

# **Отчёт по лабораторной работе №10:**

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

Касакьянц Владислав Сергеевич

# Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	12
5	Выводы	14
	Список литературы	15

## Список иллюстраций

3.1	Командный файл №1 . . . . .	7
3.2	Создание нужных файлов . . . . .	8
3.3	Результат выполнения командного файла №1 . . . . .	8
3.4	Код на СИ . . . . .	9
3.5	Командный файл . . . . .	9
3.6	Результат выполнения командного файла №2 . . . . .	9
3.7	Командный файл №3 . . . . .	10
3.8	Результат выполнения командного файла №3 . . . . .	10
3.9	Командный файл №4 . . . . .	10
3.10	Результат выполнения командного файла №4 . . . . .	11

# 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:

- `-i` - `inputfile` — прочитать данные из указанного файла;
- `-o` - `outputfile` — вывести данные в указанный файл;
- `-p` - 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 `-p`.

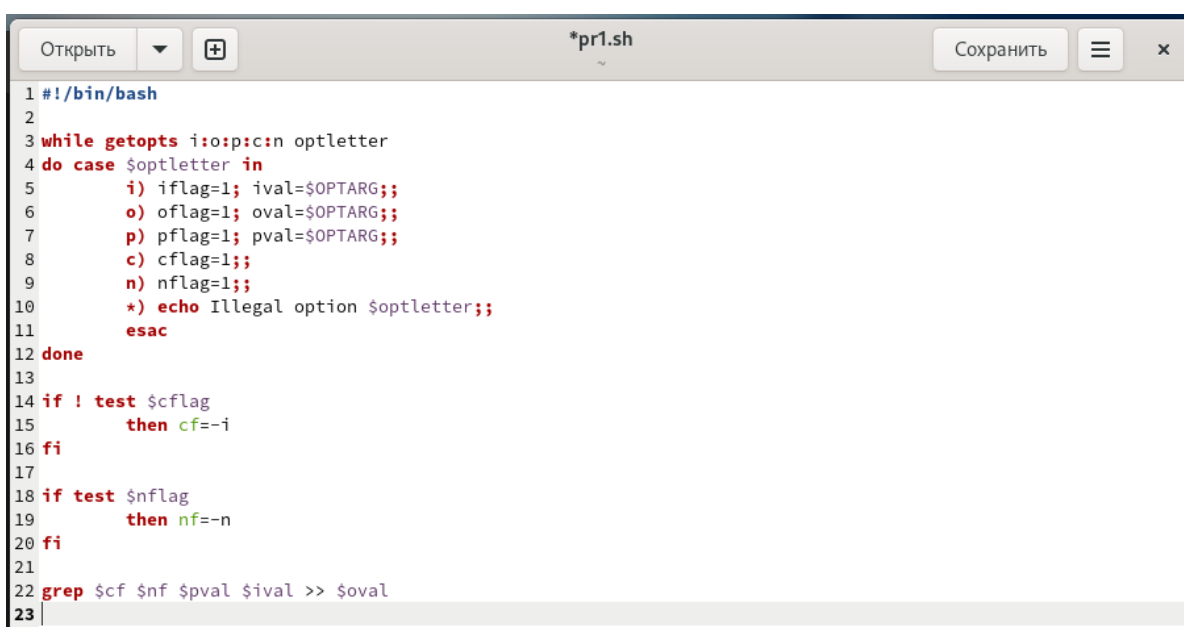
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с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

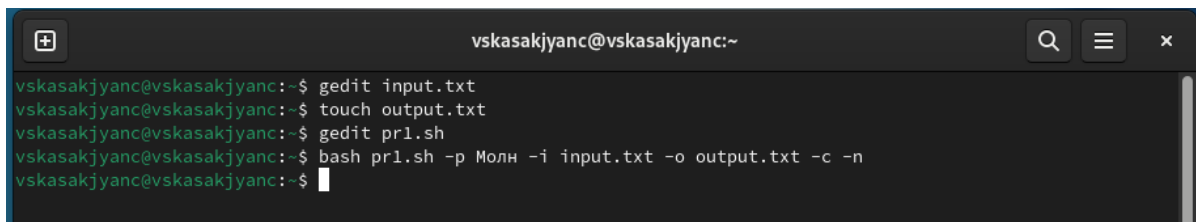
1. Используя команды `getopts` `grep`, напомним командный файл, который анализирует командную строку с ключами (`-i`, `-o`, `-p`, `-c`, `-n`), а затем ищет в указанном файле нужные строки, определяемые ключом `-p` (рис. 3.1):



```
1 #!/bin/bash
2
3 while getopts i:o:p: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
14 if ! test $cflag
15 then cf=-i
16 fi
17
18 if test $nflag
19 then nf=-n
20 fi
21
22 grep $cf $nf $pval $ival >> $oval
23
```

