

Отчёт по лабораторной работе №13

Простейший вариант

Ахмад Мд Шешир

Содержание

| | | |
|----------|---------------------------------------|-----------|
| 1 | Цель работы | 5 |
| 2 | Задания | 6 |
| 3 | Выполнение лабораторной работы | 7 |
| 3.1 | Задача 1 | 7 |
| 3.2 | Задача 2 | 9 |
| 3.3 | Задача 3 | 11 |
| 3.4 | Задача 4 | 13 |
| 4 | Выводы | 16 |

Список иллюстраций

| | | |
|------|----------------------------|----|
| 3.1 | Набираю текст | 8 |
| 3.2 | файл | 11 |
| 3.3 | программа | 12 |
| 3.4 | исполняемый файл | 12 |
| 3.5 | проверка | 13 |
| 3.6 | файл | 13 |
| 3.7 | программа | 14 |
| 3.8 | исполняемый файл | 14 |
| 3.9 | проверка | 14 |
| 3.10 | программа | 15 |
| 3.11 | проверка | 15 |

Список таблиц

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

2 Задания

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-i`inputfile — прочитать данные из указанного файла; `-o`outputfile — 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 о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `“$?”`, выдать сообщение о том, какое число было введено.

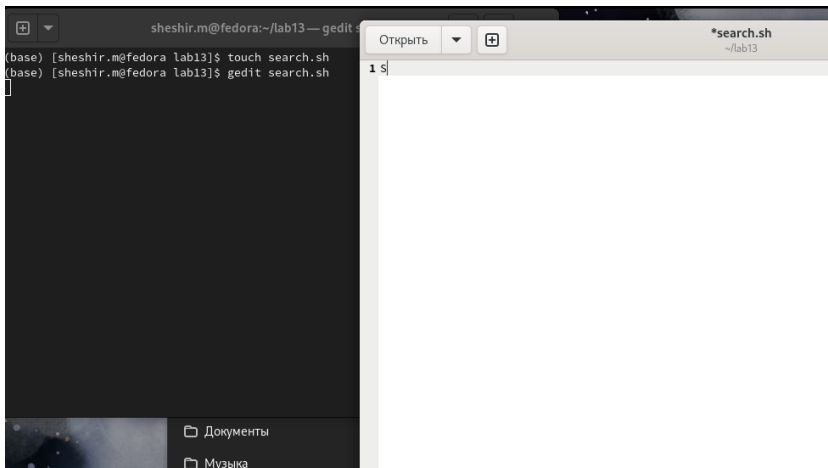
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до n (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если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

4. Написать командный файл, который с помощью команды `tag` 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`).

