H

echo -n “Cylinder radius: ”

read R

echo “Cylinder volume: `bc -l <<< 3.14*\$R^2*\$H`” – вычисления объёма и

вывод на экран.

```
kocmonavtik@kocmos:~/lab5$ ./num11
Cylinder height: 6
Cylinder radius: 12
Cylinder volume: 2712.96
kocmonavtik@kocmos:~/lab5$ _
```

Рисунок 13 – Вычисление объема цилиндра

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

Код скрипта:

echo “Name Program \$0” – вывод имени исполняемого скрипта

echo “Amount of elements: \$#” – число аргументов, передаваемых сценарию

echo “Argument value: \$*” – отображение всех аргументов, которые были

введены

```
kocmonavtik@kocmos:~/lab5$ ./num12 1 2 3 test text
Name Program ./num12
Amount of elements: 5
Argument value: 1 2 3 test text
kocmonavtik@kocmos:~/lab5$
```

Рисунок 14 – Пример работы скрипта №12

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

Код скрипта:

Result= “\$(cat \$1)” – присваивание переменной Result вывод исполнения команды “cat \$1”

echo “\${Result}” – вывод на экран значения переменной Result

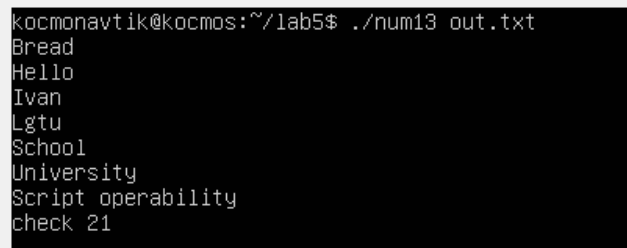
sleep 3 – Пауза при выполнении скрипта

clear – очищение экрана

Позиционные параметры – это аргументы командной строки или функции в скрипте, доступ к которым осуществляется по номеру. Возможно передавать произвольное число аргументов, но доступными являются только 9 из них. Имеют такой вид: \$1 \$2 \$3 \$4 \$5 \$6 \$7 \$8 \$9.

“\$()” является альтернативным вариантом обратных кавычек ``.

Вместо команды “echo \$Result” используется “echo “\${Result}”” для удобного вывода на экран.



```
kosmonavtik@kosmos:~/lab5$ ./num13 out.txt
Bread
Hello
Ivan
Lgtu
School
University
Script operability
check 21
```

Рисунок 15 – Пример работы скрипта №13

После паузы в 3 секунды, экран очищается.

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

Код скрипта:

for txtFiles in *.txt – циклично – поочередная запись в переменную txtFiles файлов, имеющих в конце названия текст “.txt”. От количества файлов зависит количество итераций цикла.

do – начало цикла

echo “Txt file : \$txtFiles” – Название файла, содержимое которого будет отображено на консоли

