

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

Отчет по лабораторной работе № 5

«Программирование на SHELL в ОС семейства UNIX»

по курсу «ОС Linux»

Студент
Группа ПМ-18

Полухина Е.Д.

Руководитель

Кургасов В.В.

Липецк 2020 г.

СОДЕРЖАНИЕ

Цель работы	4
Задание.....	5
Ход работы	8
1. Вывод информационных сообщений с использованием команд echo, printf.	8
2. Присвоение переменной A целочисленного значения.	9
3. Присваивание переменной B значения переменной A.....	10
4. Присваивание переменной C значения «путь до своего каталога».	11
5. Присваивание переменной D значения «имя команды» - команды date.	12
6. Присваивание переменной E значения «имя команды» просмотра содержимого файла.	13
7. Присваивание переменной F значения «имя команды» сортировки содержимого текстового файла.	14
8. Запрос программой значения переменной, а затем вывод значения этой переменной.	15
9. Запрос имени пользователя и приветствие.	16
10. Запрос значений двух переменных и алгебраические вычисления над ними.	17
11. Вычисление объема цилиндра.	18
12. Использование позиционных параметров.	19
13. Отображение содержимого текстового файла.	20
14. Отображение содержимого текстовых файлов текущего каталога, используя for.....	21
15. Запрос ввода числа и сравнение его с допустимым значение.	22

16. Определение високосного года.	23
17. Входят ли значения в диапазон данных.	25
18. Проверка введенного пароля.	27
19. Проверка существования файла.	29
20. Получение информации о процессах bash.	30
21. Анализ атрибутов файлов.	32
22. Проверка наличия файла запуска программы.	34
23. Анализ размера файла.	35
24. Использование tar и gzip.	36
25. Скрипт с использованием функции.....	37
Вывод	38

Цель работы

Целью работы является знакомство с программированием на SHELL в ОС семейства UNIX.

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. Вывод информационных сообщений с использованием команд `echo`, `printf`.

Код скрипта:

```
echo "Be the best version of you #вывод текста  
printf "Now or never" #вывод текста
```

Команда `echo` - это простая и в то же время часто используемая встроенная команда оболочки `bash`. Она имеет только одно назначение - выводить строку текста в терминал, но применяется очень часто в различных скриптах, программах, и даже для редактирования конфигурационных файлов.

Чтобы лучше контролировать форматирование вывода, можно использовать команду `printf`.



```
user@katyushaserver:~/lr5$ sh 1.sh  
Be the best version of you  
Now or neveruser@katyushaserver:~/lr5$
```

Рисунок 1 – Результат выполнения скрипта

Команда `echo` добавляет символы `"\n"` в конец строки. То есть, когда мы печатаем что-то с помощью команды `echo`, она автоматически переходит в новую строку, а `printf` этого не делает.

2. Присвоение переменной A целочисленного значения.

Код скрипта:

```
A=1 #присваивание переменной A значения 1
```

```
echo "A = $A" #вывод значения переменной A
```

Доступ к переменной осуществляется по имени (со знаком \$ перед именем).



```
user@katyushaserver:~/lr5$ sh 2.sh
A = 1
user@katyushaserver:~/lr5$
```

Рисунок 2 – Результат выполнения скрипта

3. Присваивание переменной В значения переменной А.

Код скрипта:

```
B=1 #присваивание переменной В значения 1  
A=2 #присваивание переменной А значения 2  
B=$A #присваивание переменной В значения переменной А  
echo "B = $B" #вывод значения переменной В на экран
```



```
user@katyushaserver:~/lr5$ sh 3.sh  
B = 2  
user@katyushaserver:~/lr5$
```

Рисунок 3 – Результат выполнения скрипта

4. Присваивание переменной C значения «путь до своего каталога».

Код скрипта:

```
C=~`#присваивание переменной C значения "путь до своего каталога"
```

```
cd $C #переход в директорию с названием, которое является значением переменной C
```

Знак “~” означает домашнюю директорию. С помощью команды `cd` переходим в нее.



```
user@katyushaserver:~/lr5$ sh 4.sh
4.sh: 1: /home/user: Permission denied
user@katyushaserver:~/lr5$
```

Рисунок 4 – Результат выполнения скрипта

5. Присваивание переменной D значения «имя команды» - команды date.

Код скрипта:

```
D="date" #присваивание имя команды date  
echo ` $D ` #вывод текущей даты
```

При каждом запуске скрипта будет показываться текущие дата и время, потому что в переменной D хранится имя команды, а не результат выполнения команды.



```
user@katyushaserver:~/lr5$ sh 5.sh  
Fri Nov 27 13:47:07 UTC 2020  
user@katyushaserver:~/lr5$
```

Рисунок 5 – Результат выполнения скрипта

6. Присваивание переменной E значения «имя команды» просмотра содержимого файла.

Код скрипта:

```
E="cat" #присваивание переменной D имя команды cat  
echo ` $E meow.txt ` #вывод содержимого файла
```

Вывод на экран произошел с помощью выполнения команды, названием которой является значение переменной E, а параметр задается вручную как meow.txt.



```
user@katyushaserver:~/lr5$ sh 6.sh  
meow-meow  
user@katyushaserver:~/lr5$
```

