

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

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

Федорова Наталия Артемовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>8</b>
3.1	Переменные в языке программирования bash . . . . .	8
3.2	Использование арифметических вычислений. Операторы let и read . .	9
3.3	Командные файлы и функции . . . . .	9
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>10</b>
<b>5</b>	<b>Контрольные вопросы</b>	<b>16</b>
<b>6</b>	<b>Выводы</b>	<b>18</b>

## Список иллюстраций

4.1	Командный файл №1 . . . . .	10
4.2	Создание нужных файлов . . . . .	11
4.3	Результат выполнения командного файла №1 . . . . .	11
4.4	Код на СИ . . . . .	12
4.5	Код bash . . . . .	12
4.6	Результат выполнения командного файла №2 . . . . .	13
4.7	Командный файл №3 . . . . .	13
4.8	Результат выполнения командного файла №3 . . . . .	14
4.9	Командный файл №4 . . . . .	14
4.10	Результат выполнения командного файла №4 . . . . .	15

## **Список таблиц**

# 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 запа-

ковывались только те файлы, которые были изменены менее недели тому назад (использовать команду `find`).

## 3 Теоретическое введение

### 3.1 Переменные в языке программирования bash

Командный процессор `bash` обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов.

Например, команда

```
mark=/usr/andy/bin
```

переместит файл `afle` из текущего каталога в каталог с абсолютным полным именем `/usr/andy/bin`. Использование значения, присвоенного некоторой переменной, называется подстановкой. Для того чтобы имя переменной не сливалось с символами, которые могут следовать за ним в командной строке, при подстановке в общем случае используется следующая форма записи:

```
${имя переменной}
```

Оболочка `bash` позволяет работать с массивами. Для создания массива используется команда `set` с флагом `-A`. За флагом следует имя переменной, а затем список значений, разделённых пробелами. Например,

```
set -A states Delaware Michigan "New Jersey"
```



## 3.2 Использование арифметических вычислений.

### Операторы let и read

Команда let берет два операнда и присваивает их переменной. Положительным моментом команды let можно считать то, что для идентификации переменной ей не нужен знак доллара; вы можете писать команды типа `let sum=x+7`, и let будет искать переменную x и добавлять к ней 7.

Команда let также расширяет другие выражения let, если они заключены в двойные круглые скобки. Таким способом вы можете создавать довольно сложные выражения. Команда let не ограничена простыми арифметическими выражениями.

Команда read позволяет читать значения переменных со стандартного ввода:

```
echo "Please enter Month and Day of Birth ?"  
read mon day trash
```

## 3.3 Командные файлы и функции

Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде:


```
bash командный_файл [аргументы]
```

Чтобы не вводить каждый раз последовательности символов bash, необходимо изменить код защиты этого командного файла, обеспечив доступ к этому файлу по выполнению. Это может быть сделано с помощью команды

```
chmod +x имя_файла
```

## 4 Выполнение лабораторной работы

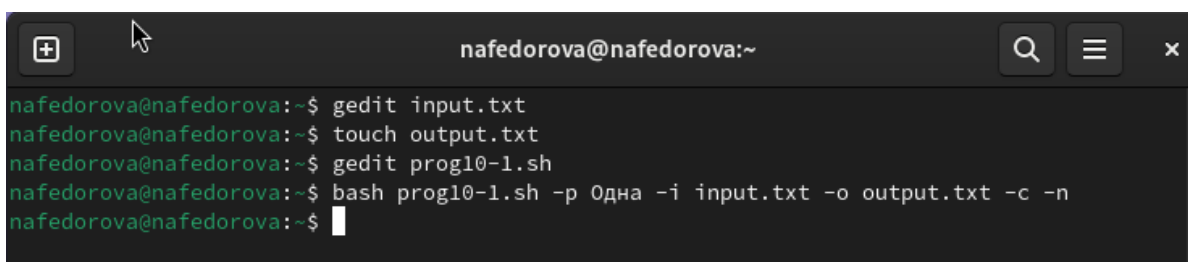
1. Используя команды `getopts` `grep`, нужно написать командный файл, который анализирует командную строку с ключами (`-i`, `-o`, `-p`, `-c`, `-n`), а затем ищет в указанном файле нужные строки, определяемые ключом `-p` (рис. 4.1):



```
1 #!/bin/bash
2
3 while getopts i:o:p:cn 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
```

