

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Отчет по Лабораторной работе №4  
по курсу “ОС Linux”

Студент

Группа ПИ-21-1

\_\_\_\_\_

подпись, дата

Красиков И.А.

Руководитель

\_\_\_\_\_

подпись, дата

Кургасов В.В.

Липецк 2023 г.

## Оглавление

Цель работы .....	3
Задание.....	4
Ход работы .....	6
I часть.....	6
II часть .....	9
Вывод.....	33
Ответы на контрольные вопросы .....	34

## **Цель работы**

- изучить основные возможности языка программирования высокого уровня Shell;
- получить навыки написания и использования скриптов.

## Задание

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 PATE. 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:

1. Программа запрашивает значение переменной, а затем выводит значение этой переменной.
2. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной.
3. Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды: а) EXPR; б) BC).
4. Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран.
5. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки.

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

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

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

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

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

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

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

13. 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.

14. 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 имена файлов и/или позиционные параметры).

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

16. 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.

Для сравнения с другими языками программирования сделайте аналогичные действия на Java, Си и Python.

## Ход работы

### I часть

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

Для начала нужно создать файл для этого используем команду vim.

```
ivan_lr@intarolr:~$ mkdir LR4
ivan_lr@intarolr:~$ cd LR4
ivan_lr@intarolr:~/LR4$ vim print.sh_
```

Рисунок 1.1 – Создание каталога и файла для скрипта

После чего можем написать скрипт.

```
ivan_lr@intarolr: ~/LR4
#!/bin/sh
echo Hello
printf "World\n"
```

Рисунок 1.2 – Написание скрипта

Для того, чтобы запустить файл как скрипт, нужно выдать ему формат исполняемого файла.

```
ivan_lr@intarolr:~/LR4$ chmod +x print.sh
ivan_lr@intarolr:~/LR4$ ls -l
total 4
-rwxrwxr-x 1 ivan_lr ivan_lr 36 ноя  6 14:29 print.sh
```

Рисунок 1.3 – Задание файлу формата исполняемого файла

После чего можем запустить скрипт.

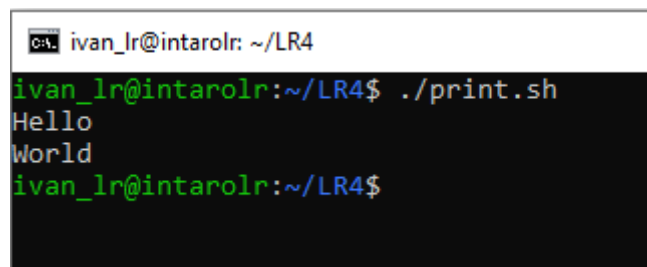
A terminal window with a title bar showing 'ivan\_lr@intarolr: ~/LR4'. The prompt is 'ivan\_lr@intarolr:~/LR4\$'. The user enters './print.sh', and the script outputs 'Hello' and 'World' on separate lines. The prompt returns to 'ivan\_lr@intarolr:~/LR4\$'.

Рисунок 1.4 – Запуск скрипта

2) Присвоить переменной А целочисленное значение. Просмотреть значение переменной А.

Также можно писать на языке Shell не в файле, а в командной строке.

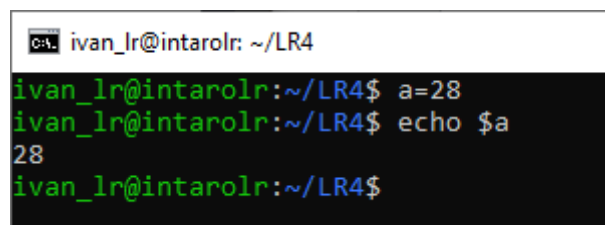
A terminal window with a title bar showing 'ivan\_lr@intarolr: ~/LR4'. The prompt is 'ivan\_lr@intarolr:~/LR4\$'. The user enters 'a=28'. The prompt returns. The user enters 'echo \$a', and the terminal outputs '28'. The prompt returns to 'ivan\_lr@intarolr:~/LR4\$'.

Рисунок 2 – Создание переменной, а также присвоение ей значение и вывод значение на экран

3) Присвоить переменной В значение переменной А. Просмотреть значение переменной В.

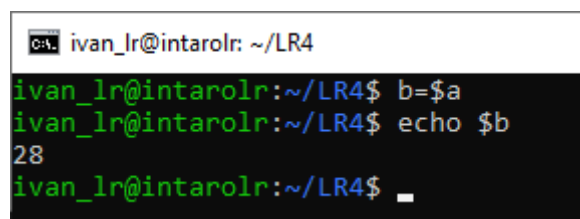
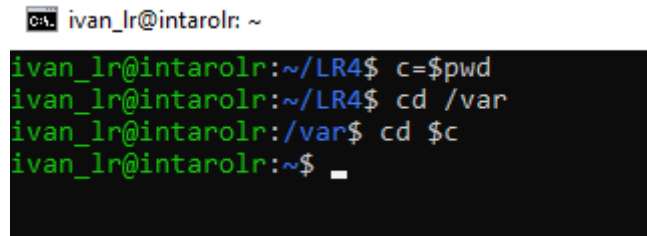
A terminal window with a title bar showing 'ivan\_lr@intarolr: ~/LR4'. The prompt is 'ivan\_lr@intarolr:~/LR4\$'. The user enters 'b=\$a'. The prompt returns. The user enters 'echo \$b', and the terminal outputs '28'. The prompt returns to 'ivan\_lr@intarolr:~/LR4\$'.

Рисунок 3 – Присвоение значения переменной В и вывод

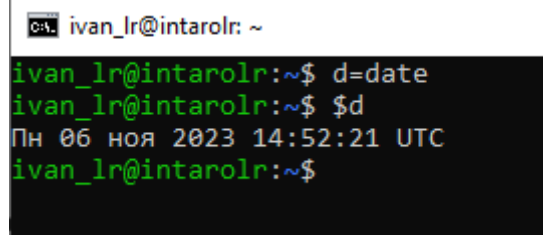
4) Присвоить переменной С значение "путь до своего каталога". Перейти в этот каталог с использованием переменной.



```
ivan_lr@intarolr: ~  
ivan_lr@intarolr:~/LR4$ c=$pwd  
ivan_lr@intarolr:~/LR4$ cd /var  
ivan_lr@intarolr:/var$ cd $c  
ivan_lr@intarolr:~$
```

Рисунок 4 – Присвоение значения переменной С и вывод

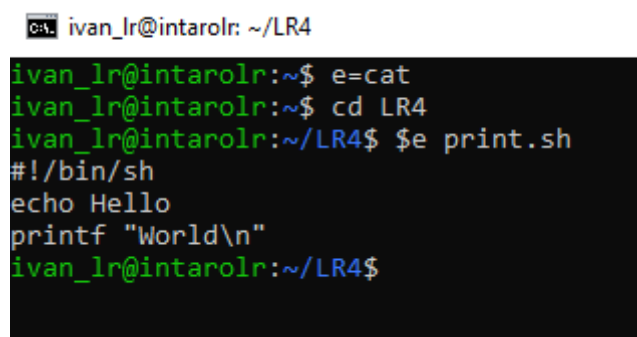
5) Присвоить переменной D значение "имя команды", а именно, команды DATE. Выполнить эту команду, используя значение переменной.



```
ivan_lr@intarolr: ~  
ivan_lr@intarolr:~$ d=date  
ivan_lr@intarolr:~$ $d  
Пн 06 ноя 2023 14:52:21 UTC  
ivan_lr@intarolr:~$
```