Рисунок 6 – Результат выполнения скрипта

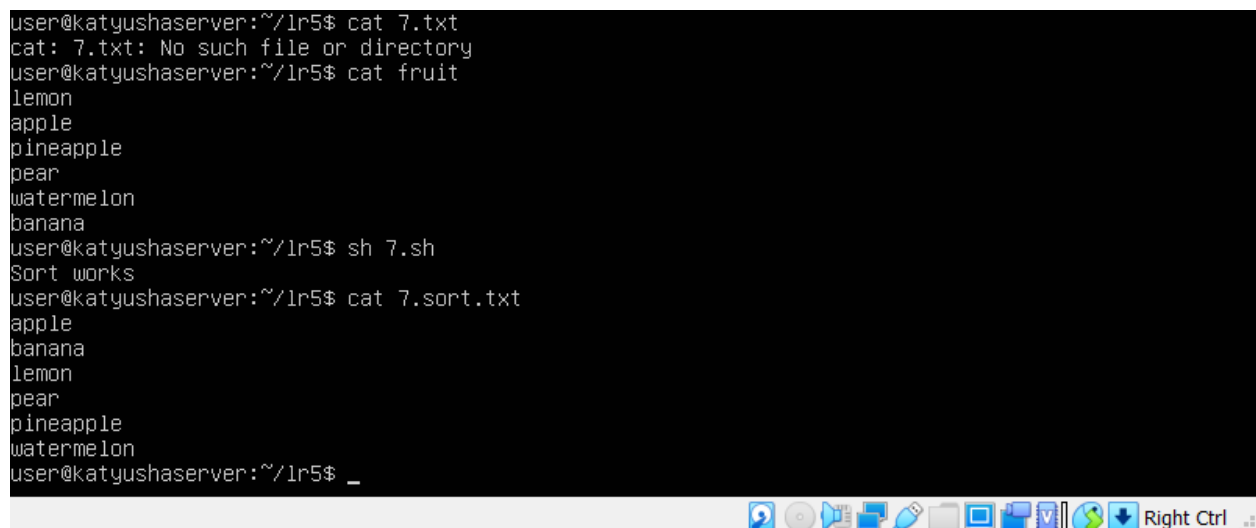
7. Присваивание переменной F значения «имя команды» сортировки содержимого текстового файла.

Код скрипта:

```
F="sort" #присваивание переменной F имя команды "sort"
`$F fruit > 7.sort.txt` #выполнение сортировки и ее
перенаправление на вывод в файл
echo "Sort works" #сообщение об успешном выполнении команды
```

Выполнили команду sort, используя обратные кавычки и обращение к переменной F, а затем сделали перенаправление вывода отсортированного файла fruit в файл 7.sort.txt.

На рисунке 7 показано содержимое исходного файла fruit, затем запуск скрипта, а затем содержимое файла 7.sort.txt, в который был перенаправлен ВЫВОД.



```
user@katyushaserver:~/lr5$ cat 7.txt
cat: 7.txt: No such file or directory
user@katyushaserver:~/lr5$ cat fruit
lemon
apple
pineapple
pear
watermelon
banana
user@katyushaserver:~/lr5$ sh 7.sh
Sort works
user@katyushaserver:~/lr5$ cat 7.sort.txt
apple
banana
lemon
pear
pineapple
watermelon
user@katyushaserver:~/lr5$ _
```

Рисунок 7 – Результат выполнения скрипта

8. Запрос программой значения переменной, а затем вывод значения этой переменной.

Код скрипта:

```
echo "Enter a number" #запрос на ввод числа, выведенный на
экран
read A #считывается одна строка данных из стандартного потока
ввода
echo "Entered number: $A" #вывод на экран приветствия и
значения переменной name
```



```
user@katyushaserver:~/1r5$ sh 8.sh
Enter a number
5
Entered number: 5
user@katyushaserver:~/1r5$ _
```

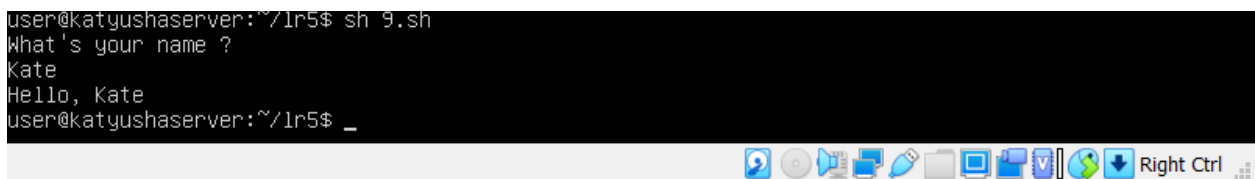
Рисунок 8 – Вывод информации о дисковой памяти

9. Запрос имени пользователя и приветствие.

Код скрипта:

```
echo "What's your name ?" #вывод запроса имени на экран
read name #считывается одна строка данных из стандартного
потока ввода
echo "Hello, $name" #вывод на экран приветствия и значения
переменной name
```

При одновременном выводе на экран приветствия «Hello, » и значения переменной name, получается предложение «Hello, “name”».



```
user@katyushaserver:~/lr5$ sh 9.sh
What's your name ?
Kate
Hello, Kate
user@katyushaserver:~/lr5$ _
```

Рисунок 9 – Результат выполнения скрипта

10. Запрос значений двух переменных и алгебраические вычисления над ними.

Код скрипта:

```
#вывод на экран сообщения о вводе данных
echo "Enter the variables"

#предложение ввести значение переменной A
echo "A:"

#считывается одна строка данных из стандартного потока ввода
read A

#предложение ввести значение переменной B
echo "B:"

#считывается одна строка данных из стандартного потока ввода
read B

#присваивание переменной minus результата выполнения команды
вычитания значения A из значения B
minus=`expr $A -$B`

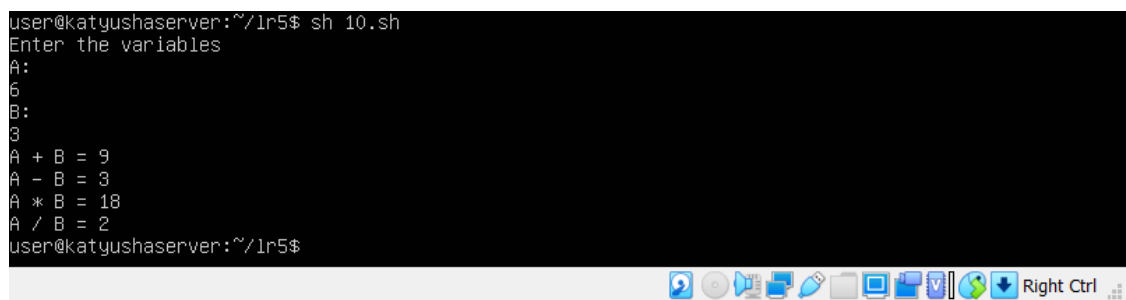
#присваивание переменной division результата выполнения
команды деления значения A на значение B
division=`echo "$A/$B" | bc`

#вывод на экран результата выполнения команды expr сложения
echo "A + B = `expr $A + $B`"

#вывод на экран значения переменной minus echo "A - B =
$minus"

#вывод на экран результата выполнения команды expr умножения
echo "A * B = `expr $A \* $B`"

#вывод на экран значения переменной division
echo "A / B = $division"
```