Рис. 4.1: Командный файл №1

Создаю один текстовый файл со стихотворением “input.txt” и файл, в который будет записываться результат “output.txt”. Делаю файл “prog10-1.sh” исполняемым и вывожу результат (рис. 4.2), (рис. 4.3).



```
nafedorova@nafedorova:~$ gedit input.txt
nafedorova@nafedorova:~$ touch output.txt
nafedorova@nafedorova:~$ gedit prog10-1.sh
nafedorova@nafedorova:~$ bash prog10-1.sh -p Одна -i input.txt -o output.txt -c -n
nafedorova@nafedorova:~$
```

Рис. 4.2: Создание нужных файлов

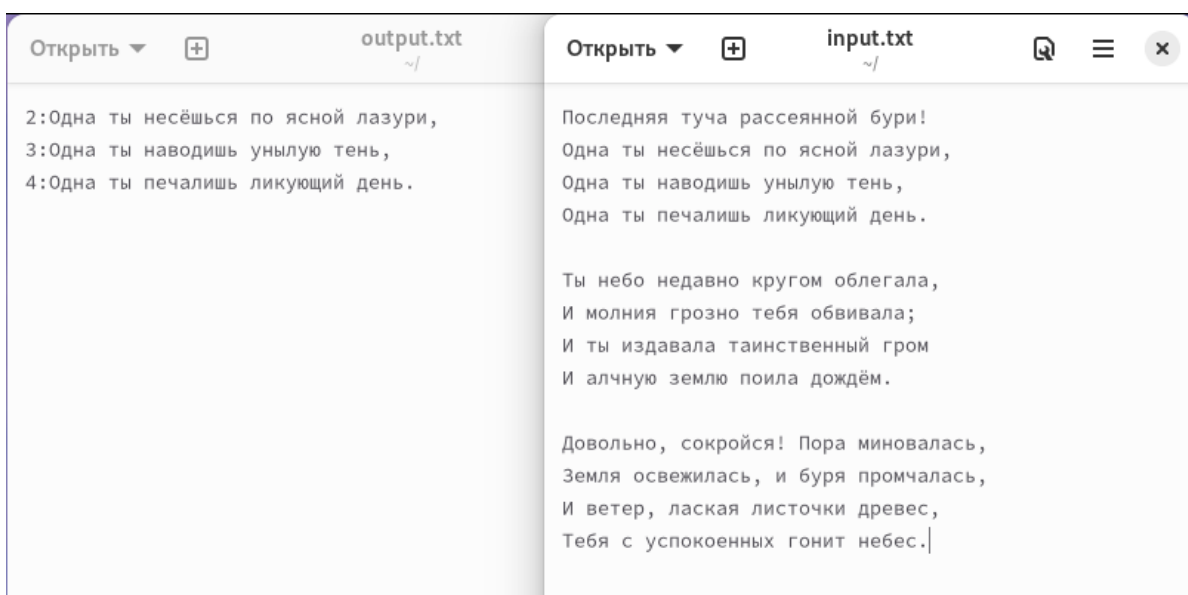
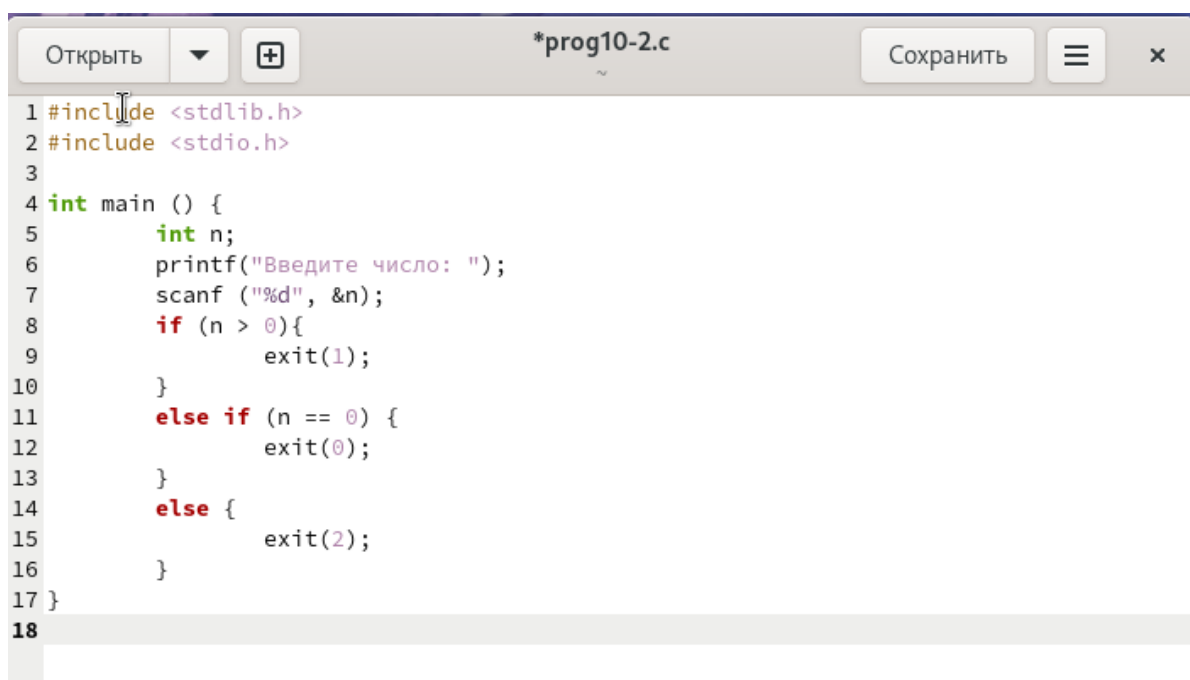


