

# **Отчёт по лабораторной работе №12**

**дисциплина: Операционные системы**

Латаева Гюзелия Андреевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>4</b>	<b>Выводы</b>	<b>17</b>
<b>5</b>	<b>Контрольные вопросы</b>	<b>18</b>
	<b>Список литературы</b>	<b>21</b>

## Список иллюстраций

3.1	Рисунок 1 . . . . .	9
3.2	Рисунок 2 . . . . .	10
3.3	Рисунок 3 . . . . .	10
3.4	Рисунок 4 . . . . .	12
3.5	Рисунок 5 . . . . .	13
3.6	Рисунок 6 . . . . .	13
3.7	Рисунок 7 . . . . .	14
3.8	Рисунок 8 . . . . .	14
3.9	Рисунок 9 . . . . .	15
3.10	Рисунок 10 . . . . .	15
3.11	Рисунок 11 . . . . .	16

## Список таблиц

# 1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 2 Задание

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами:

- `-iinputfile` — прочитать данные из указанного файла;
- `-ooutputfile` — вывести данные в указанный файл;
- `-r`шаблон — указать шаблон для поиска;
- `-C` — различать большие и малые буквы;
- `-n` — выдавать номера строк.

а затем ищет в указанном файле нужные строки, определяемые ключом `-r`.

2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в `o` коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.

3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например `1.tmp`, `2.tmp`, `3.tmp`, `4.tmp` и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

4. Написать командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду `find`)

## 3 Выполнение лабораторной работы

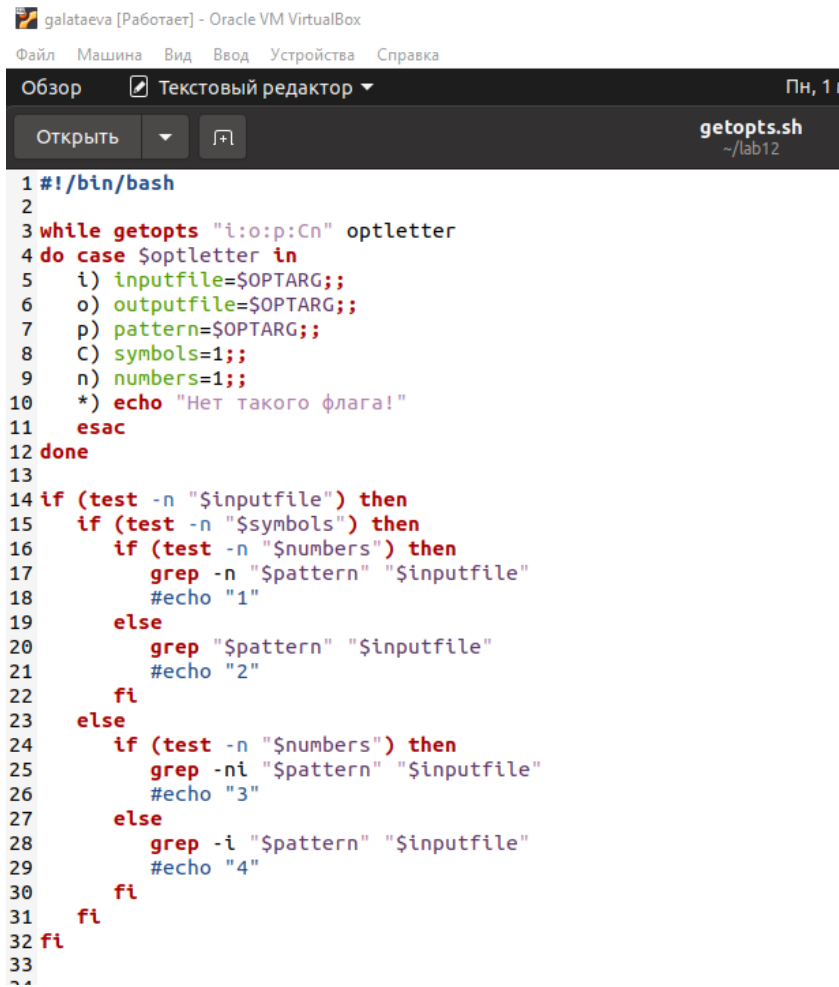
### Задание 1.