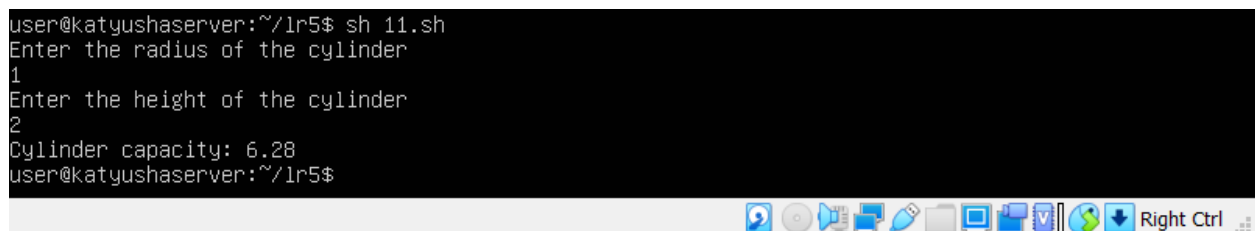
```
user@katyushaserver:~/1r5$ sh 10.sh
Enter the variables
A:
6
B:
3
A + B = 9
A - B = 3
A * B = 18
A / B = 2
user@katyushaserver:~/1r5$
```

Рисунок 10 – Результат выполнения скрипта

11. Вычисление объема цилиндра.

Код скрипта:

```
echo "Enter the radius of the cylinder" #вывод на экран
предложения ввести радиус
read radius #считывается одна строка данных из стандартного
потока ввода
echo "Enter the height of the cylinder" #вывод на экран
предложения ввести высоту
read height #считывается одна строка данных из стандартного
потока ввода
pi=3.14 #присваивание переменной значения 3.14
capacity=`echo "pi*$radius^2*height" | bc` #присваивание
переменной значения выполненной команды вычисления объема
echo "Cylinder capacity: $capacity" #вывод на экран значения
переменной capacity
```



```
user@katyushaserver:~/lr5$ sh 11.sh
Enter the radius of the cylinder
1
Enter the height of the cylinder
2
Cylinder capacity: 6.28
user@katyushaserver:~/lr5$
```

Рисунок 11 – Результат выполнения скрипта

12. Использование позиционных параметров.

Код скрипта:

```
#вывод на экран названия скрипта с помощью параметра $0
echo "This is the script name : $0"

#вывод на экран количества аргументов командной строки с
помощью параметра $#
echo "Number of command-line arguments : $#"
```

```
#вывод на экран значения каждого аргумента командной строки
с помощью параметра $*
echo "Command-line arguments : $*"
```

Если информация передается shell-сценарию, с помощью позиционных параметров можно получить доступ к ней. Каждый параметр передает сценарию заданное количество ссылок. Можно передавать произвольное число аргументов, но доступными являются только первые девять из них.

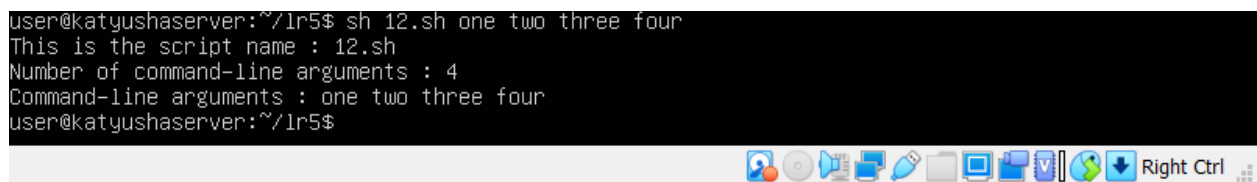
A screenshot of a terminal window. The prompt is 'user@katyushaserver:~/lr5\$'. The user has entered 'sh 12.sh one two three four'. The output of the script is displayed on the next three lines: 'This is the script name : 12.sh', 'Number of command-line arguments : 4', and 'Command-line arguments : one two three four'. The prompt returns to 'user@katyushaserver:~/lr5\$'. The terminal window has a dark background and a light-colored title bar. At the bottom of the window, there is a taskbar with various icons and the text 'Right Ctrl'.

Рисунок 12 – Результат выполнения скрипта

13. Отображение содержимого текстового файла.

Код скрипта:

```
echo `cat $1` #вывод на экран результата выполнения команды  
просмотра содержимого файла  
sleep 5 #пауза 5 секунд  
clear #очищение экрана
```

Получаем название файла, используя позиционный параметр \$1.