cat \$txtFiles /less – Отображение содержимого файла

echo “**”*** – Отделение от предыдущего цикла

done – конец цикла


```

kocmonavtik@kocmos:~/lab5$ ./num14
Txt file: ForNum23.txt
total 124
-rw-rw-r-- 1 kocmonavtik kocmonavtik 0 Nov 25 11:33 ForNum23
-rw-rw-r-- 1 kocmonavtik kocmonavtik 40 Nov 22 13:31 ForNum7.txt
-rwxrwxr-- 1 kocmonavtik kocmonavtik 15 Nov 14 07:44 num1
-rwxrwxr-- 1 kocmonavtik kocmonavtik 226 Nov 22 15:06 num10
-rwxrwxr-- 1 kocmonavtik kocmonavtik 220 Nov 22 15:33 num10Alt
-rwxrwxr-- 1 kocmonavtik kocmonavtik 131 Nov 22 15:54 num11
-rwxrwxr-- 1 kocmonavtik kocmonavtik 98 Nov 22 16:40 num12
-rwxrwxr-- 1 kocmonavtik kocmonavtik 77 Nov 22 16:57 num13
-rwxrwxr-- 1 kocmonavtik kocmonavtik 110 Nov 24 12:48 num14
-rwxrwxr-- 1 kocmonavtik kocmonavtik 184 Nov 24 13:26 num15
-rwxrwxr-- 1 kocmonavtik kocmonavtik 151 Nov 24 13:38 num16
-rwxrwxr-- 1 kocmonavtik kocmonavtik 145 Nov 24 13:39 num16Alt
-rwxrwxr-- 1 kocmonavtik kocmonavtik 341 Nov 24 14:09 num17
-rwxrwxr-- 1 kocmonavtik kocmonavtik 108 Nov 25 08:57 num18
-rwxrwxr-- 1 kocmonavtik kocmonavtik 82 Nov 25 09:34 num19
-rwxrwxr-- 1 kocmonavtik kocmonavtik 14 Nov 14 07:44 num2
-rwxrwxr-- 1 kocmonavtik kocmonavtik 97 Nov 25 09:50 num20
-rwxrwxr-- 1 kocmonavtik kocmonavtik 414 Nov 25 10:50 num21
-rwxrwxr-- 1 kocmonavtik kocmonavtik 128 Nov 25 11:09 num22
-rw-rw-r-- 1 kocmonavtik kocmonavtik 55 Nov 25 11:32 num23
-rwxrwxr-- 1 kocmonavtik kocmonavtik 20 Nov 14 07:45 num3
-rwxrwxr-- 1 kocmonavtik kocmonavtik 40 Nov 22 12:31 num4
-rwxrwxr-- 1 kocmonavtik kocmonavtik 32 Nov 24 16:01 num5
-rwxrwxr-- 1 kocmonavtik kocmonavtik 33 Nov 22 13:57 num6
-rwxrwxr-- 1 kocmonavtik kocmonavtik 50 Nov 24 11:53 num7
-rwxrwxr-- 1 kocmonavtik kocmonavtik 69 Nov 22 14:29 num7Alt
-rwxrwxr-- 1 kocmonavtik kocmonavtik 56 Nov 22 14:38 num8
-rwxrwxr-- 1 kocmonavtik kocmonavtik 69 Nov 22 14:48 num9
-rw-rw-r-- 1 kocmonavtik kocmonavtik 68 Nov 25 10:52 out.txt
-rw-rw-r-- 1 kocmonavtik kocmonavtik 40 Nov 23 19:10 output.txt
-rwxrwxr-- 1 kocmonavtik kocmonavtik 47 Nov 25 09:31 test1
-rw-rw-r-- 1 kocmonavtik kocmonavtik 28 Nov 25 10:52 testText
(END)

```

Рисунок 16 – Пример работы скрипта №14

После выхода из просмотра текстового файла, откроется следующий.

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

echo -n "Enter a number from 1 to 99: "

read Number

if [\$Number -le 99] && [\$Number -ge 1] – проверка условия

then – в случае положительной проверки

echo "\$Number – Allowed number"

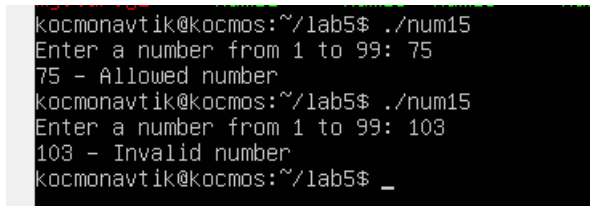
else – при отрицательном ответе проверки

echo "\$Number – Invalid number"

fi – окончание

опция “-le” у “if” означает “меньше или равно”, альтернативный вариант записи: “<=”. Опция “ge” означает “больше или равно”, альтернативный вариант записи: “>=”.

