

# **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №11**

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

Анастасия Павловна Баранова, НБИбд-01-21

# Содержание

|          |                                       |           |
|----------|---------------------------------------|-----------|
| <b>1</b> | <b>Цель работы</b>                    | <b>4</b>  |
| <b>2</b> | <b>Выполнение лабораторной работы</b> | <b>5</b>  |
| <b>3</b> | <b>Вывод</b>                          | <b>10</b> |
| <b>4</b> | <b>Ответы на контрольные вопросы</b>  | <b>11</b> |

## Список иллюстраций

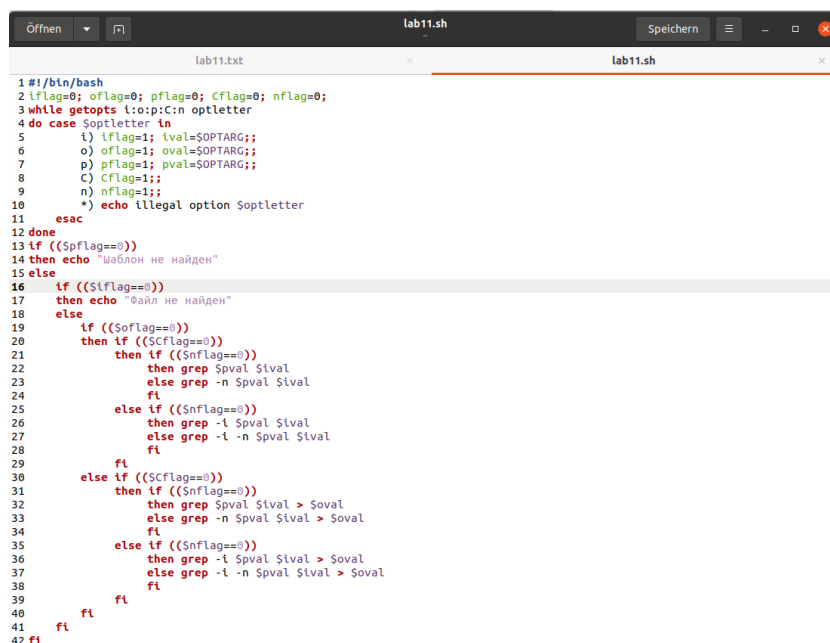
|      |   |   |
|------|---|---|
| 2.1  | Напишу командный файл. . . . .                | 5 |
| 2.2  | Указанный файл с текстом. . . . .             | 6 |
| 2.3  | Демонстрирую работу командного файла. . . . . | 6 |
| 2.4  | Демонстрирую работу командного файла. . . . . | 6 |
| 2.5  | Напишу на языке Си программу. . . . .         | 7 |
| 2.6  | Напишу командный файл. . . . .                | 7 |
| 2.7  | Демонстрирую работу командного файла. . . . . | 7 |
| 2.8  | Напишу командный файл. . . . .                | 8 |
| 2.9  | Демонстрирую работу командного файла. . . . . | 8 |
| 2.10 | Демонстрирую работу командного файла. . . . . | 8 |
| 2.11 | Демонстрирую работу командного файла. . . . . | 8 |
| 2.12 | Напишу командный файл. . . . .                | 9 |
| 2.13 | Демонстрирую работу командного файла. . . . . | 9 |

# 1 Цель работы

Целью данной работы является изучить основы программирования в оболочке ОС UNIX и научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 2 Выполнение лабораторной работы

