

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТНАЯ РАБОТА №5

по дисциплине «OS Linux»

**на тему «Программирование на SHELL. Использование командных
файлов»**

Студент

Сухоруких А.О.

Группа АС-18

Руководитель

Кургасов В.В.

к.т.н

Липецк 2020 г.

Оглавление

Цель работы	5
Задание	6
1. Используя команды ECHO, PRINTF вывести информационные сообщения на экран.....	9
В первой строке мы присваиваем переменной A значение 5. С помощью команда echo выводим это значение на экран.	10
Аналогично заданию 2 присваиваем переменной A значение 5, затем переменной B присваиваем значение A и выводим значение B на экран.	11
Присваиваем переменной D имя команды date, выводим полученную дату на экран.....	13
6. Присвоить переменной E значение “имя команды”, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной	14
7. Присвоить переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной.	15
8. Программа запрашивает значение переменной, а затем выводит значение этой переменной	17
9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной	18
10. Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) BC)	19
11. Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран.....	20

12. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки.....	21
13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается	22
14. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно	23
15. Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения	24
16. Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран	25
17. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.....	26
18. В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc	27
19. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение	28
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	29
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)	30
22. Если файл запуска программы найден, программа запускается (по выбору)	31
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	33
24. Командой TAR осуществляется сборка всех текстовых файлов текущего каталога в один архивный файл, после паузы просматривается содержимое файла, затем командой GZIP архивный файл сжимается	34
25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных	36
Вывод	37

Цель работы

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

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 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

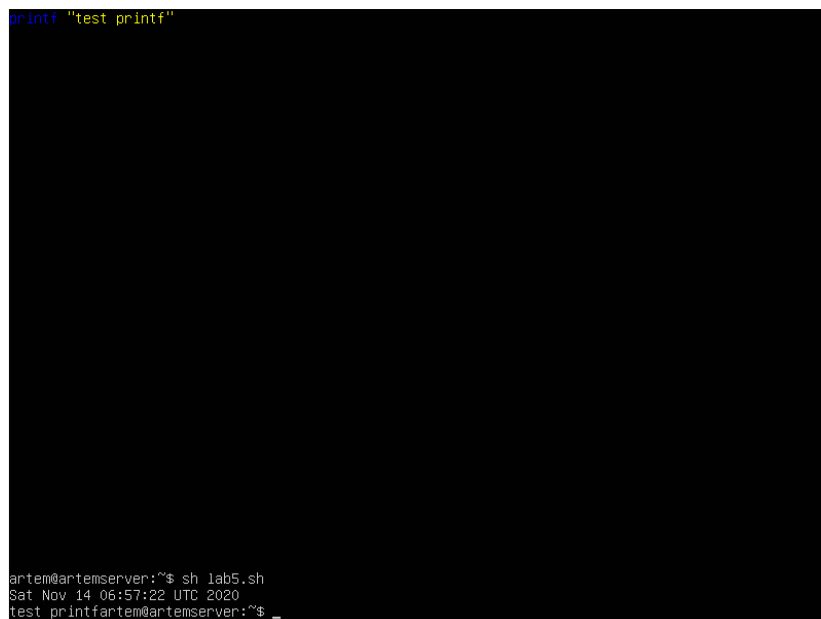
1. Используя команды ECHO, PRINTF вывести информационные сообщения на экран

С помощью команды echo выведем текущую дату на экран, а с помощью команды printf сообщение test printf. Листинг скрипта и пример выполнения показаны на рисунках 1 – 2.

A screenshot of a terminal window with the GNU nano 4.8 text editor open. The editor is editing a file named lab5.sh. The script contains two lines: echo date and printf "test printf". The nano editor's status bar at the bottom shows various keyboard shortcuts like Get Help, Write Out, Where Is, Cut Text, Justify, Cur Pos, Undo, Exit, Read File, Replace, Paste Text, To Spell, Go To Line, and Redo. The cursor is at the end of the second line.

```
GNU nano 4.8 lab5.sh
echo date
printf "test printf"
```

Рисунок 1 – Листинг скрипта для задания 1

A screenshot of a terminal window showing the execution of the script lab5.sh. The prompt is artem@artemserver:~\$. The command sh lab5.sh has been entered, and the output is Sat Nov 14 06:57:22 UTC 2020 followed by test printfartem@artemserver:~\$ on the next line.

```
artem@artemserver:~$ sh lab5.sh
Sat Nov 14 06:57:22 UTC 2020
test printfartem@artemserver:~$ _
```

Рисунок 2 – Пример выполнения задания 1

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

В первой строке мы присваиваем переменной А значение 5. С помощью команда echo выводим это значение на экран. Листинг скрипта и пример выполнения показаны на рисунках 3 – 4.

A screenshot of a terminal window with the GNU nano 4.8 text editor open. The editor is editing a file named lab5.sh. The content of the file is: A=5; echo \$A;. The status bar at the bottom shows various keyboard shortcuts like ^G Get Help, ^O Write Out, etc.

```
GNU nano 4.8 lab5.sh Modified
A=5;
echo $A;
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^G Go To Line M-E Redo
```

Рисунок 3 – Листинг скрипта для задания 2

A screenshot of a terminal window showing the execution of the script lab5.sh. The prompt is artem@artemserver:~\$. The command sh lab5.sh is entered, and the output is 5.

```
artem@artemserver:~$ sh lab5.sh
5
artem@artemserver:~$ _
```

Рисунок 4 – Пример выполнения задания 2

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

Аналогично заданию 2 присваиваем переменной А значение 5, затем переменной В присваиваем значение А и выводим значение В на экран. Листинг скрипта и пример выполнения показаны на рисунках 5 – 6.



```
GNU nano 4.8 lab5.sh
A=5;
B=$A;
echo $B;
```

The screenshot shows a terminal window with the nano text editor. The editor is editing a file named 'lab5.sh'. The content of the file is: A=5; B=\$A; echo \$B;. The nano editor's status bar at the bottom shows various shortcuts like Get Help, Write Out, Where Is, Cut Text, Justify, Cur. Pos, M-U Undo, Exit, Read File, Replace, Paste Text, To Spell, Go To Line, and M-E Redo. A message '[Wrote 3 lines]' is also visible.

Рисунок 5 – Листинг скрипта для задания 3



```
B=$A;
echo $B;

artem@artemserver:~$ sh lab5.sh
5
artem@artemserver:~$
```

The screenshot shows a terminal window where the script 'lab5.sh' is being executed. The prompt is 'artem@artemserver:~\$'. The command 'sh lab5.sh' is entered, and the output is '5'. The prompt then returns to 'artem@artemserver:~\$'.

Рисунок 6 – Пример выполнения задания 3

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

Переменной С присваиваем путь до каталога используя команду pwd, выводим получившееся значение на экран, затем перегадим в этот каталог.