Рис. 3.1: Командный файл №1

Создадим один текстовый файл со стихотворением “input.txt” и файл, в который будет записываться результат “output.txt” (рис. 3.2), (рис. 3.3).



```
vskasakjyanc@vskasakjyanc:~$ gedit input.txt
vskasakjyanc@vskasakjyanc:~$ touch output.txt
vskasakjyanc@vskasakjyanc:~$ gedit pr1.sh
vskasakjyanc@vskasakjyanc:~$ bash pr1.sh -p Молн -i input.txt -o output.txt -c -n
vskasakjyanc@vskasakjyanc:~$
```

Рис. 3.2: Создание нужных файлов

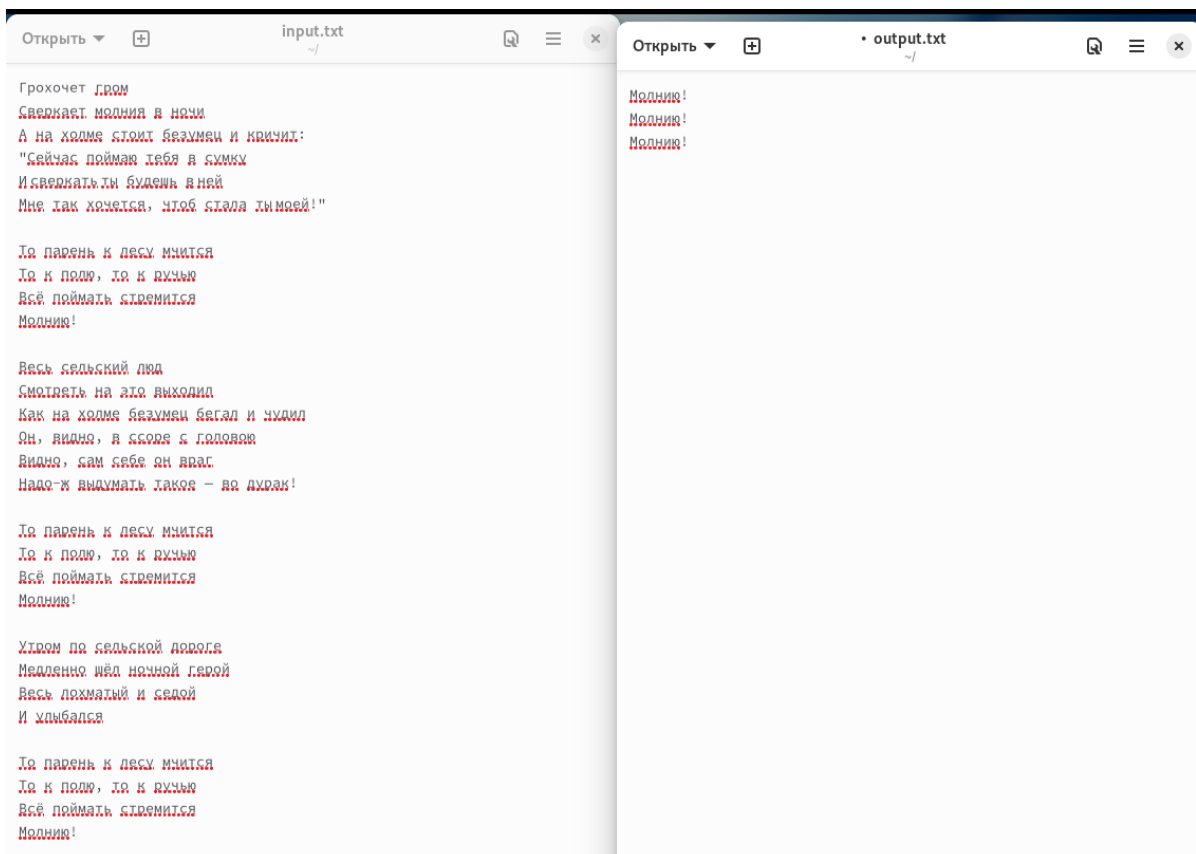
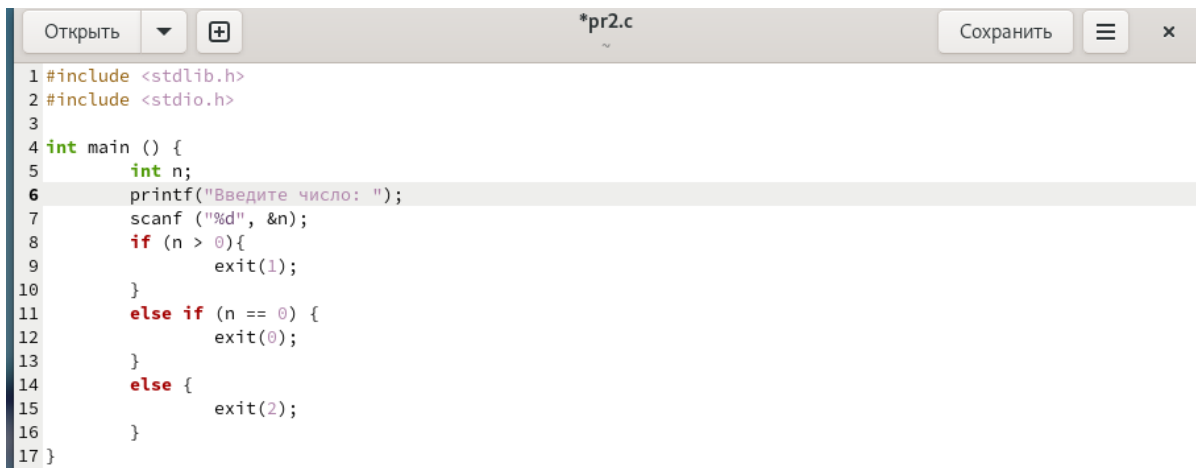