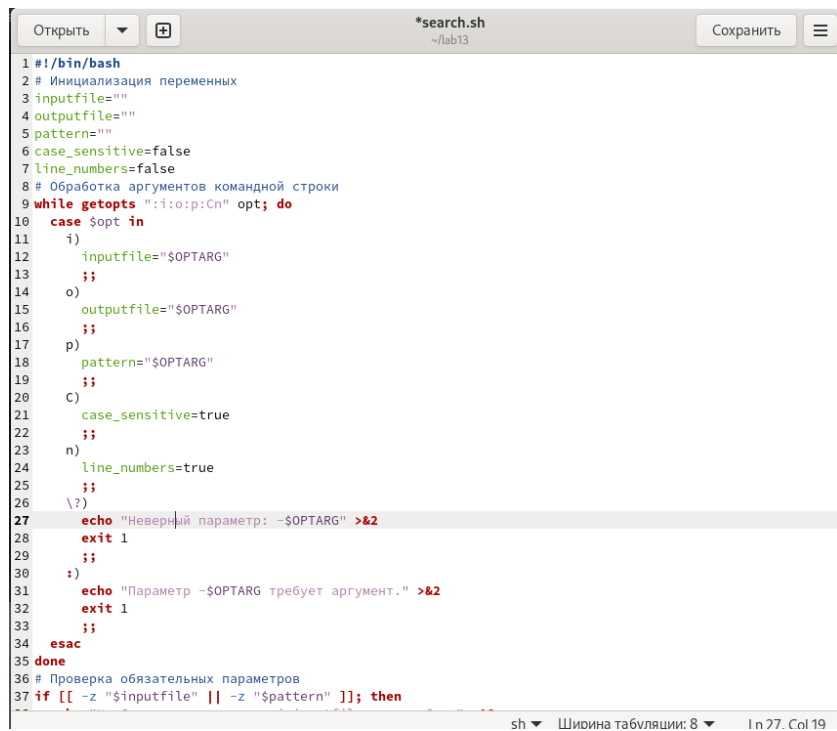
3 Выполнение лабораторной работы

3.1 Задача 1

1. Создаю файл где будет сам скрипт.(рис. ??).



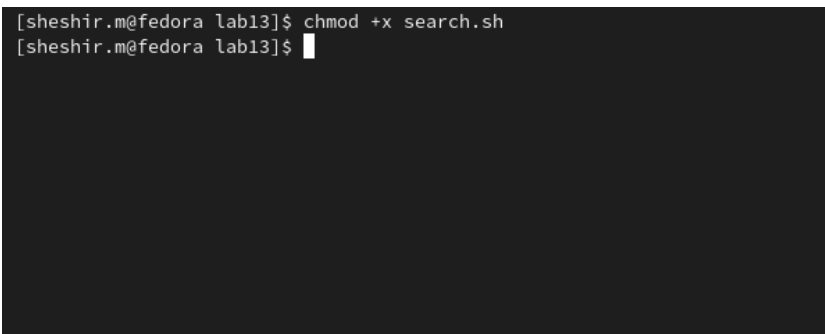
2. Набираю программу где создаю переменные с для определения аргументов для поиска (рис. 3.1).



```
1 #!/bin/bash
2 # Инициализация переменных
3 inputfile=""
4 outputfile=""
5 pattern=""
6 case_sensitive=false
7 line_numbers=false
8 # Обработка аргументов командной строки
9 while getopts "i:o:p:Cn" opt; do
10     case $opt in
11         i)
12             inputfile="$OPTARG"
13             ;;
14         o)
15             outputfile="$OPTARG"
16             ;;
17         p)
18             pattern="$OPTARG"
19             ;;
20         C)
21             case_sensitive=true
22             ;;
23         n)
24             line_numbers=true
25             ;;
26         \?)
27             echo "Неверный параметр: -$OPTARG" >&2
28             exit 1
29             ;;
30         *)
31             echo "Параметр -$OPTARG требует аргумент." >&2
32             exit 1
33             ;;
34     esac
35 done
36 # Проверка обязательных параметров
37 if [[ -z "$inputfile" || -z "$pattern" ]]; then
```

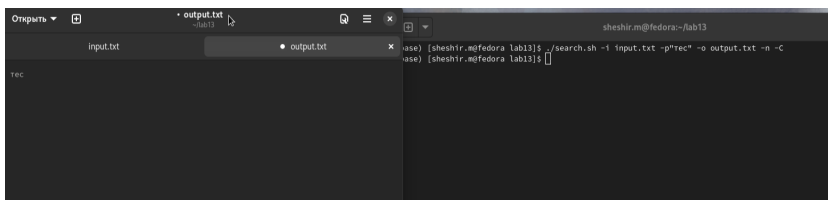
Рис. 3.1: Набираю текст

3. Делаю файл исполняемым, выполнив команду в терминале `chmod` (рис. ??).



```
[sheshir.m@fedora lab13]$ chmod +x search.sh
[sheshir.m@fedora lab13]$
```

4. Запускаю исполняемый файл и проверяю выполнение (рис. ??).