Листинг скрипта и пример выполнения показаны на рисунках 7 – 8.



```
GNU nano 4.8 lab5.sh
C= pwd ;
echo $C;
cd $C;_

[ Wrote 3 lines ]
G Get Help  O Write Out  W Where Is  K Cut Text  J Justify  C Cur Pos  M-U Undo
X Exit      R Read File  N Replace  U Paste Text T To Spell  Go To Line M-B Redo
```

Рисунок 7 – Листинг скрипта для задания 4



```
artem@artemserver:~$ sh lab5.sh
/home/artem
/home/artem
artem@artemserver:~$
```

Рисунок 8 – Пример выполнения задания 4

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

Присваиваем переменной D имя команды date, выводим полученную дату на экран. Листинг скрипта и пример выполнения показаны на рисунках 9 – 10.



```
GNU nano 4.8 lab5.sh
D=date
echo $D;
```

Рисунок 9 – Листинг скрипта для задания 5



```
echo $D
artem@artemserver:~$ sh lab5.sh
Fri Nov 27 15:56:36 UTC 2020
artem@artemserver:~$
```

Рисунок 10 – Пример выполнения задания 5

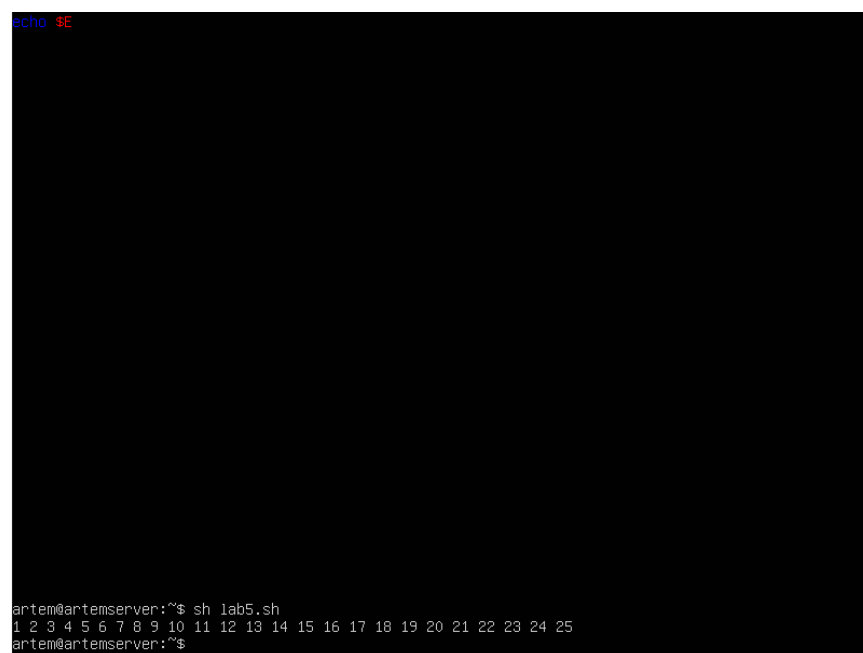
6. Присвоить переменной E значение “имя команды”, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной

Переменной E присваиваем значение команды для просмотра содержимого файла test. Выводим содержимое файла на экран. Листинг скрипта и пример выполнения показаны на рисунках 11 – 12.



```
GNU nano 4.8 lab5.sh
E= cat test
echo $E
```

Рисунок 11 – Листинг скрипта для задания 6



```
artem@artemserver:~$ sh lab5.sh
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
artem@artemserver:~$
```

Рисунок 12 – Пример выполнения задания 6

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

Создадим файл, который нам необходимо будет отсортировать. Создания файла для сортировки показан на рисунке 13.

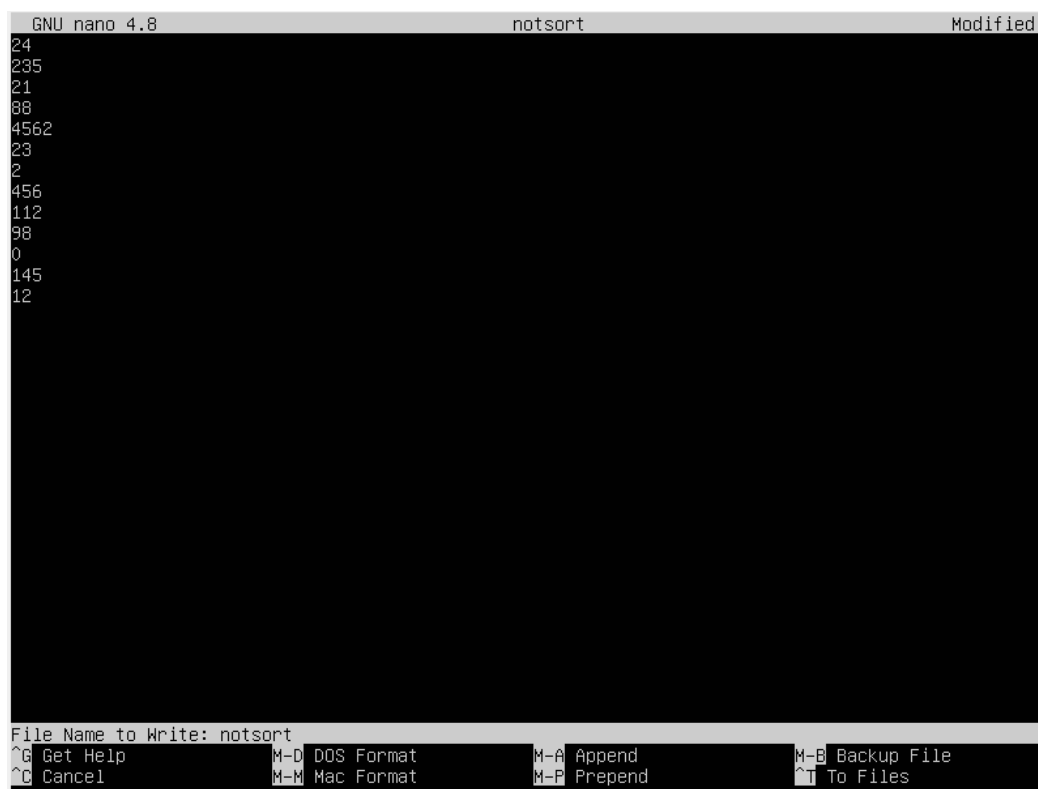


Рисунок 13 – Создание файла для сортировки

Затем в скрипте присвоим переменной E имя команды для сортировки и имя файла, который нас необходимо отсортировать. Выведем результат сортировки на экран. Листинг скрипта и пример выполнения показаны на рисунках 14 – 15.

```
GNU nano 4.8                               lab5.sh                               Modified
F= sort notsort
echo $F_

^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text      ^J Justify       ^C Cur Pos       M-U Undo
^X Exit          ^R Read File     ^N Replace       ^U Paste Text    ^T To Spell      ^G Go To Line    M-E Redo
```


