

Лабораторная работа №12

**Программирование в командном процессоре ОС UNIX. Ветвления и
циклы**

Ким Михаил Алексеевич

Содержание

1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Выводы	19
4	Термины	20

1. Цель работы

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

2. Выполнение лабораторной работы

1. Используя команды `getopts` и `grep`, написали 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. В итоге, наш скрипт ищет в указанном файле нужные строки, определяемые ключом `-r`. (рис. 2.1 - 2.5)

```
#!/bin/bash
```

```
// Структура необходимая для работы системы флагов.
```

```
// При объявлении флага переменные *flag принимают значение 1.
```

```
// Если за флагом следует информация, то эта информация принимается переменн
```

```
#!/bin/bash
```

```
while getopts i:o:p:Cn optletter
```

```
do case $optletter in
```

```
i) iflag=1; ival=$OPTARG;;
```

```
o) oflag=1; oval=$OPTARG;;
```

```
p) pflag=1; pval=$OPTARG;;
```

```
C) Cflag=1;;
```

```
n) nflag=1;;
```

```
*) echo Ключи и параметры заданы неправильно!
```

```
esac
```

done

// Пишем ветвления для обработки флагов.

// В зависимости от того, какие флаги активны выполняем действия.

// Для того чтобы обработать входной файл, выходной и шаблон обращаемся к пе

```
if [[ $Cflag == 1 ]]; then
```

```
    if [[ $nflag == 1 ]]; then
```

```
        grep $pval -n $ival
```

```
        if [[ $oflag == 1 ]]; then
```

```
            grep $pval -n $ival > $oval
```

```
            echo Данные в файл были успешно записаны
```

```
        fi
```

```
    else
```

```
        grep $pval $ival
```

```
        if [[ $oflag == 1 ]]; then
```

```
            grep $pval $ival > $oval
```

```
            echo Данные в файл были успешно записаны
```

```
        fi
```

```
    fi
```

```
else
```

```
    if [[ $nflag == 1 ]]; then
```

```
        grep $pval -n -i $ival
```

```
        if [[ $oflag == 1 ]]; then
```

```
            grep $pval -n -i $ival > $oval
```

```
            echo Данные в файл были успешно записаны
```

```
        fi
```

```
    else
```

```
        grep $pval -i $ival
```

```
if [[ $oflag == 1 ]]; then
    grep $pval -i $ival > $oval
    echo Данные в файл были успешно записаны
fi
fi
fi
```

When the days are cold
And the cards all fold
And the saints we see
Are all made of gold
when your dreams they fail
And the ones we hail
Are the worst of all
And the blood's run stale
I wanna hide the truth, I wanna shelter you
But with the beast inside
There's nowhere we can hide
No matter what we breed
We still are made of greed
This is my kingdom come
This is my kingdom come
When you feel my heat
Look into my eyes
It's where my demons hide
It's where my demons hide
Don't get too close
It's dark inside
It's where my demons hide

It's where my demons hide
When the curtain's call
It's the last of all
When the lights fade out
All the sinners crawl
So they dug your grave
And the masquerade
Will come calling out
At the mess you've made
Don't wanna let you down
But I am hell bound
Though this is all for you
Don't wanna hide the truth
No matter what we breed
We still are made of greed
This is my kingdom come
This is my kingdom come
When you feel my heat
Look into my eyes
It's where my demons hide
It's where my demons hide
Don't get too close
It's dark inside
It's where my demons hide
It's where my demons hide
They say it's what you make
I say it's up to fate
It's woven in my soul
I need to let you go