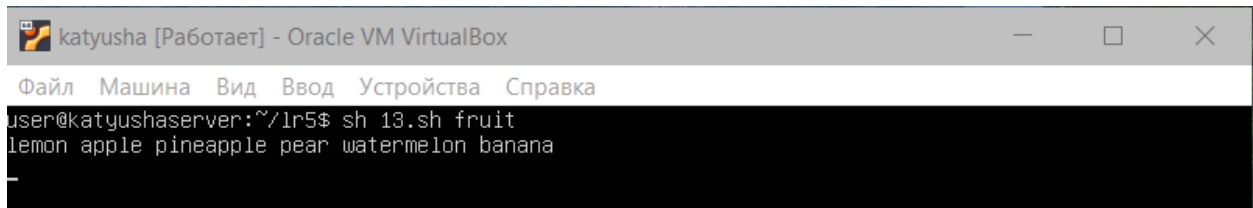


Рисунок 13 – Результат выполнения скрипта

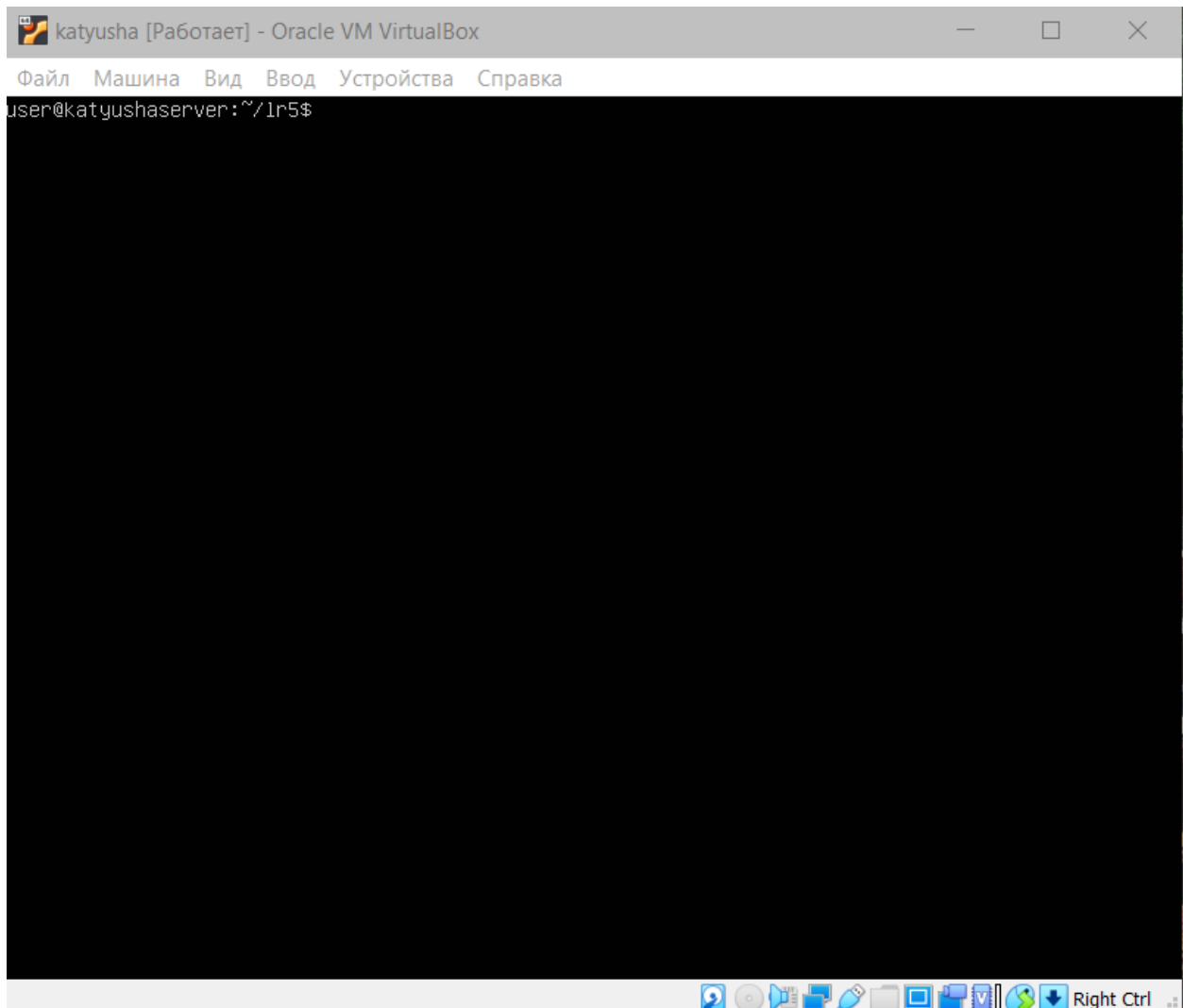


Рисунок 14 – Очищение экрана после паузы

14. Отображение содержимого текстовых файлов текущего каталога, используя for.

Код скрипта:

```
#начало цикла для всех файлов текущего каталога,
оканчивающихся на ".txt"
for file in *.txt
#выполнение цикла
do
    #вывод на экран значения переменной file
    echo "File : $file"
    #выполнение команды просмотра содержимого файла поэкранно
    cat $file | less
    #вывод на экран нижнего подчеркивания для визуального
отделения файлов
    echo "_____"
#завершение цикла
done
```

Less позволяет просматривать содержимое файлов поэкранно.



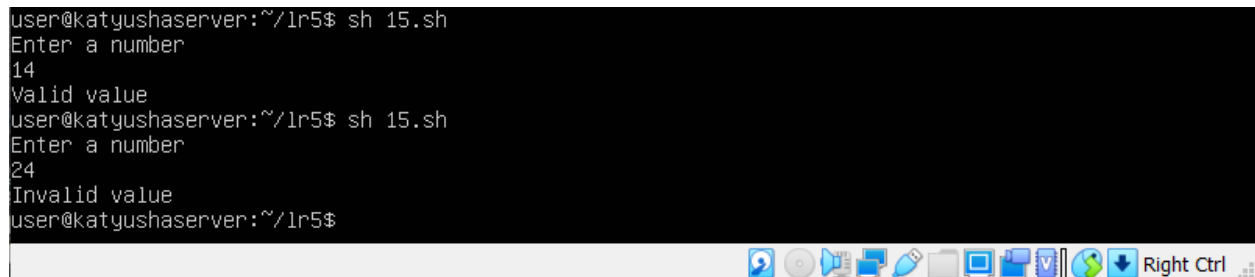
```
user@katyushaserver:~/lr5$ sh 14.sh
File : 7.sort.txt
apple
banana
lemon
pear
pineapple
watermelon
-----
File : meow.txt
meow-meow
-----
File : output.txt
apple
banana
lemon
pear
pineapple
watermelon
-----
user@katyushaserver:~/lr5$ _
```

Рисунок 15 – Результат выполнения скрипта

15. Запрос ввода числа и сравнение его с допустимым значением.

Код скрипта:

```
echo "Enter a number" #вывод на экран запроса ввести число
read A #считывается одна строка данных из стандартного потока
ввода
valid=20 #присваивание переменной значения
if [ "$A" -gt "$valid" ] # условие, если значение переменной
A меньше или равно допустимого
then #тогда выполняется
    echo "Invalid value" #вывод сообщения на экран
else #иначе
    echo "Valid value" #вывод сообщение на экран
fi #конец if
```



```
user@katyushaserver:~/lr5$ sh 15.sh
Enter a number
14
Valid value
user@katyushaserver:~/lr5$ sh 15.sh
Enter a number
24
Invalid value
user@katyushaserver:~/lr5$
```

Рисунок 16 – Результат выполнения скрипта

16. Определение високосного года.