Рисунок 14 – Листинг скрипта для задания 7

```
artem@artemserver:~$
artem@artemserver:~$ sh lab5.sh
0 112 12 145 2 21 23 235 24 456 4562 88 98
artem@artemserver:~$ _
```

Рисунок 15 – Пример выполнения задания 7

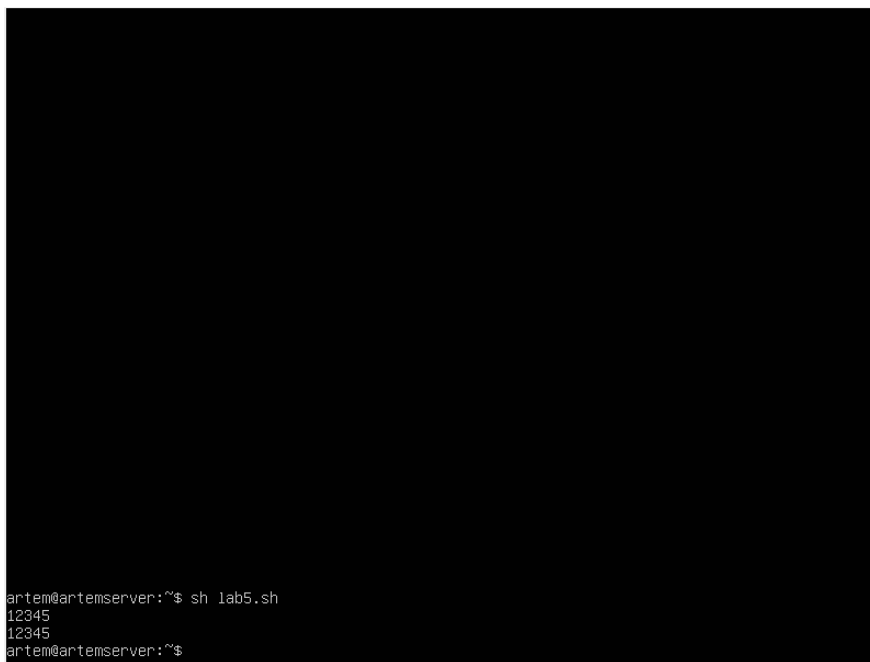
8. Программа запрашивает значение переменной, а затем выводит значение этой переменной

Для ввода значения переменной A воспользуемся командой read. Затем с помощью команды echo выведем значение переменной на экран. Листинг скрипта и пример выполнения показаны на рисунках 16 – 17.



```
GNU nano 4.8 lab5.sh Modified
read A;
echo $A;
```

Рисунок 16 – Листинг скрипта для задания 8



```
artem@artemserver:~$ sh lab5.sh
12345
12345
artem@artemserver:~$
```

Рисунок 17 – Пример выполнения задания 8

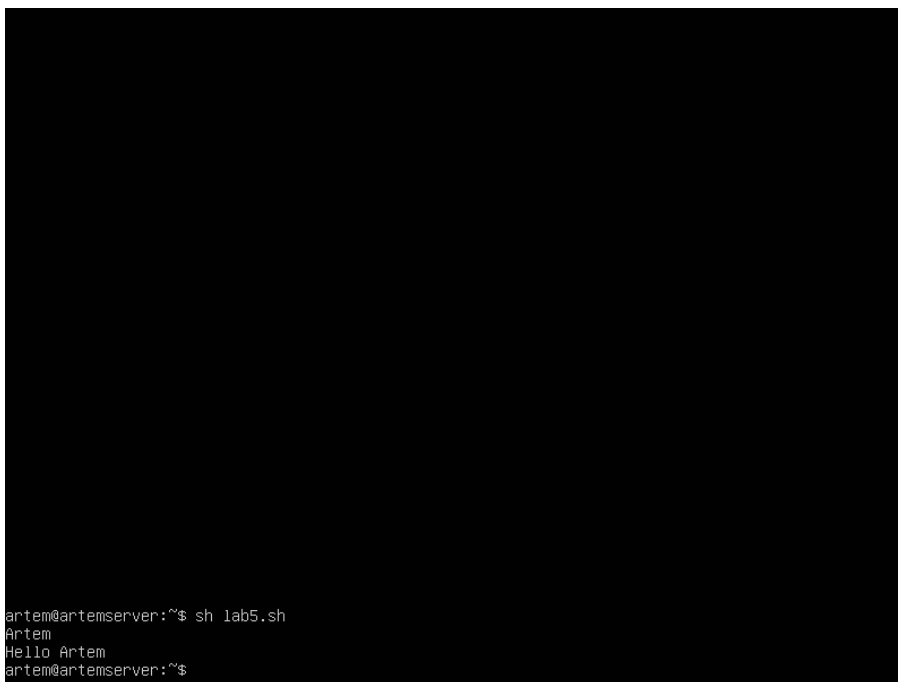
9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной

Для ввода имени в переменную A воспользуемся командой read. Затем с помощью команды echo выведем приветствие и имя на экран. Листинг скрипта и пример выполнения показаны на рисунках 18 – 19.

A screenshot of a terminal window with the GNU nano 4.8 editor open. The editor is editing a file named lab5.sh. The script contains two lines: read A; and echo Hello \$A;. The status bar at the bottom shows various nano editor commands like Get Help, Write Out, Where Is, Cut Text, Justify, Cur Pos, Undo, Exit, Read File, Replace, Paste Text, To Spell, Go To Line, and Redo. The [Read 2 lines] indicator is visible in the center of the status bar.

```
GNU nano 4.8 lab5.sh Modified
read A;
echo Hello $A;
```

Рисунок 18 – Листинг скрипта для задания 9

A screenshot of a terminal window showing the execution of the script lab5.sh. The prompt is artem@artemserver:~\$. The user enters 'sh lab5.sh'. The script prompts for a name, and the user enters 'Artem'. The script then outputs 'Hello Artem'. The prompt returns to artem@artemserver:~\$.

```
artem@artemserver:~$ sh lab5.sh
Artem
Hello Artem
artem@artemserver:~$
```

Рисунок 19 – Пример выполнения задания 9

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

Для ввода значений в переменные А и В воспользуемся командой read. Затем с помощью команды echo выведем значения полученных операций на экран. Листинг скрипта и пример выполнения показаны на рисунках 20 – 21.



```
GNU nano 4.8 lab5.sh Modified
read A;
read B;
echo `expr $A + $B`;
echo $A+$B | bc
echo `expr $A - $B`;
echo $A-$B | bc
echo `expr $A \* $B`;
echo $A*$B | bc
echo `expr $A / $B`;
echo $A/$B | bc
```

Рисунок 20 – Листинг скрипта для задания 10



```
artem@artemserver:~$ sh lab5.sh
10
2
12
12
8
8
20
20
artem@artemserver:~$ _
```