Рисунок 5 – Присвоение значения переменной D и вывод

6) Присвоить переменной E значение "имя команды", а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной.



```
ivan_lr@intarolr: ~/LR4  
ivan_lr@intarolr:~$ e=cat  
ivan_lr@intarolr:~$ cd LR4  
ivan_lr@intarolr:~/LR4$ $e print.sh  
#!/bin/sh  
echo Hello  
printf "World\n"  
ivan_lr@intarolr:~/LR4$
```

Рисунок 6 – Присвоение значения переменной E и вывод

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



```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ printf "c\nb\nd\na\n" > file.txt
ivan_lr@intarolr:~/LR4$ cat file.txt
c
b
d
a
ivan_lr@intarolr:~/LR4$ f=sort
ivan_lr@intarolr:~/LR4$ $f file.txt
a
b
c
d
ivan_lr@intarolr:~/LR4$
```

Рисунок 7 – Создание файла, вывод содержимого файла, присвоение команды переменной F, сортировка файла.

## II часть

1) Программа запрашивает значение переменной, а затем выводит значение этой переменной.

```
ivan_lr@intarolr: ~/LR4
#!/bin/sh
printf "Variable: "
read a=
printf "\n"
printf "Variable = "
echo $a
~
~
~
~
~
```

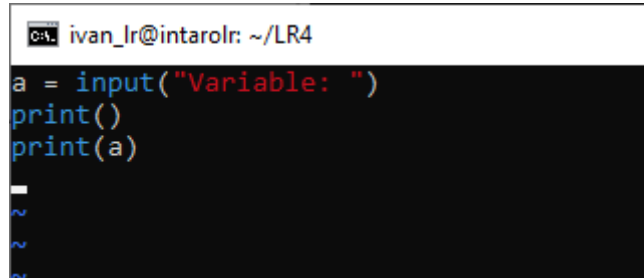
Рисунок 8.1 – Код скрипта input.sh

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./input.sh
Variable: Hello, world!!!

Variable = Hello, world!!!
ivan_lr@intarolr:~/LR4$
```

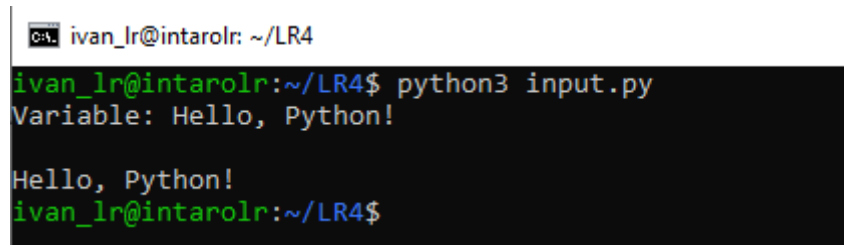
Рисунок 8.2 – Запуск скрипта input.sh

Для сравнения код на Python.

A terminal window with a black background and a title bar showing 'ivan\_lr@intarolr: ~/LR4'. The code is written in Python and uses color syntax highlighting: 'a = input("Variable: ")' in blue, 'print()' in green, and 'print(a)' in blue. There are three tilde (~) characters below the code, indicating the prompt.

```
ivan_lr@intarolr: ~/LR4
a = input("Variable: ")
print()
print(a)
~
~
~
```

Рисунок 8.3 – Код на Python

A terminal window with a black background and a title bar showing 'ivan\_lr@intarolr: ~/LR4'. It shows the execution of a Python script. The prompt is green, the command 'python3 input.py' is green, and the output 'Variable: Hello, Python!' and 'Hello, Python!' are in white. The prompt returns to 'ivan\_lr@intarolr:~/LR4\$' in green.

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 input.py
Variable: Hello, Python!
Hello, Python!
ivan_lr@intarolr:~/LR4$
```

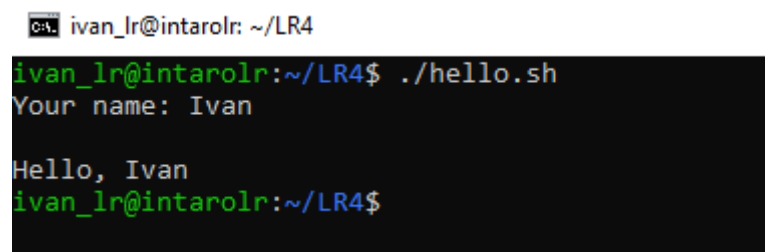
Рисунок 8.4 – Запуск кода на Python

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

A terminal window with a black background and a title bar showing 'ivan\_lr@intarolr: ~/LR4'. The code is a shell script using 'printf' and 'read' commands, with color syntax highlighting: '#!/bin/sh' in blue, 'printf' in blue, 'read name=' in blue, 'printf' in blue, 'echo \$name' in blue, and 'name' in purple. There are three tilde (~) characters below the code.

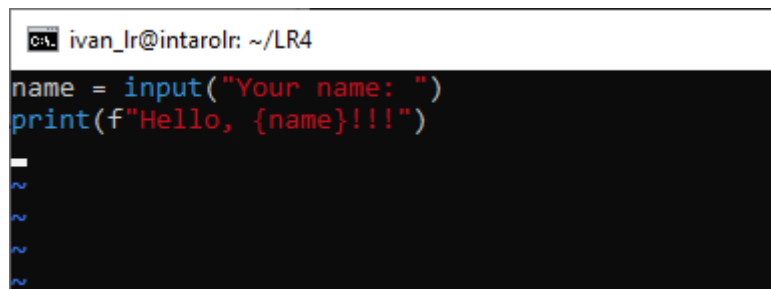
```
ivan_lr@intarolr: ~/LR4
#!/bin/sh
printf "Your name: "
read name=
printf "\n"
printf "Hello, "
echo $name
~
~
~
```

Рисунок 9.1 – Код скрипта hello.sh

A terminal window with a black background and a title bar showing 'ivan\_lr@intarolr: ~/LR4'. It shows the execution of the 'hello.sh' script. The prompt is green, the command './hello.sh' is green, and the output 'Your name: Ivan' and 'Hello, Ivan' are in white. The prompt returns to 'ivan\_lr@intarolr:~/LR4\$' in green.