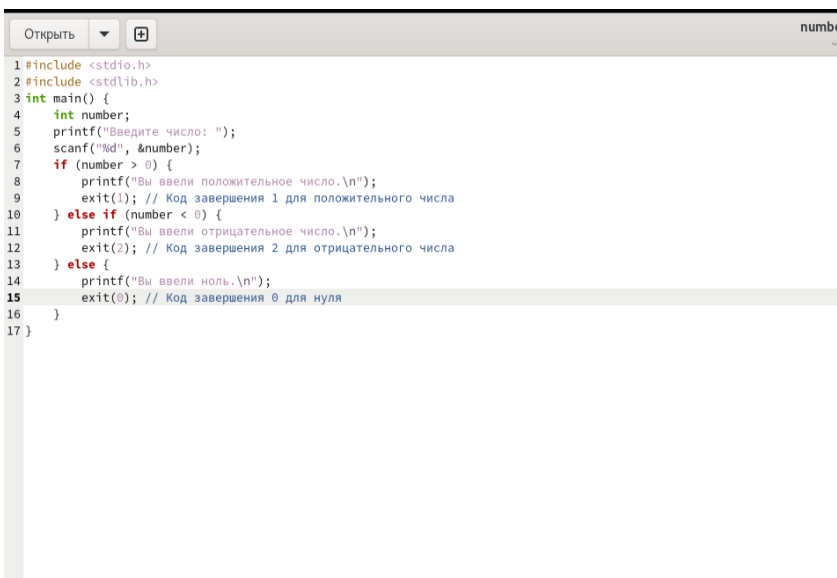
```
sheshir.m@fedora:~/lab13
x [sheshir.m@fedora lab13]$ ./search.sh -i input.txt -p"rec" -o output.txt -n -C
x [sheshir.m@fedora lab13]$
```


3.2 Задача 2

1. Открываю текстовый редактор и создаю новый файл для кода на С (рис. ??).

```
(base) [sheshir.m@fedora lab13]$ touch number_check.c
(base) [sheshir.m@fedora lab13]$ gedit number_check.c
(base) [sheshir.m@fedora lab13]$
```

2. Программа запрашивает у пользователя ввод числа, определяет является ли оно положительным отрицательным или равным нулю, а затем завершает работу. (рис. ??).



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main() {
4     int number;
5     printf("Введите число: ");
6     scanf("%d", &number);
7     if (number > 0) {
8         printf("Вы ввели положительное число.\n");
9         exit(1); // Код завершения 1 для положительного числа
10    } else if (number < 0) {
11        printf("Вы ввели отрицательное число.\n");
12        exit(2); // Код завершения 2 для отрицательного числа
13    } else {
14        printf("Вы ввели ноль.\n");
15        exit(0); // Код завершения 0 для нуля
16    }
17 }
```

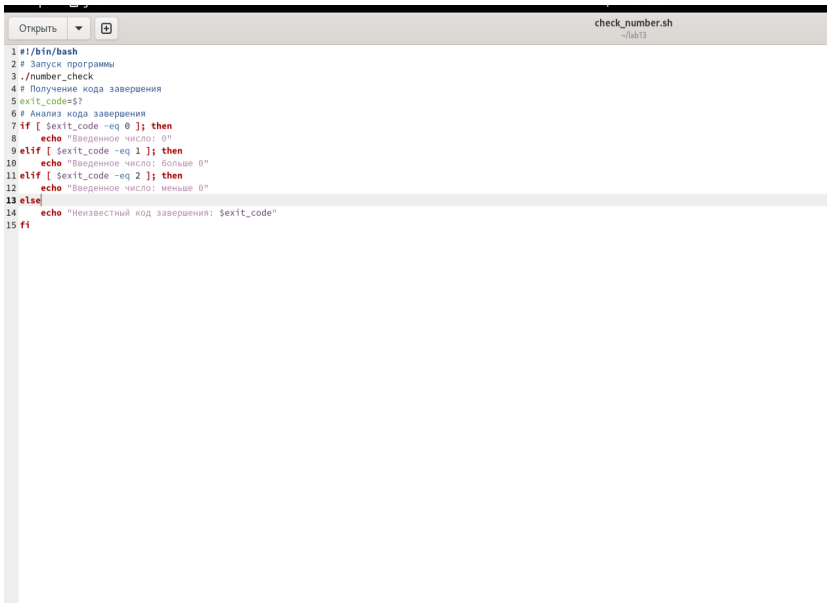
3. Создаю исполняемый файл, выполнив команду в терминале gcc для языка С. (рис. ??).