Рисунок 21 – Пример выполнения задания 10

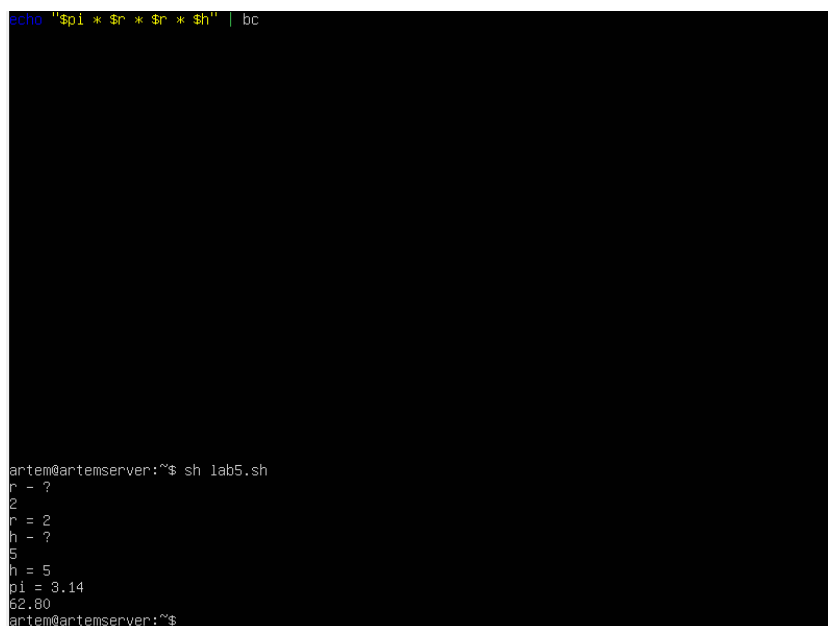
11. Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран

Для ввода значений в переменные r (радиус основания) и h (высота цилиндра) воспользуемся командой `read`. В переменную π положим значение числа π (3,14). Затем с помощью команды `echo` выведем значения полученного объема. Листинг скрипта и пример выполнения показаны на рисунках 22 – 23.



```
GNU nano 4.8 lab5.sh
echo "r - ?"
read r
echo "r = $r"
echo "h - ?"
read h
echo "h = $h"
pi=3.14
echo "pi = $pi"
echo "$pi * $r * $r * $h" | bc
```

Рисунок 22 – Листинг скрипта для задания 11



```
echo "$pi * $r * $r * $h" | bc

artem@artemserver:~$ sh lab5.sh
r - ?
2
r = 2
h - ?
5
h = 5
pi = 3.14
62.80
artem@artemserver:~$
```

Рисунок 23 – Пример выполнения задания 11

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

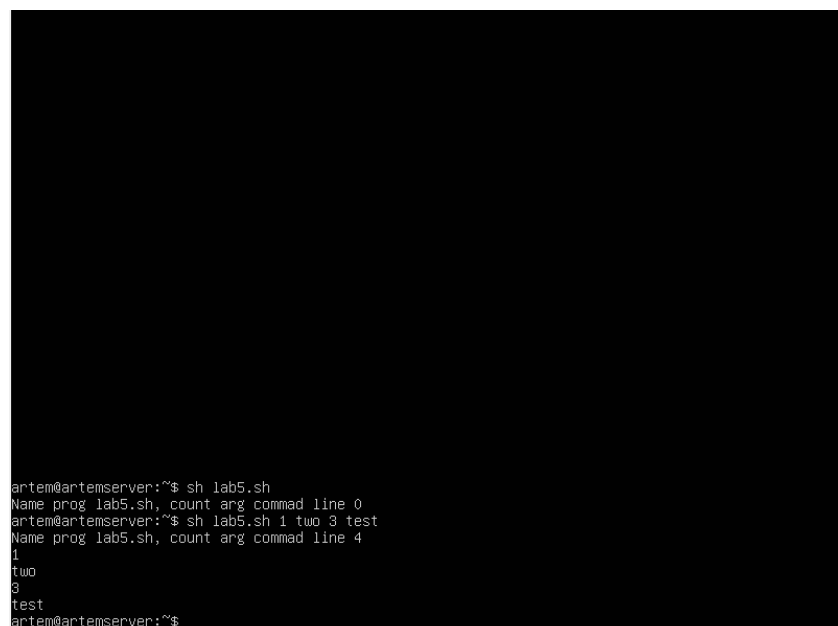
В переменной \$0 хранится имя программы, а в \$# кол-во аргументов. С помощью цикла for пройдемся по всем аргументам командной строки. Листинг скрипта и пример выполнения показаны на рисунках 24 – 25.



```
GNU nano 4.8 lab5.sh Modified
echo "Name prog $0, count arg commad line $#"
```

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

Рисунок 24 – Листинг скрипта для задания 12



```
artem@artemserver:~$ sh lab5.sh
Name prog lab5.sh, count arg commad line 0
artem@artemserver:~$ sh lab5.sh 1 two 3 test
Name prog lab5.sh, count arg commad line 4
1
two
3
test
artem@artemserver:~$ _
```

Рисунок 25 – Пример выполнения задания 12

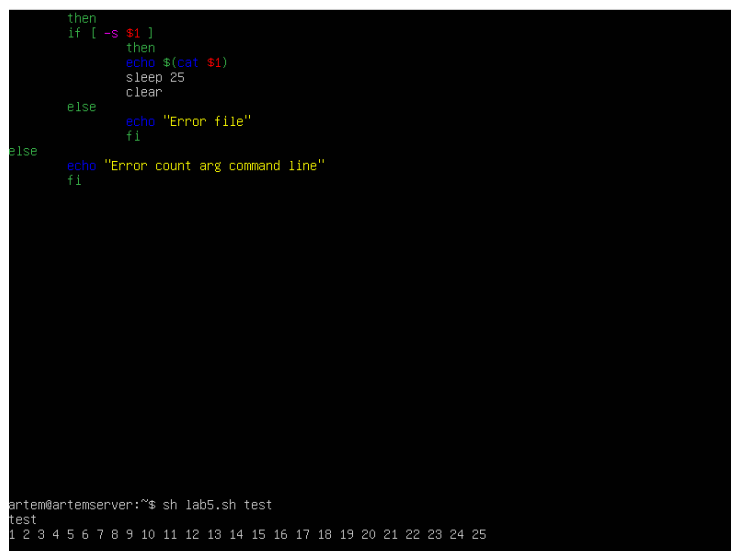
13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается

Имя текстового файла будет находиться в переменной \$1. С помощью оператора if производим необходимые проверки: 1. Количество аргументов командной строки единица; 2. Что текстовый файл существует. Выводим содержимое файла командой echo \$(cat \$1). Пауза создается командой sleep, очистка экрана – clear. Листинг скрипта и пример выполнения показаны на рисунках 25 – 26.



```
GNU nano 4.8 lab5.sh
echo $1
if [ $# -eq 1 ]
then
    if [ -s $1 ]
    then
        echo $(cat $1)
        sleep 25
        clear
    else
        echo "Error file"
    fi
else
    echo "Error count arg command line"
fi
```