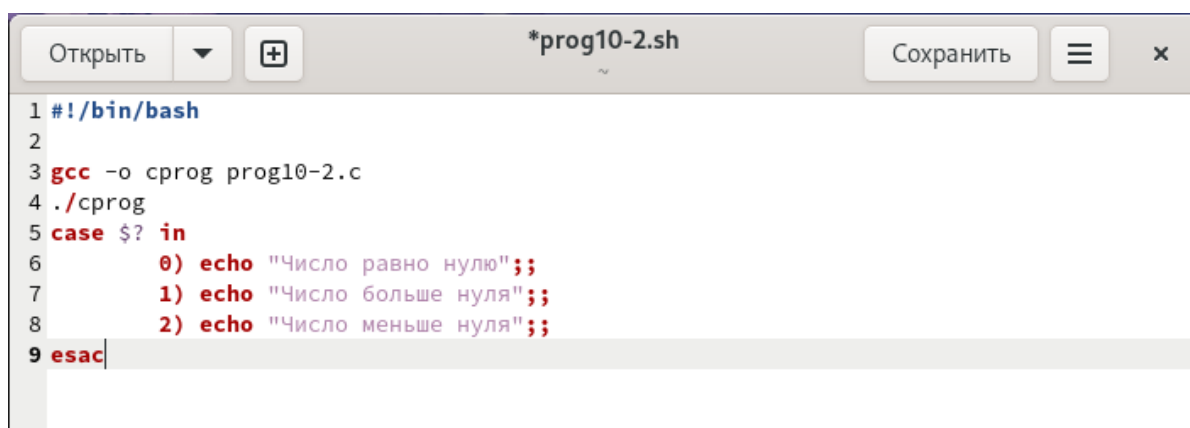
Рис. 4.3: Результат выполнения командного файла №1

2. Нужно 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 (рис. 4.4), (рис. 4.5):



```
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 }
18
```

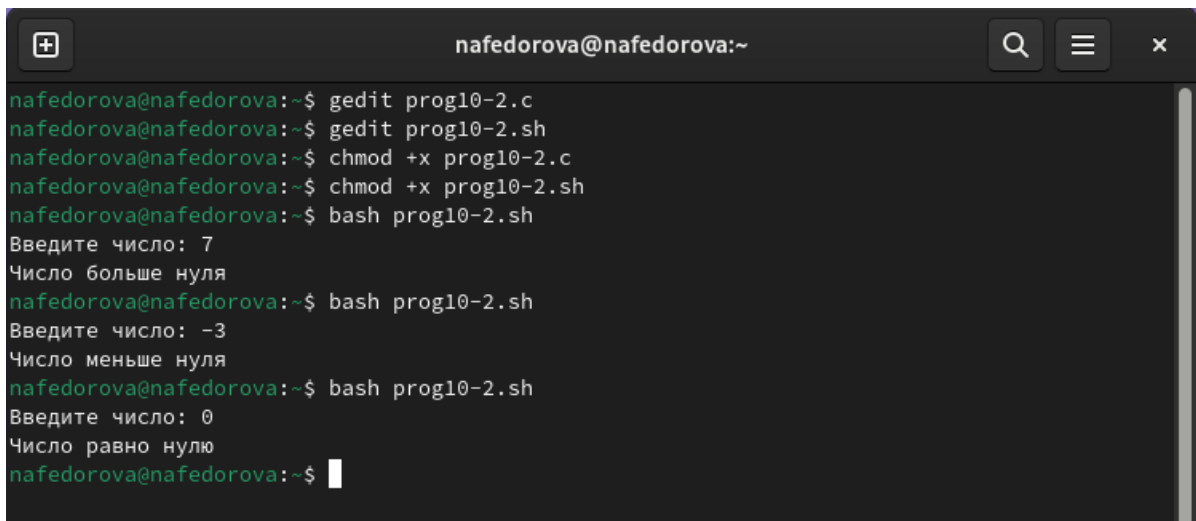
Рис. 4.4: Код на СИ



```
1 #!/bin/bash
2
3 gcc -o cprog prog10-2.c
4 ./cprog
5 case $? in
6     0) echo "Число равно нулю";;
7     1) echo "Число больше нуля";;
8     2) echo "Число меньше нуля";;
9 esac
```