1. Создала файл `getopts.sh` и устанавливаю ему права на выполнение.
2. Написала командный файл, который анализирует командную строку с ключами:

- `-iinputfile` — прочитать данные из указанного файла;
- `-ooutputfile` — вывести данные в указанный файл;
- `-ршаблон` — указать шаблон для поиска;
- `-С` — различать большие и малые буквы;
- `-n` — выдавать номера строк.

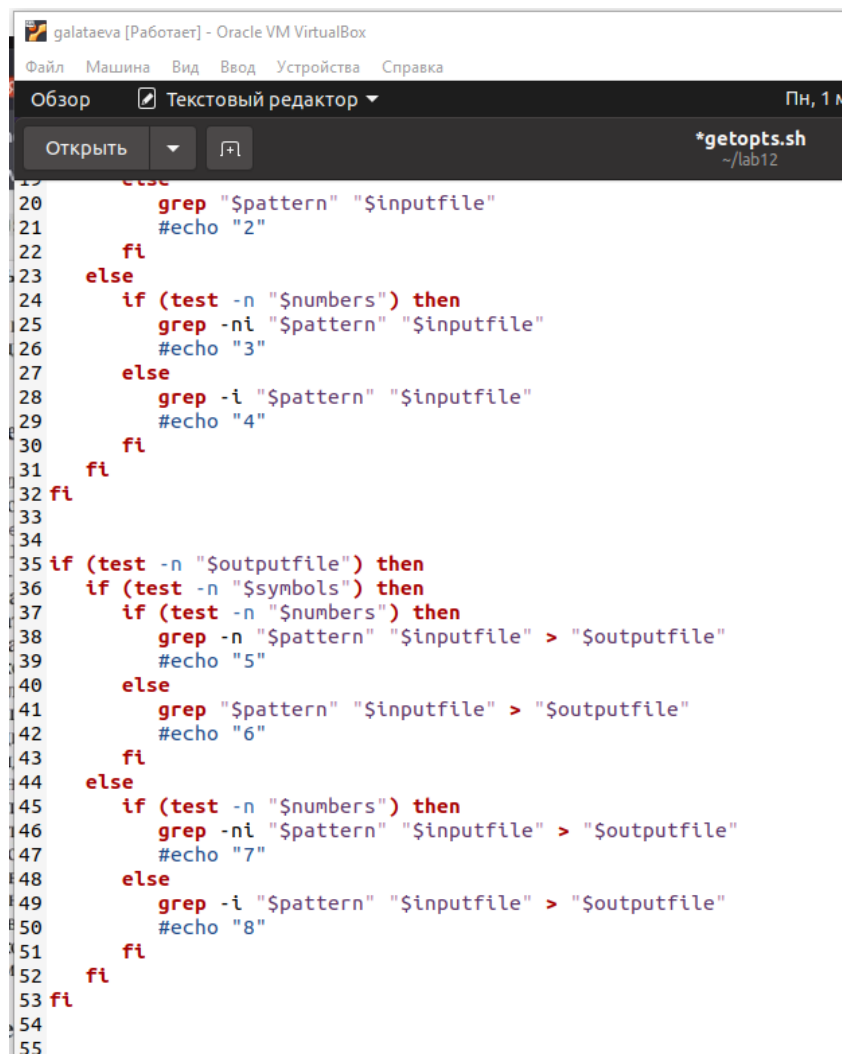
а затем ищет в указанном файле нужные строки, определяемые ключом `-р.:`  
(рис. 3.1), (рис. 3.2)





```
1 #!/bin/bash
2
3 while getopt "i:o:p:Cn" optletter
4 do case $optletter in
5 i) inputfile=$OPTARG;;
6 o) outputfile=$OPTARG;;
7 p) pattern=$OPTARG;;
8 C) symbols=1;;
9 n) numbers=1;;
10 *) echo "Нет такого флага!"
11 esac
12 done
13
14 if (test -n "$inputfile") then
15 if (test -n "$symbols") then
16 if (test -n "$numbers") then
17 grep -n "$pattern" "$inputfile"
18 #echo "1"
19 else
20 grep "$pattern" "$inputfile"
21 #echo "2"
22 fi
23 else
24 if (test -n "$numbers") then
25 grep -ni "$pattern" "$inputfile"
26 #echo "3"
27 else
28 grep -i "$pattern" "$inputfile"
29 #echo "4"
30 fi
31 fi
32 fi
33 ~.
```