Your eyes, they shine so bright
I wanna save that light
I can't escape this now
Unless you show me how
When you feel my heat
Look into my eyes
It's where my demons hide
It's where my demons hide
Don't get too close
It's dark inside
It's where my demons hide
It's where my demons hide


```
Открыть  [F4]
1 #!/bin/bash
2 while getopts i:o:p:Cn optletter
3 do case $optletter in
4 i) iflag=1; ival=$OPTARG;;
5 o) oflag=1; oval=$OPTARG;;
6 p) pflag=1; pval=$OPTARG;;
7 C) cflag=1;;
8 n) nflag=1;;
9 *) echo Ключи и параметры заданы неправильно!
10 esac
11 done
12
13 if [[ $cflag == 1 ]]; then
14     if [[ $nflag == 1 ]]; then
15         grep $pval -n $ival
16         if [[ $oflag == 1 ]]; then
17             grep $pval -n $ival > $oval
18             echo Данные в файл были успешно записаны
19         fi
20     else
21         grep $pval $ival
22         if [[ $oflag == 1 ]]; then
23             grep $pval $ival > $oval
24             echo Данные в файл были успешно записаны
25         fi
26     fi
27 else
28     if [[ $nflag == 1 ]]; then
29         grep $pval -n -i $ival
30         if [[ $oflag == 1 ]]; then
31             grep $pval -n -i $ival > $oval
32             echo Данные в файл были успешно записаны
33         fi
34     else
35         grep $pval -i $ival
36         if [[ $oflag == 1 ]]; then
37             grep $pval -i $ival > $oval
38             echo Данные в файл были успешно записаны
39         fi
40     fi
41 fi
42
```