Используя команды `getopts` `grep`, напишу командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — вывести данные в указанный файл; `-р`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 `-р`.(рис. 2.1, рис. 2.2, рис. 2.3, рис. 2.4)



```
1 #!/bin/bash
2 iflag=0; oflag=0; pflag=0; cflag=0; nflag=0;
3 while getopts i:op:C:n optletter
4 do case $optletter in
5     i) iflag=1; ival=$OPTARG;;
6     o) oflag=1; oval=$OPTARG;;
7     p) pflag=1; pval=$OPTARG;;
8     C) cflag=1;;
9     n) nflag=1;;
10    *) echo illegal option $optletter
11    esac
12 done
13 if (($pflag==0))
14 then echo "Шаблон не найден"
15 else
16     if (($iflag==0))
17     then echo "Файл не найден"
18     else
19         if (($oflag==0))
20         then if (($cflag==0))
21             then if (($nflag==0))
22                 then grep $pval $ival
23                 else grep -n $pval $ival
24                 fi
25             else if (($nflag==0))
26                 then grep -i $pval $ival
27                 else grep -i -n $pval $ival
28                 fi
29             fi
30         else if (($cflag==0))
31             then if (($nflag==0))
32                 then grep $pval $ival > $oval
33                 else grep -n $pval $ival > $oval
34                 fi
35             else if (($nflag==0))
36                 then grep -i $pval $ival > $oval
37                 else grep -i -n $pval $ival > $oval
38                 fi
39             fi
40         fi
41     fi
42 fi
```

Рис. 2.1: Напишу командный файл.

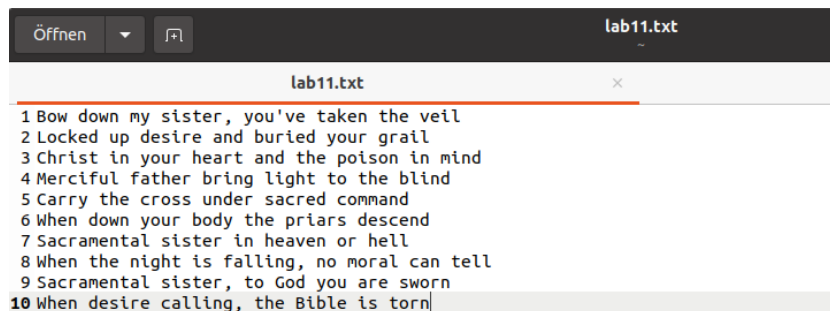


Рис. 2.2: Указанный файл с текстом.

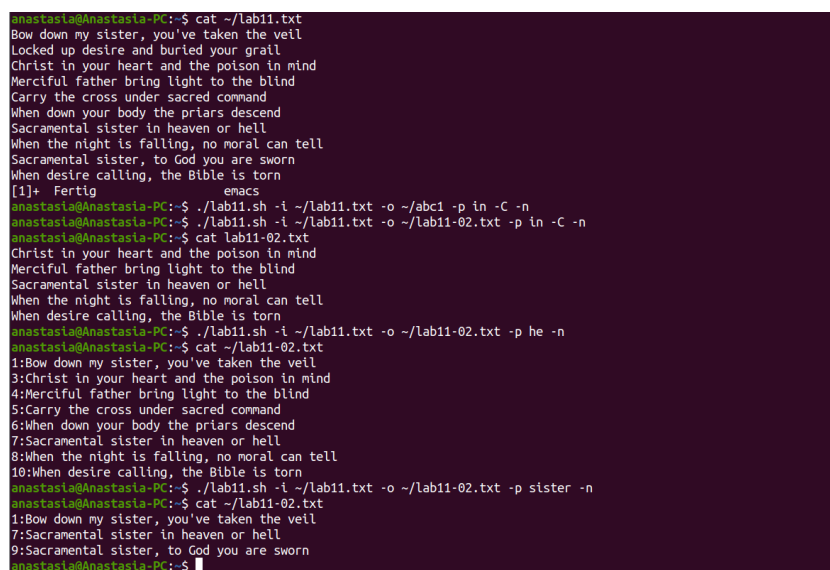


Рис. 2.3: Демонстрирую работу командного файла.

