

# **Отчет по лабораторной работе №11.**

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

Данила Андреевич Стариков

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>4</b>
2.1	Задание 1. . . . .	4
2.2	Задание 2. . . . .	5
2.3	Задание 3. . . . .	7
2.4	Задание 4. . . . .	8
<b>3</b>	<b>Выводы</b>	<b>10</b>

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

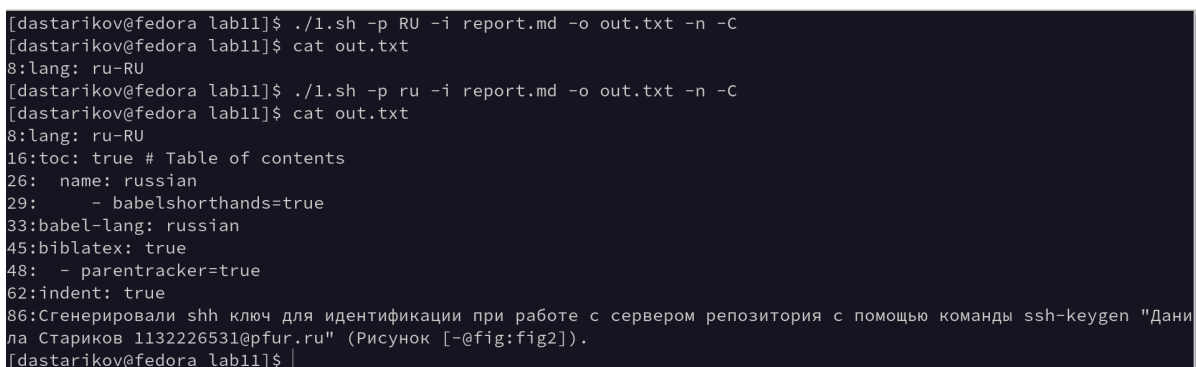
### 2.1 Задание 1.

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

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

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

Создан командный файл `1.sh` (Листинг 2.1). Примеры выполнения скрипта представлены на Рисунке 2.1.



```
[dastarikov@fedora lab11]$ ./1.sh -p RU -i report.md -o out.txt -n -C
[dastarikov@fedora lab11]$ cat out.txt
8:lang: ru-RU
[dastarikov@fedora lab11]$ ./1.sh -p ru -i report.md -o out.txt -n -C
[dastarikov@fedora lab11]$ cat out.txt
8:lang: ru-RU
16:toc: true # Table of contents
26:  name: russian
29:    - babelshorthands=true
33:babel-lang: russian
45:biblatex: true
48:  - parenttracker=true
62:indent: true
86:Сгенерировали ssh ключ для идентификации при работе с сервером репозитория с помощью команды ssh-keygen "Данила Стариков 1132226531@pfur.ru" (Рисунок [-@fig:fig2]).
[dastarikov@fedora lab11]$
```

Рис. 2.1: Пример работы программы `1.sh`.

---

**Листинг 2.1** Текст командного файла Задания №1.

---

```
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 Illegal option $optletter
10 esac
11 done
12 OPTIONS+="-e $pval $ival"
13 if [ ! $Cflag ]
14 then OPTIONS+=" -i"
15 fi
16 if [ $nflag ]
17 then OPTIONS+=" -n"
18 fi
19 grep ${OPTIONS} > $oval
```

---

## 2.2 Задание 2.

Формулировка задания: 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.

По заданию написаны 2 программы: `mod.c` на языке Си (Листинг 2.2) и командный файл `2.sh` (Листинг 2.3). Примеры выполнения скрипта представлены на Рисунке 2.2.