Рис. 2.1: Исходный код

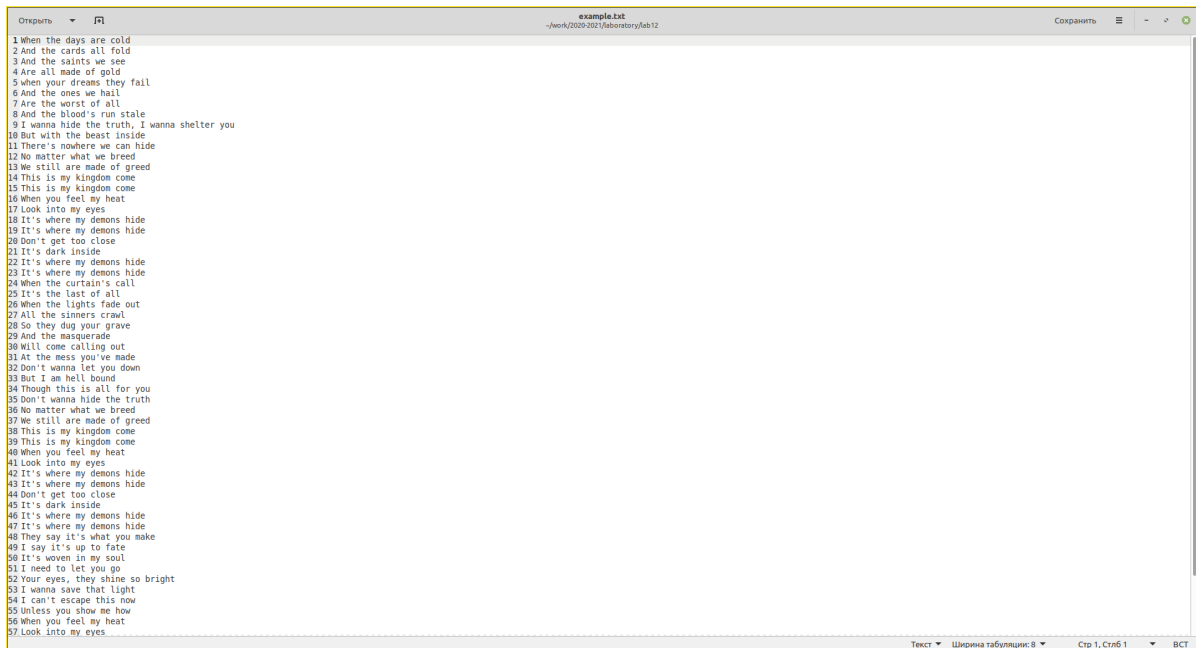


Рис. 2.2: Текст

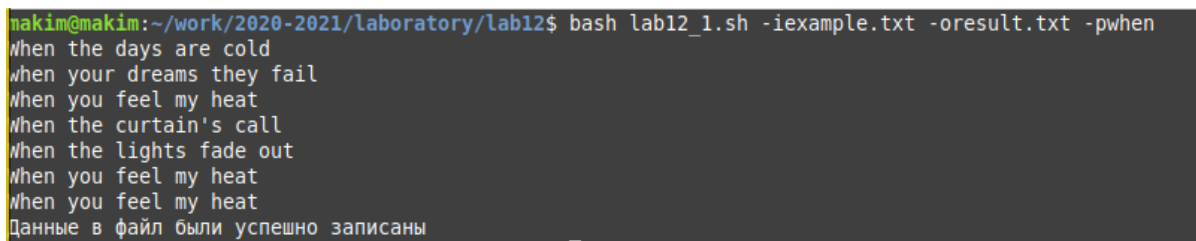


Рис. 2.3: Результат

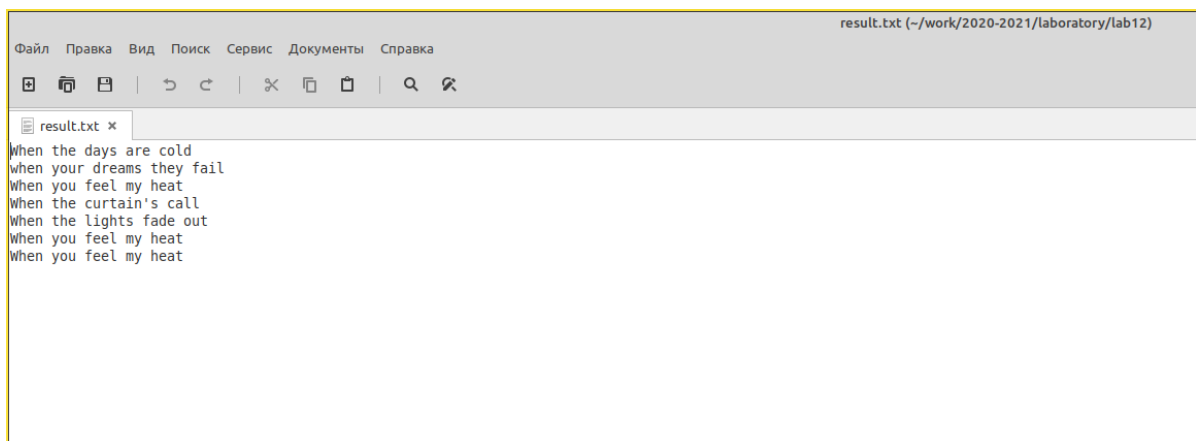


Рис. 2.4: Результат

```

makim@makim:~/work/2020-2021/laboratory/lab12$ bash lab12_1.sh -iexample.txt -pwhen -C
when your dreams they fail
makim@makim:~/work/2020-2021/laboratory/lab12$ bash lab12_1.sh -iexample.txt -pwhen -n
1:When the days are cold
5:when your dreams they fail
16:When you feel my heat
24:When the curtain's call
26:When the lights fade out
40:When you feel my heat
56:When you feel my heat
makim@makim:~/work/2020-2021/laboratory/lab12$ bash lab12_1.sh -iexample.txt -pwhen -C -n
5:when your dreams they fail

```

Рис. 2.5: Результат

2. Пишем 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 вызывает эту программу и, проанализировав с помощью команды `$?`, выдаёт сообщение о том, какое число было введено.(рис. 2.6 - 2.8)

```

// lab12_2.cpp
// Принцип работы прост. Вводим число, с помощью if определяем, в какую сторону
#include <iostream>
using namespace std;

int main()
{
    cout << "Введите число: ";
    int n;
    cin >> n;
    if (n > 0) { exit(1); };
    if (n < 0) { exit(2); };
    if (n == 0) { exit(0); };
    return 0;
}

```

```
//lab12_2.sh
```

```
#!/bin/bash
```

```
// Компилируем lab12_2.cpp.
```

```
g++ lab12_2.cpp -o output
```

```
// Открываем результат компиляции
```

```
./output
```

```
// После завершения программы анализируем результат с помощью case.
```

```
// В условие ставим $? - переменную, которая содержит код завершения последн
```

```
case $? in
```

```
1) echo Число равно 0;;
```

```
2) echo Число больше 0;;
```