Знак “&&” позволяет создавать сложные условия проверки и означает логическое И. Возвращает “true” при справедливости всех условий.



```
kosmonavtik@kosmos:~/lab5$ ./num15
Enter a number from 1 to 99: 75
75 - Allowed number
kosmonavtik@kosmos:~/lab5$ ./num15
Enter a number from 1 to 99: 103
103 - Invalid number
kosmonavtik@kosmos:~/lab5$ _
```

Рисунок 17 – Пример работы скрипта №15

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

Результат выдается на экран

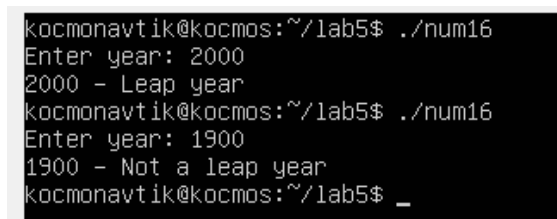
Код скрипта:

```
echo -n "Enter year: "
read Year
if [ `bc <<< $Year%4` == 0 ] – начало 1 условия
then
if [ `bc <<< $Year%100` == 0 ] – начало 2 условия
then
if [ `bc <<< $Year%400` == 0 ] – начало 3 условия
then
echo "$Year – Leap year"
else
echo "$Year – Not a leap year"
fi – конец 3 условия
else
echo "$Year – Leap year"
fi – конец 2 условия
else
echo "$Year – Not a leap year"
fi –конец первого условия
```

Знак “%” находит остаток от деления.

Чтобы определить, является ли год високосным, выполните следующие действия:

- 1) Если год делится на 4 без остатка, перейдите на шаг 2). В противном случае перейдите к выполнению действия 5).
- 2) Если год делится на 100 без остатка, перейдите на шаг 3). В противном случае перейдите к выполнению действия 4).
- 3) Если год делится на 400 без остатка, перейдите на шаг 4). В противном случае перейдите к выполнению действия 5).
- 4) Год високосный (366 дней).
- 5) Год не високосный год (365 дней).



```
kosmonavtik@kosmos:~/lab5$ ./num16
Enter year: 2000
2000 - Leap year
kosmonavtik@kosmos:~/lab5$ ./num16
Enter year: 1900
1900 - Not a leap year
kosmonavtik@kosmos:~/lab5$ _
```

Рисунок 18 – Пример выполнения скрипта №16

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

Код скрипта:

echo -n "Enter number1 from 1 to 9: "

read X

echo -n "Enter number2 from 1 to 9: "

read Y

while true – Создание бесконечного цикла

do

if [\$X -ge 1] && [\$X -le 9] && [\$Y -ge 1] && [\$Y -le 9]

then

echo "current numbers: \$X and \$Y"

X=\$((X+1)) – Увеличение значения переменной \$X на 1

Y=\$((Y+1)) - Увеличение значения переменной \$Y на 1

else

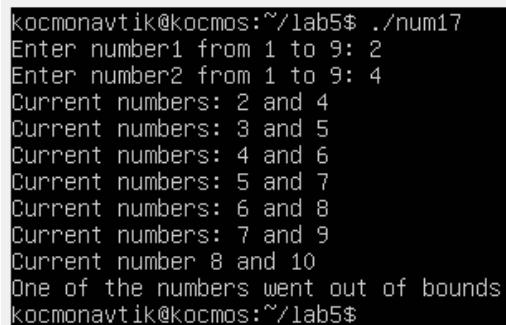
echo "Current number \$X and \$Y"

echo "One of the numbers went out of bounds"

break – команда прекращения работы цикла

fi

done



```
kocmonavtik@kocmos:~/lab5$ ./num17
Enter number1 from 1 to 9: 2
Enter number2 from 1 to 9: 4
Current numbers: 2 and 4
Current numbers: 3 and 5
Current numbers: 4 and 6
Current numbers: 5 and 7
Current numbers: 6 and 8
Current numbers: 7 and 9
Current number 8 and 10
One of the numbers went out of bounds
kocmonavtik@kocmos:~/lab5$
```

Рисунок 19 – Пример выполнения скрипта №17

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

Установим пароль в скрипте “TestPass”

Код скрипта:

if [\$1 = "TestPass"]

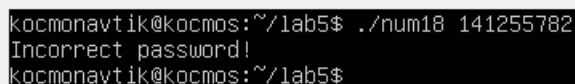
then

ls -al /etc /less – команда просмотра файлов, в том числе скрытных, каталога /etc и постраничный просмотр

else

echo "Incorrect password!"