---

**Листинг 2.2** Текст файла mod.c Задания №1.

---

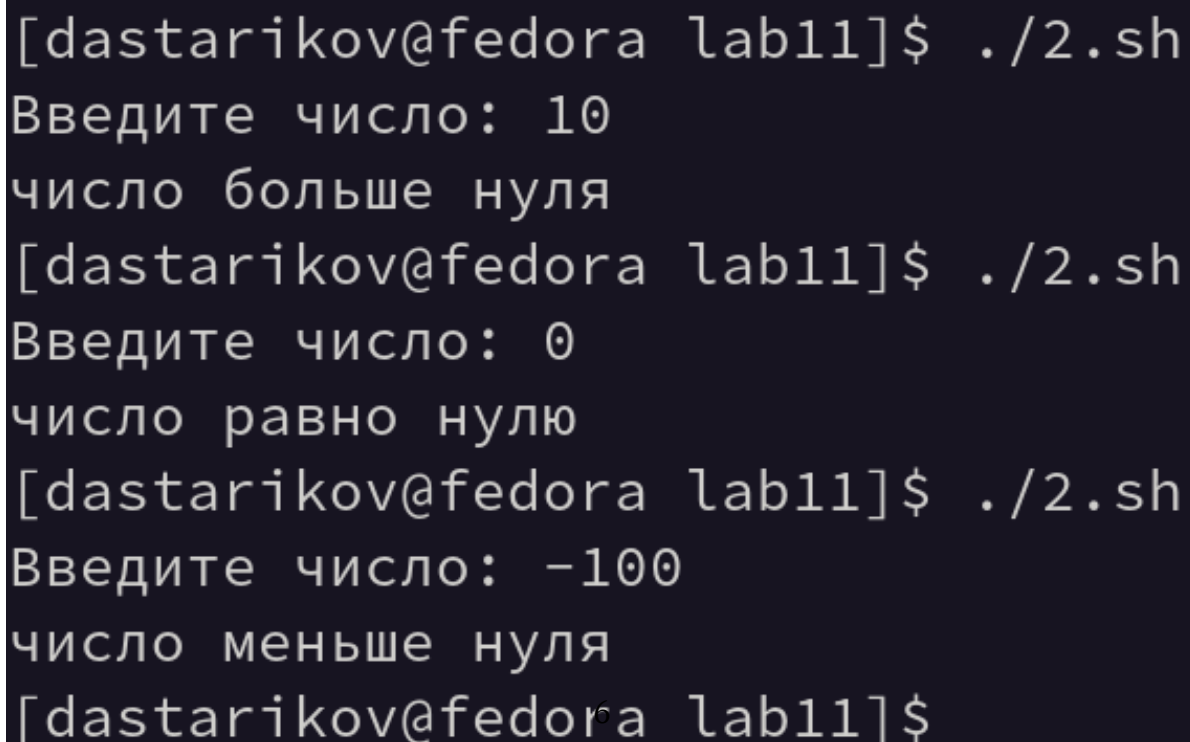
```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main(){
5     int n;
6     printf("Введите число: ");
7     scanf("%d", &n);
8     if(n > 0) {n = 1;}
9     else if (n<0) {n = -1;}
10    else {n=0;}
11    exit(n);
12    return 0;
13 }
```

---

**Листинг 2.3** Текст командного файла Задания №2.

---

```
1 #!/bin/bash
2 ./mod
3 signal=$?
4 case $signal in
5 '1') echo число больше нуля;;
6 '255') echo число меньше нуля;;
7 '0') echo число равно нулю;
8 esac
```



```
[dastarikov@fedora lab11]$ ./2.sh
Введите число: 10
число больше нуля
[dastarikov@fedora lab11]$ ./2.sh
Введите число: 0
число равно нулю
[dastarikov@fedora lab11]$ ./2.sh
Введите число: -100
число меньше нуля
[dastarikov@fedora lab11]$
```

Рис. 2.2: Пример работы программы 2.sh.

## 2.3 Задание 3.

Формулировка задания: 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).

Создан командный файл 1.sh (Листинг 2.4). Примеры выполнения скрипта представлены на Рисунке 2.3.

---

**Листинг 2.4** Текст командного файла Задания №3.

---

```
1 #!/bin/bash
2 N=$1
3 FILES=
4 EXTENSION=.tmp
5 DIR=temp
6 for (( a=1; a <= N; a++ ))
7 do
8     FILES+=$DIR/${a}${EXTENSION}" "
9 done
10
11 if (! find $DIR -type d 2>/dev/null)
12 then
13     mkdir $DIR
14 fi
15
16 touch $FILES
```

---

← ~/work/misc/lab11/temp • [↓] →			
↓И	Имя	Размер	Время правки
	/..	-ВВЕРХ-	апр 21 22:32
	1.tmp	0	апр 21 22:33
	2.tmp	0	апр 21 22:33
	3.tmp	0	апр 21 22:33
	4.tmp	0	апр 21 22:33
-ВВЕРХ-			
66G / 79G (83%)			

Рис. 2.3: Пример работы программы 3.sh.

## 2.4 Задание 4.

Формулировка задания: 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).