Рисунок 25 – Листинг скрипта для задания 13



```
then
if [ -s $1 ]
then
    echo $(cat $1)
    sleep 25
    clear
else
    echo "Error file"
fi
else
    echo "Error count arg command line"
fi

artem@artemserver:~$ sh lab5.sh test
test
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
```

Рисунок 26 – Пример выполнения задания 13

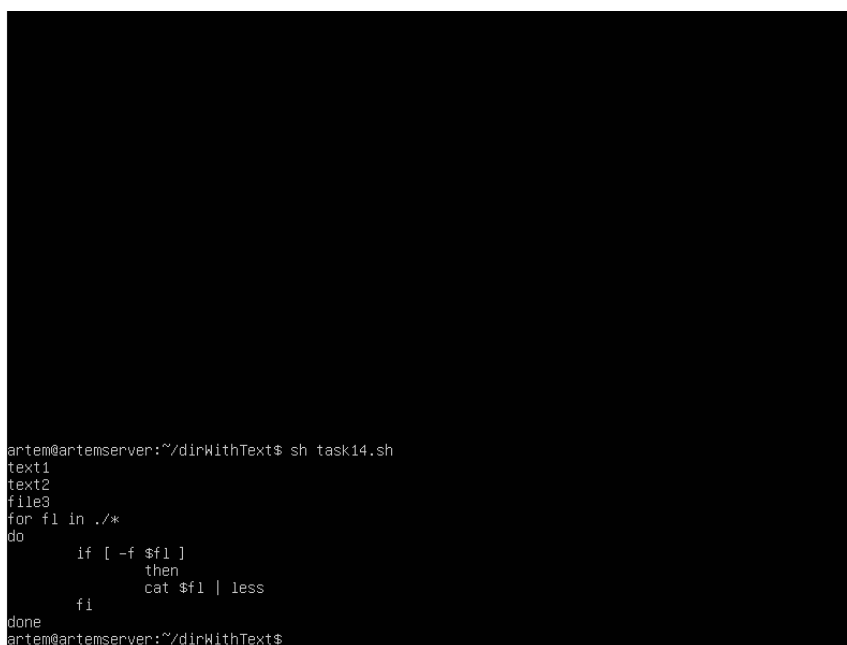
14. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно

С помощью цикла for проходимся по всем файлам, если файл текстовый, то используем команду cat. Листинг скрипта и пример выполнения показаны на рисунках 27 – 28.



```
GNU nano 4.8 task14.sh
for fi in .*
do
    if [ -f $fi ]
    then
        cat $fi | less
    fi
done
```

Рисунок 27 – Листинг скрипта для задания 14



```
artem@artemserver:~/dirWithText$ sh task14.sh
text1
text2
file3
for fi in .*
do
    if [ -f $fi ]
    then
        cat $fi | less
    fi
done
artem@artemserver:~/dirWithText$
```

Рисунок 28 – Пример выполнения задания 14

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

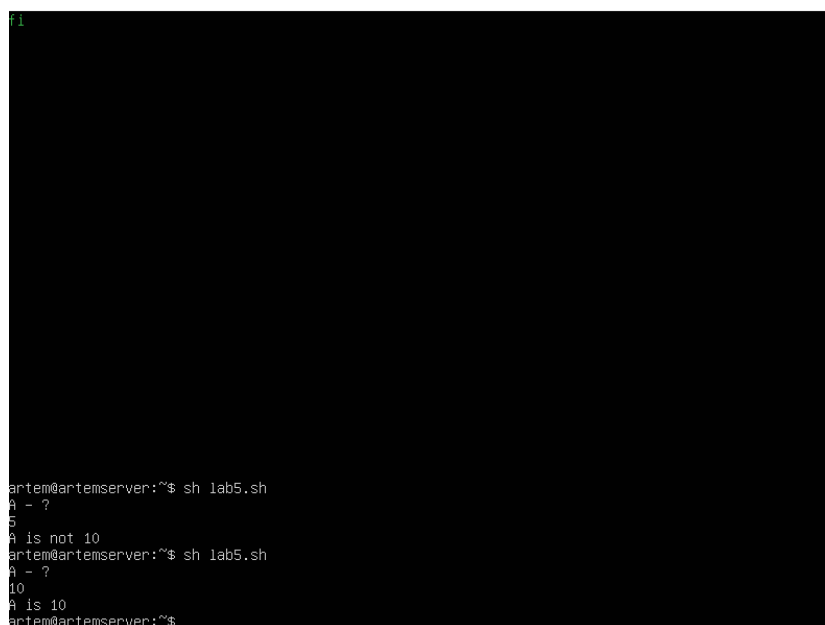
Для ввода значения в переменную A воспользуемся командой read. Затем с помощью if сравниваем значения переменной A с числом 10. В зависимости от сравнения выводим соответствующее сообщение на экран. Листинг скрипта и пример выполнения показаны на рисунках 29 – 30.



```
GNU nano 4.8 lab5.sh
echo "A - ?"
read a
if [ $a -eq 10 ]
then
    echo "A is 10"
else
    echo "A is not 10"
fi

[ Read 8 lines ]
Get Help Write Out Where Is Cut Text Justify Cur Pos M-U Undo
Exit Read File Replace Paste Text To Spell Go To Line M-E Redo
```

Рисунок 29 – Листинг скрипта для задания 15




```
artem@artemserver:~$ sh lab5.sh
A - ?
5
A is not 10
artem@artemserver:~$ sh lab5.sh
A - ?
10
A is 10
artem@artemserver:~$ _
```

Рисунок 30 – Пример выполнения задания 15

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

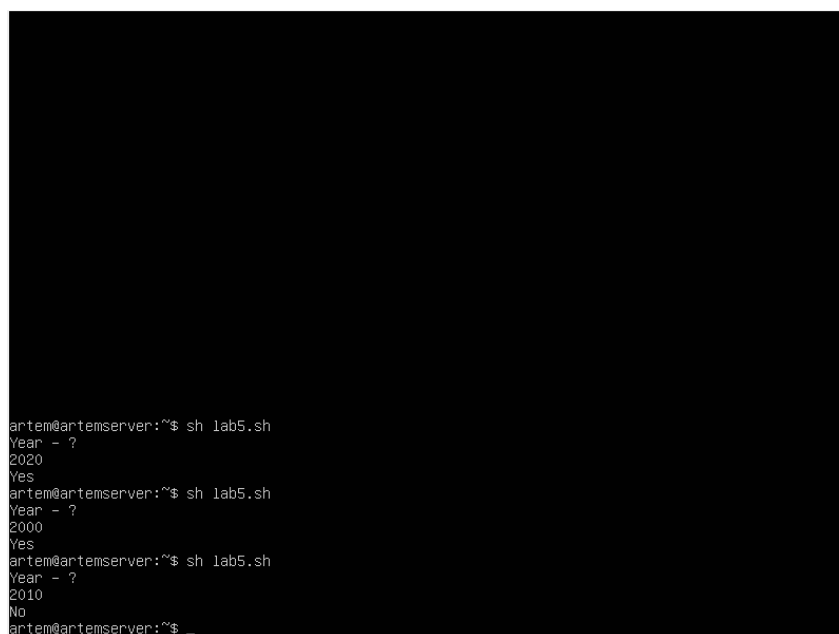
В переменную `y` будет помещен введенный год, проверяем введенный год с помощью условий `if`. В зависимости от сравнения выводим соответствующее сообщение на экран. Листинг скрипта и пример выполнения показаны на рисунках 31 – 32.



```
GNU nano 4.8 lab5.sh
echo "Year - ?"
read y
if [ `expr $y % 4` -eq 0 -a `expr $y % 100` -ne 0 ]
then
    echo "Yes"
elif [ `expr $y % 400` -eq 0 ]
then
    echo "Yes"
else
    echo "No"
fi

[ Wrote 11 lines ]
Get Help Write Out Where Is Cut Text Justify Cur Pos M-U Undo
Exit Read File Replace Paste Text To Spell Go To Line M-E Redo
```