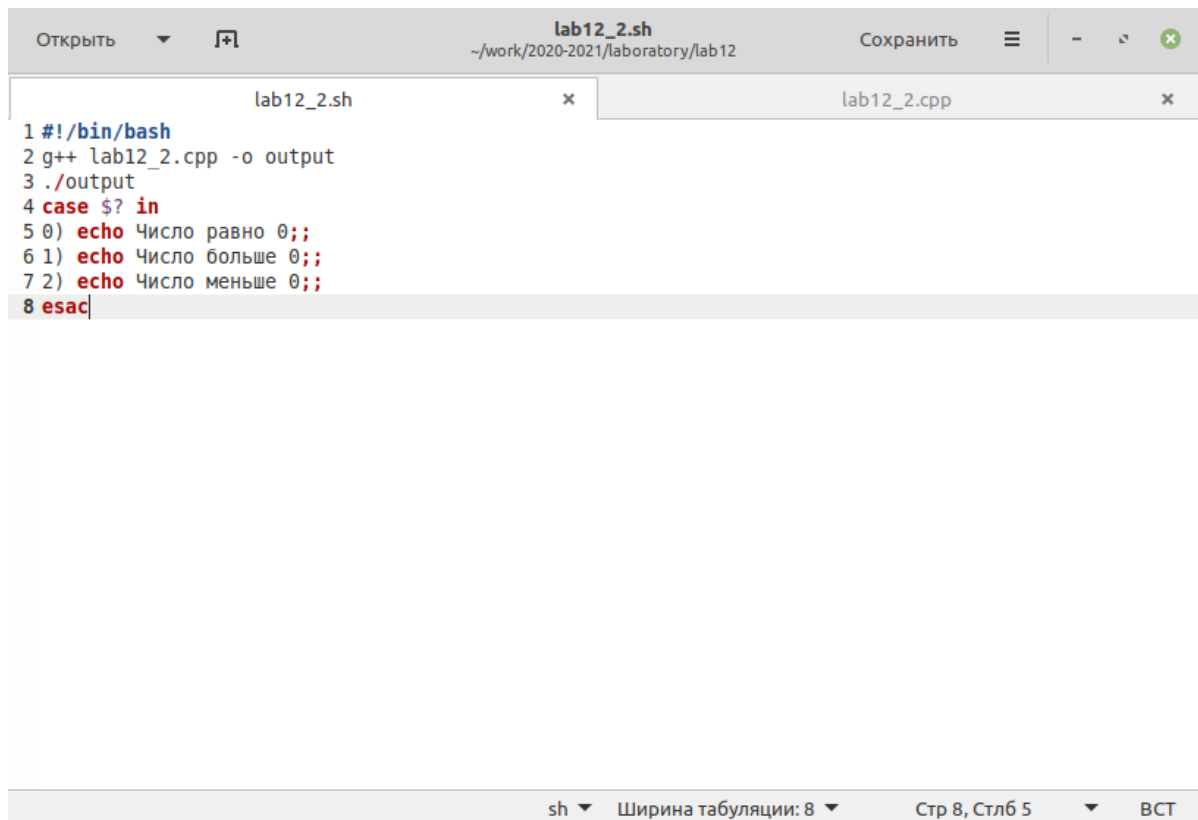
```
3) echo Число меньше 0;;
```

```
esac
```