Код скрипта:

```
echo "Enter the year" #вывод на экран запроса ввести год
read year #считывается одна строка данных из стандартного
потокa ввода
ost4=$((year % 4)) #вычисление остатка от деления
ost100=$((year % 100)) #вычисление остатка от деления
ost400=$((year % 400)) #вычисление остатка от деления
if [ "$ost4" -eq "0" ] && [ "$ost100" -eq "0" ] && [ "$ost400"
-eq "0" ] #проверка, високосный ли год
then #тогда
    echo "Leap year" #вывести на экран сообщение о том, что год
високосный
elif [ "$ost4" -eq "0" ] && [ "$ost100" -ne "0" ] #иначе
другая проверка, високосный ли год
then #тогда
    echo "Leap year" #вывести на экран сообщение о том, что год
високосный
else
    echo "Common year" #вывести на экран сообщение о том, что
год обычный
fi #конец if
```

Знаки сравнения:

-eq – равно

-ne – не равно

Проверка, високосный ли год:

- Если год не делится на 4, значит он обычный.
- Иначе надо проверить не делится ли год на 100.
- Если не делится, значит это не столетие и можно сделать вывод, что год високосный.
- Если делится на 100, значит это столетие и его следует проверить его делимость на 400.

- Если год делится на 400, то он високосный.

```
user@katyushaserver:~/1r5$ sh 16.sh
Enter the year
2020
Leap year
user@katyushaserver:~/1r5$ sh 16.sh
Enter the year
2018
Common year
user@katyushaserver:~/1r5$ _
```

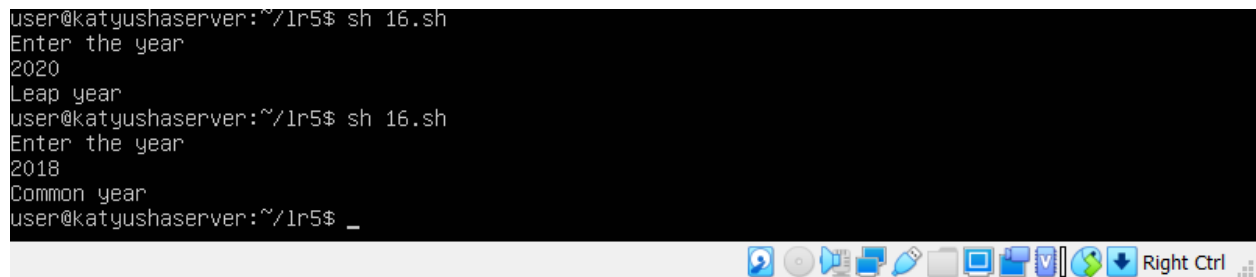


Рисунок 17 – Результат выполнения скрипта

17. Входят ли значения в диапазон данных.

Код скрипта:

```
echo "Enter the variables" #вывод на экран запроса ввести значения
echo "Enter A" #вывод на экран запроса ввести A
read A #считывается одна строка данных из стандартного потока ввода
echo "Enter B" #вывод на экран запроса ввести B
read B #считывается одна строка данных из стандартного потока ввода
echo "Enter the left edge" #вывод на экран запроса ввести левую границу
read left #считывается одна строка данных из стандартного потока ввода
echo "Enter the right edge" #вывод на экран запроса ввести правую границу
read right #считывается одна строка данных из стандартного потока ввода
A1=`echo "$A+1" | bc`#присваивание переменной A инкрементированного значения A
B1=`echo "$B+1" | bc` `#присваивание переменной B инкрементированного значения B
if [ $A -ge $left ] && [ $B -ge $left ] && [ $A -le $right ] && [ $B -le $right ] #проверка, входят ли значения в диапазон
then #тогда выполняем
    echo "A = " $A1 #вывод на экран значения переменной A
    echo "B = " $B1 #вывод на экран значения переменной B
else #иначе
    echo "Error" #вывод на экран сообщения об ошибке
fi #конец if
```

Знаки сравнения:

-ge – больше или равно

-le – меньше или равно

18. Проверка введенного пароля.

Код скрипта:

```
password=12345 #присваивание переменной значения
echo "Enter the password" #вывод на экран запроса ввести
пароль
read passw #считывается одна строка данных из стандартного
потока ввода
if [ "$password" -eq $passw ] #проверка, совпадает ли пароль
then #тогда
    cd /etc #переход в каталог /etc
    ls -al /etc | less #постраничное отображение содержимого
каталога /etc
else #иначе
    echo "Incorrect password" #вывод на экран сообщения об ошибке
fi #конец if
```

В команде «ls -al /etc | less» параметры -al позволяют смотреть содержимое в длинном формате и со скрытыми файлами, а less делает доступным постраничный просмотр.