Рисунок 31 – Листинг скрипта для задания 16



```
artem@artemserver:~$ sh lab5.sh
Year - ?
2020
Yes
artem@artemserver:~$ sh lab5.sh
Year - ?
2000
Yes
artem@artemserver:~$ sh lab5.sh
Year - ?
2010
No
artem@artemserver:~$ _
```

Рисунок 32 – Пример выполнения задания 16

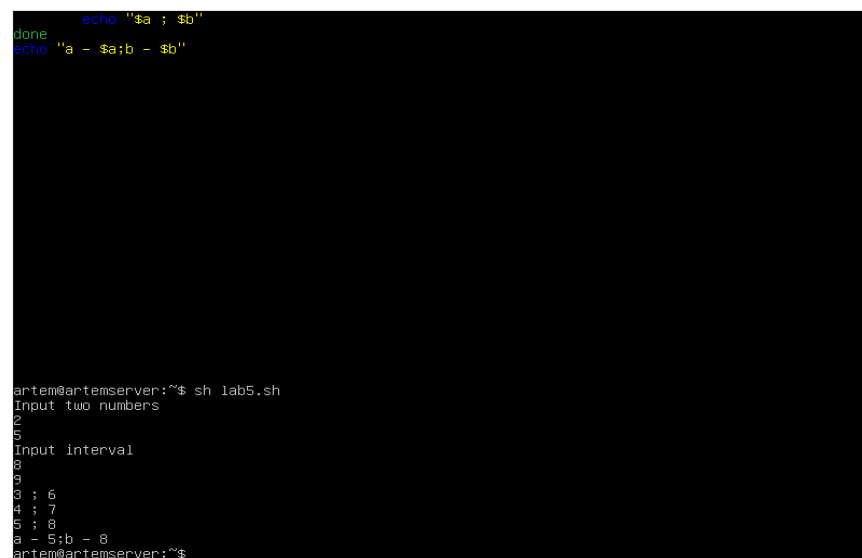
17. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются

С помощью команды `read` вводим значения. С помощью цикла `while` инкрементируем переменные `a` и `b` пока выполняется условие. Листинг скрипта и пример выполнения показаны на рисунках 33 – 34.



```
GNU nano 4.8 lab5.sh
echo "Input two numbers"
read a
read b
echo "Input interval"
read c
read d
while [ $c -gt $a ] && [ $c -gt $b ] && [ $d -gt $a ] && [ $d -gt $b ]
do
    a=`expr $a + 1`
    b=`expr $b + 1`
    echo "$a ; $b"
done
echo "a - $a;b - $b"
```

Рисунок 33 – Листинг скрипта для задания 17



```
done
echo "$a ; $b"
echo "a - $a;b - $b"

artem@artemserver:~$ sh lab5.sh
Input two numbers
2
5
Input interval
3
4
5
8
a - 5;b - 8
artem@artemserver:~$
```

Рисунок 34 – Пример выполнения задания 17

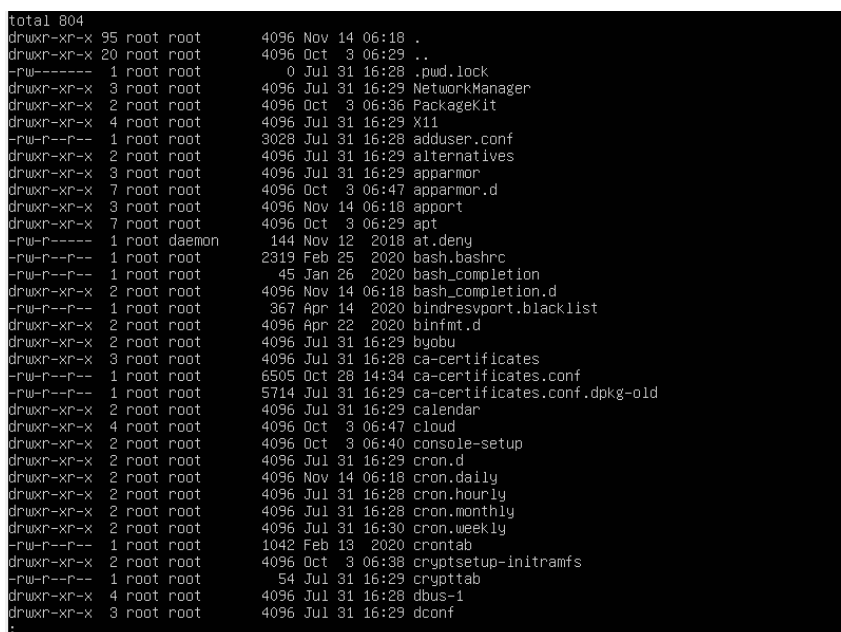
18. В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc

Пароль будет храниться в переменной \$1, в переменной rN хранится пароль с которым будет сравниваться введенный. Если пароли совпадают, то будет выполнена команда для отображения в длинном формате с указанием скрытых файлов содержимое каталога /etc. Листинг скрипта и пример выполнения показаны на рисунках 35 – 36.



```
GNU nano 4.8 lab5.sh
spN='querty123'
echo $1
if [ "$1" = "$spN" ]
then
    cd / & ls -al /etc | less
else
    echo "Wrong password"
fi
```