Рис. 3.3: Результат выполнения командного файла №1

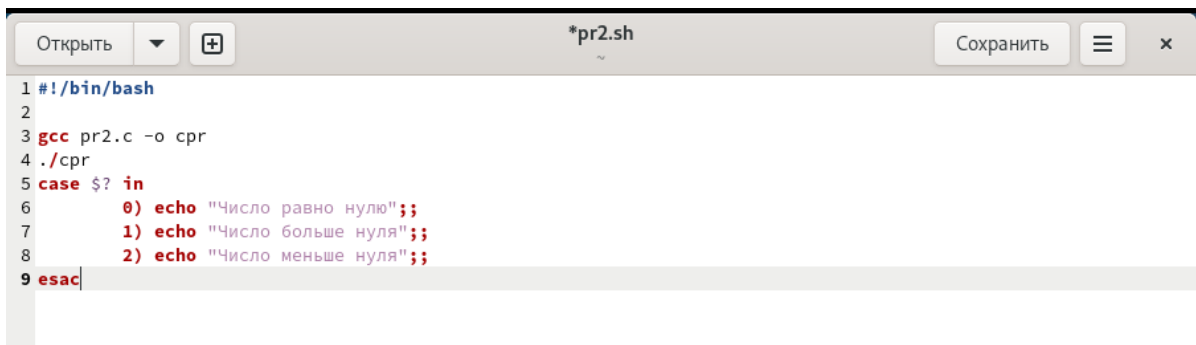
2. Напишем на языке Си программу (рис. 3.4), которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию о коде завершения в оболочку. Командный файл (рис. 3.5) должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено (рис. 3.6):





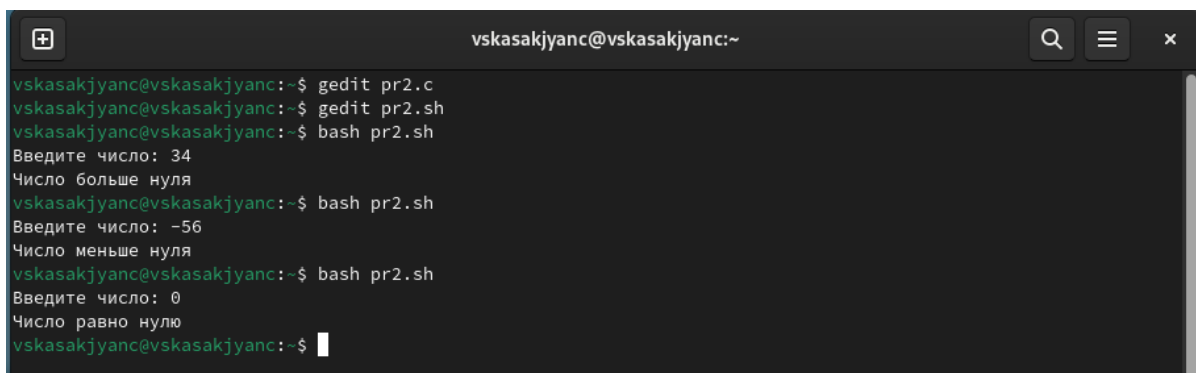
```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main () {
5     int n;
6     printf("Введите число: ");
7     scanf ("%d", &n);
8     if (n > 0){
9         exit(1);
10    }
11    else if (n == 0) {
12        exit(0);
13    }
14    else {
15        exit(2);
16    }
17 }
```