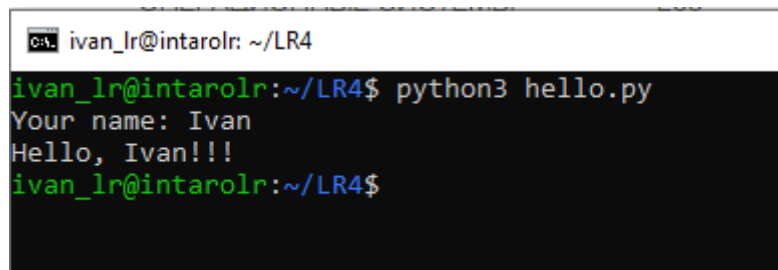
```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./hello.sh
Your name: Ivan
Hello, Ivan
ivan_lr@intarolr:~/LR4$
```

Рисунок 9.2 – Запуск скрипта hello.sh



```
ivan_lr@intarolr: ~/LR4
name = input("Your name: ")
print(f"Hello, {name}!!!")
~
~
~
```

Рисунок 9.3 – Код на Python (hello.py)

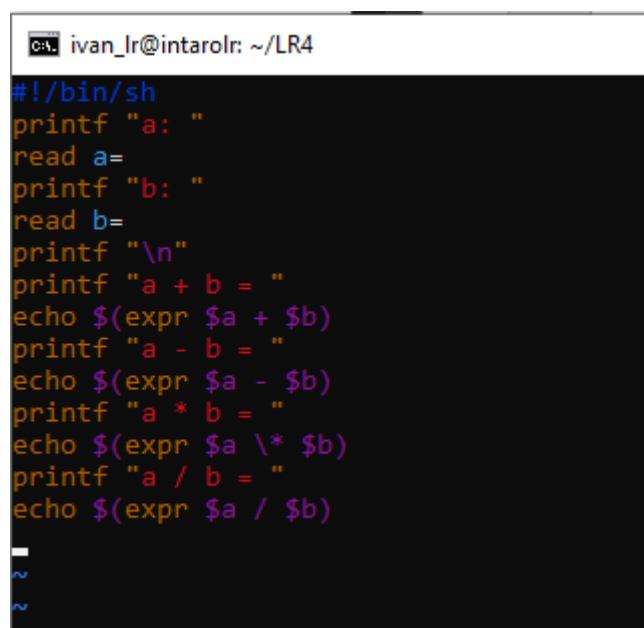


```
ivan_lr@intarolr: ~/LR4$ python3 hello.py
Your name: Ivan
Hello, Ivan!!!
ivan_lr@intarolr:~/LR4$
```

Рисунок 9.4 – Запуск кода на Python (hello.py)

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

expr – базовый целочисленный калькулятор. Поддерживает операции сложения/деления/умножения/вычитания



```
ivan_lr@intarolr: ~/LR4
#!/bin/sh
printf "a: "
read a=
printf "b: "
read b=
printf "\n"
printf "a + b = "
echo $(expr $a + $b)
printf "a - b = "
echo $(expr $a - $b)
printf "a * b = "
echo $(expr $a \* $b)
printf "a / b = "
echo $(expr $a / $b)
~
~
```

Рисунок 10.1 – Код скрипта expr.sh

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./expr.sh
a: 8
b: 5

a + b = 13
a - b = 3
a * b = 40
a / b = 1
ivan_lr@intarolr:~/LR4$ _
```

Рисунок 10.2 – Запуск скрипта expr.sh

bc – интерактивный интерпретатор Си-подобного языка, позволяет выполнять вычисления с произвольно заданной точностью.

```
ivan_lr@intarolr: ~/LR4
#!/bin/sh
printf "a: "
read a=
printf "b: "
read b=
printf "\n"
printf "a + b = "
echo "$a + $b" | bc
printf "a - b = "
echo "$a - $b" | bc
printf "a * b = "
echo "$a * $b" | bc
printf "a / b = "
echo "$a / $b" | bc_
~
```

Рисунок 10.3 – Код скрипта bc.sh

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./bc.sh
a: 10
b: 3

a + b = 13
a - b = 7
a * b = 30
a / b = 3
ivan_lr@intarolr:~/LR4$
```

Рисунок 10.4 – Запуск скрипта bc.sh

Программа на Python:

```
ivan_lr@intarolr: ~/LR4
a = int(input("a: "))
b = int(input("b: "))
print()
print(f"a + b = {a + b}")
print(f"a - b = {a - b}")
print(f"a * b = {a * b}")
print(f"a / b = {a // b}")
_
~
~
~
~
```

Рисунок 10.5 – Код на Python (calc.py)

```
ivan_lr@intarolr: ~/LR4$ vim calc.py
ivan_lr@intarolr:~/LR4$ python3 calc.py
a: 9
b: 4

a + b = 13
a - b = 5
a * b = 36
a / b = 2
ivan_lr@intarolr:~/LR4$ _
```

Рисунок 10.6 – Запуск кода на Python (calc.py)

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

```
ivan_lr@intarolr: ~/LR4
#!/bin/sh
pi=3.14
printf "Radius: "
read radius=
printf "Height: "
read height=
printf "\n"
res=`echo "scale=3; $pi * $height * $radius * $radius" | bc`
printf "Volume = "
echo $res
```

Рисунок 11.1 – Код скрипта vol.sh

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ chmod +x vol.sh
ivan_lr@intarolr:~/LR4$ ./vol.sh
Radius: 4
Height: 3

Volume = 150.72
ivan_lr@intarolr:~/LR4$
```

Рисунок 11.2 – Запуск скрипта vol.sh

Код на Python:

```
ivan_lr@intarolr: ~/LR4
pi = 3.14
r = int(input("Radius: "))
h = int(input("Height: "))

print(f"Volume = {round(r*r*h*pi, 3)}")
```

Рисунок 11.3 – Код на Python (vol.py)

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 vol.py
Radius: 3
Height: 2
Volume = 56.52
ivan_lr@intarolr:~/LR4$
```

Рисунок 11.4 – Запуск программы на Python

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

```
#!/bin/sh
echo "Prorgamm name: $0"
echo "Arg count: $#"
```

```
for i in $@
do
    echo "$i"
done
```

```
~
```

Рисунок 12.1 – Код скрипта arg.sh

```
ivan_lr@intarolr:~/LR4$ chmod +x arg.sh
ivan_lr@intarolr:~/LR4$ ./arg.sh a b c d
Prorgamm name: ./arg.sh
Arg count: 4
a
b
c
d
ivan_lr@intarolr:~/LR4$
```

Рисунок 12.2 – Запуск скрипта arg.sh

Код на Python для сравнения:

```
ivan_lr@intarolr: ~/LR4
```

```
import sys
print(f"Programm name: {sys.argv[0]}")
print(f"Arg count: {len(sys.argv)}")
for i in range(1, len(sys.argv)):
    print(sys.argv[i])
```

```
~
```

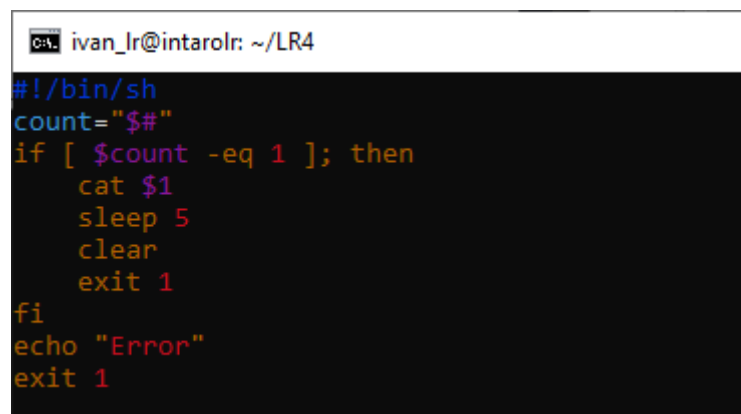
Рисунок 12.3 – Код программы на Python (arg.py)

```
ivan_lr@intarolr: ~/LR4
```

```
ivan_lr@intarolr:~/LR4$ python3 arg.py q w e r t y
Programm name: arg.py
Arg count: 7
q
w
e
r
t
y
ivan_lr@intarolr:~/LR4$
```