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << "Введите число: ";
7     int n;
8     cin >> n;
9     if (n > 0) { exit(1); };
10    if (n < 0) { exit(2); };
11    if (n == 0) { exit(0); };
12    return 0;
13 }
```

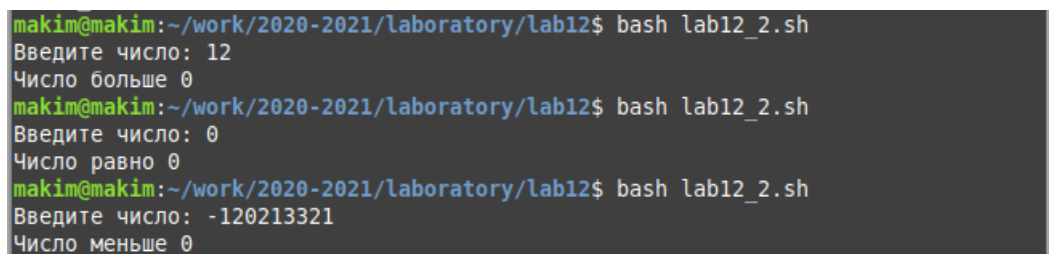
Парная скобка найдена в строке: 5 C++ Ширина табуляции: 8 Стр 13, Стлб 2 ВСТ

Рис. 2.6: Исходный код .cpp



```
lab12_2.sh
1 #!/bin/bash
2 g++ lab12_2.cpp -o output
3 ./output
4 case $? in
5 0) echo Число равно 0;;
6 1) echo Число больше 0;;
7 2) echo Число меньше 0;;
8 esac
```

Рис. 2.7: Исходный код .sh



```
makim@makim:~/work/2020-2021/laboratory/lab12$ bash lab12_2.sh
Введите число: 12
Число больше 0
makim@makim:~/work/2020-2021/laboratory/lab12$ bash lab12_2.sh
Введите число: 0
Число равно 0
makim@makim:~/work/2020-2021/laboratory/lab12$ bash lab12_2.sh
Введите число: -120213321
Число меньше 0
```

Рис. 2.8: Результат

3. Пишем командный файл, который создаст указанное число файлов, пронумерованных последовательно от 0 до N (например 0.tmp, 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 умеет удалять все созданные им файлы.(рис. 2.8 - 2.11)

```
#!/bin/bash
```

```
// Используем цикл for, чтобы пробежаться по всем числам от 1 до нашего аргу
// Для каждого числа будет создаваться файл
for (( i = 1; i <= $1; i++ ))
    do
        touch ${i}.tmp
    done

// Спрашиваем пользователя, нужно ли удалять созданные файлы
// Если да, то таким же образом их удаляем
read -p 'Нужно ли удалить созданные файлы?y/n: ' marker
if [[ $marker == 'y' ]]; then
    for (( i = 1; i <= $1; i++ ))
        do
            rm ${i}.tmp
        done
fi
```