fi



```
kocmonavtik@kocmos:~/lab5$ ./num18 141255782
Incorrect password!
kocmonavtik@kocmos:~/lab5$
```

Рисунок 20 – Вывод сообщения в случае неправильного пароля

```
total 824
drwxr-xr-x 96 root root      4096 Nov 22 13:06 .
drwxr-xr-x 20 root root      4096 Nov  9 16:50 ..
-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  3 07:18 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 Nov  9 14:28 alternatives
drwxr-xr-x 8 root root      4096 Nov  9 14:28 apache2
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 Nov 22 13:06 appport
drwxr-xr-x 7 root root      4096 Oct  3 07:08 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     2319 Oct  7 09:17 bash.bashrc.backup
-rw-r--r-- 1 root root        45 Jan 26 2020 bash_completion
drwxr-xr-x 2 root root      4096 Nov 22 13:06 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     6505 Nov  8 15:59 ca-certificates.conf
-rw-r--r-- 1 root root     5714 Jul 31 16:29 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  3 07:15 cloud
drwxr-xr-x 2 root root      4096 Oct  3 07:20 console-setup
drwxr-xr-x 2 root root      4096 Nov  9 14:28 cron.d
drwxr-xr-x 2 root root      4096 Nov 22 13:06 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  3 07:17 cryptsetup-initramfs
-rw-r--r-- 1 root root        54 Jul 31 16:29 crypttab
:~
```

Рисунок 21 – Пример работы скрипта №18

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

Код скрипта:

```
if [ -e "$1" ]
then cat $1 |less
else
echo "File does not exist!"
fi
```

Опция “-e” у “if” проверяет существование файла.

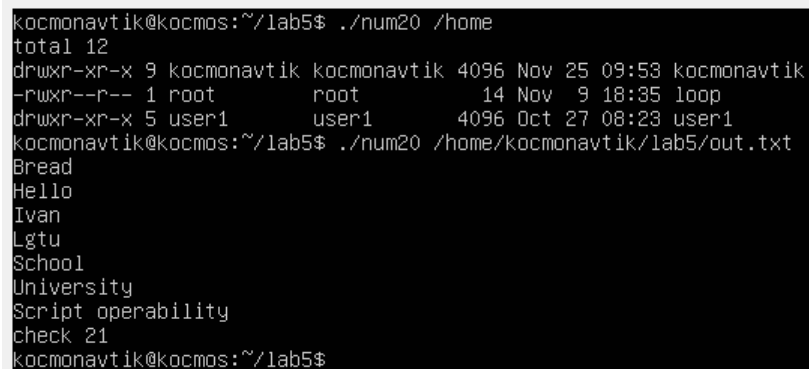
```
kocmonavtik@kocmos:~/lab5$ ./num19
File does not exist!
kocmonavtik@kocmos:~/lab5$ ./num19 ergirh
File does not exist!
kocmonavtik@kocmos:~/lab5$ ./num19 out.txt
Bread
Hello
Ivan
Lgtu
School
University
Script operability
check 21
(END)
```

Рисунок 22 – Пример работы скрипта №19

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

Код скрипта:

```
if [ -e $1 ] – Проверка на существование файла  
then  
if [ -r $1 ] – Проверка на права чтения файла  
then  
if [ -d $1 ] – Проверка, является ли файл каталогом  
then  
ls -l $1  
else  
cat $1 |less  
fi  
else  
echo “No permission to read the file”  
fi  
else  
mkdir $1  
fi
```



```
kocmonavtik@kocmos:~/lab5$ ./num20 /home  
total 12  
drwxr-xr-x 9 kocmonavtik kocmonavtik 4096 Nov 25 09:53 kocmonavtik  
-rwxr--r-- 1 root root 14 Nov 9 18:35 loop  
drwxr-xr-x 5 user1 user1 4096 Oct 27 08:23 user1  
kocmonavtik@kocmos:~/lab5$ ./num20 /home/kocmonavtik/lab5/out.txt  
Bread  
Hello  
Ivan  
Lgtu  
School  
University  
Script operability  
check 21  
kocmonavtik@kocmos:~/lab5$
```