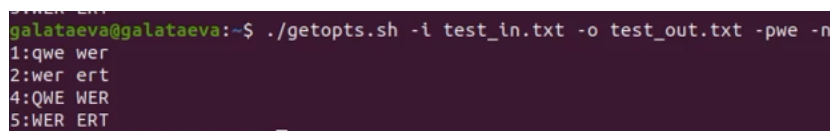
Рис. 3.1: Рисунок 1



```
19 else
20     grep "$pattern" "$inputfile"
21     #echo "2"
22 fi
23 else
24     if (test -n "$numbers") then
25         grep -ni "$pattern" "$inputfile"
26         #echo "3"
27     else
28         grep -i "$pattern" "$inputfile"
29         #echo "4"
30     fi
31 fi
32 fi
33
34
35 if (test -n "$outputfile") then
36     if (test -n "$symbols") then
37         if (test -n "$numbers") then
38             grep -n "$pattern" "$inputfile" > "$outputfile"
39             #echo "5"
40         else
41             grep "$pattern" "$inputfile" > "$outputfile"
42             #echo "6"
43         fi
44     else
45         if (test -n "$numbers") then
46             grep -ni "$pattern" "$inputfile" > "$outputfile"
47             #echo "7"
48         else
49             grep -i "$pattern" "$inputfile" > "$outputfile"
50             #echo "8"
51         fi
52     fi
53 fi
54
55
```

Рис. 3.2: Рисунок 2

3. Результат после выполнения: (рис. 3.3)



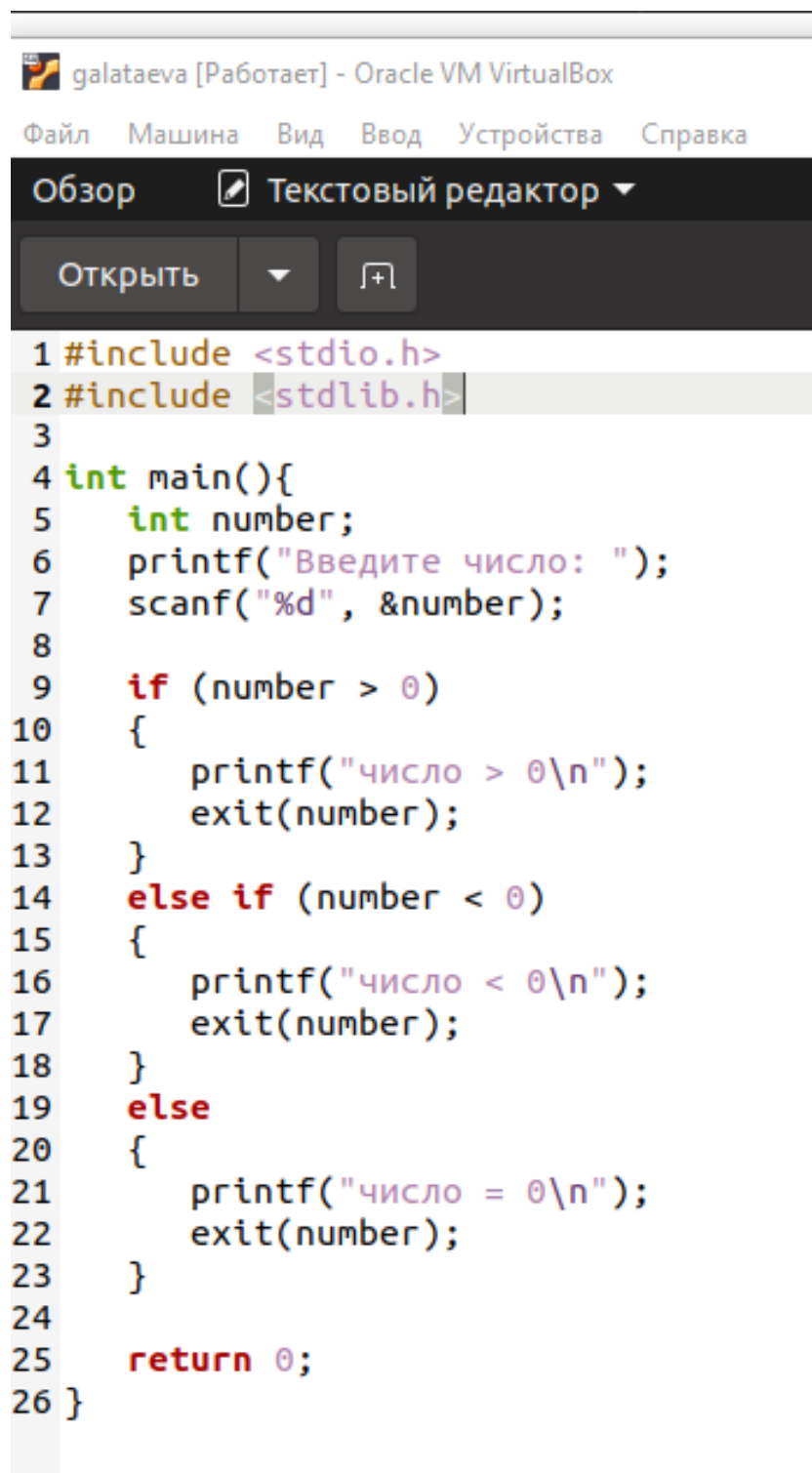
```
galataeva@galataeva:~$ ./getopts.sh -i test_in.txt -o test_out.txt -pwe -n
1:qwe wer
2:wer ert
4:QWE WER
5:WER ERT
```