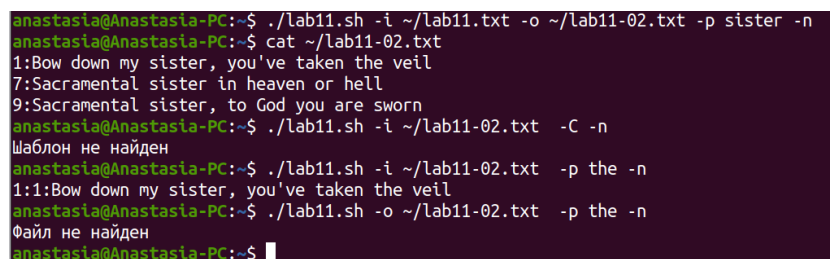


Рис. 2.4: Демонстрирую работу командного файла.

Напишу 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

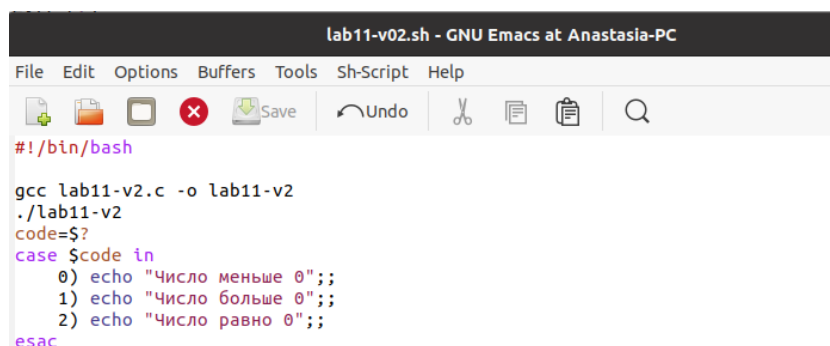
оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено. (рис. 2.5, рис. 2.6, рис. 2.7)



The screenshot shows a code editor window titled 'lab11-v2.c'. It contains a C program that prompts the user to enter a number and then checks if it is less than, greater than, or equal to zero. The code is as follows:

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

Рис. 2.5: Напишу на языке Си программу.

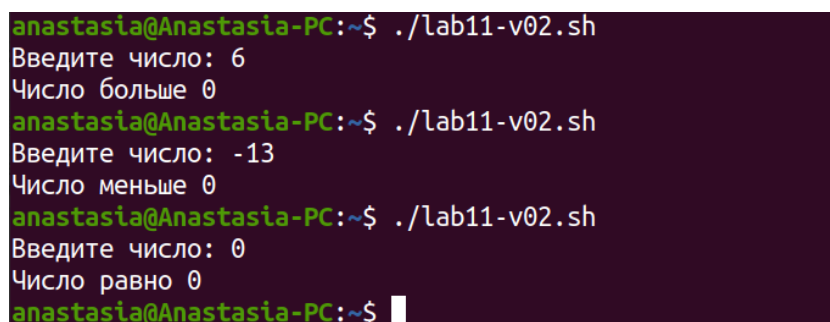


The screenshot shows a shell script editor window titled 'lab11-v02.sh - GNU Emacs at Anastasia-PC'. The script uses the `case` statement to check the exit code of the C program and print the appropriate message. The code is as follows:

```
#!/bin/bash

gcc lab11-v2.c -o lab11-v2
./lab11-v2
code=$?
case $code in
  0) echo "Число меньше 0";;
  1) echo "Число больше 0";;
  2) echo "Число равно 0";;
esac
```

Рис. 2.6: Напишу командный файл.



The screenshot shows a terminal window with the following output:

```
anastasia@Anastasia-PC:~$ ./lab11-v02.sh
Введите число: 6
Число больше 0
anastasia@Anastasia-PC:~$ ./lab11-v02.sh
Введите число: -13
Число меньше 0
anastasia@Anastasia-PC:~$ ./lab11-v02.sh
Введите число: 0
Число равно 0
anastasia@Anastasia-PC:~$
```

Рис. 2.7: Демонстрирую работу командного файла.