```
(base) [sheshir.m@fedora lab13]$ gcc number_check.c -o number_check
(base) [sheshir.m@fedora lab13]$
```

4. Запускаю исполняемый файл и проверяю работу кода на С (рис. ??).

```
(base) [sheshir.m@fedora lab13]$ ./number_check
Введите число: 5
Вы ввели положительное число.
(base) [sheshir.m@fedora lab13]$
```

5. Создаю новый файл для кода для BASH и набираю программу (рис. ??).



```
1 #!/bin/bash
2 # Запуск программы
3 ./number_check
4 # Получение кода завершения
5 exit_code=$?
6 # Анализ кода завершения
7 if [ $exit_code -eq 0 ]; then
8     echo "Введенное число: 0"
9 elif [ $exit_code -eq 1 ]; then
10    echo "Введенное число: больше 0"
11 elif [ $exit_code -eq 2 ]; then
12    echo "Введенное число: меньше 0"
13 else
14    echo "Неизвестный код завершения: $exit_code"
15 fi
```

6. Делаю файл исполняемым, выполнив команду в терминале chmod (рис. ??).

```
(base) [sheshir.m@fedora lab13]$ chmod +x check_number.sh
(base) [sheshir.m@fedora lab13]$
```

7. Запускаю исполняемый файл, и пишу любое число(рис. ??).