Рис. 3.3: Рисунок 3

## Задание 2.

1. Создала файл exit\_prog.sh и устанавливаю ему права на выполнение.

2. Написала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. (рис. 3.4). Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено (рис. 3.5):



The screenshot shows a text editor window titled "galataeva [Работает] - Oracle VM VirtualBox". The menu bar includes "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The toolbar has "Обзор", "Текстовый редактор", "Открыть", a dropdown arrow, and a "New File" icon. The code is a C program that prompts the user to enter a number and then checks if it is greater than, less than, or equal to zero.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     int number;
6     printf("Введите число: ");
7     scanf("%d", &number);
8
9     if (number > 0)
10    {
11        printf("число > 0\n");
12        exit(number);
13    }
14    else if (number < 0)
15    {
16        printf("число < 0\n");
17        exit(number);
18    }
19    else
20    {
21        printf("число = 0\n");
22        exit(number);
23    }
24
25    return 0;
26 }
```

Рис. 3.4: Рисунок 4

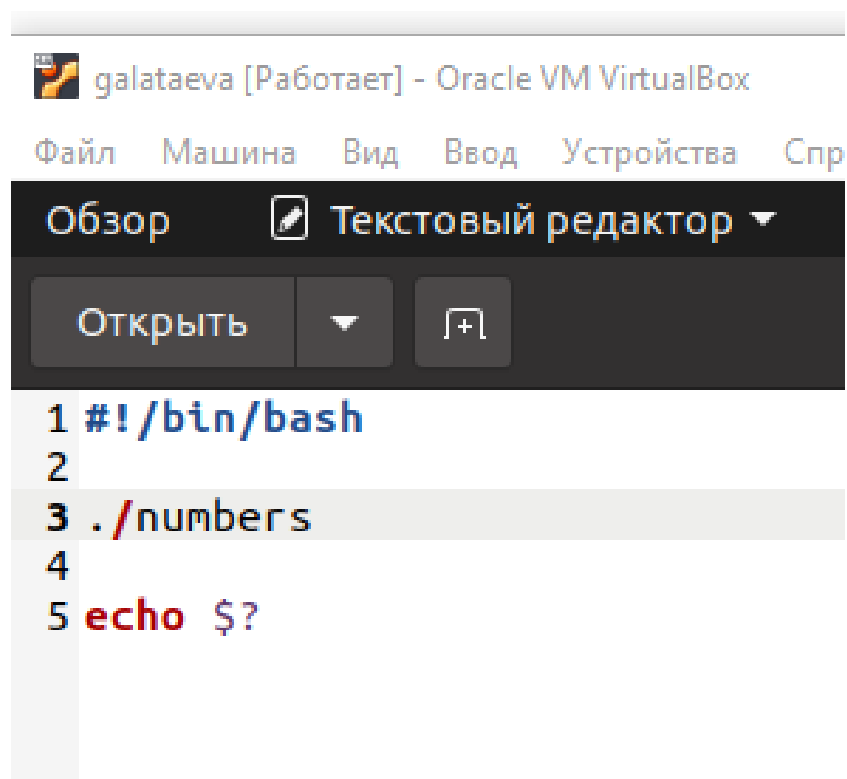


Рис. 3.5: Рисунок 5

3. Выполнила скрипт и получила результат: (рис. 3.6)

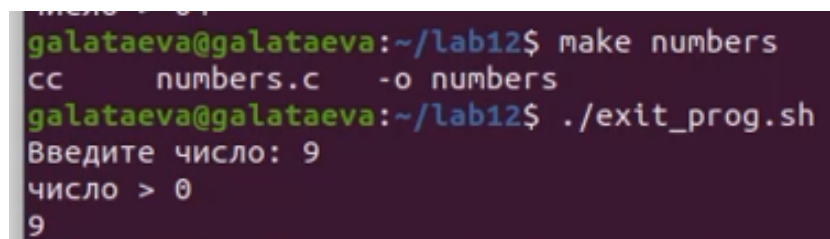
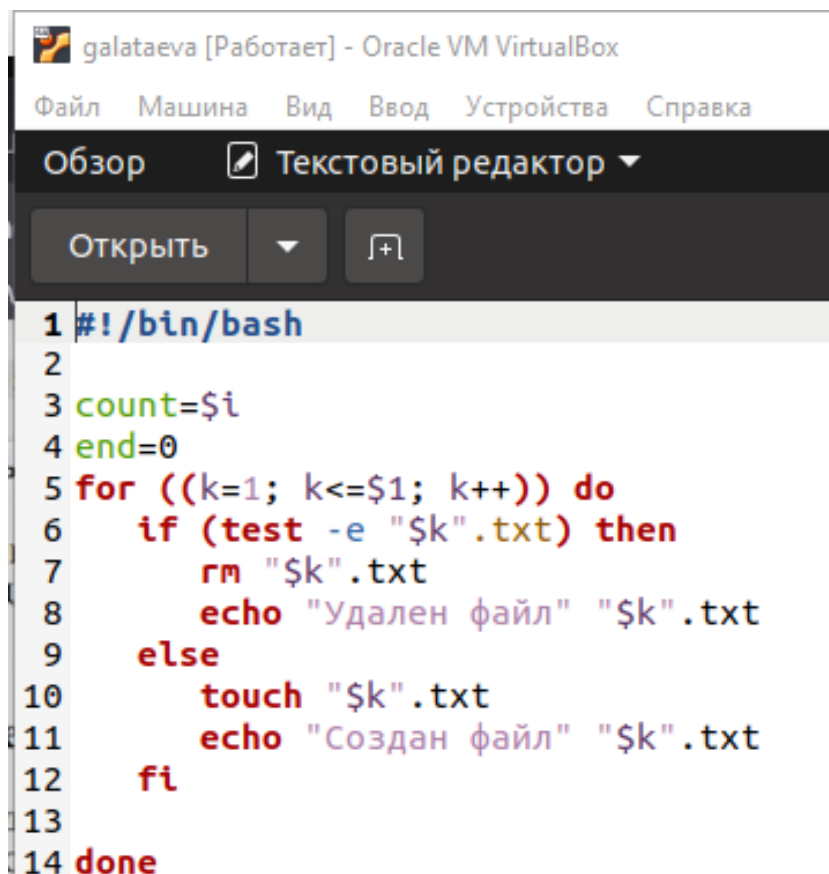


Рис. 3.6: Рисунок 6

### Задание 3.

1. Создала файл `create_file.sh` и устанавливаю ему права на выполнение.
2. Написала командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp,

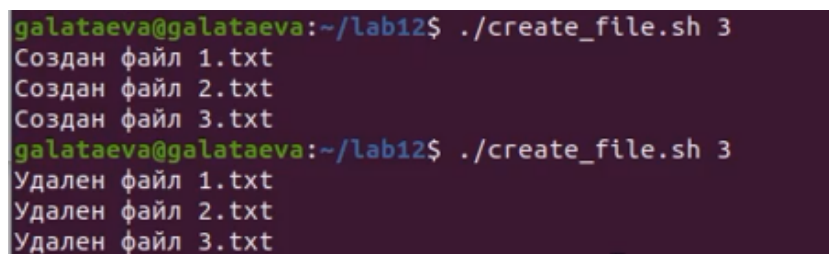
3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).: (рис. 3.7)