Напишу 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. 2.8, рис. 2.9, рис. 2.10, рис. 2.11)

```

1#!/bin/bash
2
3opt=$1;
4form=$2;
5num=$3;
6function Files() {
7    for ((i=1; i<=$num; i++)) do
8        file=$(echo $form | tr '#' '$i')
9        if [ $opt == "-r" ]
10        then
11            rm -f $file
12        elif [ $opt == "-c" ]
13        then
14            touch $file
15        fi
16    done
17}
18Files

```

Рис. 2.8: Напишу командный файл.

```

anastasia@Anastasia-PC:~$ ./lab11-v03.sh -c y#.txt 4
anastasia@Anastasia-PC:~$ ls

```

Рис. 2.9: Демонстрирую работу командного файла.

```

y1.txt
y2.txt
y3.txt
y4.txt

```

Рис. 2.10: Демонстрирую работу командного файла.

```

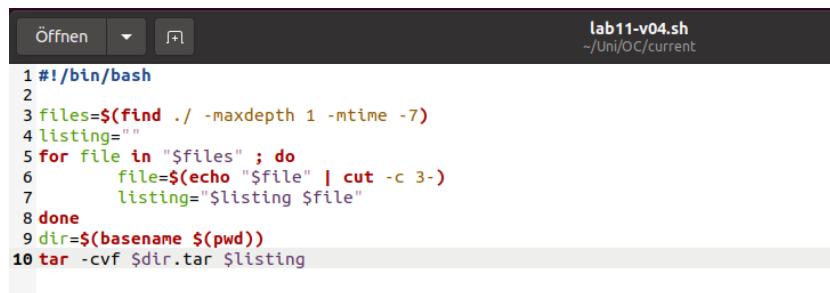
anastasia@Anastasia-PC:~$ ./lab11-v03.sh -r y#.txt 4
anastasia@Anastasia-PC:~$ ls

```

Рис. 2.11: Демонстрирую работу командного файла.

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

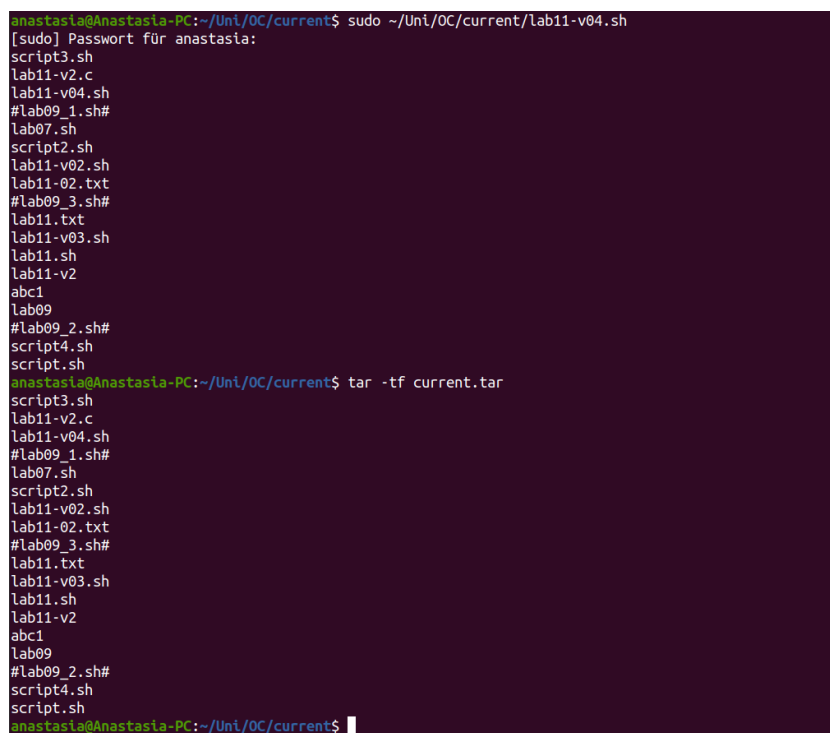




```
lab11-v04.sh
~/Uni/OC/current

1 #!/bin/bash
2
3 files=$(find ./ -maxdepth 1 -mtime -7)
4 listing=""
5 for file in "$files" ; do
6     file=$(echo "$file" | cut -c 3-)
7     listing="$listing $file"
8 done
9 dir=$(basename $(pwd))
10 tar -cvf $dir.tar $listing
```