Рисунок 23 – Пример работы скрипта №20

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)

Код скрипта:

if [-e \$1] && [-f \$1] – проверка первого файла на существование и является ли файл обычным

then

if [-r \$1] – проверка первого файла на право чтения текущим пользователем

then

if [-e \$2] && [-f \$2] – проверка второго файла на существование и является ли файл обычным

then

if [-w \$2] – проверка второго файла на право записи текущим пользователем

then

cat \$1 >>\$2

else

echo "User does not have write permission to \$2 file"

fi

else

echo "\$2 file does not exist or is not a regular file"

fi

else

echo "User does not have read permission to \$1 file"

fi

else

echo "\$1 file does not exist or is not a regular file"

fi

```
kocmonavtik@kocmos:~/lab5$ ls
ForNum23.txt  num1      num12     num16     num19     num22     num3      num7      out.txt
ForNum7.txt   num10     num13     num16Alt  num2      num23     num4      num7Alt   output.txt
OutNum23.txt  num10Alt  num14     num17     num20     num24     num5      num8      test1
my.tar.gz     num11     num15     num18     num21     num25     num6      num9      testText
kocmonavtik@kocmos:~/lab5$ ./num21 output.txt out.txt
kocmonavtik@kocmos:~/lab5$ cat output.txt
Bread
Hello
Ivan
Lgtu
School
University
kocmonavtik@kocmos:~/lab5$ cat out.txt
Bread
Hello
Ivan
Lgtu
School
University
Script operability
check 21
Bread
Hello
Ivan
Lgtu
School
University
kocmonavtik@kocmos:~/lab5$ _
```

Рисунок 24 – Пример выполнения скрипта №21

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

Код скрипта:

if [-f \$1] \$\$ *[-x \$1]* – проверка, является ли файл обычным и доступен ли файл для исполнения

then

. \$1 \$2 \$3 \$4 \$5 \$6 \$7 \$8 \$9 – запуск исполняемого файла с позиционными параметрами, которые указали при запуске основного скрипта.

else

echo "File does not exist or is not executable"

fi

```
kocmonavtik@kocmos:~/lab5$ ./num22 num1
Number1
kocmonavtik@kocmos:~/lab5$ ./num22 num12 7 2 6 8 test
Name Program ./num22
Amount of elements: 5
Argument value: 7 2 6 8 test
kocmonavtik@kocmos:~/lab5$
```

Рисунок 25 – Пример работы скрипта №22

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

Код скрипта:

if [-e \$1] && [-f \$1] - проверка на существование файла и является ли он обычным

then

if [\$(stat -c %s \$1) -gt 1] – проверка размера файла.

then

sort -k1 \$1 >OutNum23.txt – сортировка по 1 столбцу перенаправление вывода в файл.

cat OutNum23.txt |less

else

echo "\$1 is empty"

fi

else

echo "\$1 does not exist or is not a text file"