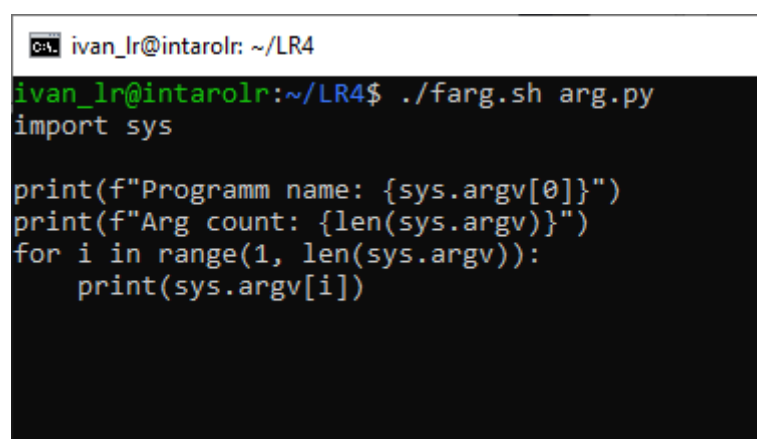
Рисунок 12.4 – Запуск программы на Python (arg.py)

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



```
ivan_lr@intarolr: ~/LR4
#!/bin/sh
count="$#"
if [ $count -eq 1 ]; then
    cat $1
    sleep 5
    clear
    exit 1
fi
echo "Error"
exit 1
```

Рисунок 13.2 – Код скрипта farg.sh



```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./farg.sh arg.py
import sys

print(f"Programm name: {sys.argv[0]}")
print(f"Arg count: {len(sys.argv)}")
for i in range(1, len(sys.argv)):
    print(sys.argv[i])
```

Рисунок 13.3 – Запуск скрипта farg.sh

Код на Python:



```
ivan_lr@intarolr: ~/LR4
import sys
import os

if len(sys.argv) == 2:
    with open(sys.argv[1], "r") as f:
        for line in f:
            print(line, end='')
    os.system('sleep 5')
    os.system('clear')
```

Рисунок 13.4 – Код программы на Python (farg.py)

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 farg.py arg.sh
#!/bin/sh
echo "Prorgamm name: $0"
echo "Arg count: $#"
```

Рисунок 13.5 - Запуск программы на Python (farg.py)

7) Используя FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно.

```
ivan_lr@intarolr: ~/LR4
#!/bin/sh

for file in /*
do
    if [ -f $file ]; then
        cat $file | less -N
        echo ""
    fi
done
```

Рисунок 13.6 – Код скрипта filesdir.sh

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./filesdir.sh
```

Рисунок 13.7 – Запуск скрипта filesdir.sh

```

ivan_lr@intarolr: ~/LR4
1 import sys
2
3 print(f"Programm name: {sys.argv[0]}")
4 print(f"Arg count: {len(sys.argv)}")
5 for i in range(1, len(sys.argv)):
6     print(sys.argv[i])
(END)

```

```

ivan_lr@intarolr: ~/LR4
1 #!/bin/sh
2 echo "Prorgamm name: $0"
3 echo "Arg count: $# "
4 for i in $@
5 do
6     echo "$i"
7 done
(END)

```

Рисунки 13.8 и 13.9 – Результат запуска

Код на Python:

```

ivan_lr@intarolr: ~/LR4
import os

dir = os.walk('/home/ivan_lr/LR4')
for address, dirs, files in dir:
    for file in files:
        os.system(f"cat {file} | less -N")
~

```

Рисунок 13.10 – Код программы на Python (filesdir.py)

```

ivan_lr@intarolr: ~/LR4
1 import sys
2
3 print(f"Programm name: {sys.argv[0]}")
4 print(f"Arg count: {len(sys.argv)}")
5 for i in range(1, len(sys.argv)):
6     print(sys.argv[i])
(END)(END)

```

Рисунок 13.11 – Результат запуска программы на Python

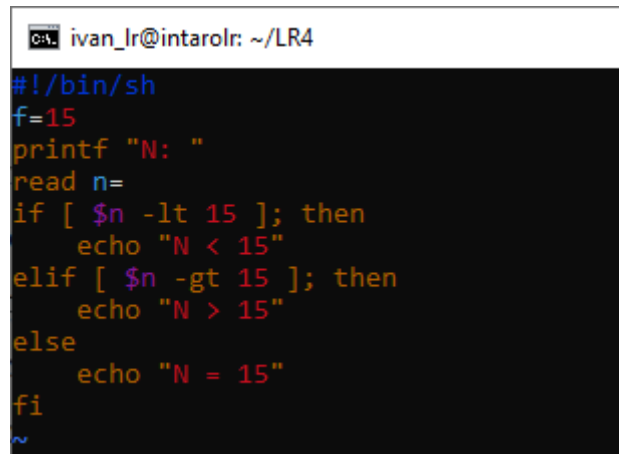
```

ivan_lr@intarolr: ~/LR4
1 #!/bin/sh
1 #!/bin/sh
2
3 for file in ./*
4 do
5     if [ -f $file ]; then
6         cat $file | less -N
7         echo ""
8     fi
9 done
(END)

```

Рисунок 13.12 – Результат запуска программы на Python

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



```
ivan_lr@intarolr: ~/LR4
#!/bin/sh
f=15
printf "N: "
read n=
if [ $n -lt 15 ]; then
    echo "N < 15"
elif [ $n -gt 15 ]; then
    echo "N > 15"
else
    echo "N = 15"
fi
~
```

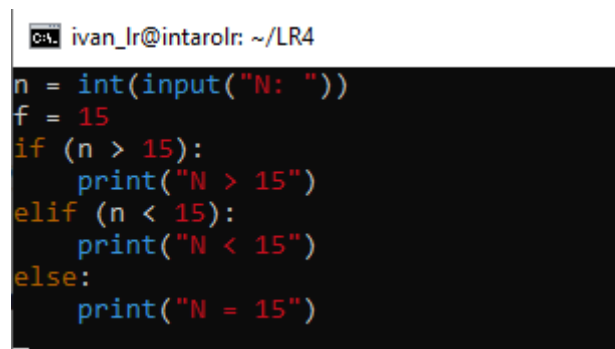
Рисунок 14.1 – Код скрипта compar.sh



```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./compar.sh
N: 17
N > 15
ivan_lr@intarolr:~/LR4$ ./compar.sh
N: 12
N < 15
ivan_lr@intarolr:~/LR4$ ./compar.sh
N: 15
N = 15
ivan_lr@intarolr:~/LR4$
```

Рисунок 14.2 – Запуск скрипта compar.sh

Код на Python:



```
ivan_lr@intarolr: ~/LR4
n = int(input("N: "))
f = 15
if (n > 15):
    print("N > 15")
elif (n < 15):
    print("N < 15")
else:
    print("N = 15")
```

Рисунок 14.3 – Код программы на Python (compar.py)