```
(base) [sheshir.m@fedora lab13]$ ./check_number.sh
Введите число: 5
Вы ввели положительное число.
Введенное число: больше 0
(base) [sheshir.m@fedora lab13]$
```

3.3 Задача 3

1. Открываю редактор gedit и создаю новый файл .sh (рис. 3.2).

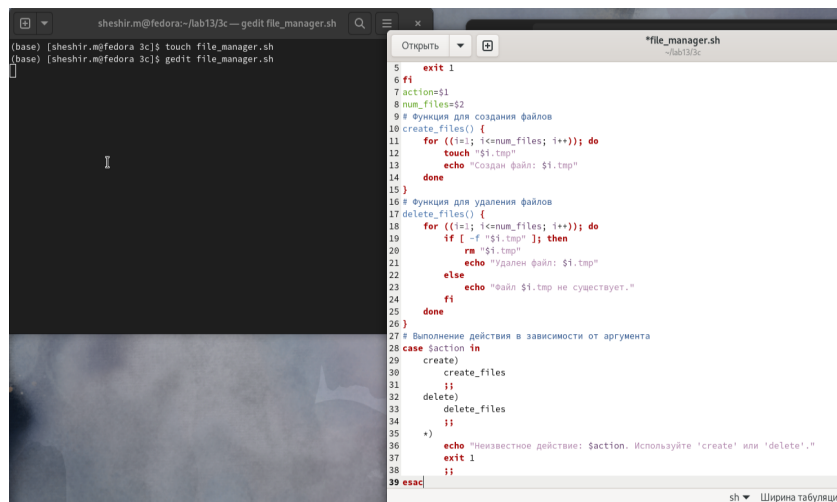
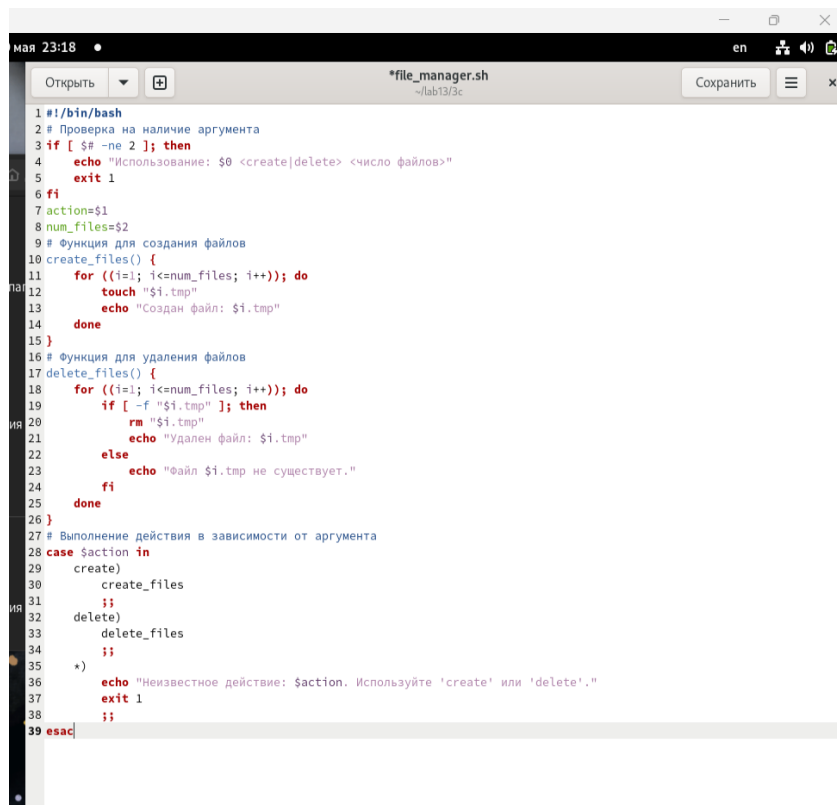


Рис. 3.2: файл

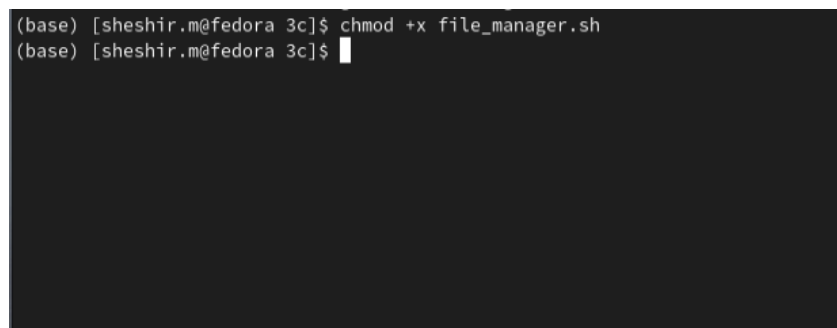
2. Пишу программу которая создает указанное количество файлов, пронумерованных от 1 до N, и также может их удалять. . (рис. 3.3).



```
1 #!/bin/bash
2 # Проверка на наличие аргумента
3 if [ $# -ne 2 ]; then
4     echo "Использование: $0 <create|delete> <число файлов>"
5     exit 1
6 fi
7 action=$1
8 num_files=$2
9 # Функция для создания файлов
10 create_files() {
11     for ((i=1; i<=num_files; i++)); do
12         touch "$i.tmp"
13         echo "Создан файл: $i.tmp"
14     done
15 }
16 # Функция для удаления файлов
17 delete_files() {
18     for ((i=1; i<=num_files; i++)); do
19         if [ -f "$i.tmp" ]; then
20             rm "$i.tmp"
21             echo "Удален файл: $i.tmp"
22         else
23             echo "Файл $i.tmp не существует."
24         fi
25     done
26 }
27 # Выполнение действия в зависимости от аргумента
28 case $action in
29     create)
30         create_files
31         ;;
32     delete)
33         delete_files
34         ;;
35     *)
36         echo "Неизвестное действие: $action. Используйте 'create' или 'delete'."
37         exit 1
38         ;;
39 esac
```

Рис. 3.3: программа

3. Делаю файл исполняемым, выполнив команду в терминале `chmod` (рис. 3.4).