Рисунок 19 – Результат выполнения скрипта при правильном пароле

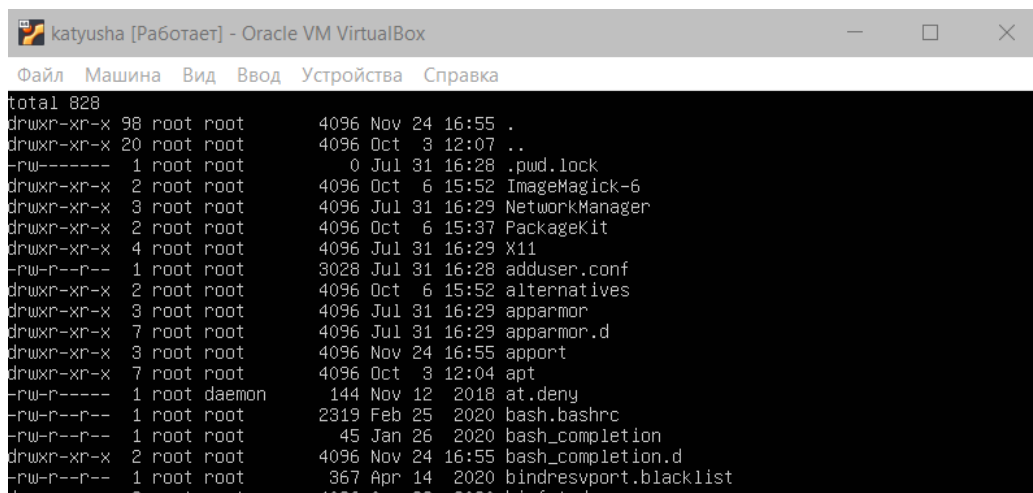


Рисунок 20 – Содержимое каталога /etc

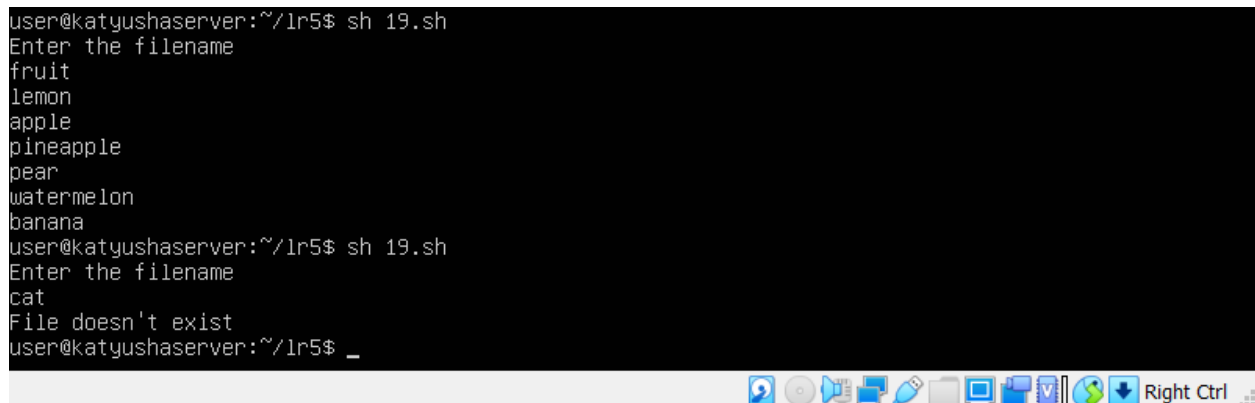


Рисунок 21 – Результат выполнения при неправильном пароле

19. Проверка существования файла.

Код скрипта:

```
echo "Enter the filename" #вывод на экран запроса ввести
название файла
read name #считывается одна строка данных из стандартного
потока ввода
if test -f "$name" #если файл существует
then #тогда выполняется
    cat $name #просмотр содержимого найденного файла
else #иначе
    echo "File doesn't exist" # вывод на экран сообщения, что
файла не существует
fi #конец if
```



```
user@katyushaserver:~/lr5$ sh 19.sh
Enter the filename
fruit
lemon
apple
pineapple
pear
watermelon
banana
user@katyushaserver:~/lr5$ sh 19.sh
Enter the filename
cat
File doesn't exist
user@katyushaserver:~/lr5$ _
```

Рисунок 22 – Результат выполнения скрипта при различных введенных данных

20. Получение информации о процессах bash.

```
user@katyushaserver:~/lr5$ ls
1.sh      11.sh  14.sh  17.sh  2.sh   22.sh  25.sh  5.sh   7.sort.txt  apple  my.tar  rf
10.sh     12.sh  15.sh  18.sh  20.sh  23.sh  3.sh   6.sh   8.sh        fruit  my.tar.gz  what
10.sh.save 13.sh  16.sh  19.sh  21.sh  24.sh  4.sh   7.sh   9.sh        meow.txt output.txt
user@katyushaserver:~/lr5$ sh 20.sh
Enter the filename
apple
total 8
drwxrwxr-x 2 user user 4096 Nov 26 14:21 .
drwxrwxr-x 4 user user 4096 Nov 27 16:59 ..
This is the directory
user@katyushaserver:~/lr5$ sh 20.sh
Enter the filename
lemon
Create the directory
user@katyushaserver:~/lr5$ ls
1.sh      12.sh  16.sh  2.sh   23.sh  4.sh   7.sort.txt  fruit  my.tar.gz
10.sh     13.sh  17.sh  20.sh  24.sh  5.sh   8.sh        lemon  output.txt
10.sh.save 14.sh  18.sh  21.sh  25.sh  6.sh   9.sh        meow.txt rf
11.sh     15.sh  19.sh  22.sh  3.sh   7.sh   apple       my.tar  what
user@katyushaserver:~/lr5$ sh 20.sh
Enter the filename
fruit
lemon
apple
pineapple
pear
watermelon
banana
user@katyushaserver:~/lr5$
```

Рисунок 23 – Результат выполнения скрипта при различных введенных данных

Код скрипта:

```
echo "Enter the filename" #вывод на экран запроса ввести
название файла
read name #считывается одна строка данных из стандартного
потока ввода
read name
if [ -d "$name" ] && [ -r "$name" ] #проверка, является ли
файл каталогом и доступен ли для чтения
then #тогда выполняется
    ls -al "$name" #просмотр содержимого каталога
    echo "This is the directory" # вывод на экран сообщения о
том, что это директория
elif [ ! -f "$name" ] #иначе если файл не является обычным
файлом
then #тогда выполняется
```

```
    echo "Create the directory" #вывод на экран сообщения о том,  
    что директория создана  
else #иначе  
    cat "$name" #вывод содержимого файла  
fi #конец if
```

21. Анализ атрибутов файлов.