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 compar.py
N: 20
N > 15
ivan_lr@intarolr:~/LR4$ python3 compar.py
N: 0
N < 15
ivan_lr@intarolr:~/LR4$ python3 compar.py
N: 15
N = 15
ivan_lr@intarolr:~/LR4$ _
```

Рисунок 14.4 – Запуск программы на Python (compar.py)

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

```
ivan_lr@intarolr: ~/LR4
#!/bin/sh

printf "Year: "
read year=
if [ `expr $year % 4` -eq 0 ] && [ `expr $year % 100` -gt 0 ]; then
    echo "Leap year\n"
else
    echo "Non-Leap year\n"
fi
~
~
```

Рисунок 15.1 – Код скрипта year.sh

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./year.sh
Year: 2022
Non-Leap year

ivan_lr@intarolr:~/LR4$ ./year.sh
Year: 2024
Leap year

ivan_lr@intarolr:~/LR4$
```

Рисунок 15.2 – Запуск скрипта year.sh

Код на Python:

```
ivan_lr@intarolr: ~/LR4
year = int(input("Year: "))
if (year % 4 == 0) and (year % 100 != 0):
    print("Leep Year\n")
else:
    print("Non-leep year\n")
```

Рисунок 15.3 – Код программы на Python (year.py)

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 year.py
Year: 2021
Non-leep year

ivan_lr@intarolr:~/LR4$ python3 year.py
Year: 2020
Leep Year

ivan_lr@intarolr:~/LR4$
```

Рисунок 15.4 – Запуск программы на Python (year.py)

10) Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значение инкрементируется.

```
ivan_lr@intarolr: ~/LR4
#!/bin/sh
printf "x: "
read x=
printf "y: "
read y=
printf "Left border: "
read a=
printf "Right border: "
read b=

while [ $x -le $b ] && [ $x -ge $a ]
do
    x=$((expr $x - 1))
done
while [ $y -le $b ] && [ $y -ge $a ]
do
    y=$((expr $y + 1))
done

printf "x = "
echo $x
printf "y = "
echo $y
```

Рисунок 16.1 – Код скрипта diap.sh

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./diap.sh
x: 4
y: 5
Left border: 1
Right border: 10
x = 0
y = 11
ivan_lr@intarolr:~/LR4$ _
```

Рисунок 16.2 – Запуск скрипта diap.sh

Код на Python:

```
ivan_lr@intarolr: ~/LR4
x = int(input("x: "))
y = int(input("y: "))
a = int(input("Left border: "))
b = int(input("Right border: "))

while ((x >= a) and (x <= b)):
    x -= 1
while ((y >= a) and (y <= b)):
    y += 1

print("x = ", x)
print("y = ", y)
```

Рисунок 16.3 – Код программы на Python (diap.py)

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 diap.py
x: 51
y: 42
Left border: 1
Right border: 100
x = 0
y = 101
ivan_lr@intarolr:~/LR4$
```

Рисунок 16.4 – Запуск программы на Python (diap.py)

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

```
ivan_lr@intarolr: ~/LR4

#!/bin/sh
password="1234"
if [ $1 -eq $password ]; then
    ls -la /etc | less
else
    echo "Permission denied"
fi
```

Рисунок 17.1 – Код скрипта pswd.sh

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./pswd.sh 4321
Permission denied
ivan_lr@intarolr:~/LR4$
```

Рисунок 17.2 – Результат запуска скрипта с неверным паролем.

```
ivan_lr@intarolr: ~/LR4

-rw-r--r--  1 root root      5892 авг 10 00:20 ca-certificates.conf
drwxr-xr-x 103 root root      4096 ноя  6 19:58 .
drwxr-xr-x 19 root root      4096 сен 25 12:14 ..
-rw-r--r--  1 root root     3028 авг 10 00:17 adduser.conf
drwxr-xr-x  2 root root      4096 окт  7 16:24 alternatives
drwxr-xr-x  3 root root      4096 авг 10 00:21 apparmor
drwxr-xr-x  8 root root      4096 авг 10 00:21 apparmor.d
drwxr-xr-x  3 root root      4096 авг 10 00:21 apport
drwxr-xr-x  8 root root      4096 сен 25 12:09 apt
```

Рисунок 17.3 – Результат запуска скрипта с верным паролем.

Код на Python:

```
ivan_lr@intarolr: ~/LR4

import os
import sys

password = "1234"
if (sys.argv[1] == password):
    os.system("ls -la /etc | less")
else:
    print("Permission denied")
```

Рисунок 17.4 – Код программы на Python (pswd.py)

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 pswd.py 4312
Permission denied
ivan_lr@intarolr:~/LR4$ _
```

Рисунок 17.5 – Результат запуска программы на Python с неверным паролем

```
ivan_lr@intarolr: ~/LR4
-rw-r--r--  1 root root      5892 авг 10 00:20 ca-certificates.conto
drwxr-xr-x 103 root root      4096 ноя  6 19:58 .
drwxr-xr-x  19 root root      4096 сен 25 12:14 ..
-rw-r--r--  1 root root      3028 авг 10 00:17 adduser.conf
drwxr-xr-x  2 root root      4096 окт  7 16:24 alternatives
drwxr-xr-x  3 root root      4096 авг 10 00:21 apparmor
drwxr-xr-x  8 root root      4096 авг 10 00:21 apparmor.d
drwxr-xr-x  3 root root      4096 авг 10 00:21 apport
drwxr-xr-x  8 root root      4096 сен 25 12:09 apt
-rw-r--r--  1 root root      2319 янв  6 2022 bash.bashrc
```

Рисунок 17.6 – Результат запуска программы на Python с верным паролем

12) Проверить существует ли файл. Если да, выводится на экран его содержимое, если нет – выдается соответствующее сообщение.

```
ivan_lr@intarolr: ~/LR4
#!/bin/sh

if [ -f $1 ]; then
    echo "there is a file"
    cat $1
else
    echo "the file is not there"
fi

~
```

Рисунок 18.1 – Код скрипта checkfile.sh



```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ ./checkfile.sh passwd.py
the file is not there
ivan_lr@intarolr:~/LR4$ ./checkfile.sh pswd.py
there is a file
import os
import sys

password = "1234"
if (sys.argv[1] == password):
    os.system("ls -la /etc | less")
else:
    print("Permission denied")

ivan_lr@intarolr:~/LR4$ _
```

Рисунок 18.2 – Запуск скрипта checkfile.sh

Код на Python:

```
ivan_lr@intarolr: ~/LR4
import os
import sys

flag = False
for address, dirs, files in os.walk('/home/ivan_lr/LR4'):
    if sys.argv[1] in files:
        flag = True
if flag:
    print("there is a file")
    os.system(f"cat {sys.argv[1]}")
else:
    print("the file is not there")

~
~
```

Рисунок 18.3 – Код программы на Python (checkfile.py)

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 ckeckfile.py checkfile.sh
there is a file
#!/bin/sh

if [ -f $1 ]; then
    echo "there is a file"
    cat $1
else
    echo "the file is not there"
fi

ivan_lr@intarolr:~/LR4$ python3 ckeckfile.py 3123.sh
the file is not there
ivan_lr@intarolr:~/LR4$ _
```

Рисунок 18.4 – Запуск программы на Python (checkfile.py)