```
(base) [sheshir.m@fedora 3c]$ chmod +x file_manager.sh
(base) [sheshir.m@fedora 3c]$
```

Рис. 3.4: исполняемый файл

4. Запускаю исполняемы файл, передаю ей в качестве аргумента 5 Проверяю работу (рис. 3.5).

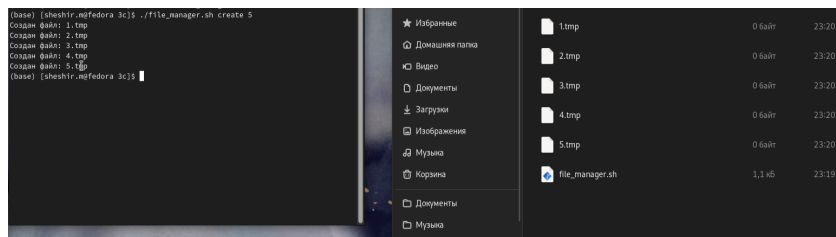


Рис. 3.5: проверка

3.4 Задача 4

1. Открываю редактор gedit и создаю новый файл count_files.sh (рис. 3.6).

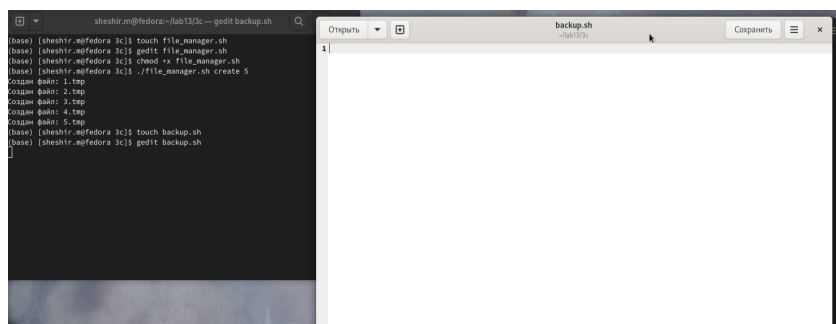


Рис. 3.6: файл

2. Пишу программу которая запаковывает все файлы в архив tar с помощью команды tar. (рис. 3.7).

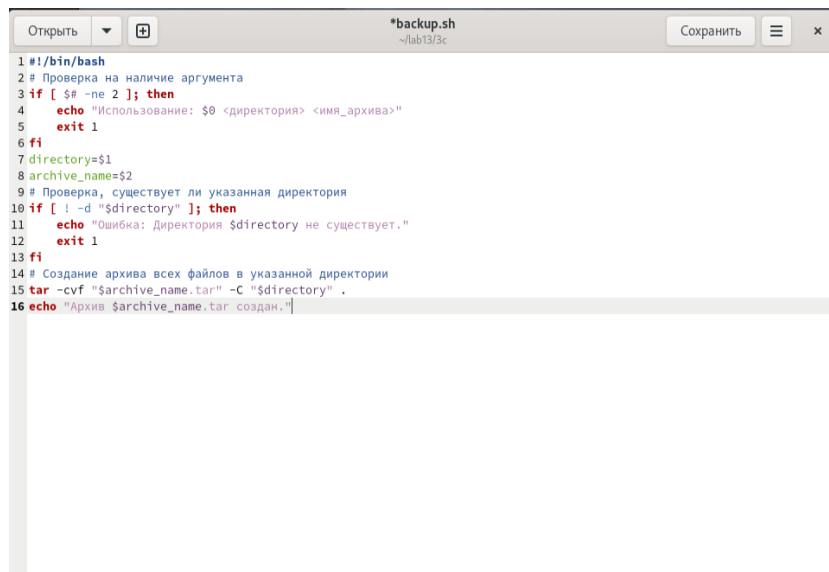


Рис. 3.7: программа

3. Делаю файл исполняемым, выполнив команду в терминале chmod (рис. 3.8).

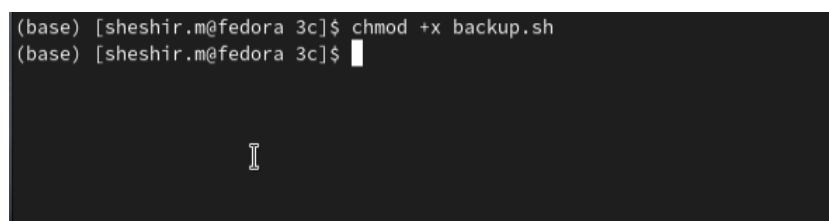