Создан командный файл 4.sh (Листинг 2.5). Примеры выполнения скрипта представлены на Рисунке 2.4.



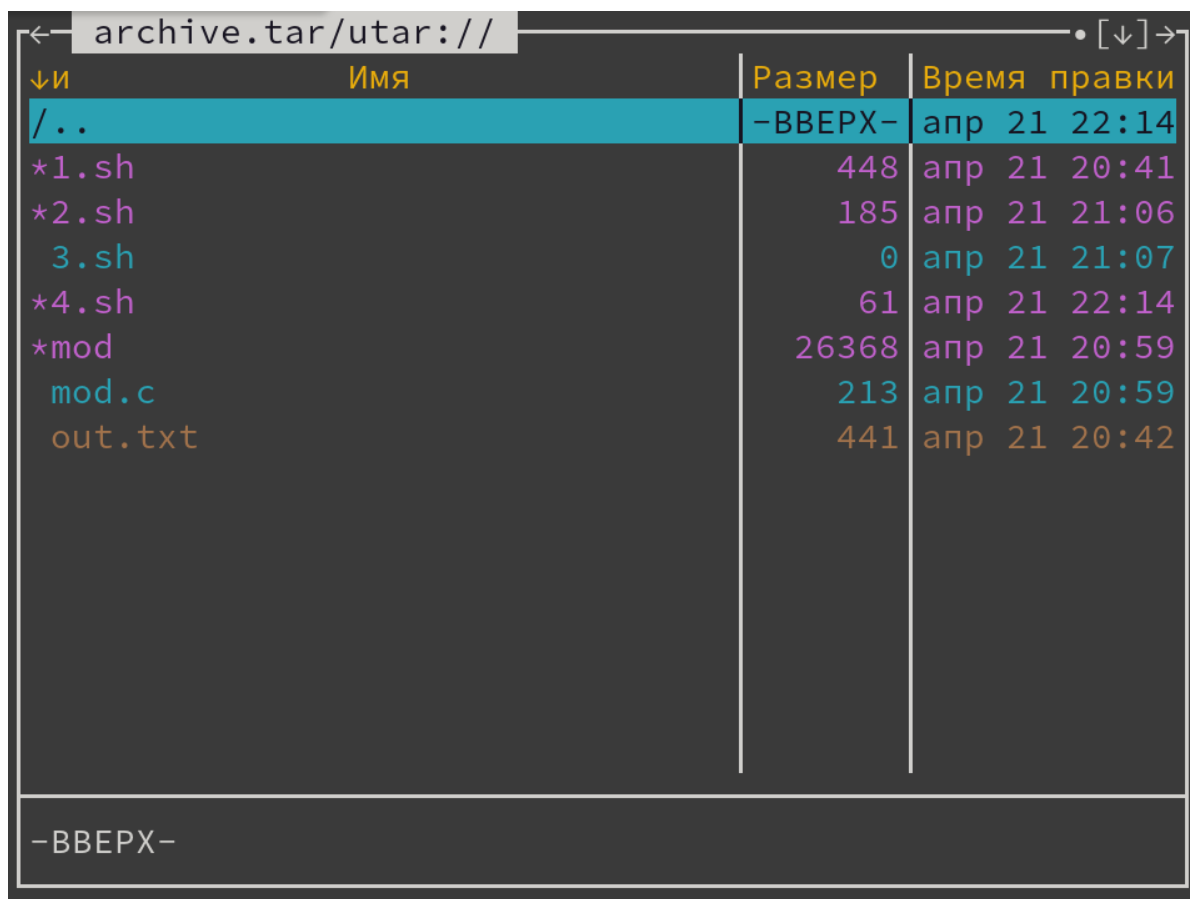
---

**Листинг 2.5** Текст командного файла Задания №4.

---

```
1 #!/bin/bash
2 tar -c -f archive.tar $(find -type f -mtime -7)
```

---



Имя	Размер	Время правки
/..	-ВВЕРХ-	апр 21 22:14
*1.sh	448	апр 21 20:41
*2.sh	185	апр 21 21:06
3.sh	0	апр 21 21:07
*4.sh	61	апр 21 22:14
*mod	26368	апр 21 20:59
mod.c	213	апр 21 20:59
out.txt	441	апр 21 20:42

-ВВЕРХ-

Рис. 2.4: Пример работы программы 4.sh.

## **3 Выводы**

В рамках лабораторной работы научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.