Рисунок 35 – Листинг скрипта для задания 18



```
total 804
drwxr-xr-x 95 root root 4096 Nov 14 06:18 .
drwxr-xr-x 20 root root 4096 Oct 3 06:29 ..
-rw-r--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 06:36 PackageKit
drwxr-xr-x 4 root root 4096 Jul 31 16:29 X11
-rw-r--r-- 1 root root 3028 Jul 31 16:28 adduser.conf
drwxr-xr-x 2 root root 4096 Jul 31 16:29 alternatives
drwxr-xr-x 3 root root 4096 Jul 31 16:29 apparmor
drwxr-xr-x 7 root root 4096 Oct 3 06:47 apparmor.d
drwxr-xr-x 3 root root 4096 Nov 14 06:18 apport
drwxr-xr-x 7 root root 4096 Oct 3 06:29 apt
-rw-r--r-- 1 root daemon 144 Nov 12 2018 at.deny
-rw-r--r-- 1 root root 2319 Feb 25 2020 bash.bashrc
-rw-r--r-- 1 root root 45 Jan 26 2020 bash_completion
drwxr-xr-x 2 root root 4096 Nov 14 06:18 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 Oct 28 14:34 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 06:47 cloud
drwxr-xr-x 2 root root 4096 Oct 3 06:40 console-setup
drwxr-xr-x 2 root root 4096 Jul 31 16:29 cron.d
drwxr-xr-x 2 root root 4096 Nov 14 06:18 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 06:38 cryptsetup-initramfs
-rw-r--r-- 1 root root 54 Jul 31 16:29 crpyttab
drwxr-xr-x 4 root root 4096 Jul 31 16:28 dbus-1
drwxr-xr-x 3 root root 4096 Jul 31 16:29 dconf
:
```

Рисунок 36 – Пример выполнения задания 18

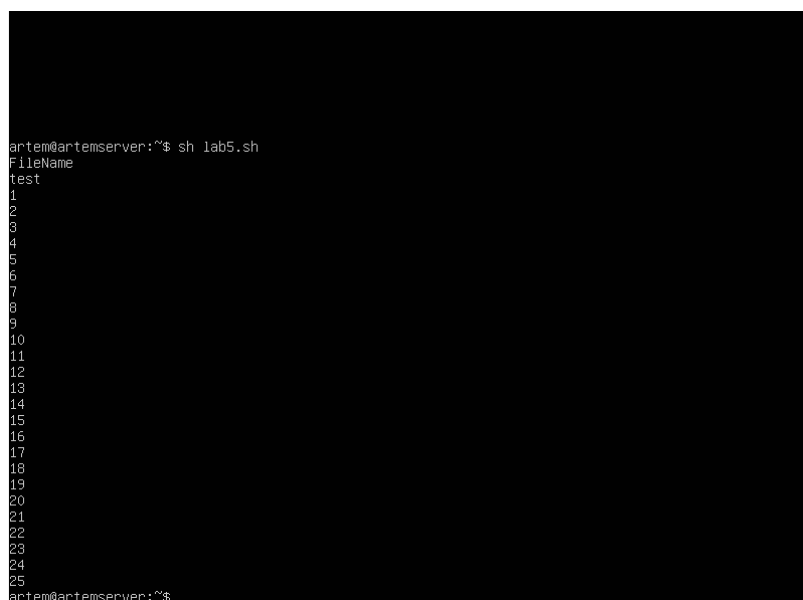
19. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение

С помощью команды `read` вводим имя файла. Затем с помощью оператора `if` проверяем существует ли файл. Если существует выполняем команду для просмотра содержимого файла. Листинг скрипта и пример выполнения показаны на рисунках 37 – 38.



```
GNU nano 4.8 lab5.sh
echo "FileName"
read fName
if [ -e $fName ]
then
    cat $fName
else
    echo "No such file"
fi
```

Рисунок 37 – Листинг скрипта для задания 19



```
artem@artemserver:~$ sh lab5.sh
FileName
test
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
artem@artemserver:~$ _
```

Рисунок 37 – Пример выполнения задания 19

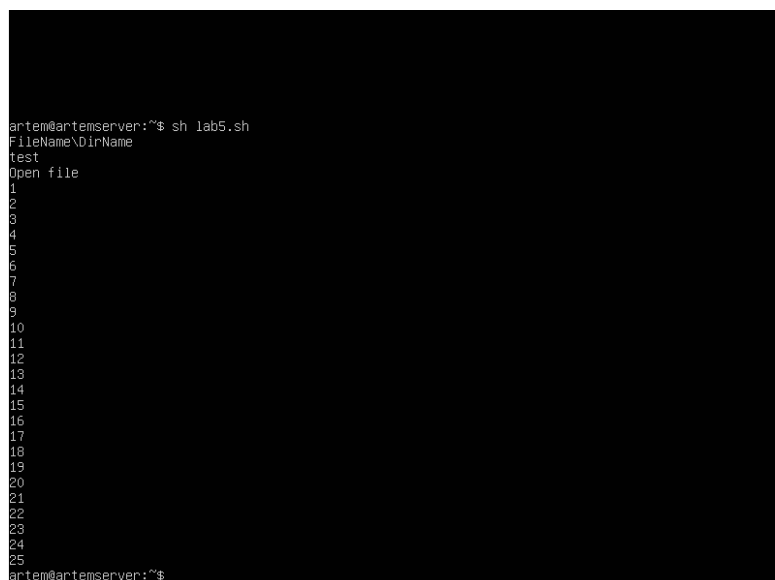
20. Если файл есть каталог и этот каталог можно читать, просматривается содержимое этого каталога. Если каталог отсутствует, он создается. Если файл не есть каталог, просматривается содержимое файла

Вводим имя файла или каталога. С помощью операторов if осуществляем необходимые проверки (существование каталога с таким именем, является ли файл каталогом, возможность чтения содержимого каталога). Листинг скрипта и пример выполнения показаны на рисунках 38 – 40.



```
GNU nano 4.8 lab5.sh
echo "FileName\DirName"
read fName
if [ -e $fName ]
then
  if [ -d $fName ]
  then
    if [ -r $fName ]
    then
      ls $fName
    else
      echo "Can not read"
    fi
  else
    echo "Open file"
    cat $fName
  fi
else
  "Make new dir"
  mkdir $fName
fi
```

Рисунок 38 – Листинг скрипта для задания 20



```
artem@artemserver:~$ sh lab5.sh
FileName\DirName
test
Open file
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
artem@artemserver:~$ _
```

Рисунок 39 – Пример открытия файла

```
FileName\DirName
test
Open file
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
artem@artemserver:~$ ls
dirWithText  lab5.sh  loop3  newdir  nowdate logs.log  result_new
errloop.log  loop    ls      notsort nowloop.log  sort_result
errnowdate.log  loop2  newchanel  nowdate.sh  result  test
artem@artemserver:~$ sh lab5.sh
FileName\DirName
dirWithText
1 2 3 task14.sh
artem@artemserver:~$ _
```

Рисунок 40 – Пример открытие каталога

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)

Вводим с помощью команды read имена файлов. С помощью if проверим существование файлов, и права доступа. Перенаправляем содержимое файла 1 в файл 2 с помощью команды cat и перенаправления вывода. Листинг скрипта и пример выполнения показаны на рисунках 41 – 42.

```
GNU nano 4.8 lab5.sh Modified
echo "FileName1 - ?"
read fName1
echo "FileName2 - ?"
read fName2
if [ -e $fName1 -a -e $fName2 -a -r $fName1 -a -w $fName2 ]
then
    echo "File1"
    cat $fName1
    echo "File2"
    cat $fName2
    cat $fName1 > $fName2
    echo "New File2"
    cat $fName2_
else
    echo "Error"
fi
```