Код скрипта:

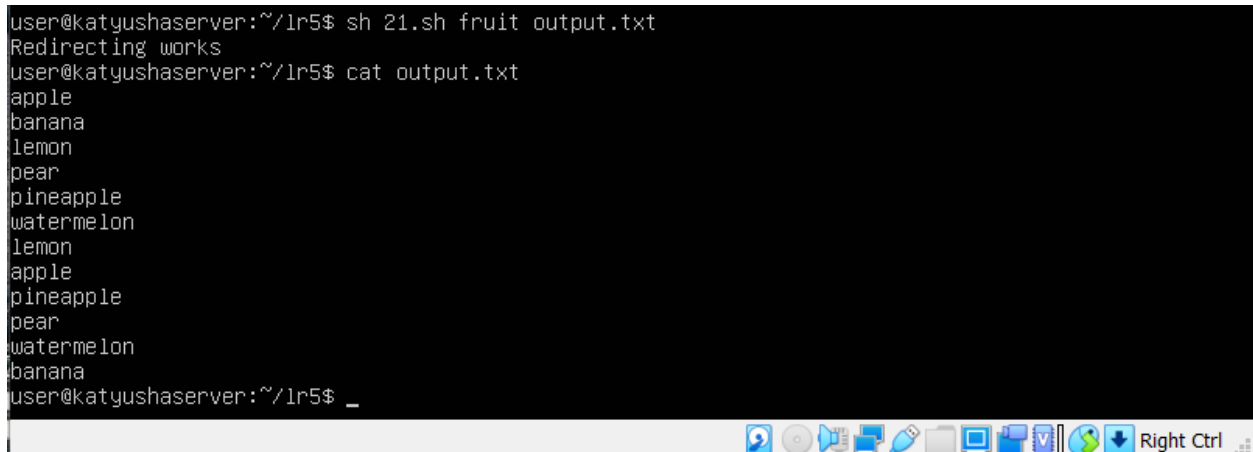
```
#проверка, если оба файла существуют, являются обычными, и
один из них доступен для чтения, а другой для записи
if [ -f "$1" ] && [ -r "$1" ] && [ -f "$2" ] && [ -w "$2" ]
#тогда выполняется
then
    #содержимое первого файла перенаправляется во второй файл
    cat $1 >> $2
    #вывод на экран сообщения о том, что перенаправление прошло
    успешно
    echo "Redirecting works"
fi #конец if
#проверка, если первый файл не существует
if [ -! -f "$1" ]
then
    #вывод на экран сообщения, что файла не существует
    echo "File $1 doesn't exist"
fi #конец if
#проверка, если второй файл не существует
if [ -! -f "$2" ] #проверка, если второй файл не существует
then
    #вывод на экран сообщения, что файла не существует
    echo "File $2 doesn't exist"
fi #конец if
#проверка, если первый файл не доступен для чтения и
существует
if [ ! -r "$1" ] && [ -f "$1" ]
then
    #вывод на экран сообщения
    echo "The user doesn't have permission to "read" the file
    $1"
fi #конец if
#проверка, если второй файл не доступен записи и существует
```



```

if [ ! -w "$2" ] && [ -f "$2" ]
then
    #вывод на экран сообщения
    echo "The user doesn't have permission to "write" the file
    $2"
fi #конец if

```

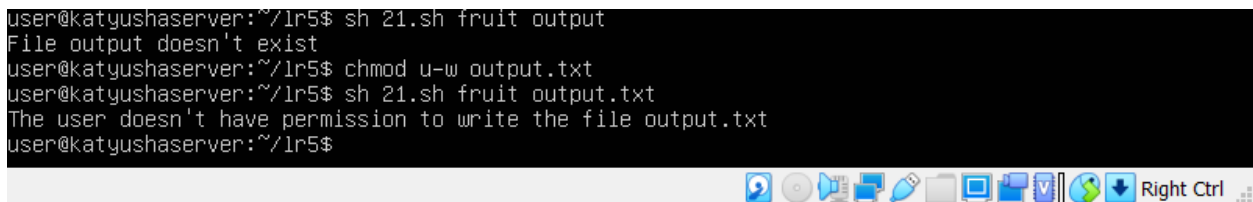


```

user@katyushaserver:~/lr5$ sh 21.sh fruit output.txt
Redirecting works
user@katyushaserver:~/lr5$ cat output.txt
apple
banana
lemon
pear
pineapple
watermelon
lemon
apple
pineapple
pear
watermelon
banana
user@katyushaserver:~/lr5$ _

```

Рисунок 24 – Результат выполнения скрипта при разных введенных данных



```

user@katyushaserver:~/lr5$ sh 21.sh fruit output
File output doesn't exist
user@katyushaserver:~/lr5$ chmod u-w output.txt
user@katyushaserver:~/lr5$ sh 21.sh fruit output.txt
The user doesn't have permission to write the file output.txt
user@katyushaserver:~/lr5$

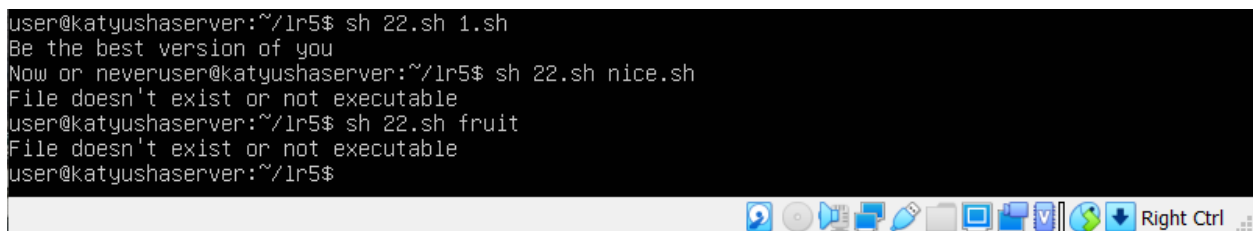
```

Рисунок 25 – Результат выполнения скрипта при разных введенных данных

22. Проверка наличия файла запуска программы.