fi

```

kocmonavtik@kocmos:~/lab5$ cat ForNum23.txt
total 124
-rw-rw-r-- 1 kocmonavtik kocmonavtik 0 Nov 25 11:33 ForNum23
-rw-rw-r-- 1 kocmonavtik kocmonavtik 40 Nov 22 13:31 ForNum7.txt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 15 Nov 14 07:44 num1
-rwxrw-r-- 1 kocmonavtik kocmonavtik 226 Nov 22 15:06 num10
-rwxrw-r-- 1 kocmonavtik kocmonavtik 220 Nov 22 15:33 num10Alt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 131 Nov 22 15:54 num11
-rwxrw-r-- 1 kocmonavtik kocmonavtik 98 Nov 22 16:40 num12
-rwxrw-r-- 1 kocmonavtik kocmonavtik 77 Nov 22 16:57 num13
-rwxrw-r-- 1 kocmonavtik kocmonavtik 110 Nov 24 12:48 num14
-rwxrw-r-- 1 kocmonavtik kocmonavtik 184 Nov 24 13:26 num15
-rwxrw-r-- 1 kocmonavtik kocmonavtik 151 Nov 24 13:38 num16
-rwxrw-r-- 1 kocmonavtik kocmonavtik 145 Nov 24 13:39 num16Alt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 341 Nov 24 14:09 num17
-rwxrw-r-- 1 kocmonavtik kocmonavtik 108 Nov 25 08:57 num18
-rwxrw-r-- 1 kocmonavtik kocmonavtik 82 Nov 25 09:34 num19
-rwxrw-r-- 1 kocmonavtik kocmonavtik 14 Nov 14 07:44 num2
-rwxrw-r-- 1 kocmonavtik kocmonavtik 97 Nov 25 09:50 num20
-rwxrw-r-- 1 kocmonavtik kocmonavtik 414 Nov 25 10:50 num21
-rwxrw-r-- 1 kocmonavtik kocmonavtik 128 Nov 25 11:09 num22
-rw-rw-r-- 1 kocmonavtik kocmonavtik 55 Nov 25 11:32 num23
-rwxrw-r-- 1 kocmonavtik kocmonavtik 20 Nov 14 07:45 num3
-rwxrw-r-- 1 kocmonavtik kocmonavtik 40 Nov 22 12:31 num4
-rwxrw-r-- 1 kocmonavtik kocmonavtik 32 Nov 24 16:01 num5
-rwxrw-r-- 1 kocmonavtik kocmonavtik 33 Nov 22 13:57 num6
-rwxrw-r-- 1 kocmonavtik kocmonavtik 50 Nov 24 11:53 num7
-rwxrw-r-- 1 kocmonavtik kocmonavtik 69 Nov 22 14:29 num7Alt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 56 Nov 22 14:38 num8
-rw-rw-r-- 1 kocmonavtik kocmonavtik 69 Nov 22 14:48 num9
-rw-rw-r-- 1 kocmonavtik kocmonavtik 68 Nov 25 10:52 out.txt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 40 Nov 23 19:10 output.txt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 47 Nov 25 09:31 test1
-rw-rw-r-- 1 kocmonavtik kocmonavtik 28 Nov 25 10:52 testText
kocmonavtik@kocmos:~/lab5$

```

Рисунок 26 – Файл для сортировки

```

kocmonavtik@kocmos:~/lab5$ ./num23 ForNum23.txt
-rw-rw-r-- 1 kocmonavtik kocmonavtik 0 Nov 25 11:33 ForNum23
-rw-rw-r-- 1 kocmonavtik kocmonavtik 28 Nov 25 10:52 testText
-rw-rw-r-- 1 kocmonavtik kocmonavtik 40 Nov 22 13:31 ForNum7.txt
-rw-rw-r-- 1 kocmonavtik kocmonavtik 40 Nov 23 19:10 output.txt
-rw-rw-r-- 1 kocmonavtik kocmonavtik 55 Nov 25 11:32 num23
-rw-rw-r-- 1 kocmonavtik kocmonavtik 68 Nov 25 10:52 out.txt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 14 Nov 14 07:44 num2
-rwxrw-r-- 1 kocmonavtik kocmonavtik 15 Nov 14 07:44 num1
-rwxrw-r-- 1 kocmonavtik kocmonavtik 20 Nov 14 07:45 num3
-rwxrw-r-- 1 kocmonavtik kocmonavtik 32 Nov 24 16:01 num5
-rwxrw-r-- 1 kocmonavtik kocmonavtik 33 Nov 22 13:57 num6
-rwxrw-r-- 1 kocmonavtik kocmonavtik 40 Nov 22 12:31 num4
-rwxrw-r-- 1 kocmonavtik kocmonavtik 47 Nov 25 09:31 test1
-rwxrw-r-- 1 kocmonavtik kocmonavtik 50 Nov 24 11:53 num7
-rwxrw-r-- 1 kocmonavtik kocmonavtik 56 Nov 22 14:38 num8
-rwxrw-r-- 1 kocmonavtik kocmonavtik 69 Nov 22 14:29 num7Alt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 69 Nov 22 14:48 num9
-rwxrw-r-- 1 kocmonavtik kocmonavtik 77 Nov 22 16:57 num13
-rwxrw-r-- 1 kocmonavtik kocmonavtik 82 Nov 25 09:34 num19
-rwxrw-r-- 1 kocmonavtik kocmonavtik 97 Nov 25 09:50 num20
-rwxrw-r-- 1 kocmonavtik kocmonavtik 98 Nov 22 16:40 num12
-rwxrw-r-- 1 kocmonavtik kocmonavtik 108 Nov 25 08:57 num18
-rwxrw-r-- 1 kocmonavtik kocmonavtik 110 Nov 24 12:48 num14
-rwxrw-r-- 1 kocmonavtik kocmonavtik 128 Nov 25 11:09 num22
-rwxrw-r-- 1 kocmonavtik kocmonavtik 131 Nov 22 15:54 num11
-rwxrw-r-- 1 kocmonavtik kocmonavtik 145 Nov 24 13:39 num16Alt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 151 Nov 24 13:38 num16
-rwxrw-r-- 1 kocmonavtik kocmonavtik 184 Nov 24 13:26 num15
-rwxrw-r-- 1 kocmonavtik kocmonavtik 220 Nov 22 15:33 num10Alt
-rwxrw-r-- 1 kocmonavtik kocmonavtik 226 Nov 22 15:06 num10
-rwxrw-r-- 1 kocmonavtik kocmonavtik 341 Nov 24 14:09 num17
-rwxrw-r-- 1 kocmonavtik kocmonavtik 414 Nov 25 10:50 num21
total 124
(END)