13) 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.

```
ivan_lr@intarolr: ~/LR4

#!/bin/sh

if [ -d $1 ]; then
    echo "This is directory"
    ls $1
elif [ -f $1 ]; then
    echo "This is file"
    cat $1
else
    echo "Create directory"
    mkdir -p $1
fi
```

Рисунок 19.1 – Код скрипта checkdir.sh

```
ivan_lr@intarolr: ~/LR4

ivan_lr@intarolr:~/LR4$ ./checkdir.sh dirtest
This is directory
1.txt 2.txt 3.txt
ivan_lr@intarolr:~/LR4$ ./checkdir.sh ckeckfile.py
This is file
import os
import sys

flag = False
for adress, dirs, files in os.walk('/home/ivan_lr/LR4'):
    if sys.argv[1] in files:
        flag = True
if flag:
    print("there is a file")
    os.system(f"cat {sys.argv[1]}")
else:
    print("the file is not there")

ivan_lr@intarolr:~/LR4$ ./checkdir.sh testd
Create directory
ivan_lr@intarolr:~/LR4$ ls
:      arg.py  checkdir.sh  compar.sh  expr.sh    filesdir.sh  input.py  pswd.sh  year.py
1.txt  arg.sh  checkfile.sh  diap.py   farg.py    file.txt    input.sh  testd   year.sh
2.txt  bc.sh  ckeckfile.py  diap.sh   farg.sh    hello.py    print.sh  vol.py
3.txt  calc.py compar.py     dirtest   filesdir.py hello.sh    pswd.py   vol.sh
ivan_lr@intarolr:~/LR4$
```

Рисунок 19.2 – Запуск скрипта checkdir.sh

Код на Python:

```
ivan_lr@intarolr: ~/LR4

import sys
import os

flag = 'c'
for address, dirs, files in os.walk('/home/ivan_lr/LR4'):
    if sys.argv[1] in dirs:
        flag = 'd'
    elif sys.argv[1] in files:
        flag = 'f'
if flag == 'd':
    print("This is directory")
    os.system(f"ls {sys.argv[1]}")
elif flag == 'f':
    print("This is file")
    os.system(f"cat {sys.argv[1]}")
else:
    print("Create directory")
    os.system(f"mkdir -p {sys.argv[1]}")
```

Рисунок 19.3 – Код программы на Python (checkdir.py)

```
ivan_lr@intarolr: ~/LR4

ivan_lr@intarolr:~/LR4$ python3 checkdir.py dirtest
This is directory
1.txt 2.txt 3.txt
ivan_lr@intarolr:~/LR4$ python3 checkdir.py checkfile.sh
This is file
#!/bin/sh

if [ -f $1 ]; then
    echo "there is a file"
    cat $1
else
    echo "the file is not there"
fi

ivan_lr@intarolr:~/LR4$ python3 checkdir.py testp
Create directory
ivan_lr@intarolr:~/LR4$
```

Рисунок 19.4 – Запуск программы на Python (checkdir.py)

14) 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 имена файлов и/или позиционные параметры).

 cat \$1 > \$2  
elif [ ! -r \$1 ] && [ -w \$2 ]; then  
 echo "First file is not for reading"  
elif [ -r \$1 ] && [ ! -w \$2 ]; then  
 echo "Second file is not for writing"  
else  
 echo "Error"  
fi" data-bbox="267 217 792 423"/>

```
ivan_lr@intarolr: ~/LR4  
#!/bin/sh  
  
if [ -r $1 ] && [ -w $2 ]; then  
    echo "Write suscessfully"  
    cat $1 > $2  
elif [ ! -r $1 ] && [ -w $2 ]; then  
    echo "First file is not for reading"  
elif [ -r $1 ] && [ ! -w $2 ]; then  
    echo "Second file is not for writing"  
else  
    echo "Error"  
fi
```

Рисунок 20.1 – Код скрипта analys.sh

 cat \$1  
else  
 echo "the file is not there"  
fi  
ivan\_lr@intarolr:~/LR4\$" data-bbox="203 477 859 700"/>

```
ivan_lr@intarolr:~/LR4$ chmod +x analys.sh  
ivan_lr@intarolr:~/LR4$ ./analys.sh checkfile.sh 1.txt  
Write suscessfully  
ivan_lr@intarolr:~/LR4$ cat 1.txt  
#!/bin/sh  
  
if [ -f $1 ]; then  
    echo "there is a file"  
    cat $1  
else  
    echo "the file is not there"  
fi  
ivan_lr@intarolr:~/LR4$
```

Рисунок 20.2 – Запуск скрипта analys.sh

Код на Python:

```
ivan_lr@intarolr: ~/LR4
import sys
import os

if (os.access(sys.argv[1], os.F_OK) and os.access(sys.argv[2], os.F_OK)):
    if (os.access(sys.argv[1], os.R_OK) and os.access(sys.argv[2], os.W_OK)):
        print("Write succsessfully")
        os.system(f"cat {sys.argv[1]} > {sys.argv[2]}")
    elif (not os.access(sys.argv[1], os.R_OK) and os.access(sys.argv[2], os.W_OK)):
        print("First file is not for reading")
    elif (os.access(sys.argv[1], os.R_OK) and not os.access(sys.argv[2], os.W_OK)):
        print("Second file is not for writing")
    else:
        print("Error")
else:
    print("Error")
```

Рисунок 20.3 – Код программы на Python (analys.py)

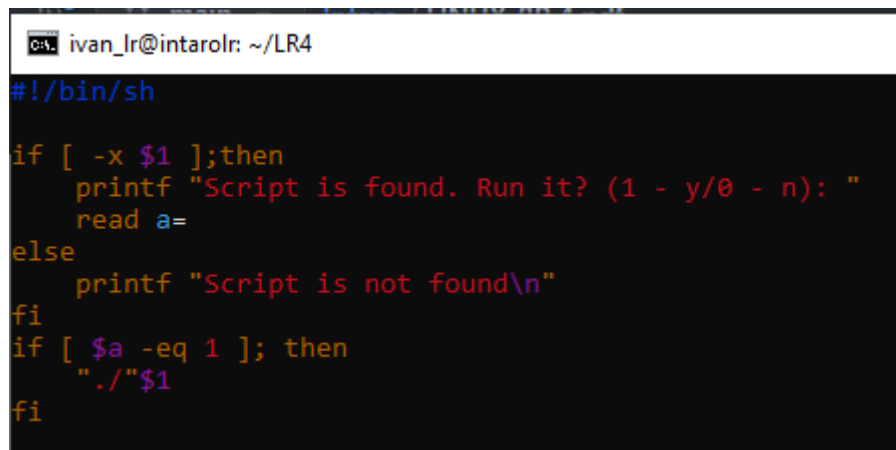
```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 analys.py ccheckfile.py 2.txt
Write succsessfully
ivan_lr@intarolr:~/LR4$ cat 2.txt
import os
import sys

flag = False
for adress, dirs, files in os.walk('/home/ivan_lr/LR4'):
    if sys.argv[1] in files:
        flag = True
if flag:
    print("there is a file")
    os.system(f"cat {sys.argv[1]}")
else:
    print("the file is not there")

ivan_lr@intarolr:~/LR4$ _
```

Рисунок 20.4 – Запуск программы на Python (analys.py)