```
1 #!/bin/bash
2
3 count=$1
4 end=0
5 for ((k=1; k<=$1; k++)) do
6     if (test -e "$k".txt) then
7         rm "$k".txt
8         echo "Удален файл" "$k".txt
9     else
10        touch "$k".txt
11        echo "Создан файл" "$k".txt
12    fi
13
14 done
```

Рис. 3.7: Рисунок 7

3. Выполнила скрипт и получила результат: (рис. 3.8):

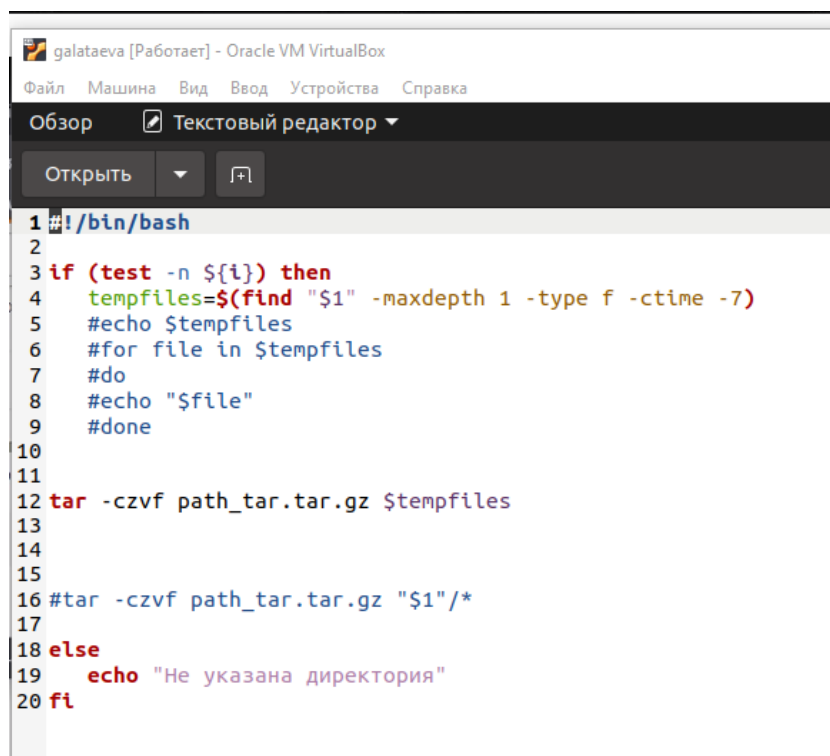


```
galataeva@galataeva:~/lab12$ ./create_file.sh 3
Создан файл 1.txt
Создан файл 2.txt
Создан файл 3.txt
galataeva@galataeva:~/lab12$ ./create_file.sh 3
Удален файл 1.txt
Удален файл 2.txt
Удален файл 3.txt
```

Рис. 3.8: Рисунок 8

#### Задание 4.

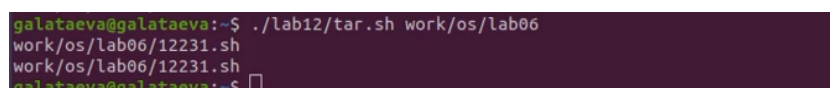
1. Создала файл tar.sh и устанавливаю ему права на выполнение.
2. Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории и модифицировала его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (рис. 3.9):



```
1 #!/bin/bash
2
3 if (test -n ${1}) then
4     tempfiles=$(find "$1" -maxdepth 1 -type f -ctime -7)
5     #echo $tempfiles
6     #for file in $tempfiles
7     #do
8     #echo "$file"
9     #done
10
11
12 tar -czvf path_tar.tar.gz $tempfiles
13
14
15
16 #tar -czvf path_tar.tar.gz "$1"/*
17
18 else
19     echo "Не указана директория"
20 fi
```

Рис. 3.9: Рисунок 9

3. Выполнила скрипт и получила результат: (рис. 3.10):