```

Рисунок 27 – Пример работы скрипта №23

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

Код скрипта:

`tar -cf my.tar *.txt` – архивирование всех текстовых файлов

`sleep 2` – пауза в 2 секунды

`tar -tf my.tar` – просмотр содержимого архива tar

`gzip my.tar` – посредством GZIP сжатия архива tar

```
kocmonavtik@kocmos:~/lab5$ ls
ForNum23.txt  num10  num13  num16Alt  num2  num23  num4  num7Alt  output.txt
ForNum7.txt   num10Alt num14  num17     num20 num24  num5  num8     test1
OutNum23.txt  num11   num15  num18     num21 num25  num6  num9     testText
num1          num12   num16  num19     num22 num3    num7  out.txt
kocmonavtik@kocmos:~/lab5$ ./num24
ForNum23.txt
ForNum7.txt
OutNum23.txt
out.txt
output.txt
kocmonavtik@kocmos:~/lab5$ ls
ForNum23.txt  num1  num12  num16  num19  num22  num3  num7  out.txt
ForNum7.txt   num10  num13  num16Alt  num2  num23  num4  num7Alt  output.txt
OutNum23.txt  num10Alt  num14  num17     num20  num24  num5  num8     test1
my.tar.gz     num11   num15  num18     num21  num25  num6  num9     testText
kocmonavtik@kocmos:~/lab5$
```

Рисунок 28 – Пример работы скрипта №24

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

Код скрипта:

Summ() – инициализация функции с названием ***Summ***

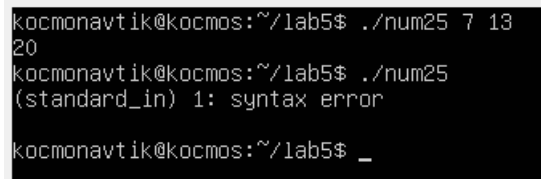
{ -начало функции

echo `bc <<< \$1+\$2`

} -конец функции

Summ \$1 \$2 – обращение к функции и передача аргументов

Так же, стоит учесть, что в функции используются локальные позиционные параметры, которые не связаны с программой.



```
kocmonavtik@kocmos:~/lab5$ ./num25 7 13
20
kocmonavtik@kocmos:~/lab5$ ./num25
(standard_in) 1: syntax error
kocmonavtik@kocmos:~/lab5$ _
```

Рисунок 29 – Пример работы скрипта №25

Вывод

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