Рисунок 41 – Листинг скрипта для задания 21

```
17
18
19
20
21
22
23
24
25
File2
New File2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
artem@artemserver:~$ _
```

Рисунок 42 – Пример выполнения задания 21

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

Проверяем количество аргументов командной строки, если файл \$1 существует и может быть выполнен, то выполняем данный файл. Листинг скрипта и пример выполнения показаны на рисунках 43 – 43.

```
GNU nano 4.8 lab5.sh Modified
if [ $# -eq 1 ]
then
    if [ -e $1 -a -x $1 ]
    then
        sh $1
    else
        echo "Error"
    fi
else
    echo "ArgCount Error"
fi

File Name to Write: lab5.sh
^G Get Help      M-D DOS Format  M-A Append      M-B Backup File
^C Cancel        M-M Mac Format  M-P Prepend     ^T To Files
```

Рисунок 43 – Листинг скрипта для задания 22

```
if [ -e $1 ] && [ -x $1 ]
then
    sh $1
else
    echo "Error"
fi
else
    echo "ArgCount Error"
fi

artem@artemserver:~$ sh lab5.sh 9.sh
artem
Hello artem
artem@artemserver:~$ _
```

Рисунок 44 – Пример выполнения задания 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

Проверяем количество аргументов, в переменной \$1 будет находится имя файла. С помощью if проверяем, что файл существует и не пуст. Если размер файла больше нуля, содержимое файла сортируется по первому столбцу по возрастанию, отсортированная информация помещается в файл task23result. Листинг скрипта и пример выполнения показаны на рисунках 45 – 46.



```
GNU nano 4.8 lab5.sh
if [ $# -eq 1 ]
then
  if [ -s $1 ]
  then
    sort -k1 $1 > "task23result"
    cat "task23result"
  else
    echo "Error"
  fi
else
  echo "ArgCount Error"
fi
```

[Wrote 12 lines]

Get Help Write Out Where Is Cut Text Justify Cur Pos M-U Undo
Exit Read File Replace Paste Text To Spell Go To Line M-E Redo

Рисунок 45 – Листинг скрипта для задания 23

A terminal window with a black background and white text. The prompt is 'artem@artemserver:~\$'. The command 'sh lab5.sh test' has been entered. The output consists of a vertical list of line numbers from 1 to 25, with some numbers appearing multiple times (e.g., 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25). The prompt 'artem@artemserver:~\$' is visible at the bottom.

```
artem@artemserver:~$ sh lab5.sh test
1
10
11
12
13
14
15
16
17
18
19
2
20
21
22
23
24
25
3
4
5
6
7
8
9
artem@artemserver:~$
```

Рисунок 46 – Пример выполнения задания 23

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

Командой `find . -type f` поиск всех текстовых файлов. В переменной `archName` хранится название архивного файла. Командой `tar` осуществляется сборка всех текстовых файлов текущего каталога в архивный файл, после паузы просматривается содержимое файла, затем командой `gzip` архивный файл сжимается. Листинг скрипта и пример выполнения показаны на рисунках 47 – 48.

```
GNU nano 4.8                               lab5.sh
find= find . -type f
archName="task24.tar"
tar -cf $archName $find
sleep 10
tar -tf $archName
gzip $archName

[ Read 6 lines ]
Get Help  Write Out  Where Is  Cut Text  Justify  Cur Pos  Undo
Exit      Read File  Replace  Paste Text  To Spell  Go To Line  Redo
```

Рисунок 47 – Листинг скрипта для задания 24

```
artem@artemserver:~$ sh lab5.sh
./bashrc
./sudo_as_admin_successful
./nowdate.sh
./ls
./result_new
./sort_result
./bash_logout
./errloop.log
./newdir/newchanel
./newdir/newdir2/file2.txt
./newdir/file1.txt
./viminfo
./loop3
./loop2
./loop
./result
./9.sh
./selected_editor
./profile
./nowdatelogs.log
./nowloop.log
./cache/motd.legal-displayed
./test
./notsort
./bash_history
./dirWithText/1
./dirWithText/3
./dirWithText/2
./dirWithText/task14.sh
./errnowdate.log
./lab5.sh
./task23result
artem@artemserver:~$
```

Рисунок 48 – Пример выполнения задания 24

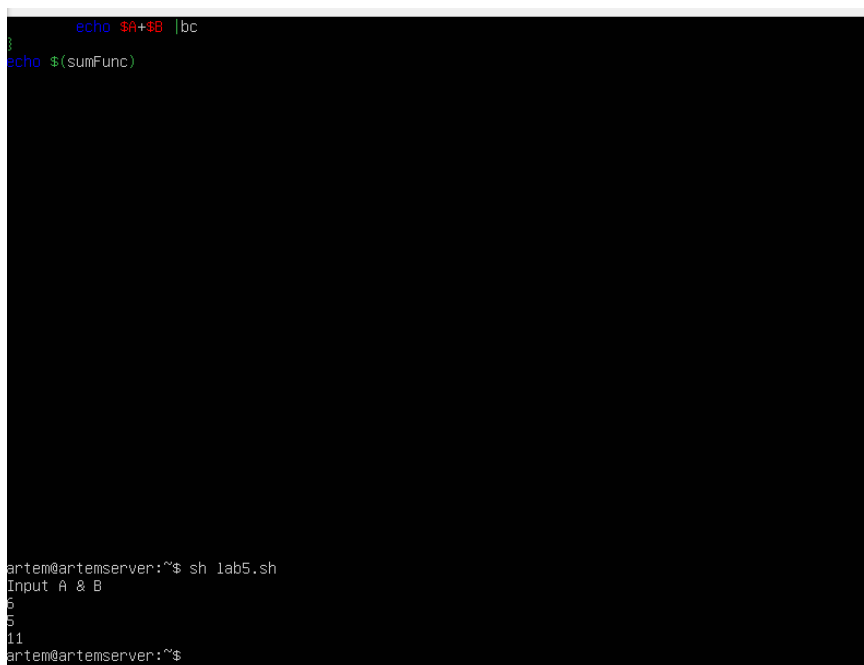
25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных

С помощью `read` вводим значения цифр для суммирования. Затем создаем функцию и вызываем её. Листинг скрипта и пример выполнения показаны на рисунках 49 – 50.



```
GNU nano 4.8 lab5.sh
echo "Input A & B"
read A
read B
sumFunc(){
    echo $A+$B |bc
}
echo $(sumFunc)_
```

Рисунок 49 – Листинг скрипта для задания 25



```
artem@artemserver:~$ sh lab5.sh
Input A & B
5
11
16
artem@artemserver:~$
```

Рисунок 50 – Пример выполнения задания 25

Вывод

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