```
galataeva@galataeva:~$ ./lab12/tar.sh work/os/lab06
work/os/lab06/12231.sh
work/os/lab06/12231.sh
galataeva@galataeva:~$
```

Рис. 3.10: Рисунок 10

4. Результат: (рис. 3.11):

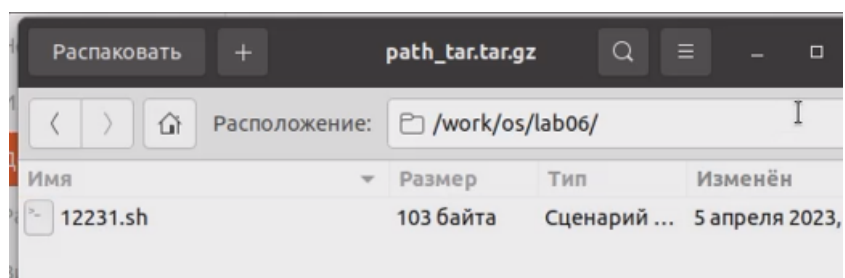


Рис. 3.11: Рисунок 11



## 4 Выводы

Я изучила основы программирования в оболочке ОС UNIX и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 5 Контрольные вопросы

### 1. Каково предназначение команды `getopts`?

Команда `getopts` в скриптах оболочки Unix/Linux предназначена для обработки опций командной строки. Она позволяет скрипту получать аргументы командной строки в определенном формате, что делает его более удобным для использования конечными пользователями. Команда `getopts` используется в цикле `while`, который обрабатывает каждый аргумент командной строки по очереди. Кроме того, `getopts` позволяет обрабатывать несколько опций в одном аргументе командной строки, а также определить опцию по умолчанию, которая будет использоваться, если пользователь не указал никаких опций.

### 2. Какое отношение метасимволы имеют к генерации имён файлов?

Метасимволы в UNIX-подобных операционных системах используются для генерации имен файлов и путей к ним. Например, символ звездочки (\*) соответствует любой последовательности символов в имени файла, а символ вопросительного знака (?) соответствует одному любому символу. Также метасимволы используются для выполнения поиска и замены текста в файле при использовании утилит, таких как `sed` и `awk`.

### 3. Какие операторы управления действиями вы знаете?

В языке программирования `bash` есть много операторов управления действиями, вот некоторые из них:

if-then-else: оператор условия, позволяет выполнять определенные действия в зависимости от условия.

for: оператор цикла, позволяет выполнить набор инструкций несколько раз, итерируясь по списку значений.

while: оператор цикла, позволяет выполнять набор инструкций, пока определенное условие остается истинным.

case: оператор выбора, позволяет выбрать действие, которое должно быть выполнено, исходя из значения переменной.

until: оператор цикла, выполняет набор инструкций до тех пор, пока определенное условие не станет истинным.

select: оператор выбора, позволяет создать меню выбора для пользователя.

break: оператор, который прерывает выполнение цикла или выбора.

continue: оператор, который пропускает текущую итерацию цикла или выбора и переходит к следующей.

trap: оператор, который позволяет устанавливать обработчики сигналов и выполнять определенные действия при получении сигнала.

shift: оператор, который позволяет сдвигать значения параметров командной строки на одну позицию.

Это только некоторые из операторов управления действиями в языке bash.

#### 4. Какие операторы используются для прерывания цикла?

break: используется для немедленного выхода из цикла. continue: используется для пропуска текущей итерации цикла и перехода к следующей.

#### 5. Для чего нужны команды false и true?

Команды false и true являются базовыми утилитами в UNIX-подобных операционных системах, которые возвращают код возврата 1 и 0 соответственно. Обычно эти команды используются в комбинации с другими командами и операторами для контроля над процессами, запускаемыми в скриптах.

6. Что означает строка `if test -f mans/i.s`, встречающаяся в командном файле?

Данная строка используется для проверки наличия файла в определенном месте в файловой системе. Она проверяет, существует ли файл с именем `mans/i.$s`, где `$s`, `$i` - переменные, заданные ранее в скрипте.

7. Объясните различия между конструкциями `while` и `until`.

Конструкция `while` используется для выполнения цикла, пока условие истинно. Цикл выполняется, пока результат выражения внутри скобок `while` истинен.

Конструкция `until` похожа на `while`, но выполняет цикл, пока условие ложно. Цикл будет выполняться, пока результат выражения внутри скобок `until` ложен.

## **Список литературы**