Код скрипта:

```
if [ -x "$1" ] #проверка, доступен ли файл для исполнения
then
    ./"$1" #запуск файла
else
    echo "File doesn't exist or not executable" #вывод на экран
    сообщения о том, что файла не существует или он не является
    исполняемым
fi #конец if
```



```
user@katyushaserver:~/lr5$ sh 22.sh 1.sh
Be the best version of you
Now or neveruser@katyushaserver:~/lr5$ sh 22.sh nice.sh
File doesn't exist or not executable
user@katyushaserver:~/lr5$ sh 22.sh fruit
File doesn't exist or not executable
user@katyushaserver:~/lr5$
```

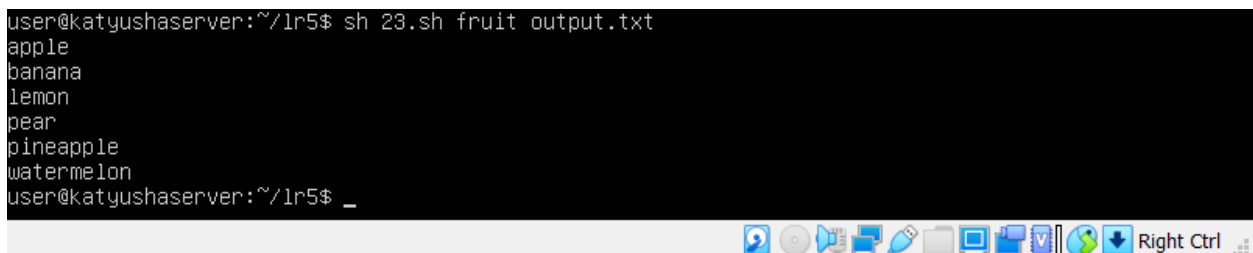
Рисунок 26 – Результат выполнения скрипта

23. Анализ размера файла.

Код скрипта:

```
if [ -s $1 ] #если файл имеет ненулевой размер
then
    sort -k 1 $1 > $2 #информация об отсортированном по первому
    столбцу по возрастанию содержимому первого файла помещается
    во второй файл
    cat $2 #просмотр содержимого второго файла
else
    echo "File doesn't exist or file has a size zero" #вывод на
    экран сообщения о том, что файла не существует или он имеет
    нулевой размер
fi #конец if
```

На рисунке **Error! Reference source not found.** показано получение информации о процессах bash с использованием команды «ps lax | grep bash».



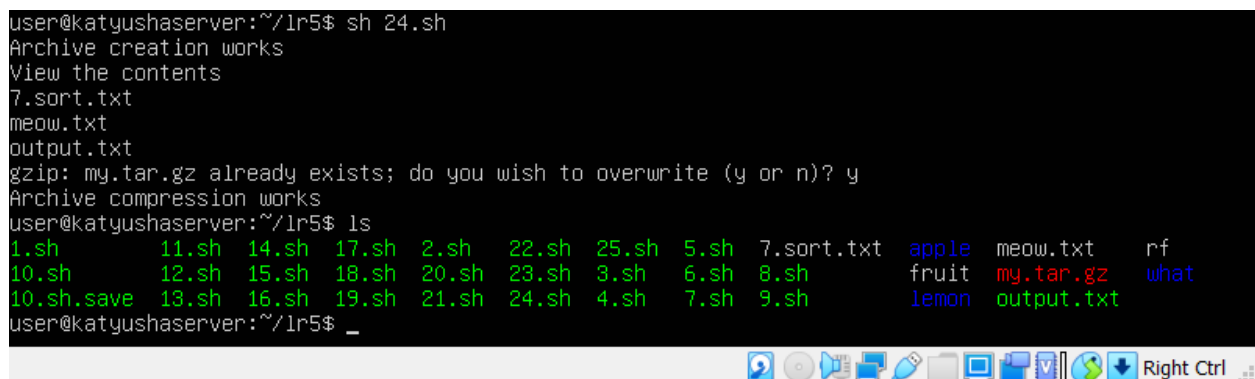
```
user@katyushaserver:~/lr5$ sh 23.sh fruit output.txt
apple
banana
lemon
pear
pineapple
watermelon
user@katyushaserver:~/lr5$ _
```

Рисунок 27 – Результат выполнения скрипта

24. Использование tar и gzip.

Код скрипта:

```
tar -cf my.tar *.txt #архивирование всех текстовых файлов
текущего каталога
echo "Archive creation works" #вывод на экран сообщения о
том, создание архива выполнено успешно
sleep 5 #пауза 5 секунд
echo "View the contents" #вывод на экран сообщения о том,
просматривается содержимое
tar -tf my.tar #просмотр содержимого файла
gzip my.tar #сжатие архивного файла
echo "Archive compression works" #вывод на экран сообщения о
том, сжатие архивного файла выполнено успешно
```



```
user@katyushaserver:~/lr5$ sh 24.sh
Archive creation works
View the contents
7.sort.txt
meow.txt
output.txt
gzip: my.tar.gz already exists; do you wish to overwrite (y or n)? y
Archive compression works
user@katyushaserver:~/lr5$ ls
1.sh      11.sh  14.sh  17.sh  2.sh   22.sh  25.sh  5.sh   7.sort.txt  apple  meow.txt  rf
10.sh     12.sh  15.sh  18.sh  20.sh  23.sh  3.sh   6.sh   8.sh       fruit  my.tar.gz  what
10.sh.save 13.sh  16.sh  19.sh  21.sh  24.sh  4.sh   7.sh   9.sh       lemon  output.txt
```

Рисунок 28 – Результат выполнения скрипта

25. Скрипт с использованием функции.

Код скрипта:

```
minus() #объявление функции
{
echo " $1 - $2 " | bc " #вывод на экран результата выполнения
вычитания
}
minus $1 $2 #запуск функции
```



```
user@katyushaserver:~/lr5$ sh 25.sh 5 3
2
user@katyushaserver:~/lr5$ sh 25.sh -10 -1
-9
user@katyushaserver:~/lr5$ _
```

Рисунок 29 – Результат выполнения скрипта

Вывод

В результате выполнения лабораторной работы я получила знания по программированию на SHELL в ОС Linux Ubuntu. Научилась выводить информацию на экран, присваивать значения переменным и получать доступ к переменным. Также научилась сравнивать значения друг с другом и узнавать, каков тип файла и какие у него права.

На практике совершала алгебраические преобразования с переменными, пользовалась циклом `for`. Узнала назначение позиционных параметров и научилась создавать архив с использованием скрипта.