```
1 #!/bin/bash
2 for (( i = 1; i <= $1; i++ ))
3 do
4     touch ${i}.tmp
5 done
6
7 read -p 'Нужно ли удалить созданные файлы?y/n: ' marker
8 if [[ $marker == 'y' ]]; then
9     for (( i = 1; i <= $1; i++ ))
10 do
11     rm ${i}.tmp
12 done
13 fi
```

Рис. 2.9: Исходный код

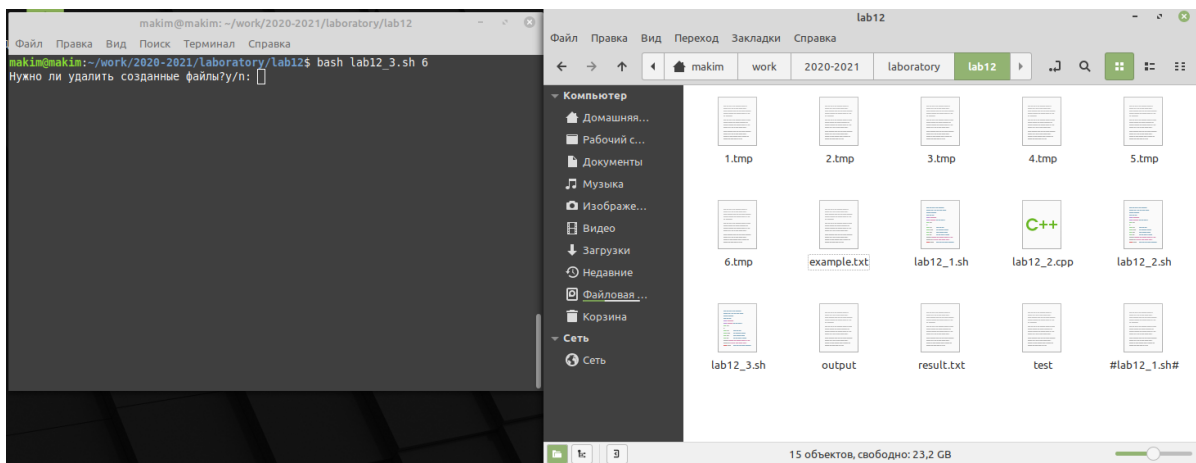


Рис. 2.10: Результат

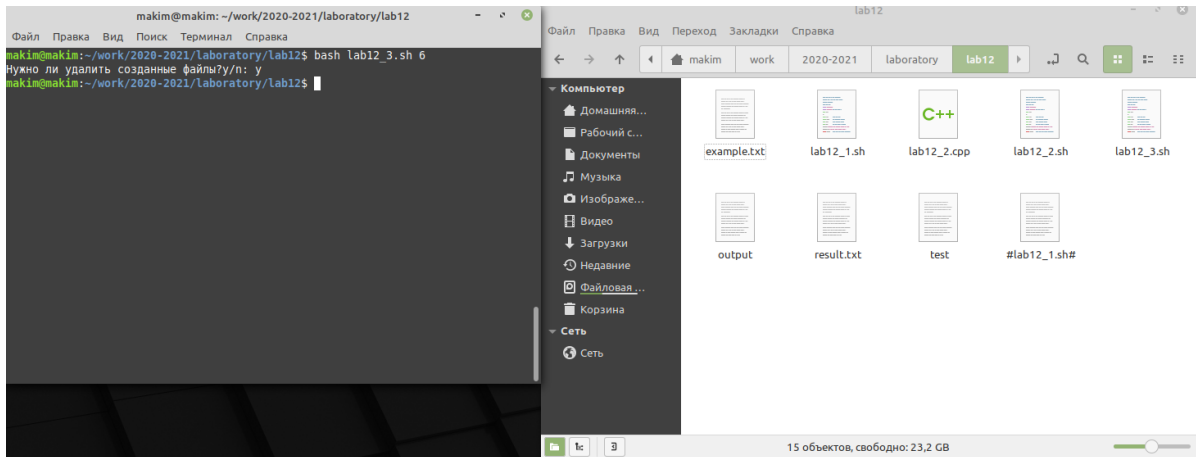


Рис. 2.11: Результат

4. Пишем командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицируем его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (используем команду `find`). (рис. 2.12 - 2.13)

```
#!/bin/bash
```

```
// Считываем директорию.
```

```
read -p "Enter directory: " folder
```

```
// Создаём пустой архив.
```

```
tar -cf backup7.tar -T /dev/null
```

```
// Используем find с опцией -mtime, которая позволяет отсеивать файлы, в зав
```

```
// также используем -exec, который позволяет к каждому найденному файлу прим
```

```
// после -exec ставим команду tar rf backup7.tar, чтобы все найденные файлы
```