Рис. 2.12: Напишу командный файл.



```
anastasia@Anastasia-PC:~/Uni/OC/current$ sudo ~/Uni/OC/current/lab11-v04.sh
[sudo] Passwort für anastasia:
script3.sh
lab11-v2.c
lab11-v04.sh
#lab09_1.sh#
lab07.sh
script2.sh
lab11-v02.sh
lab11-02.txt
#lab09_3.sh#
lab11.txt
lab11-v03.sh
lab11.sh
lab11-v2
abc1
lab09
#lab09_2.sh#
script4.sh
script.sh
anastasia@Anastasia-PC:~/Uni/OC/current$ tar -tf current.tar
script3.sh
lab11-v2.c
lab11-v04.sh
#lab09_1.sh#
lab07.sh
script2.sh
lab11-v02.sh
lab11-02.txt
#lab09_3.sh#
lab11.txt
lab11-v03.sh
lab11.sh
lab11-v2
abc1
lab09
#lab09_2.sh#
script4.sh
script.sh
anastasia@Anastasia-PC:~/Uni/OC/current$
```

Рис. 2.13: Демонстрирую работу командного файла.

## **3 Вывод**

В ходе данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX/Linux, научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 4 Ответы на контрольные вопросы

1. Каково предназначение команды `getopts`? Ответ: Команда `getopts` является встроенной командой командной оболочки `bash`, предназначенной для разбора параметров сценариев. Она обрабатывает исключительно однокбуквенные параметры как с аргументами, так и без них и этого вполне достаточно для передачи сценариям любых входных данных.
2. Какое отношение метасимволы имеют к генерации имён файлов? Ответ: При генерации имен используют метасимволы:

произвольная (возможно пустая) последовательность символов; ? один произвольный символ; [...] любой из символов, указанных в скобках перечислением и/или с указанием диапазона; `cat f*` выдаст все файлы каталога, начинающиеся с “f”; `cat f` выдаст все файлы, содержащие “f”; `cat program.?` выдаст файлы данного каталога с однобуквенными расширениями, скажем “program.c” и “program.o”, но не выдаст “program.com”; `cat [a-d]*` выдаст файлы, которые начинаются с “a”, “b”, “c”, “d”. Аналогичный эффект дадут и команды “`cat [abcd]`” и “`cat [bdac]`”.

3. Какие операторы управления действиями вы знаете? Ответ: Операторы `&&` и `||` являются управляющими операторами. Если в командной строке стоит `command1 && command2`, то `command2` выполняется в том, и только в том случае, если статус выхода из команды `command1` равен нулю, что говорит об успешном ее завершении. Аналогично, если командная строка имеет вид `command1 || command2`, то команда `command2` выполняется тогда, и только тогда, когда статус выхода из команды `command1` отличен от нуля.

4. Какие операторы используются для прерывания цикла? Ответ: Оператор `break` завершает выполнение ближайшего включающего цикла или условного оператора, в котором он отображается.
5. Для чего нужны команды `false` и `true`? Ответ: Команда `true` всегда возвращает ноль в качестве выходного статуса для индикации успеха. Команда `false` всегда возвращает не-ноль в качестве выходного статуса для индикации неудачи. Во всех управляющих конструкциях в качестве логического значения используется код возврата из программы, указанной в качестве условия. Код возврата 0 – истина, любое другое значение – ложь. Программа `true` – всегда завершается с кодом 0, `false` – всегда завершается с кодом 1.
6. Что означает строка `if test -f mans/i.s, ? : mans/i.s`
7. Объясните различия между конструкциями `while` и `until`. Ответ: Цикл `While` выполняется до тех пор, пока указанное в нем условие истинно. Когда указанное условие становится ложным - цикл завершается. Цикл `Until` выполняется до тех пор, пока указанное в нем условие ложно.