Рис. 4.5: Код bash

Делаю файлы исполняемыми и вывожу результат (рис. 4.6).



```
nafedorova@nafedorova:~$ gedit prog10-2.c
nafedorova@nafedorova:~$ gedit prog10-2.sh
nafedorova@nafedorova:~$ chmod +x prog10-2.c
nafedorova@nafedorova:~$ chmod +x prog10-2.sh
nafedorova@nafedorova:~$ bash prog10-2.sh
Введите число: 7
Число больше нуля
nafedorova@nafedorova:~$ bash prog10-2.sh
Введите число: -3
Число меньше нуля
nafedorova@nafedorova:~$ bash prog10-2.sh
Введите число: 0
Число равно нулю
nafedorova@nafedorova:~$
```

Рис. 4.6: Результат выполнения командного файла №2

3. Нужно написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N. Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (рис. 4.7):



```
*prog10-3.sh
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
```

Рис. 4.7: Командный файл №3

Делаю файлы исполняемыми и вывожу результат (рис. 4.8).

```
nafedorova@nafedorova:~$ gedit prog10-3.sh
nafedorova@nafedorova:~$ chmod +x prog10-3.sh
nafedorova@nafedorova:~$ bash prog10-3.sh 3
nafedorova@nafedorova:~$ ls
1.tmp      cprog      output.txt  prog1.sh    work        Музыка
2.tmp      file.txt   play        prog2.sh    Видео       Общедоступные
3.tmp      id_rsa.pub prog10-1.sh  prog3.sh    Документы   'Рабочий стол'
australia  input.txt  prog10-2.c  prog4.sh    Загрузки    Шаблоны
backup     monthly   prog10-2.sh reports      Изображения
conf.txt   my_os     prog10-3.sh ski.places  Каталог

nafedorova@nafedorova:~$ bash prog10-3.sh 3
nafedorova@nafedorova:~$ ls
australia  id_rsa.pub  play        prog1.sh    ski.places  Изображения  Шаблоны
backup     input.txt   prog10-1.sh prog2.sh    work        Каталог
conf.txt   monthly     prog10-2.c  prog3.sh    Видео       Музыка
cprog      my_os       prog10-2.sh prog4.sh    Документы   Общедоступные
file.txt   output.txt  prog10-3.sh reports      Загрузки    'Рабочий стол'
nafedorova@nafedorova:~$
```

Рис. 4.8: Результат выполнения командного файла №3

4. Нужно 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, используя команду find (рис. 4.9):

```
Открыть  prog10-4.sh  Сохранить
1 #!/bin/bash
2
3 find $* -mtime -7 -mtime +0 -type f > Files.txt
4 tar -cf archive.tar -T Files.txt
```

Рис. 4.9: Командный файл №4

Делаю файлы исполняемыми и вывожу результат (рис. 4.10).

```
nafedorova@nafedorova:~$ gedit prog10-4.sh
nafedorova@nafedorova:~$ chmod +x prog10-4.sh
nafedorova@nafedorova:~$ ls
australia    input.txt    prog10-2.c   prog3.sh     Документы    'Рабочий стол'
backup       monthly     prog10-2.sh  prog4.sh     Загрузки     Шаблоны
conf.txt     my_os       prog10-3.sh  reports      Изображения
cprog       output.txt  prog10-4.sh  ski.plases   Каталог
file.txt     play        prog1.sh     work         Музыка
id_rsa.pub   prog10-1.sh prog2.sh     Видео        Общедоступные
nafedorova@nafedorova:~$ bash prog10-4.sh
nafedorova@nafedorova:~$ ls
archive.tar  file.txt    play         prog1.sh     work         Музыка
australia    id_rsa.pub  prog10-1.sh  prog2.sh     Видео        Общедоступные
backup       input.txt   prog10-2.c   prog3.sh     Документы    'Рабочий стол'
conf.txt     monthly    prog10-2.sh  prog4.sh     Загрузки     Шаблоны
cprog       my_os      prog10-3.sh  reports      Изображения
Files.txt    output.txt  prog10-4.sh  ski.plases   Каталог
```

Рис. 4.10: Результат выполнения командного файла №4

## 5 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.

## **6 Выводы**

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