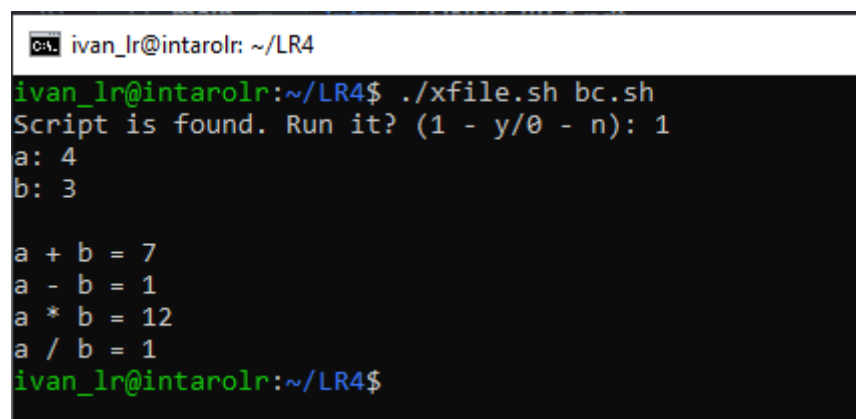
15) Если файл запуска программы найден, программа запускается (по выбору).



```
ivan_lr@intarolr: ~/LR4
#!/bin/sh

if [ -x $1 ];then
    printf "Script is found. Run it? (1 - y/0 - n): "
    read a=
else
    printf "Script is not found\n"
fi
if [ $a -eq 1 ]; then
    "./"$1
fi
```

Рисунок 21.1 – Код скрипта xfile.sh

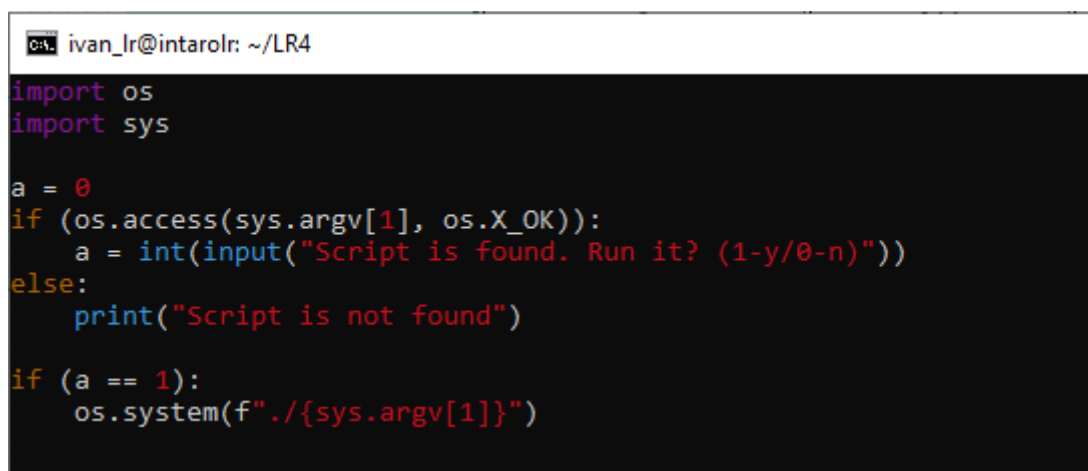


```
ivan_lr@intarolr: ~/LR4$ ./xfile.sh bc.sh
Script is found. Run it? (1 - y/0 - n): 1
a: 4
b: 3

a + b = 7
a - b = 1
a * b = 12
a / b = 1
ivan_lr@intarolr:~/LR4$
```

Рисунок 21.2 – Запуск скрипта xfile.sh

Код на Python:



```
ivan_lr@intarolr: ~/LR4
import os
import sys

a = 0
if (os.access(sys.argv[1], os.X_OK)):
    a = int(input("Script is found. Run it? (1-y/0-n)"))
else:
    print("Script is not found")

if (a == 1):
    os.system(f"./{sys.argv[1]}")
```

Рисунок 21.3 – Код программы на Python (xfile.py)

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 xfile.py expr.sh
Script is found. Run it? (1-y/0-n)1
a: 5
b: 4

a + b = 9
a - b = 1
a * b = 20
a / b = 1
ivan_lr@intarolr:~/LR4$ python3 xfile.py req.sh
Script is not found
ivan_lr@intarolr:~/LR4$
```

Рисунок 21.4 – Запуск программы на Python (xfile.py)

16) 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.

```
ivan_lr@intarolr: ~/LR4
#!/bin/sh

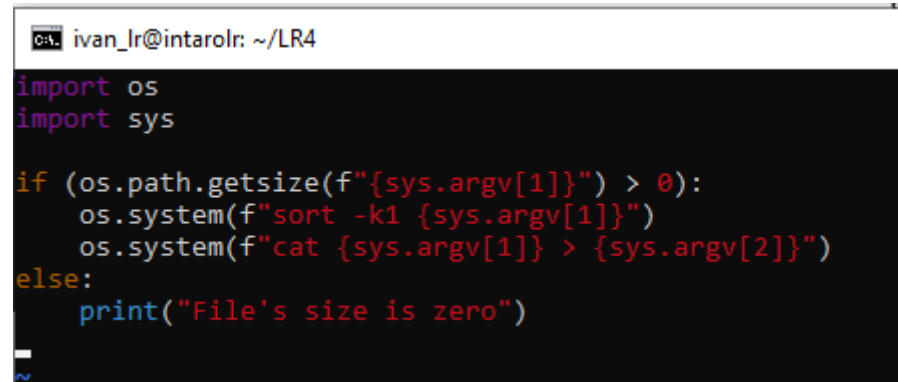
size=$(wc -c $1 | awk '{print $1}')
if [ $size -ge 0 ]; then
    sort -k1 $1
    cat $1 > $2
else
    echo "File's size is zero"
fi
```

Рисунок 22.1 – Код скрипта sort.sh

```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ cat testsort.txt
1 g 4
6 a 0
2 c 1
ivan_lr@intarolr:~/LR4$ ./sort.sh testsort.txt 3.txt
1 g 4
2 c 1
6 a 0
ivan_lr@intarolr:~/LR4$ cat 3.txt
1 g 4
6 a 0
2 c 1
ivan_lr@intarolr:~/LR4$
```

Рисунок 22.2 – Запуск скрипта sort.sh

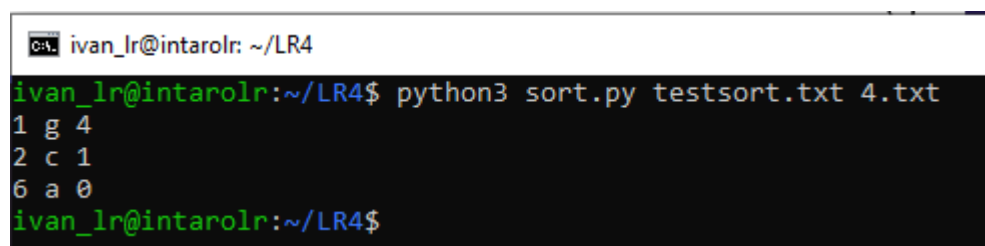
Код на Python:



```
ivan_lr@intarolr: ~/LR4
import os
import sys

if (os.path.getsize(f"{sys.argv[1]}") > 0):
    os.system(f"sort -k1 {sys.argv[1]}")
    os.system(f"cat {sys.argv[1]} > {sys.argv[2]}")
else:
    print("File's size is zero")
```

Рисунок 22.3 – Код программы на Python (sort.py)



```
ivan_lr@intarolr: ~/LR4
ivan_lr@intarolr:~/LR4$ python3 sort.py testsort.txt 4.txt
1 g 4
2 c 1
6 a 0
ivan_lr@intarolr:~/LR4$
```

Рисунок 22.4 – Запуск программы на Python (sort.py)



## **Вывод**

По итогу лабораторной работы я изучил основные возможности языка программирования высокого уровня Shell и получил навыки написания и использования скриптов.

## Ответы на контрольные вопросы

### 1) В чем отличие пользовательских переменных от переменных среды?

Пользовательские переменные - это переменные, определенные и использованные конкретным пользователем внутри его рабочей среды.

Переменные среды - это переменные, которые определены на уровне операционной системы и доступны для всех пользователей и приложений, работающих на компьютере.

### 2) Математические операции в Shell

«+» - сложение ( $a + b$ ), «-» - вычитание ( $a - b$ ), «\*» - умножение ( $a * b$ ), «/» - деление ( $a / b$ ), «%» - остаток от деления ( $a \% b$ ), «\*\*» - возведение в степень ( $a ** b$ ).

Пример использования

```
result=$((a + b))
```

### 3) Условные операторы Shell

```
if [ УСЛОВИЕ1 ] ; then
```

```
    # Блок кода, если условие1 выполнено
```

```
elif [ УСЛОВИЕ2 ] ; then
```

```
    # Блок кода, если условие2 выполнено
```

```
else
```

```
    # Блок кода, если ни одно из описанных условий не выполнено
```

```
fi
```

### 4) Принципы построения простых и составных условий.

«&&» или «-a» - логическое И; «||» или «-o» - логическое ИЛИ; «=» или «==» - проверка на равенство; «!=» или «-ne» - проверка на неравенство; «-z» - проверяет, что значение переменной пусто; «-n» - проверяет, что значение переменной НЕпустое; также поддерживает все операции сравнения типа больше/меньше/равно: «>», «<», «>=», «<=» или «-gt» (greater than), «-lt» (less than), «-ge» (greater or equal), «-le» (less or equal).

## 5) Циклы в Shell.

```
while [ УСЛОВИЕ ] do
```

```
    # Блок кода, если условие выполнено
```

```
done
```

```
for ПЕРЕМЕННАЯ in МНОЖЕСТВО_ЭЛЕМЕНТОВ
```

```
    # Блок кода, использующий элемент из множества
```

```
done
```

## 6) Массивы и модули в Shell.

- Определение массива:

```
myArray=(“value1” “value2” “value3”)
```

- Доступ к элементу по индексу:

```
array=(“a” “b” “c”)
```

```
echo ${array[0]}
```

Все элементы массива:

```
echo ${array[@]}
```

Количество элементов в массиве:

```
echo ${#array[@]}
```

- Удаление элемента массива:

```
unset array[2]
```

- Добавление элементов в конец массива:

```
$array+=(“e” “f”)
```

## 7) Чтение параметров командной строки.

Работа с параметрами происходит следующим образом:

\$0 - всегда **имя файла скрипта** (в данном случае script.sh)

\$1 - первый позиционный параметр (в данном случае par1)

\$2 - второй позиционный параметр (par2) и тд

#### 8) Как различать ключи и параметры?

Перед ключами ставится “-”, а параметры ставятся в конце команды.

Пример: `ls -la dir`

#### 9) Чтение данных из файлов.

Чтение из файлов происходит с помощью символа “<”

Пример: `readarray lines < file.txt`

Или с помощью команды `cat`

#### 10) Стандартные дескрипторы файлов.

- Дескриптор стандартного ввода (stdin) - используется для чтения данных из стандартного входного потока.

- Дескриптор стандартного вывода (stdout) - используется для вывода данных в стандартный поток вывода.

- Дескриптор стандартной ошибки (stderr) - используется для вывода сообщений об ошибках и диагностической информации.

числовые значения:

- stdin: 0

- stdout: 1

- stderr: 2

#### 11) Перенаправление вывода.

`ls > result.txt` - Перенаправит вывод команды `ls` в файл `result.txt`. Если файл существует, его содержимое будет перезаписано.

- `ls >> result.txt` - Сделает то же, что и «>», но если файл уже существует, результат будет записан в конец файла.

- `ls | grep “test”` - Перенаправит результат вывода команды `ls` в команду `grep`, которая отфильтрует полученный результат и выведет только те результаты, которые содержат слово «test».

## 12) Подавление вывода.

Команда будет подавлять вывод ошибок и стандартного вывод команды ls:

```
ls > /dev/null 2>$1
```

## 13) Отправка сигналов скриптам.

Скриптам в процессе выполнения можно отправлять сигналы так же, как и другим процессам, например kill, killall и другие.

## 14) Использование функций.

Синтаксис функций в shell:

```
function myfunc {  
    # do smth  
}
```

## 15) Отправка сообщений в терминал пользователя.

Это происходит с помощью команды echo:

```
echo "Hello, it is script file $0"
```

## 16) BASH и SHELL – синонимы?

SHELL (интерпретатор командной оболочки) - это общий термин, который описывает программу, предоставляющую пользователю интерфейс для взаимодействия с операционной системой. Она обрабатывает команды, вводимые пользователем, и выполняет их.

BASH (Bourne Again SHell) - это одна из наиболее распространенных командных оболочек в UNIX-подобных системах. BASH является расширением исходного кода оригинальной командной оболочки Unix - Bourne shell (sh), и предоставляет дополнительные функции и улучшения.

Таким образом, BASH - это конкретная реализация командной оболочки, в то время как SHELL - это более широкое понятие,

описывающее класс программ, осуществляющих взаимодействие пользователя с операционной системой.

17) PowerShell в операционных системах семейства Windows: назначение и особенности.

PowerShell - это мощный интерактивный оболочечный язык и среда командной строки, разработанные компанией Microsoft для операционных систем Windows. Он был выпущен в 2006 году и является продолжением более старой командной оболочки cmd.exe.

Назначение PowerShell заключается в автоматизации задач администрирования и управления компьютером. Он предоставляет мощные средства для создания и выполнения сценариев, управления файлами и папками, настройки и управления службами, установки и удаления программных пакетов, редактирования реестра и многого другого. PowerShell также поддерживает управление удаленными компьютерами и Active Directory.

#### Особенности:

- Объектно-ориентированность: PowerShell представляет результаты команд в виде объектов, которые можно легко фильтровать, сортировать и передавать на вход других команд.
- Расширяемость: PowerShell поддерживает создание пользовательских модулей, которые добавляют новые команды и функциональность.
- Интеграция с другими технологиями Microsoft: PowerShell может использоваться для автоматизации работы с продуктами Microsoft, такими как SQL Server, Exchange, SharePoint и другими.
- Кросс-платформенность: начиная с версии PowerShell 6.0, Microsoft выпустила кросс-платформенную версию PowerShell, которая работает также на Linux и macOS.