Рис. 3.8: исполняемый файл

4. Запускаю исполняемый файл, передаю ей аргументы пути к директории и имя сделанного архива. (рис. 3.9).

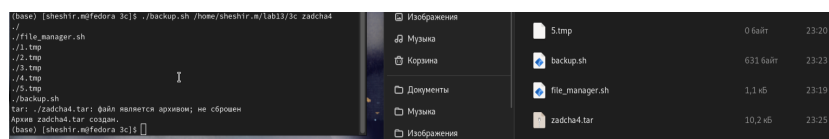


Рис. 3.9: проверка

5. Модифицирую программу так чтоб запаковывала все файлы в архив tar которые были изменены до недели назад. (рис. 3.10).

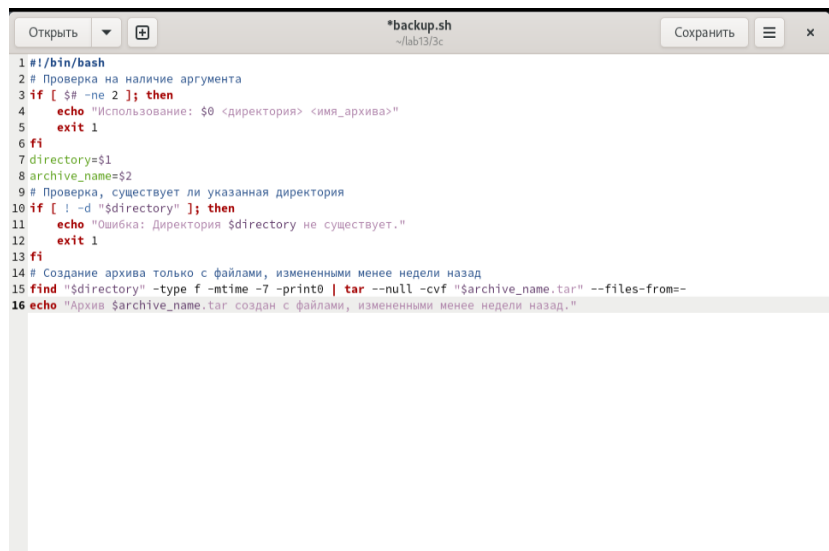


Рис. 3.10: программа

7. Запускаю исполняемый файл, передаю ей аргументы пути к директории и имя сделанного архива. (рис. 3.11).

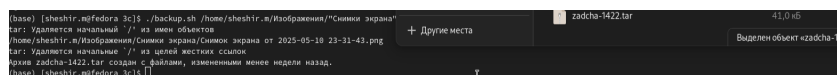


Рис. 3.11: проверка

4 Выводы

В ходе работы я познакомился с основами программирования в оболочке ОС UNIX/Linux. Научился писать небольшие командные файлы.