```
find $folder -mtime -7 -exec tar rf backup7.tar '{}' '+'
```

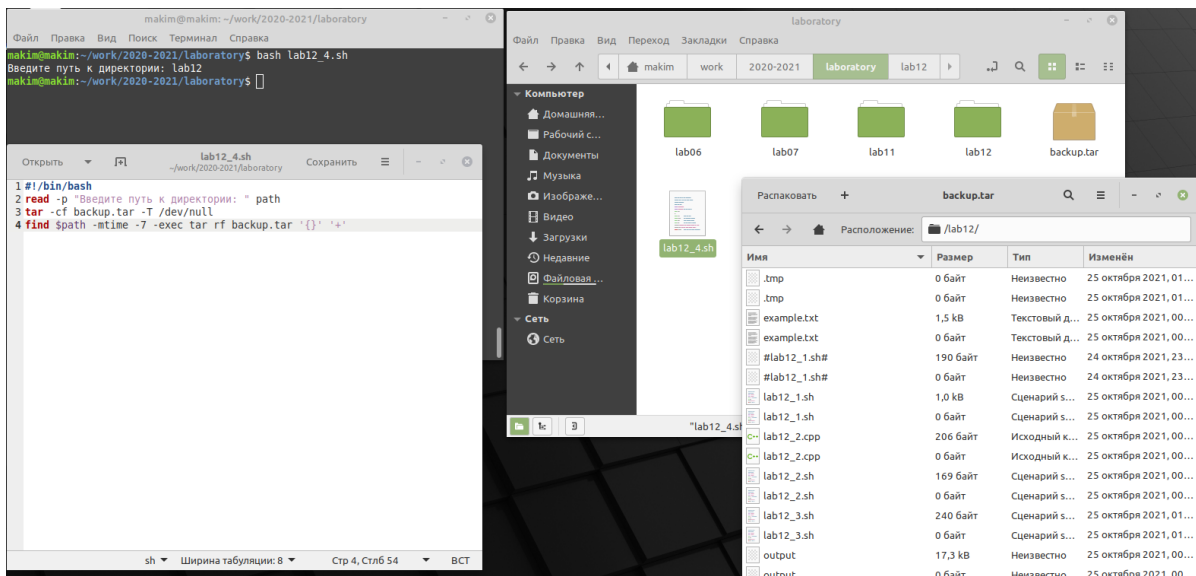


Рис. 2.12: Исходный код

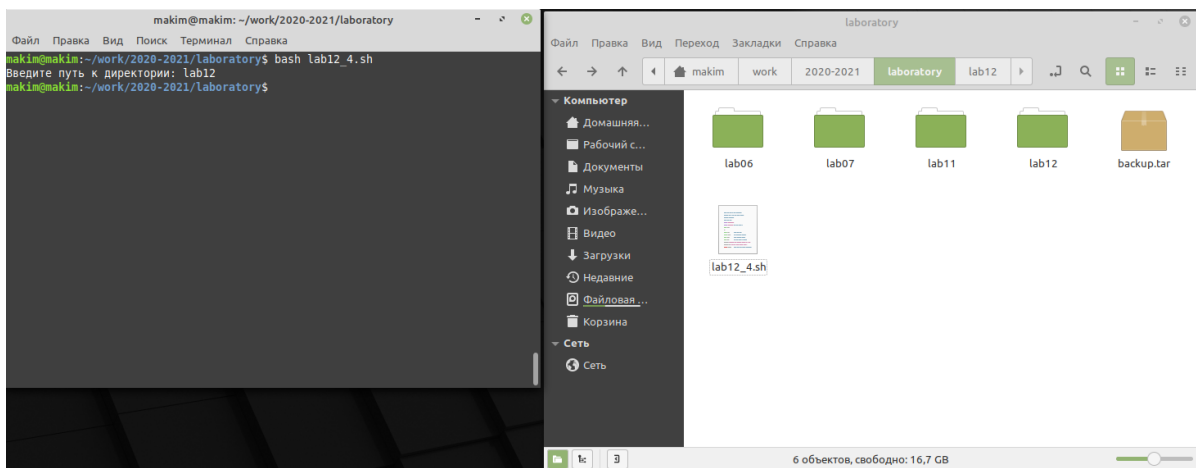


Рис. 2.13: Результат

3. Выводы

Мы изучили основы программирования в оболочке ОС UNIX. Закрепили знания, полученные в прошлых работах. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

4. Термины

- Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.
- POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.
- Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным.
- Флаги — это опции командной строки, обычно помеченные знаком минус; Например, для команды ls флагом может являться -F.
- Каталог, он же директория, (от английского Directory) – это объект в ФС (файловой системе), необходимый для того, чтобы упростить работу с файлами.
- Домашний каталог - каталог, предназначенный для хранения собственных данных пользователя Linux. Как правило, является текущим непосредственно после регистрации пользователя в системе.
- Команда - записанный по специальным правилам текст (возможно с аргументами), представляющий собой указание на выполнение какой-либо функций (или действий) в операционной системе.