Рис. 3.4: Код на СИ



```
1 #!/bin/bash
2
3 gcc pr2.c -o cpr
4 ./cpr
5 case $? in
6     0) echo "Число равно нулю";;
7     1) echo "Число больше нуля";;
8     2) echo "Число меньше нуля";;
9 esac
```

Рис. 3.5: Командный файл

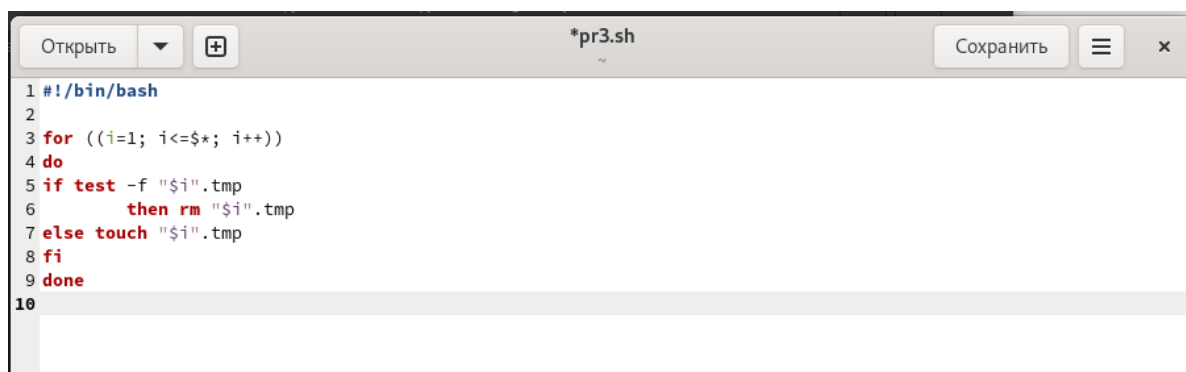


```
vskasakjyanc@vskasakjyanc:~$ gedit pr2.c
vskasakjyanc@vskasakjyanc:~$ gedit pr2.sh
vskasakjyanc@vskasakjyanc:~$ bash pr2.sh
Введите число: 34
Число больше нуля
vskasakjyanc@vskasakjyanc:~$ bash pr2.sh
Введите число: -56
Число меньше нуля
vskasakjyanc@vskasakjyanc:~$ bash pr2.sh
Введите число: 0
Число равно нулю
vskasakjyanc@vskasakjyanc:~$
```

Рис. 3.6: Результат выполнения командного файла №2

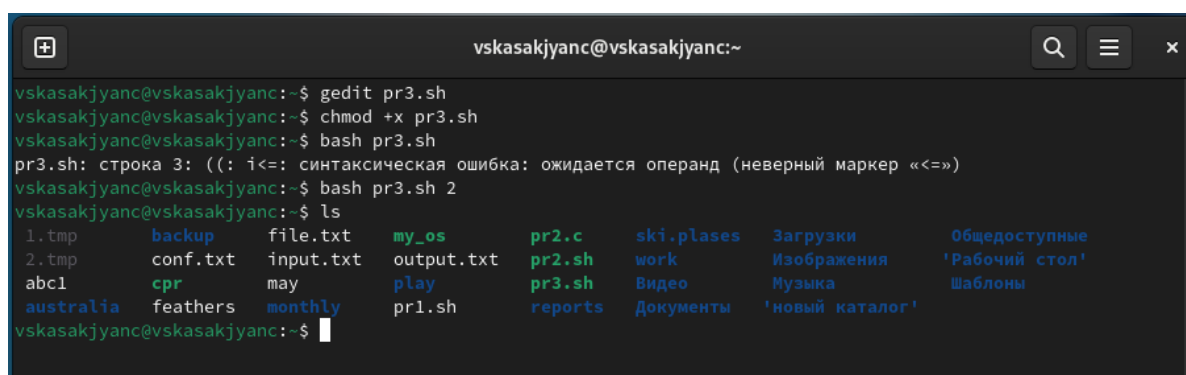
3. Напишем командный файл (рис. 3.7), создающий указанное число файлов, пронумерованных последовательно от 1 до N. Число файлов, которые необ-

ходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (рис. 3.8):



```
1 #!/bin/bash
2
3 for ((i=1; i<=$*; i++))
4 do
5 if test -f "$i".tmp
6 then rm "$i".tmp
7 else touch "$i".tmp
8 fi
9 done
10
```

Рис. 3.7: Командный файл №3



```
vskasakjyanc@vskasakjyanc:~$ gedit pr3.sh
vskasakjyanc@vskasakjyanc:~$ chmod +x pr3.sh
vskasakjyanc@vskasakjyanc:~$ bash pr3.sh
pr3.sh: строка 3: ((: i<=: синтаксическая ошибка: ожидается операнд (неверный маркер «<=»)
vskasakjyanc@vskasakjyanc:~$ bash pr3.sh 2
vskasakjyanc@vskasakjyanc:~$ ls
1.tmp      backup    file.txt  my_os     pr2.c     ski.plases  Загрузки  Общедоступные
2.tmp      conf.txt  input.txt output.txt pr2.sh     work        Изображения  'Рабочий стол'
abc1      cpr       may       play      pr3.sh     Видео       Музыка       Шаблоны
australia feathers  monthly   pr1.sh     reports    Документы  'новый каталог'
```

Рис. 3.8: Результат выполнения командного файла №3

4. Напишем командный файл (рис. 3.9), 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, используя команду find (рис. 3.10):



```
1 #!/bin/bash
2
3 find $* -mtime -7 -mtime +0 -type f > files.txt
4 tar -cf archive.tar -T files.txt
5
```

Рис. 3.9: Командный файл №4

```
vskasakjyanc@vskasakjyanc:~  
vskasakjyanc@vskasakjyanc:~$ gedit pr4.sh  
vskasakjyanc@vskasakjyanc:~$ bash pr4.sh  
vskasakjyanc@vskasakjyanc:~$ ls  
1.tmp      backup      file.txt    output.txt  pr3.sh      Видео      'новый каталог'  
2.tmp      conf.txt    input.txt   play        pr4.sh      Документы  Общедоступные  
abc1       cpr         may         pr1.sh      reports     Загрузки   'Рабочий стол'  
archive.tar feathers    monthly     pr2.c       ski.places  Изображения Шаблоны  
australia  files.txt   my_os       pr2.sh      work        Музыка  
vskasakjyanc@vskasakjyanc:~$
```

Рис. 3.10: Результат выполнения командного файла №4

## 4 Контрольные вопросы

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

Команда `getopts` используется для обработки аргументов командной строки. Она позволяет извлекать опции и их значения из списка аргументов.

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

Метасимволы используются в генерации имён файлов для сопоставления шаблонов. Например, звездочка (\*) сопоставляет любое количество символов, а знак вопроса (?) сопоставляет любой один символ.

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

Операторы управления действиями используются для изменения потока выполнения скрипта. Вот некоторые из наиболее распространенных операторов управления действиями:

- **if...then...else:** Выполняет блок кода, если условие истинно. Если условие ложно, выполняется блок кода `else` (необязательно).
- **case...esac:** Выполняет блок кода в зависимости от значения переменной.
- **for...do...done:** Выполняет блок кода для каждого элемента в списке.
- **while...do...done:** Выполняет блок кода, пока условие истинно.
- **until...do...done:** Выполняет блок кода, пока условие ложно.

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

- **break:** Немедленно выходит из цикла.
- **continue:** Переходит к следующей итерации цикла, пропуская оставшиеся операторы в текущей итерации.

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

Команды **false** и **true** используются для возврата кода выхода, указывающего на успех (**true**) или неудачу (**false**).

#### 6. Что означает строка **if test -f man*s*/i.\$s**, встреченная в командном файле?

Эта строка проверяет, существует ли файл с именем **man*s*/i.\$s**. Если файл существует, выполняется оператор **then**.

#### 7. Объясните различия между конструкциями **while** и **until**.

- **while:** Выполняет блок кода, пока условие истинно.
- **until:** Выполняет блок кода, пока условие ложно.

## 5 Выводы

В данной лабораторной работе мы изучили основы программирования в оболочке ОС UNIX, а также научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## **Список литературы**